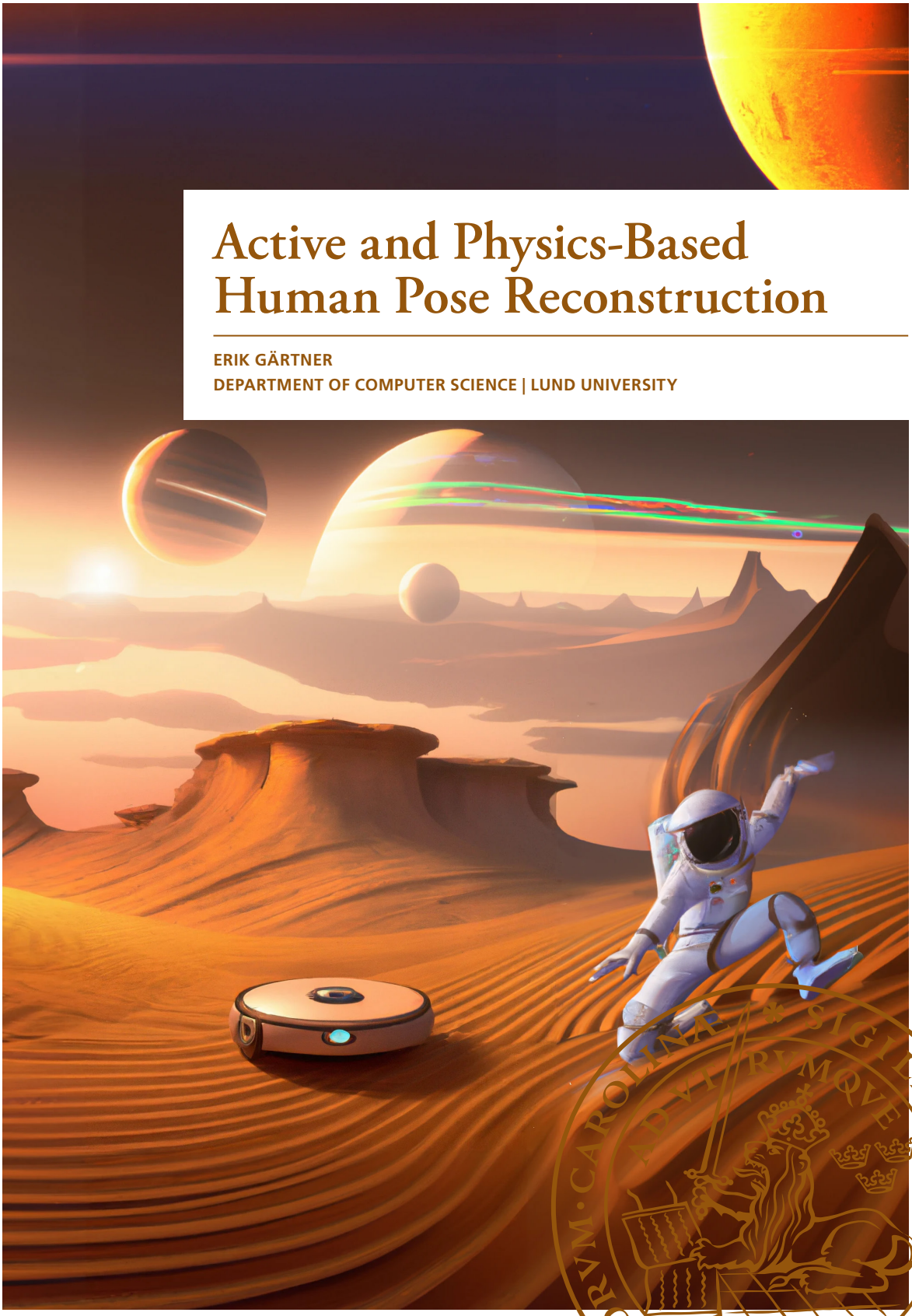
Active and Physics-Based Human Pose Reconstruction

Gärtner, Erik

2023

Document Version:
Publisher's PDF, also known as Version of record



Active and Physics-Based Human Pose Reconstruction

ERIK GÄRTNER
DEPARTMENT OF COMPUTER SCIENCE | LUND UNIVERSITY

Active and Physics-Based Human Pose Reconstruction

Active and Physics-Based Human Pose Reconstruction

by Erik Gärtner

LUND
UNIVERSITY

Thesis for the degree of Doctor of Philosophy

Prof. Cristian Sminchisescu	Supervisor
Dr. Elin A. Topp	Co-supervisor
Prof. Kalle Åström	Co-supervisor
Dr. Fahad Shahbaz Khan	Faculty opponent

To be presented, with the permission of the Faculty of Engineering of Lund University, for public criticism in the lecture hall MH:Hörmander at the Centre for Mathematical Sciences, Sölvegatan 18, Lund on Friday, the 13th of January 2023 at 10:15.

Organization
LUND UNIVERSITY

Department of Computer Science
Box 118
SE–221 00 LUND
Sweden

Author(s)
Erik Gärtner

Document name
DOCTORAL THESIS

Date of disputation
2023-01-13

Sponsoring organization
The Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Title and subtitle
Active and Physics-Based Human Pose Reconstruction

Abstract
Perceiving humans is an important and complex problem within computer vision. Its significance is derived from its numerous applications, such as human-robot interaction, virtual reality, markerless motion capture, and human tracking for autonomous driving. The difficulty lies in the variability in human appearance, physique, and plausible body poses. In real-world scenes, this is further exacerbated by difficult lighting conditions, partial occlusions, and the depth ambiguity stemming from the loss of information during the 3d to 2d projection. Despite these challenges, significant progress has been made in recent years, primarily due to the expressive power of deep neural networks trained on large datasets. However, creating large-scale datasets with 3d annotations is expensive, and capturing the vast diversity of the real world is demanding. Traditionally, 3d ground truth is captured using motion capture laboratories that require large investments. Furthermore, many laboratories cannot easily accommodate athletic and dynamic motions. This thesis studies three approaches to improving visual perception, with emphasis on human pose estimation, that can complement improvements to the underlying predictor or training data.

The first two papers present *active human pose estimation*, where a reinforcement learning agent is tasked with selecting informative viewpoints to reconstruct subjects efficiently. The papers discard the common assumption that the input is given and instead allow the agent to move to observe subjects from desirable viewpoints, e.g., those which avoid occlusions and for which the underlying pose estimator has a low prediction error.

The third paper introduces the task of *embodied visual active learning*, which goes further and assumes that the perceptual model is not pre-trained. Instead, the agent is tasked with exploring its environment and requesting annotations to refine its visual model. Learning to explore novel scenarios and efficiently request annotation for new data is a step towards life-long learning, where models can evolve beyond what they learned during the initial training phase. We study the problem for segmentation, though the idea is applicable to other perception tasks.

Lastly, the final two papers propose improving human pose estimation by integrating physical constraints. These regularize the reconstructed motions to be physically plausible and serve as a complement to current kinematic approaches. Whether a motion has been observed in the training data or not, the predictions should obey the laws of physics. Through integration with a physical simulator, we demonstrate that we can reduce reconstruction artifacts and enforce, e.g., contact constraints.

Key words
computer vision, human pose estimation, reinforcement learning, physics-based human pose estimation, active learning

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404-1219

ISBN
978-91-8039-471-0 (print)
978-91-8039-472-7 (pdf)

Recipient's notes

Number of pages
231

Price

Security classification

Active and Physics-Based Human Pose Reconstruction

by Erik Gärtner

LUND
UNIVERSITY

Cover illustration front: The author partially generated this image with DALL·E 2, OpenAI's large-scale image-generation model. The image is a composition of multiple outputs from DALL·E 2, together with manual edits by the author.

Funding information: The thesis work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Department of Computer Science
Faculty of Engineering
Lund University
Box 118
SE-221 00 Lund
Sweden

ISBN: 978-91-8039-471-0 (print)
ISBN: 978-91-8039-472-7 (pdf)
ISSN: 1404-1219
Dissertation 70, 2023
LU-CS-DISS: 2023-01

Typeset using LaTeX
Printed in Sweden by Media-Tryck, Lund University, Lund 2023

Dedicated to Anders and Cecilia.

Contents

Part I

Background

1 Abstract

Perceiving humans is an important and complex problem within computer vision. Its significance is derived from its numerous applications, such as human-robot interaction, virtual reality, markerless motion capture, and human tracking for autonomous driving. The difficulty lies in the variability in human appearance, physique, and plausible body poses. In real-world scenes, this is further exacerbated by difficult lighting conditions, partial occlusions, and the depth ambiguity stemming from the loss of information during the 3d to 2d projection. Despite these challenges, significant progress has been made in recent years, primarily due to the expressive power of deep neural networks trained on large datasets. However, creating large-scale datasets with 3d annotations is expensive, and capturing the vast diversity of the real world is demanding. Traditionally, 3d ground truth is captured using motion capture laboratories that require large investments. Furthermore, many laboratories cannot easily accommodate athletic and dynamic motions. This thesis studies three approaches to improving visual perception, with emphasis on human pose estimation, that can complement improvements to the underlying predictor or training data.

The first two papers present *active human pose estimation*, where a reinforcement learning agent is tasked with selecting informative viewpoints to reconstruct subjects efficiently. The papers discard the common assumption that the input is given and instead allow the agent to move to observe subjects from desirable viewpoints, e.g., those which avoid occlusions and for which the underlying pose estimator has a low prediction error.

The third paper introduces the task of *embodied visual active learning*, which goes further and assumes that the perceptual model is not pre-trained. Instead, the agent is tasked with exploring its environment and requesting annotations to refine its visual model. Learning to explore novel scenarios and efficiently request annotation for new data is a step towards life-long learning, where models can evolve beyond what they learned during the initial training phase. We study the problem for segmentation, though the idea is applicable to other perception tasks.

Lastly, the final two papers propose improving human pose estimation by integrating physical constraints. These regularize the reconstructed motions to be physically plausible and serve as a complement to current kinematic approaches. Whether a motion has been observed in the training data or not, the predictions should obey the laws of physics. Through integration with a physical simulator, we demonstrate that we can reduce reconstruction artifacts and enforce, e.g., contact constraints.

2 List of Publications

This thesis is based on the following publications, referred to by their Roman numerals:

I **Domes to Drones: Self-Supervised Active Triangulation for 3D Human Pose Reconstruction**
Aleksis Pirinen*, **Erik Gärtner***, Cristian Sminchisescu
Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2019

II **Deep Reinforcement Learning for Active Human Pose Estimation**
Erik Gärtner*, Aleksis Pirinen*, Cristian Sminchisescu
Association for the Advancement of Artificial Intelligence (AAAI), New York, USA, 2020

III **Embodied Visual Active Learning for Semantic Segmentation**
David Nilsson, Aleksis Pirinen, **Erik Gärtner**, Cristian Sminchisescu
Association for the Advancement of Artificial Intelligence (AAAI), Virtual conference, 2021

IV **Trajectory Optimization for Physics-Based Reconstruction of 3d Human Pose from Monocular Video**
Erik Gärtner, Mykhaylo Andriluka, Hongyi Xu, Cristian Sminchisescu
Computer Vision and Pattern Recognition (CVPR), New Orleans, USA, 2022

V **Differentiable Dynamics for Articulated 3d Human Motion Reconstruction**
Erik Gärtner, Mykhaylo Andriluka, Erwin Coumans, Cristian Sminchisescu
Computer Vision and Pattern Recognition (CVPR), New Orleans, USA, 2022

All papers are reproduced with permission of their respective publishers.

*Equal contribution.

Publications not included in this thesis, referred to by lowercase Roman numerals:

vi **Improving Usability of Search and Rescue Decision Support Systems: WARA-PS Case Study**
Veronika Domova, **Erik Gärtner**, Fredrik Präntare, Martin Pallin, Johan Källström, Nikita Korzhitskii
25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020

vii **Embodied Learning for Lifelong Visual Perception**
David Nilsson, Aleksis Pirinen, **Erik Gärtner**, Cristian Sminchisescu
arXiv preprint arXiv:2112.14084, 2021

viii **Transformer-Based Learned Optimization**
Erik Gärtner, Luke Metz, Mykhaylo Andriluka, C. Daniel Freeman, Cristian Sminchisescu
arXiv preprint arXiv:2212.01055, 2022

3 Acknowledgements

My five-year journey toward a doctoral degree has come to an end. In many ways, I took the first step when I peeked through the door to the world of research during my Master's project. Back then, I had no idea what hard work and satisfying rewards awaited behind that door. In my experience, pursuing a Ph.D. is an arduous journey that tests one's composure and necessitates the help of others. Therefore, I'm grateful I had Cristian as my supervisor and mentor throughout that journey. He inspired me with his passion for research and motivated me to strive toward impactful research. Thank you for staying up late at night, helping me write papers, and for the thoughtful discussion during dinners and in the alps. Lastly, I'm grateful for the internship where I had the chance to work with brilliant researchers in an international environment.

Next, I would like to thank my co-supervisors. Elin for her continued support and generous open-door policy. In particular, I would like to thank her for her guidance and feedback concerning this thesis. I also want to thank my co-supervisor, Kalle, for his infectious positivity, advice on various matters, and interest in my research.

My five years at the department would have been much less enjoyable without my great colleagues. In particular, I would like to thank the members of Cristian's research lab. Aleksis for his support and collaboration during my initial years, which resulted in two publications and innumerable internal jokes. Whether or not you realized it, you were a focal point in the group that drew us together with your many after-work activities. David, for the excellent collaboration that led to the third paper and the camaraderie during our time in Zürich. Henning for being a thoughtful yet immensely funny and warm person. I thoroughly enjoyed our late-night talks and competing together in the league. Maria, for bringing life to any discussion and always taking the time to help me. Martin, for being a funny and considerate office roommate. Ted, for the spirited conversations and for generously hosting BBQs in his beautiful home.

At conferences and through my internship, I met many fantastic researchers and collaborators. I want to thank Andrei, Mihai Z., Alin, and Elisabeta at IMAR for their generous help over the years. In addition, I would like to thank my collaborators at Google Research. Edi, Hongyi, and Mihai F. in Zürich, as well as Erwin, Daniel, and Luke in the U.S. Lastly, I would like to extend a special thanks to Misha for our long-standing and fruitful collaboration.

While scientific discoveries are wondrous, sharing them with others, especially those close to one's heart, gives them meaning. Therefore, I would be remiss if I didn't extend my sincere gratitude to Caroline, my friends, and of course, my parents for their unwavering support throughout these stressful years.

Finally, thanks to the Knut and Alice Wallenberg Foundation for financing AI research in general and my doctoral studies in particular. Its financial support gave me five exciting years of research, friendships, and personal development.

Erik Gärtner
Lund, November 2022

4 Popular Summary

Recent advances in artificial intelligence have opened the gate for transformative technologies such as self-driving cars, robot collaborators, and virtual reality. However, technologies such as these rely on computers being able to see humans. For example, a self-driving car must be able to track pedestrians as they move to avoid potentially fatal accidents. Likewise, virtual reality applications require capturing human motion to drive the virtual avatar's movement.

Unfortunately, the fundamental task of perceiving humans is difficult for computers. One reason is that humans vary greatly in appearance; another is that humans can perform a wide variety of actions and fast-paced motions. To make matters worse, the camera might not be able to fully see the human subject due to occlusions or poor lighting conditions. Despite these challenges, impressive progress has been made thanks to so-called artificial neural networks. These mathematical models are inspired by how neurons function in biological brains and learn how to solve tasks from labeled data. For perceiving humans, labeled data usually means videos of humans with corresponding motion capture data in the form of 3d body joint locations tracked over time. Unfortunately, collecting this type of data is time-consuming and requires expensive motion capture equipment. In addition, the capture takes place in indoor studios, limiting the kind of actions that can be captured. Together these factors make it both expensive and challenging to create large and diverse datasets of human motion. As a result, the neural networks often make mistakes when given a new video of humans where the motion or camera viewpoint is different from the examples they were trained on.

This thesis presents methods for improving visual perception through three main ideas. First, we study how an agent equipped with a neural network for perceiving humans should move around to better view the subjects. Imagine a drone tasked with capturing a human athlete. It must move to avoid occlusions and observe the subject from viewpoints seen in the training data to get accurate results. We present neural network-based agents capable of intelligently moving to observe the subjects.

Next, we focus on how an agent should continue to train its neural network once deployed. We study this problem for neural networks performing semantic segmentation (labeling each pixel in an image with an object class). Imagine, for example, an autonomous household robot coming to a new home. The robot might encounter many new objects not present in the training data. In those cases, it would be beneficial if the agent could realize what it does not understand and request explanations from its owner. In this thesis, we present an agent that learns to explore virtual 3d houses and ask for labels for objects it does not recognize. To promote efficiency, we give the agent a budget of how much help it can receive so that it only asks for help when it is crucial.

Lastly, we improve the reconstruction of human motion by integrating the laws of physics. Despite being trained on large video datasets of human motion, neural networks tend to make physically implausible mistakes. For example, the network might predict humans sliding along the ground rather than walking or penetrating the floor. We present a method to combine physics simulation with a neural network perception module to make the results physically plausible – without requiring additional training data.

5 Populärvetenskaplig sammanfattning

Den senaste tidens framgångar inom artificiell intelligens har öppnat dörren till många transformativa tekniker, exempelvis självkörande bilar, robotassistenter och virtuell verklighet. Dessa tekniker bygger på att datorer kan "se människor". En självkörande bil måste kunna hålla uppsikt över fotgängare för att undvika allvarliga olyckor. Likaså kräver datorsimulerad verklighet att datorn kan fånga användarens rörelser för att kunna driva avataren i den simulerade verkligheten.

Tyvärr är det mycket svårt för datorer att uppfatta och urskilja människor. En av anledningarna är att vi människor skiljer oss mycket från varandra när det kommer till utseendet. En annan anledning är att vi kan utföra många olika typer av snabba rörelser. Dessutom kan robotens kamera vara utsatt för dåliga ljusförhållanden eller vara så inställd att delar av objektet faller utanför kamerans synfält. Trots dessa utmaningar har artificiella neurala nätverk lett till stora framgångar. Neurala nätverk är matematiska modeller som bygger på hur biologiska hjärnor attackerar problemen för att lösa dem. När det gäller människors rörelser exemplifieras detta oftast i videor med tillhörande "motion capture" data, där man har spelat in människans leder i 3D. Detta är tyvärr en tidskrävande process som kräver en dyr specialutrustning. Dessutom sker inspelningen av exempel oftast i laboratorier inomhus, vilket begränsar urvalet av rörelser som kan komma ifråga. Detta resulterar i att det är både dyrt och svårt att spela in stora och varierade dataset av mänsklig rörelse. Resultatet blir att de artificiella neurala nätverken, som lär sig av denna ofullständiga data, gör misstag när de observerar människor som utför rörelser som inte finns med i träningsdatan eller när de ser människor från en kameravinkel som avviker från den som användes vid inspelningen i laboratoriet.

Denna avhandling presenterar tre metoder för hur neurala nätverk bättre kan uppfatta omgivningen. I den första delen studeras hur en robot, som är utrustad med ett neutralt nätverk designat för att uppfatta en människa, bör röra sig. Föreställ er en drönare vars uppgift är att spela in en idrottsman. Drönaren bör röra sig så att den har full uppsikt och ser den som idrottar ur de vinklar som det neurala nätverket har tränats för. I min avhandling presenterar jag en artificiell agent som lär sig hur den bäst ska röra sig för att uppfatta en människas rörelser.

I den andra delen studeras hur en agent succesivt förbättrar sitt neurala nätverk när den väl har satts i bruk. Föreställ er en hushållsrobot som har kommit till ett nytt hem. Det kan finnas många nya föremål som roboten aldrig tränats att känna igen. Det vore då fördelaktigt om roboten kunde konkludera, att den inte känner dessa föremål och be ägaren om hjälp med identifikationen. Denna del av avhandlingen presenterar en artificiell agent som lär sig att utforska virtuella hus i 3D och ber om hjälp för att känna igen nya föremål. För att roboten ska arbeta så effektivt som möjligt tillåts den endast att ställa ett begränsat antal

frågor. Därigenom tvingas den välja sina frågor väl.

Den sista delen behandlar hur fysikaliska lagar kan tillämpas för att underlätta för ett artificiellt neuralt nätverk att känna igen en människas rörelse. Trots att nätverken har tränats med omfattande dataset av mänskliga rörelser tenderar de ge orealistiska resultat. Nätverken kan exempelvis göra fel som får det att se ut att en människa svävar istället för att gå framåt. De sista artiklarna presenterar en metod som kombinerar en fysiksimulator med ett neuralt nätverk, vilket gör resultaten mer realistiska. Metoden kräver ingen ytterligare träningsdata.

Chapter 1

Overview

1 Introduction

As humans, it is natural that we have a particular interest in computer vision tasks related to perceiving humans. Thus, human detection, identification, tracking, and reconstruction are all key problems in computer vision. These problems come together in human pose estimation, the process of estimating the body pose configuration from, typically, monocular images.

Solving human pose estimation enables a plethora of applications in the areas of human-computer interaction, animation, and motion analysis. For example, a robot interacting with humans must perceive them, human digitization for virtual reality requires markerless motion capture, and clinical analysis of gait pathologies depends on accurately tracking the body motion.

Recent progress in large and expressive deep neural networks has enabled impressive results in reconstructing human body poses. However, human pose estimation remains a largely unsolved problem in many unconstrained real-world settings. The challenge partly stems from the difficulty of modeling the variability of human appearance, physique, and body poses. Furthermore, difficult lighting conditions, partial occlusions, and loss of information during the projection from 3d to 2d image plane acerbate the problem. Additionally, training deep neural networks requires large annotated human motion datasets that are often recorded in motion capture laboratories. Creating these datasets is expensive, and the resulting datasets are often limited with respect to motion, environment, body shape, and camera viewpoint diversity. One way of addressing these issues is by utilizing self-supervised learning or weakly-labeled data. The five papers of this thesis take a different approach and present three methods to improve visual perception, related but not limited to human pose estimation. An overview of the problems studied in this thesis is presented in Fig. 1.1.

The first two papers study the problem of *active human pose estimation*. We imagine a moving observer that must move to reconstruct the subjects in a scene efficiently using a small set of observations. By selecting informative (e.g., non-occluded) viewpoints, the agents can maximize the reconstruction performance of a pre-trained pose predictor. Next, the third paper proposes *embodied visual active learning*, where an agent must explore an environment and request annotation to learn a perception model efficiently. The agent is initialized with an untrained perception model and must learn to find and request annotation for the most informative viewpoints. Thus, the paper discards the common assumption that the dataset is already collected and annotated. Instead, the agent explores, requests annotation, and refines its perceptual model at test time. We study the problem in the context of semantic segmentation, but learning when to request annotations depending on viewpoint applies to many other tasks, such as human pose estimation and object detection.

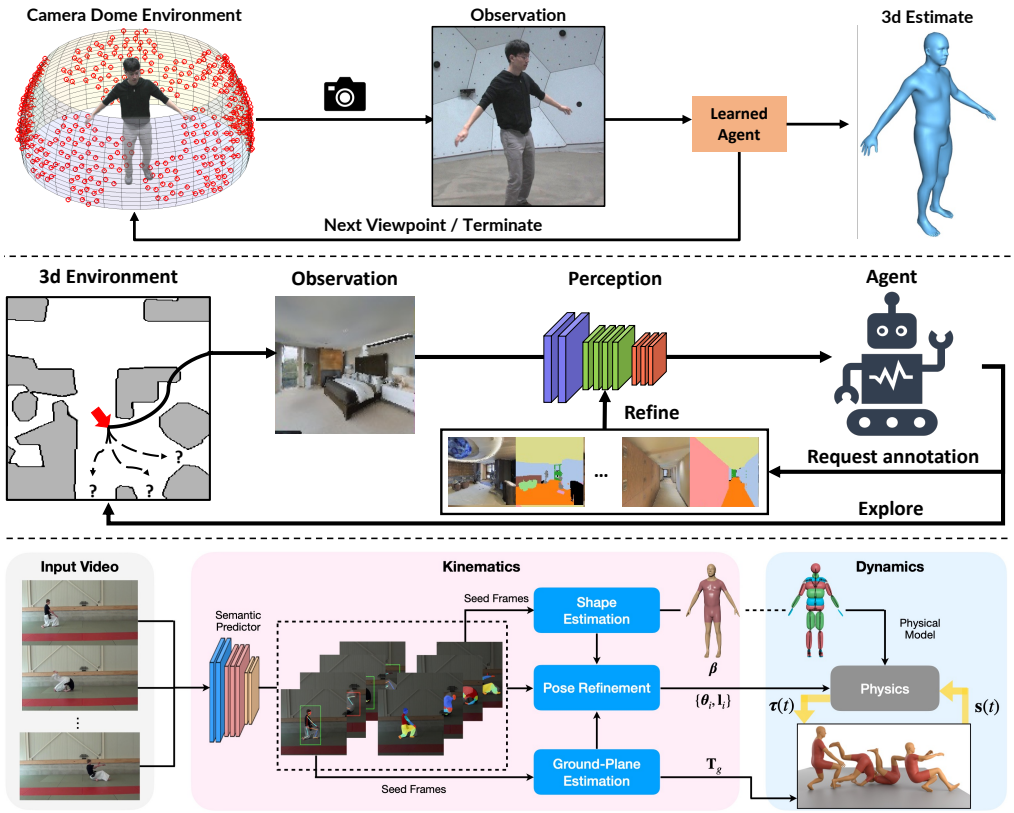


Figure 1.1: Overview of the three problems studied in this thesis. **Top:** *Active human pose estimation*, where an agent is tasked with selecting viewpoints to reconstruct a subject given a pre-trained pose prediction network. The agent sequentially selects viewpoints on a sphere around the subject and terminates the search either through a "stop" action or based on a heuristic criterion. **Middle:** *Embodied visual active learning* in which an agent learns to refine a perception model by exploring its environment and requesting annotation. **Bottom:** *Physics-based human pose estimation* where monocular human pose estimation from video is improved through the inclusion of a reconstruction loss constrained by the dynamics of physical simulation.

Finally, the last two papers present a methodology for integrating physical constraints into a human pose estimation process to enable physically plausible pose predictions and richer outputs with estimated physical quantities. By incorporating physics as a constraint, we improve the reconstruction in a principled manner complementary to any improvements to the underlying reconstruction model and the training set.

The thesis is structured as follows. First, §2 presents the central research questions of the thesis, followed by a summary of the papers' contributions in §3. Next, §4-6 introduces prior work and relevant concepts, §7 presents the conclusions and possible avenues of future work, and finally §8 specifies the author's contributions to each paper.

2 Research Questions

The overall research question of this thesis is how visual perception, in particular human pose estimation, can be improved through active perception and physical constraints. Entailed in the main question are the following questions:

[**RQ1**] How can an active observer learn to seek out informative viewpoints?

[**RQ2**] How can an active observer learn when it has gathered enough information from diverse viewpoints?

[**RQ3**] How can an active observer learn to refine its visual system in the presence of novel percepts?

[**RQ4**] How can physics be to used guide human pose reconstruction to more physically plausible results?

[**RQ5**] Can differentiable physics be leveraged to make physics-based human pose estimation more efficient?

These questions are studied in the papers in this thesis. RQ1 is investigated in the active perception tasks (see §5), where agents must seek out either informative viewpoints or novel object instances. RQ2 is studied in paper Paper II, when the active observer should be efficient when selecting viewpoints to reduce computation and noise from poor viewpoints. RQ3 is addressed in Paper III, where the agent is given an untrained semantic segmentation network and must efficiently query for annotations until it can reconstruct its environment. RQ4 is studied in Paper IV and V, when two different physics simulators are integrated into the reconstruction process to ensure physical plausibility by incorporating the physical constraints of the simulator. Finally, Paper V answers RQ5 by leveraging a differentiable physics simulator to significantly speed up inference compared to the non-differentiable approach of Paper IV.

3 Summary of Contributions

Before introducing relevant concepts and background material, this section summarizes the contributions of each paper.

Paper I studies how an active observer, e.g., a drone equipped with a pre-trained visual perception model (a 2d body joint predictor), should move to gather information necessary to reconstruct human subjects in 3d through triangulation. In the paper, we formalize this by introducing the *active triangulation* task, wherein an agent selects multiple viewpoints so that it may triangulate the body joints of human subjects from 2d to 3d. We consider the problem an egocentric sequential decision-making problem where the agent selects the next viewpoint based on the current and past viewpoints. We propose the reinforcement learning-based model *ACTOR* to solve the task together with a set of heuristic baselines. The results demonstrate that our learned model outperforms the handcrafted baselines at finding sets of viewpoints that accurately reconstruct the subjects in the scene. Furthermore, the presented model is trained in a self-supervised fashion, thus requiring no additional 3d ground truth data to train.

Two limitations of ACTOR were the need to observe each body joint from at least two viewpoints and the stopping criterion being based on a heuristic. Paper II addresses the first shortcoming by relying on a 3d rather than a 2d body joint predictor. Using such an underlying estimator, the agent needs only to see each subject in a single camera to obtain a prediction of the 3d joints. However, as monocular 3d joint estimation is an ill-posed problem, it suffers more from estimation errors than 2d joint prediction. Thus a joint *may* benefit from being viewed from multiple viewpoints in order to reduce the error. The paper introduces *Pose-DRL*, a reinforcement learning-based agent that learns to select informative viewpoints and when to terminate viewpoint selection. The model learns to terminate viewpoint selection when selecting additional viewpoints is expected not to reduce the reconstruction error significantly.

In Paper III, we no longer focus on maximizing the performance of a pre-trained predictor by moving to the most informative viewpoints. Instead, we focus on learning and refining a visual perceptual model efficiently. We formulate the *embodied visual active learning* task wherein an embodied agent must explore its environment and request annotations for unknown elements in the scene to learn its visual perception model. The focus on adaptive behavior for learning a perceptual model is of great interest as no dataset will contain all possible objects, scenes, and viewpoints. Therefore a system must be able to continue learning once deployed into the real world. We instantiate the perception model as a semantic segmentation predictor in the paper. However, the core idea of learning how to explore novel scenarios and request annotations could be applied to other tasks, such as object detection or human pose estimation.

In Paper IV and V, we exploit the seemingly obvious insight that reconstructs of human motions must adhere to the laws of physics. Paper IV, therefore, presents a method for integrating a physics simulator into the human pose estimation process. After using a neural network to estimate the body shape and motion, we perform trajectory optimization in physical simulation, constraining the final motion prediction to be physically plausible. In addition, the integration of physics into the reconstruction process enables a more rich reconstruction that includes estimated physical quantities such as body torques and contact forces. Our results demonstrate that our approach removes unnatural movements and artifacts commonly produced by neural network predictors. However, as the reconstruction loss is non-differentiable, we resort to gradient-free optimization, which is computationally very expensive. Paper V addresses the high computational cost by formulating the reconstruction loss using a differentiable physics simulator, thus allowing gradient-based trajectory optimization, which reduces the computation by orders of magnitude.

4 Human Pose Estimation

The human pose estimation task is concerned with estimating the body pose of humans from, most commonly, monocular video or images. The pose can be represented as either a set of (2d or 3d) body joints or through an articulated body model (see §4.1).

One of the main challenges of human pose estimation is correctly recognizing the subject despite the great variability in human appearance and body physique. This motivates inclusive body models that can accommodate this wondrous variety while acting as a useful prior during inference. For example, the body model might include learned priors over possible body poses, anatomical joint limits, and even muscle force limits. Such constraints and priors may act as regularizers that improve the reconstruction result despite the underdetermined nature of the problem.

The ill-posedness of the problem stems partly from the loss of information during the projection from our 3d world to the 2d image plane. This leads to, e.g., ambiguities when determining the depth ordering of body parts. Tough lighting conditions, as well as partial occlusions due to object occlusions and subject self-occlusion from the perspective of the camera, further exacerbates the difficulty of estimating the body pose (see examples in Fig. 1.2). The latter issues may be addressed by changing the viewpoint and observing the subject from a less occluded vantage point. Furthermore, in complex and crowded scenes, no single viewpoint may contain all necessary information, and multiple viewpoints may be required. An active observer thus requires scene understanding to select informative, unoccluded, and complementary viewpoints such that they together yield accurate estimates of the subject's body pose.

Figure 1.2: Examples of challenges in human pose estimation. **a)** The camera viewpoint poorly captures the scene, e.g., the woman with the hat occludes her right arm and occludes her friends. Furthermore, there is great variety in appearance and clothes. **b)** Loose-fitting clothes can make estimation difficult by obscuring body parts, such as a dress obscuring the lower body. **c)** There exist many plausible rare and complex poses seldom present in motion capture datasets, for example, somersaults. **d)** Fast motions may produce motion blur, which makes even 2d detections hard without incorporating additional information, such as consecutive frames of the video. **e)** The combination of self-similarity between the legs and depth ambiguity caused by the camera projection makes estimation of the legs very difficult.

Finally, most human pose approaches are kinematics-based and do not consider physical quantities when reconstructing motions (series of human body poses) from video. This lack of physical awareness leads to results that violate the laws of physics. For example, the reconstructed motion may contain foot skating, that is, the person skates along the ground rather than walks. Including physics in the reconstruction process may reduce artifacts while simultaneously estimating internal and external forces acting on the human body. The increased richness of the output could enable, for example, applications in medicine

Figure 1.3: Three examples of human representations from monocular reconstructions of the same picture of a tourist in a jumping pose at Dune du Pilat in France. The first image shows 2d landmarks produced by OpenPose [20], the second image shows a GHUM [143] body model predicted by THUNDR [154], and the last image shows a 3d volume represented by an implicit function produced by PIFuHD [109]. The 2d landmark network produces the most accurate reconstruction though this lacks any depth information, the parametric GHUM model captures the pose in 3d together with the body shape, and finally, the body volume produced by the PIFuHD model produces a more detailed but incomplete surface reconstruction. The work in this thesis relies on both 2d landmarks and parametric body models such as the GHUM body model.

studying joint stress during exercise. Physics also provides non-learned regularization applicable to fast and dynamic motions (such as acrobatics) that are rare in motion capture datasets and yield blurry results when recorded in the wild with consumer-grade cameras.

4.1 Human Pose and Shape Representations

There exist many different approaches to representing a human pose. This thesis will focus mainly on human pose estimation in 3d, and below follows a selection of popular approaches. For a more comprehensive overview, refer to, for example, Wang et al. [131] and Sigal [117].

Landmarks. The simplest representation of a pose is a set of independent landmarks or keypoints. These may correspond to anatomical joints, such as the knees or landmarks on the surface, such as the eyes. The main advantage of landmarks is that they are easy to implement and usually fast to predict. The downside, however, is the lack of a coherent structure between the points, resulting in no guarantees regarding the body's symmetry or consistent bone length throughout a sequence of poses. The number of landmarks recognized varies from a sparse skeleton (i.e., 18 body joints) in COCO [72] to skeletons with detailed face and hand keypoints [20].

Kinematic Tree. Rather than representing the body as a set of independent landmarks, it is common to use a kinematic tree representation. In this representation, the pose is expressed through a set of rotations of the joints. One advantage of this representation is that it preserves bone length for a given subject. Furthermore, the representation allows for

an anatomically correct number of degrees of freedom for the joints (e.g., 3-DoF for the neck but only 1-DoF for the knee joint).

Part-Based Models. A less common approach is to divide the human body into a graph of "parts" and express the body pose as the position, orientation, and size of these parts [118]. The body shape is represented by the volume of the geometric primitives that represent the body parts. While mesh-based models have largely superseded part-based models, they are worth mentioning as they are related to the physical body model presented in this thesis.

Mesh-Based Body Models. The most commonly used representation today is mesh-based statistical body models such as SCAPE [9], SMPL [75], GHUM [143] and STAR [93]. These parametric models represent the pose as a kinematic skeleton with a pose-deformed mesh representing the body volume. The mesh is expressed as a function $\boldsymbol{M}(\boldsymbol{\theta}, \boldsymbol{\beta}) \in \mathbb{R}^{3\times N}$, where $\boldsymbol{\theta}$ is a pose vector, $\boldsymbol{\beta}$ is a person-specific shape vector, and N is the number of vertices on the triangular mesh. Using the pose and shape parameters, the model predicts the person's body mesh in articulated poses. These models are trained using large datasets of human body scans to learn how human bodies deform during articulation. Furthermore, modeling not only the pose but the entire body enables, for example, differentiable rendering losses [32, 153] during pose estimation. Representing the body surface also enables modeling object and inter-person contacts [38, 87, 141]. Paper IV-V rely on monocular reconstructions of the subject in the form of statistical body model meshes.

Just as mesh-based models give more informative reconstructions than 3d skeletons, active research exists to increase the richness even further. As an example, imGHUM [4] models the body mesh using a signed distance field which enables the model to represent non-normative body shapes such as one-armed people. Another active research direction is reconstructing not only pose and body shape but also the clothes (e.g., CAPE [77]) and the texture (e.g., PIFu [109] and SCALE [78]). This thesis focuses on improving the realism and richness of the output through the usage of a physical body model, which includes quantities such as masses, joint torques, and contact forces. A description of our physical representation, which bears some resemblance to classical part-based models, can be found in §6.2.

4.2 Neural Networks for Human Pose Estimation

This section briefly reviews neural networks in the context of human pose estimation. Early work in human pose estimation relied on hand-crafted features such as silhouettes [1], edges [105], SIFT [60], or spatial pyramids [60]. However, following the success of convolutional neural networks on the ImageNet [31] challenge pioneered by AlexNet [66] the field started to make heavy use of deep neural networks. Prominently Toshev and Szegedy [124] demonstrated that the AlexNet architecture could be applied to 2d human pose es-

timation with their DeepPose model. Today, there exists a myriad of neural network approaches. See, for example, Chen et al. [25] for an in-depth review.

On a high level, a discriminative human pose network may be viewed as a function $f_{\boldsymbol{\theta}}$ which maps an input image $\mathbf{I}$ to an output $\boldsymbol{y}$

$$\boldsymbol{y} = f_{\boldsymbol{\theta}}(\mathbf{I}), \tag{1.1}$$

where $\boldsymbol{\theta}$ are the parameters of the neural network. Depending on the specific model, $\boldsymbol{y}$ may, for example, represent 2d or 3d keypoints[1]. In the case of a mesh-based body model, the network might directly regress both the pose and shape parameters [153]. The parameters $\boldsymbol{\theta}$ are learned using a gradient descent method such as ADAM [63] to minimize one or more loss functions. The most standard loss is the L_2 regression loss which is commonly used for keypoints

$$L(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} ||\hat{\boldsymbol{y}}_i^j - \boldsymbol{y}_i^j||_2^2 \tag{1.2}$$

where $\hat{\boldsymbol{y}}_i$ is a ground-truth pose from a labeled dataset $\mathbb{D} = \{(\mathbf{I}_i, \hat{\boldsymbol{y}}_i)\}_{i=1}^N$ and $\boldsymbol{y}_i$ is the predicted pose using (1.1). The loss is averaged for each joint j in the set of body joints $\mathbb{J}$ and then over the entire dataset of N frames.

While neural networks yield impressive results on difficult tasks such as monocular 3d human pose estimation, their performance is dependent on access to large training datasets. Typically these are recorded using expensive motion capture equipment in laboratories [54, 58, 80, 125], which gives high accuracy but often leads to data with limited subject diversity, pose complexity, and scene composition. Using, e.g., inertial sensors, human motion can be captured in natural scenes [127], which allows for datasets with more complex scenes. However, inertial sensors do not readily capture facial expressions, and as with optical markers, the inertial sensors need to be carefully placed and calibrated. The training of 3d human pose networks may also be augmented with ground-truth 2d keypoints [61, 151, 153] as a form of weak supervision. This is advantageous as 2d keypoints are easier to annotate and available for complex real-world scenes and in large quantities [7, 72]. However, state-of-the-art methods tend to require labeled 3d data in addition to any weak 2d annotations. Finally, the use of synthetic datasets [12, 52, 94, 126] shows promising results as these allow for highly accurate 3d ground-truth data together with realistically rendered scenes. However, procedurally generating semantically meaningful 3d scenes with human-object interactions and realistic motion is still an active area of research [47, 48, 141, 147, 157].

[1] It is commonplace to also predict a confidence score $c_i \in [0, 1]$ for each keypoint i as a way of handling occlusions.

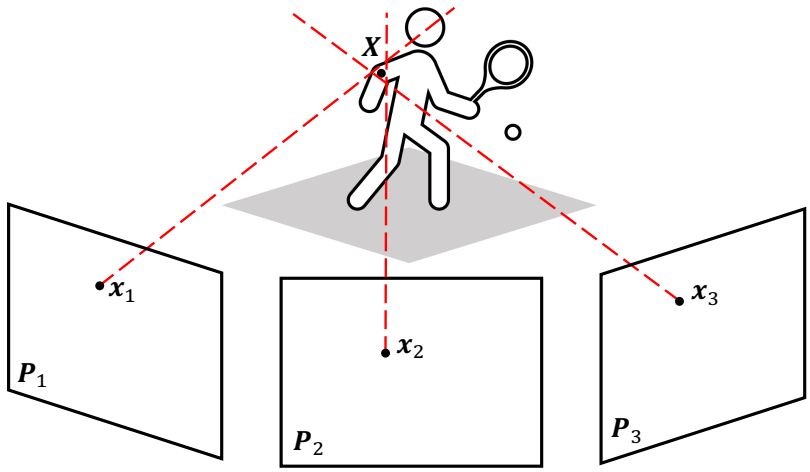


Figure 1.4: Schematic overview of 3d human pose estimation from multiple 2d estimates using triangulation. Given two or more cameras with known camera matrices $\boldsymbol{P}_i$ and the corresponding detection of a joint (e.g., the right elbow) in each camera $\boldsymbol{x}_i$, triangulation is the process of estimating the joint's 3d position $\boldsymbol{X}$ by finding the intersection of the back-projected rays. In the presence of measurement noise or distortions, the rays might not intersect exactly, and the estimated 3d location may differ from the actual location.

Monocular 3d Human Pose Estimation from Video. While a human motion usually is represented as a series of poses that may be predicted frame-by-frame, this usually results in jittery and temporally inconsistent results. Hence, multiple approaches exist to leverage the temporal information present in video data to reduce artifacts and improve individual per-frame reconstructions. For example, Kocabas et al. [64] learn a neural network predictor which treats each frame as input to a gated recurrent unit (GRU) to capture temporal information. In Paper IV-V, we follow Zanfir et al. [152] and perform post-optimization of the per-frame neural network predictions with a temporal smoothness loss that promotes temporal coherence and reduces jitter.

4.3 Triangulation for Multiple View Human Pose Estimation

In computer vision, triangulation [46] refers to the problem of recovering a 3d point given its (possibly noisy) projection in two or more cameras (see Fig. 1.4). In Paper I, the learned agent reconstructs the 3d human pose through triangulation of 2d poses from different cameras. The 2d poses are represented as 2d joint locations and estimated using a neural network predictor [20]. Triangulation is commonly used to create datasets with pseudo 3d ground-truth from multiple camera setups [58, 70]. This approach is less expensive than professional motion capture systems and does not require the subjects to wear motion capture markers. However, it is often less accurate and requires a large set of time-synchronized and calibrated cameras.

Multiple methods exist for triangulating a 3d point from 2d points given known cameras. We use the direct linear transform [46] (DLT) in homogeneous coordinates. This method is fast and does not require non-linear optimization, which is an advantage as we must perform the triangulation many times when training the reinforcement learning agent in Paper 1.

Given a camera matrix[1] $\boldsymbol{P}_i \in \mathbb{R}^{3\times 4}$ of the calibrated camera i, the projection of a homogeneous 3d point $\boldsymbol{X}$ is given by

$$\lambda_i \boldsymbol{x}_i = \boldsymbol{P}_i \boldsymbol{X} \tag{1.3}$$

where $\boldsymbol{x}_i = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^\top$ is the corresponding homogeneous 2d point and λ_i is a scale factor. During triangulation, we wish to estimate the 3d point $\boldsymbol{X}$ given a set of N 2d points $\boldsymbol{x}_1 \dots \boldsymbol{x}_N$ all of which are projections of $\boldsymbol{X}$. For a single point, we may form the cross product $\lambda_i \boldsymbol{x}_i \times \boldsymbol{P}_i \boldsymbol{X} = \boldsymbol{0}$, which gives

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{p}_i^{1\top} \boldsymbol{X} \\ \boldsymbol{p}_i^{2\top} \boldsymbol{X} \\ \boldsymbol{p}_i^{3\top} \boldsymbol{X} \end{bmatrix} = \lambda_i \begin{bmatrix} v_i \boldsymbol{p}_i^{3\top} \boldsymbol{X} - \boldsymbol{p}_i^{2\top} \boldsymbol{X} \\ \boldsymbol{p}_i^{1\top} \boldsymbol{X} - u_i \boldsymbol{p}_i^{3\top} \boldsymbol{X} \\ u_i \boldsymbol{p}_i^{2\top} \boldsymbol{X} - v_i \boldsymbol{p}_i^{1\top} \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{1.4}$$

where $\boldsymbol{p}_i^{j\top} \in \mathbb{R}^{1\times 4}$ is the j:th row of $\boldsymbol{P}_i$. As these equations are linear in $\boldsymbol{X}$ we may form the system $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{0}$. Where each of the N points gives two linearly independent equations as

$$\boldsymbol{A} = \begin{bmatrix} u_1 \boldsymbol{p}_1^{3\top} - \boldsymbol{p}_1^{1\top} \\ v_1 \boldsymbol{p}_1^{3\top} - \boldsymbol{p}_1^{2\top} \\ \vdots \\ u_N \boldsymbol{p}_N^{3\top} - \boldsymbol{p}_N^{1\top} \\ v_N \boldsymbol{p}_N^{3\top} - \boldsymbol{p}_N^{2\top} \end{bmatrix}. \tag{1.5}$$

We are interested in a non-trivial solution $\boldsymbol{X} \neq \boldsymbol{0}$. If we observe a point from more than two cameras, then $\boldsymbol{A}$ will be overdetermined. We find the approximate solution that minimizes $||\boldsymbol{A}\boldsymbol{X}||$ where $||\boldsymbol{X}|| = 1$ using singular value decomposition (see Hartley and Zisserman [45, p. 312]). In other words, we take the unit singular vector corresponding to the smallest singular value as the solution.

[1] The camera matrix is defined as $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$, where $\boldsymbol{K}$ describes the intrinsic camera parameters (such as focal length), and $\boldsymbol{R}$ and $\boldsymbol{t}$ describe the rotation and translation of the projection, respectively.

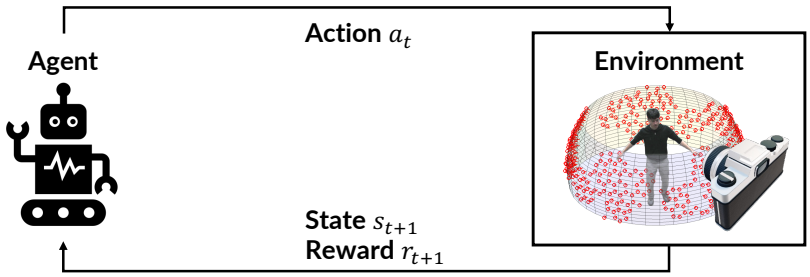


Figure 1.5: Schematic overview of reinforcement learning. Given an environment, the agent should learn to select actions to maximize the cumulative reward. At each time step, the agent should select an action a_t based on an observation of the environment's current state s_t. When the agent selects an action, it receives a reward r_{t+1} together with an observation of the following state s_{t+1}. The environment can, for example, model the active human pose estimation task as introduced in Paper I-II, where the agent should select informative camera viewpoints.

5 Active Perception Tasks

Paper I-III introduce and study two active perception tasks. We formulate these problems in the reinforcement learning framework, briefly introduced in §5.1. Next, §5.2 describes *active human pose estimation* where an active observer should move to observe and reconstruct a human in a scene. Finally, §5.3 presents a variant on active learning called *embodied visual active learning*, where an agent should explore and request annotations in a 3d environment to learn its perception module efficiently.

5.1 Reinforcement Learning

We propose solutions to the active perception tasks using reinforcement learning (RL), see Fig. 1.5. At a high level, reinforcement learning aims to train an agent that solves a sequential decision problem. Rather than learning from labeled examples, the agent learns through interaction with an environment with the goal encoded in a *reward function*. It provides the agent with an immediate reward after each action, and the agent's goal is to maximize the expected cumulative reward. Below is a brief introduction to reinforcement learning. Refer to the textbook of Sutton and Barto [121] for a more in-depth explanation.

Reinforcement learning formulates the sequential decision problem as a Markov Decision Process (MDP) [14]. Formally, an MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, R, p, p_0, \gamma)$ where $\mathcal{S}$ is the set of all states, $\mathcal{A}$ is the set of all actions, p_0 is the initial state distribution, $R(s, a, s') \in \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The MDP's dynamics are defined by the state transition distribution $p(s'|s, a)$ where $s' \in \mathcal{S}$ is a successor state of state $s \in \mathcal{S}$ following action $a \in \mathcal{A}$. When the agent takes an action it also receives a reward from r from the reward function. We can write a series of interactions

$$s_0 \xrightarrow{\pi(\cdot|s_0)} a_0 \xrightarrow{p(\cdot|s_0,a_0),\, R(s_0,a_0,\cdot)} s_1, r_1 \xrightarrow{\pi(\cdot|s_1)} \ldots \rightarrow s_T, r_T \tag{1.6}$$

as the trajectory

$$\tau = (s_0, a_0, s_1, r_1, a_1, s_2, r_2, \ldots, a_{T-1}, s_T, r_T), \tag{1.7}$$

where the actions are sampled from the agent's policy distribution $a_t \sim \pi(\cdot|s_t)$. Note that, according to the Markov property of the MDP, the successor state s' depends only on the previous state s and action a, not the full history of states and actions. Thus the probability of a trajectory τ, assuming discrete state and action spaces, when following the policy π can be expressed as

$$p_\pi(\tau) = p_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t). \tag{1.8}$$

The goal of reinforcement learning is to learn a policy $\pi(a|s)$ that maximizes the expected[1] discounted cumulative return $\mathbb{E}[G_0]$ where

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{1.9}$$

is the sum of the rewards from time t until time T discounted by γ. A large γ implies that the agent should value future rewards higher and vice versa. Furthermore, (1.9) implies that the discounted reward G_t associated with action a_{t-1} only depends on actions from time $t-1$ until time T but no action prior to that. The intuition is that the agent can only affect its future, not its past.

There exist two paradigms within reinforcement learning for how to learn the policy. Following the *model-based* approach, we first seek to estimate the MDP dynamics (that is $p(s'|s,a)$ and $R(s,a,s')$), then find the policy that solves our estimated MDP. In many cases, it is difficult to estimate the dynamics, and the subsequent policy may be suboptimal due to modeling errors of the estimated MDP. The *model-free* approaches attempt to directly learn the policy without first estimating the dynamics of the MDP.

A function of interest related to learning the policy is the q-value function. The *optimal* q-value function is recursively defined as

[1] We take the expectation as $p(s'|s,a)$, $R(s,a,s')$, and $\pi(a|s)$ may all be stochastic.

$$q^*(s,a) = \sum_{s'} p(s'|s,a) \left[R(s,a,s') + \gamma \max_a q^*(s',a') \right] \tag{1.10}$$

and describes the expected discounted return for the action a in state s. If we knew it, we could define the optimal policy $\pi^*(a_t|s_t)$ as selecting the action with highest expected return as

$$\pi^*(a|s) = \operatorname*{argmax}_a q^*(s,a). \tag{1.11}$$

Therefore, a popular approach is to estimate the q-value function [85]. We can also formulate the optimal state value function

$$v^*(s) = \max_a q^*(s,a) = \max_a \sum_{s'} p(s'|s,a) \left[R(s,a,s') + \gamma v^*(s) \right], \tag{1.12}$$

which gives the expected return of starting in state s and then act optimal. In practice it is difficult to obtain the optimal value functions and we usually study the value functions while following a policy π

$$q^\pi(s,a) = \sum_{s'} p(s'|s,a) \left[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a') \right] \tag{1.13}$$

$$v^\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \left[R(s,a,s') + \gamma v_\pi(s) \right]. \tag{1.14}$$

If the size of our state and action space $|\mathcal{S} \times \mathcal{A}|$ is small, we may use tabular representations for $q(s,a)$ and $v(s)$. However, for complex problems, it is common to use neural networks to approximate the functions. For example, we approximate the value function as $v^\pi_{\boldsymbol{\theta}}(s) = f_{\boldsymbol{\theta}}(s)$, where $f_{\boldsymbol{\theta}}$ is a neural network parametrized by $\boldsymbol{\theta}$.

5.1.1 Policy Gradient Methods

In Paper I-III, we employ a class of methods called *Policy Gradient Methods*. They directly attempt to learn the parameters $\boldsymbol{\theta}$ of the neural network $f_{\boldsymbol{\theta}}$ that represent the policy $\pi_{\boldsymbol{\theta}}(a|s) = f_{\boldsymbol{\theta}}(s)$. To do so using a gradient-based method, we must define the gradient of the "performance" of the agent with respect to the policy parameters. This is difficult

since the performance depends on the MDP's unknown and potentially non-differentiable dynamics.

REINFORCE. The most straightforward method for directly learning the policy parameters is the REINFORCE [136] algorithm[1]. It does so by maximizing the expected return (1.9) through the objective

$$J(\boldsymbol{\theta}) = \mathbb{E}_{p_\pi(\tau)}\left[\sum_{t=0}^{T} \gamma^t r_{t+1}\right] = \mathbb{E}_{p_\pi(\tau)}\left[G_0\right] \tag{1.15}$$

under the expectation of a trajectory, see (1.8). The idea is to update the parameters $\boldsymbol{\theta}$ using gradient ascent. However, as we cannot compute the exact gradient of the expression, we resort to a few tricks that enable us to get a sample-based estimate of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. We begin by noting the log-derivative trick

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{f_{\boldsymbol{\theta}}}\left[g(x)\right] = \nabla_{\boldsymbol{\theta}} \int f_{\boldsymbol{\theta}}(x) g(x)\,\mathrm{d}x \tag{1.16}$$

$$= \int \frac{f_{\boldsymbol{\theta}}(x)}{f_{\boldsymbol{\theta}}(x)} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x) g(x)\,\mathrm{d}x \qquad \text{(multiply with } \frac{f_{\boldsymbol{\theta}}(x)}{f_{\boldsymbol{\theta}}(x)}\text{)} \tag{1.17}$$

$$= \int f_{\boldsymbol{\theta}}(x) \nabla_{\boldsymbol{\theta}} \log f_{\boldsymbol{\theta}}(x) g(x)\,\mathrm{d}x \qquad \text{(using } \nabla \log f(x) = \frac{\nabla f(x)}{f(x)}\text{)} \tag{1.18}$$

$$= \mathbb{E}_{f_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \log f_{\theta}(x) g(x)\right] \tag{1.19}$$

which shows how we may take the gradient of the expectation of $g(x)$ under the distribution $f_{\boldsymbol{\theta}}(x)$ with respect to $\boldsymbol{\theta}$ through sampling. It then follows that (1.15) may be differentiated as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{p_\pi(\tau)}\left[\nabla_{\boldsymbol{\theta}} \log p_\pi(\tau) G_0\right] \tag{1.20}$$

Then we may rewrite $\nabla_{\boldsymbol{\theta}} \log p_\pi(\tau)$ using (1.8) as

[1] Also known as Monte Carlo Policy Gradient as it estimates the policy gradient through Monte Carlo sampling.

$$\nabla_{\boldsymbol{\theta}} \log p_\pi(\tau) = \nabla_{\boldsymbol{\theta}} \log \left[p_0(s_0) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t|s_t) p(s_{t+1}|s_t, a_t) \right] \tag{1.21}$$

$$= \nabla_{\boldsymbol{\theta}} \left[p_0(s_0) + \sum_{t=0}^{T-1} (\log \pi_{\boldsymbol{\theta}}(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)) \right] \tag{1.22}$$

$$= \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{1.23}$$

where we can simplify (1.22) to (1.24) as only the policy depends on the neural network parameters $\boldsymbol{\theta}$. Finally, we get to the gradient expression

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{p_\pi(\tau)} \left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) G_t \right] \tag{1.24}$$

$$= \mathbb{E}_{p_\pi(\tau)} \left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \left(\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'+1} \right) \right] \tag{1.25}$$

which may be written as the Monte Carlo estimate

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^i|s_t^i) G_t^i \tag{1.26}$$

where N is the number of trajectories used to estimate $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Note the change from G_0 in (1.20) to G_t in (1.24) for action a_t. This is common practice since the return of an action is based on the future outcome, not what happened in the past. Finally, we have now derived the REINFORCE loss

$$L(\boldsymbol{\theta}) = -J(\boldsymbol{\theta}) \approx -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(a_t^i|s_t^i) G_t^i. \tag{1.27}$$

In practice, there exist many improvements to REINFORCE. For example, we may subtract an action-independent *baseline* $b(s_t)$ [121] from the return

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\log\pi_{\boldsymbol{\theta}}(a_t^i|s_t^i)(G_t^i - b(s_t^i)) \tag{1.28}$$

which for a suitable choice of b, can decrease the variance. A natural baseline is the value function in (1.14) approximated with a neural network

$$L_{\text{baseline}}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\log\pi_{\boldsymbol{\theta}}(a_t^i|s_t^i)(G_t^i - v_{\boldsymbol{w}}(s_t^i)) \tag{1.29}$$

where $\boldsymbol{w}$ are the neural network parameters. During training, we would alternate between updating $\boldsymbol{\theta}$ and $\boldsymbol{w}$. Commonly, the mean squared error regression loss is used to learn $\boldsymbol{w}$ based on the observed returns from the sampled trajectories.

Proximal Policy Optimization. In Paper III we use the popular Proximal Policy Optimization [112] (PPO) algorithm. It belongs to a class of methods called Actor-Critic methods that jointly learn an approximation of the value function (1.14) (the critic) to improve convergence when learning the policy $\pi_{\boldsymbol{\theta}}$ (the actor). During test time, only the policy is used for selecting actions. PPO is more stable and has a lower variance compared to REINFORCE. The loss[1] is given by

$$L_{\text{PPO}}(\boldsymbol{\theta}) = \mathbb{E}_{p_\pi(\tau)}\left[\sum_{t=0}^{T-1}\min\left(\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)}A(s_t,a_t), g(\epsilon, A(s_t,a_t))\right)\right] \tag{1.30}$$

where

$$g(\epsilon, A) = \begin{cases}(1+\epsilon)A & \text{if } A \geq 0\\ (1-\epsilon)A & \text{if } A < 0\end{cases} \tag{1.31}$$

and $A(s_t, a_t) = q_\pi(s_t, a_t) - v_\pi(s_t)$ is the *advantage* function which defines how beneficial an action is relative to the state's value. Using the advantage rather than return is a form of baseline subtraction used to reduce the variance. As PPO is an actor-critic method, it learns a neural network approximation of v_π.

Rewriting the inner expression when the advantage $A(s_t, a_t)$ is positive versus negative gives

[1] This is not the original formulation of the loss from the paper [112] but an equivalent and simplified formulation from OpenAI.

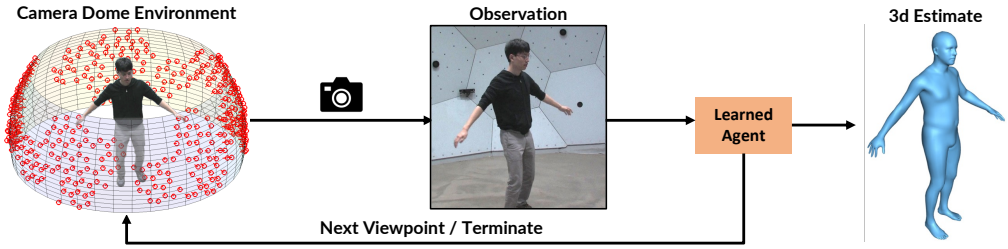


Figure 1.6: *Active human pose estimation*, where an agent is tasked with selecting viewpoints to reconstruct a subject given a pre-trained pose prediction network. The agent sequentially selects viewpoints on a sphere around the subject and terminates the search either through a "stop" action or based on a heuristic criterion. This process is repeated for each frame in a video and may feature multiple subjects.

$$\min\left(\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)}A(s_t,a_t), g(\epsilon, A(s_t,a_t))\right) = \tag{1.32}$$

$$= \begin{cases} \min\left(\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)}, 1+\epsilon\right) A(s_t,a_t) & \text{if } A \geq 0 \\ \max\left(\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)}, 1-\epsilon\right) A(s_t,a_t) & \text{if } A < 0 \end{cases} \tag{1.33}$$

which can be interpreted as when the advantage is positive (the action is considered good), it will increase $\pi_{\boldsymbol{\theta}}(a_t|s_t)$. However, the min operator limits the size of the update since when $\pi_{\boldsymbol{\theta}}(a_t|s_t) > (1+\epsilon)\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)$, we clip the expression to $(1+\epsilon)A(a_t|s_t)$ and vice versa if $A < 0$. The clipping makes learning more stable by preventing too large updates and acting as a trust region around the old policy parameters $\boldsymbol{\theta}_{\text{old}}$. The clip ratio hyperparameter is usually small, for example, $\epsilon \in [0.1, 0.3]$.

Armed with basic knowledge about reinforcement learning, we may now apply this methodology to solve the active perception tasks in Paper I-III, discussed in the following sections.

5.2 Active Human Pose Estimation

Paper I tasks an agent with selecting a small set of viewpoints that can be used to accurately triangulate 2d body joint estimates into 3d for subjects in a scene. The agent selects viewpoints until each joint has been seen from at least two cameras since this is the fewest number of viewpoints required for triangulation. Paper II studies the task of an agent equipped with a 3d monocular human pose estimation network, where the agent should again select informative viewpoints to reconstruct the subjects. As the 3d estimation network estimates even unseen joints, the agent can theoretically reconstruct the subject using a single viewpoint. However, as predictions for occluded joints may be noisy, the agent may wish to select multiple viewpoints to improve the reconstruction. Therefore there is

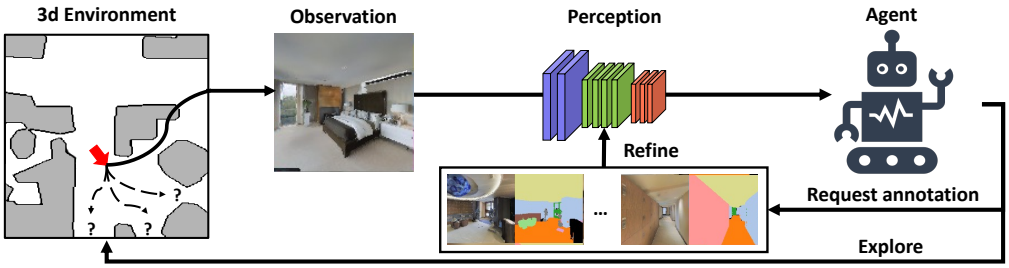


Figure 1.7: *Embodied visual active Learning* in which an agent learns to refine a perception model by exploring its environment and requesting annotation.

no heuristic stopping criterion. Instead, the agent learns when it has observed the subject from sufficiently many viewpoints and should stop. See Fig. 1.6 for a schematic overview.

Active human pose estimation is related to the "next-best-view" problem [27], where an observer should sequentially select viewpoints to reconstruct a scene [27, 43] or an object [81]. Unlike prior work, our task focuses on viewpoint selection for articulated 3d human pose estimation from video. This is of particular interest as humans tend not to be static. Furthermore, unlike object surface reconstruction, human pose estimation does not require observing the subject exhaustively from all sides of the object. Furthermore, recent advances in drone-based markerless motion capture[1] [51, 89, 90, 108, 122, 145, 159] merit research into viewpoint selection for human pose estimation. Unlike the optimization-based approach of Kiciroglu et al. [62], our approaches focus on *training* agents, which select viewpoints that minimize reconstruction errors while considering both estimator biases and occlusions in the scene. The concurrent work of Yang et al. [146] studies how to maximize a recognition system's performance through movement to localize unobstructed viewpoints. Unlike our work, their method focuses on object detection rather than articulated pose reconstruction.

Since active human pose estimation is a sequential decision-making problem, we present two reinforcement learning-based solutions. We use the Panoptic dataset [58] to address the vast amount of samples required during training. It consists of video recordings of human interactions from a large dome with ∼500 time-synchronized inward-facing cameras. We use the dataset as a proxy for an active observer and simulate movement by switching camera viewpoints. While controlling real drones poses additional challenges, our setup facilitates reproducible experiments into learned viewpoint selection on real image data.

5.3 Embodied Visual Active Learning

The *embodied visual active learning* (EVAL) task (see Fig. 1.7) for an introduced in Paper III is a research problem that sits at the intersection of recent research into learning embodied agents [30, 135, 160] and active learning [8, 107, 114].

In this context, embodied learning refers to the study of learning intelligent behaviors when controlling a physical body in an (often simulated) 3d environment. That is different from, for example, the classical object detection task where the learned algorithm can only observe a given image and must make a prediction based solely on that input. In contrast, in embodied learning, the agent may move around and explore its environment while completing its tasks. Prior work studied embodied learning in the context of point-goal navigation [135] (learning how to navigate to a target location), question answering [30] (learning how to navigate an environment to answer questions about it), object-goal navigation (learning how to navigate to an object class in the environment), and learning how to move to improve the accuracy of a pre-trained recognition system [146].

Active learning studies how a learning algorithm can achieve high accuracy with few labeled examples provided it intelligently selects what examples should be labeled. Rather than training a model on a predetermined labeled dataset, the learner *queries* an *oracle* (often a human annotator) to annotate a selected subset of examples. Intuitively this makes sense, as annotations may be expensive, and not all examples are equally informative. For example, in the case of semantic segmentation, many pixels are occupied by background classes such as pavement, and most tasks are interested in the classification of foreground objects. There exist multiple approaches to active learning, such as pool-based [69] where the algorithm is given a set of unlabeled examples to query from, membership query synthesis [8] where the algorithm generates new instances to query labels for, and stream-based [26] where the algorithm is given a stream of instances and given a choice to either discard or query the oracle for annotation.

Our task combines these two research areas. It can be viewed as a membership query synthesis task where embodiment constrains the queries. The agent should efficiently improve its visual perception system but must explore its environment to obtain suitable instances to query the oracle. The task may also be viewed as an extension of the active human pose estimation task introduced in §5.2. Except, rather than moving to increase the reconstruction accuracy, it may now move to *improve* the underlying perception system through querying for annotations. Such a system would be of interest as Paper I-II find that the pose estimation system provides poor estimates for viewpoint, unlike those seen during training. Enabling the agent to refine its perception would be a logical next step.

[1]Essentially equipping drones with human pose estimators that follow and reconstruct subjects in outdoor environments where traditional motion capture technology is not feasible.

In Paper III we study the EVAL task in the context of semantic segmentation using the Habitat simulator [110] and the photorealistic Matterport3D dataset [23]. We resort to simulation rather than a physical robot partly because simulation makes experimentation and model development significantly easier. The other reason is the abundance of simulators [65, 110, 120] which provide ground-truth segmentation masks while affording complex exploration behavior beyond the Panoptic dataset used in the active human pose estimation task. While experimenting with a physical robot would be of great interest, this is a significantly more challenging task, especially when training reinforcement learning agents, which generally require many samples during training. Despite prior work noting that generalizing an exploration policy from simulation to the real world is non-trivial [59], we believe simulation provides a valuable platform for reproducible experiments that may lead to future real-world advances. Recent advances in sim-to-real transfer for embodied agents support this belief. For example, Anderson et al. [6] deploy an embodied vision-and-language navigation network to the real-world using domain randomization and an adaptation module that account for the change from a discrete to a continuous world[1]. They conclude that this approach is effective, given that a map of the new environment is provided. Such a map may be collected in a pre-processing step using a traditional Simultaneous Localization and Mapping (SLAM) approach.

5.3.1 Semantic Segmentation

Semantic segmentation is the fundamental computer vision task of predicting the semantic class of each pixel in an image [84] from an application-dependent set of labels. This is commonly formulated as: given an RGB image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$

$$\mathbf{Y} = f_{\boldsymbol{\theta}}(\mathbf{I}), \tag{1.34}$$

find the parameters $\boldsymbol{\theta}$ of a function $f_{\boldsymbol{\theta}}$ (for example, a deep convolutional neural network) that predicts a tensor with the per-pixel class probabilities $\mathbf{Y} \in \mathbb{R}^{H \times W \times C}$, where C is the number of object classes, H and W are the image height and width, respectively. From $\mathbf{Y}$ we obtain the predicted class for each pixel by taking the argmax over the class dimension.

The performance of a semantic segmentation algorithm is commonly measured using class-balanced metrics. The reason is that most datasets tend to have large class imbalances. For example, most pixels in a photo of a room will be dominated by "background" classes, such as the ceiling, rather than object classes, such as "TV". In Paper III we use the class balanced metric *mean intersection over union* (mIoU) defined as

[1]Datasets such as Matterport3D consist of a grid of discrete positions, unlike the real world, which is continuous.

$$\text{mIoU} = \frac{1}{N}\sum_{i=1}^{N}\text{IoU}_i = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbb{P}_i \cap \mathbb{T}_i|}{|\mathbb{P}_i \cup \mathbb{T}_i|} \tag{1.35}$$

where, $\mathbb{P}_i$ is the set of pixels predicted as belonging to class i, and $\mathbb{T}_i$ is the set of pixels actually belonging to class i. This promotes the algorithm to value the classes equally. Note that, as any single image in a dataset may contain only a few semantic classes, mIoU should be computed for the entire dataset for the metric to be meaningful.

6 Physics-Based 3d Human Pose Estimation

While recent advances in monocular 3d human pose estimation have been propelled forward significantly thanks to the advent of deep neural networks, it has been observed that these tend to produce physically implausible results [106, 115]. For example, the subject may penetrate the ground plane, the feet may slide unnaturally along the ground as if on ice (commonly called foot sliding or foot skating), or the motion may contain jitter. Methods that attempt to improve the reconstruction quality and augment the output with physical quantities by explicitly incorporating physical constraints into the reconstruction process are referred to as *physics-based human pose estimation methods* in this thesis[1].

Including physics in the reconstruction process has clear advantages. For example, it may enable the estimation of physical quantities such as body torques and contact forces. Additionally, it constrains the reconstructed motion to obey the laws of (the simulated) physics. However, these advantages come at the cost of increased modeling complexity, as the methods often require that the subject's mass, moment of inertia, and muscle strength are known. Furthermore, including physics in the reconstruction process tends to increase the computational cost compared to purely kinematic methods.

Computer vision has a long tradition of physics-based human pose estimation [17, 18, 19, 71, 82, 128, 133, 155]. For example, the early work by Metaxas and Terzopoulos [82] presents a method for tracking an articulated upper body from 3d marker data using simulation of Lagrange equations of motion and Kalman filtering. Brubaker et al. [18] perform trajectory optimization using a reduced-dimension body model that tracks the center-of-mass location rather than the rotations in the kinematic skeleton. Li et al. [71] estimate ground and contact forces during human-object interactions from videos of subjects interacting with tools-like objects.

[1] These methods also go by other names such as "physically plausible", "physically aware", or "physionical".

More recently and closely related to our approach, Rempe et al. [106] present a physics-based reconstruction method with the primary goal of reducing artifacts such as foot skating compared to kinematics-based methods. Their method first reconstructs the subject from monocular video using a 3d pose estimation network and a 2d body keypoint network. From the 2d keypoints, a foot contact network predicts the contact timings between the feet and the ground plane. Given the 3d poses, 2d body keypoints, and contact timings, their method optimizes a simplified centroidal dynamics model introduced by Winkler et al. [137] to match the estimated visual evidence. Their contact model only considers feet-ground contacts, and the binary contact state (i.e., contact or not in contact) is considered fixed. However, the exact contact timings are updated during optimization. Furthermore, their method does not simulate the upper body pose with physics but infers it using inverse kinematics from the lower body pose.

Shimada et al. [115] introduce the real-time *PhysCap* model, which, similarly to Rempe et al. [106], first estimates 2d and 3d body joint positions using neural network predictors. Their method then recovers smooth per-frame 3d joint body angles using inverse kinematics coupled with a temporal smoothness loss. Next, they detect foot-ground contact labels and a binary stationary/non-stationary pose indicator for each frame using a neural network predictor from the 2d body keypoints. The predicted contacts are used to inform their physics optimization step which frames should contain ground reaction forces. The stationary pose indicator is used in a pre-processing step to heuristically adjust the body poses to such that the center of gravity is inside the subject's base of support[1] [33]. Given these quantities, PhysCap performs multi-stage energy-based optimization to recover joint torques and ground reaction forces such that the physical motion is close to the estimated visual evidence while subject to the equation of motion introduced in Featherstone [37]. As the visual evidence (2d and 3d body keypoints) obtained by the neural network predictors may contain errors and be physically implausible, the authors introduce a root residual force [149] that allows their solution to contain a (usually small) physical force applied to the center of the character. This unrealistic force is a component that recurs in other physics-based human pose estimation approaches [116, 150] as it makes optimization easier in the presence of, for example, visual estimation and body modeling errors.

In their work on human motion synthesis, Xie et al. [142] present a physics-based human pose estimation method from monocular video. They include physics into their reconstruction pipeline to improve the quality of monocular pose reconstruction to high-quality training data for their synthesis model. Like prior work, they first extract visual evidence in the form of 2d and 3d body joint positions and then recover body joint rotations skeleton using inverse kinematics. They do not, however, pre-detect foot contacts but instead re-

[1] Aligning the subject's center of gravity with its base of support intuitively means that the subject should not lean excessively and that their center of gravity should be roughly above their feet when in a stationary pose (e.g., not running).

cover these during physics optimization. Their method supports feet-ground contacts and models these through a soft contact penalty introduced by Mordatch et al. [86]. Contacts are then optimized jointly with joint torques and ground reaction forces. The advantage of their physics formulation is that it is differentiable, and their physics loss can be directly minimized by a quasi-Newton solver such as L-BFGS [92]. However, their formulation includes physics as a loss term among others and thus does not guarantee that the final motion corresponds to physically plausible motion.

Similar to the approach in Paper IV-V, the *SimPoE* model [150] eschews using simpler custom-made physics formulations but instead relies on a readily available physics simulator. In the case of SimPoE, they simulate physics using the rigid multibody simulator MuJoCo [123]. Following prior work [97] on learning human motion controllers from video, the authors learn a motion controller using reinforcement learning which predicts joint torques[1], rather than recovering them through optimization. Using a neural network controller makes their method applicable in real-time while still allowing it to take full advantage of the features of a full-fledged physics simulator, such as full-body contracts. However, training a reinforcement learning model is very costly - and as prior work on motion controllers [96, 98, 99, 140] has shown - learning a general motion controller is a challenging and open problem. Moreover, as the authors do not evaluate their model on real-world videos or highly dynamic motions (e.g., backflips or somersaults), its ability to generalize to novel types of motions is unproven. Besides, their method relies on using a non-physical root residual force. Still, the authors propose an attractive method because it is real-time and does not simplify the physics formulation. Yu et al. [148] similarly take a reinforcement learning-based approach, training a DeepMimic [96] motion controller for each video they wish to reconstruct. While computationally costly, their system can reconstruct fast-paced motions recorded by a moving camera with unknown camera parameters.

Shimada et al. [116] take a different approach from prior work. They embed physics-based constraints into a custom layer of the neural network predictor, thus removing the need for a multi-stage approach. Like PhysCap, their approach runs at real-time speed but only supports foot-ground contacts, relies on unrealistic root residual forces, and does not adapt the physical body model to match the subject's body shape.

Aside from utilizing physics in human pose estimation, there exists a plethora of adjacent research within other computer vision tasks. For example, Peng et al. [97] attempted to learn motion controllers from real-world video examples using the DeepMimic [96] model. Unlike human pose estimation - where the goal is reconstructing the motion as accurately as possible - motion controller research often aims to reproduce motions of similar types or characteristics. This research closely relates to decades of physics-based motion generation research within computer graphics [16, 35, 57, 74, 98, 99, 101, 130]. The goal is to synthesize

[1]To be precise, their network predicts control targets to PD controllers, which in turn generate the joint torques, see (1.55).

realistic and physically plausible motion rather than reconstruct motions from videos.

In addition to physical constraints, several recent works use different types of scene constraints to improve the reconstruction. Zou et al. [161] present a method for reducing reconstruction artifacts, specifically foot sliding, through enforcing zero velocity in the foot joints during moments when there are foot-ground contacts. The contacts are predicted by a neural network given the 2d joint estimates. However, this approach has clear limitations, such as the need for the feet to be visible and that the approach considers only foot-ground contacts. Fieraru et al. [38] condition the reconstruction based on inter-personal contacts, Zanfir et al. [151] consider ground-plane when reconstructing multiple people in a scene, Zhang et al. [156] jointly reconstruct people interacting with objects, and Yi et al. [147] propose a method that considers images of the scene, objects in the scene as well as examples of human-object interactions to reconstruct a physically plausible 3d scene. However, none of these works consider the use of dynamics to improve the physical plausibility of a reconstructed motion nor attempt to recover physical quantities such as contact forces and body torques, as is the focus of physics-based human pose estimation.

To summarize, the proposed approaches in this thesis integrate a physical simulator into a video reconstruction system. Our methods support dynamic in-the-wild motions, and since they rely on full-fledged physics simulators, they can handle full-body contacts and self-contacts. Furthermore, unlike several other methods, our approaches do not require unrealistic residual forces to decrease the reconstruction error at the cost of realism. While the approach presented in Paper IV is computationally costly, Paper V demonstrates that using a differentiable physics simulator can reduce computation by orders of magnitude.

6.1 Physically Based Modeling

Paper IV and V utilize off-the-shelf physics engines to simulate the physics of human dynamics, enforce physical constraints such as contacts, and approximate Newton's laws of motion[1]. Physics-based modeling is an active research field with applications in computer vision and computer graphics. Due to the breadth and complexity of the subject, the following sections will be brief and focus on building an intuition of the core concepts of a physics engine rather than thoroughly describing its inner workings[2]. The two main components of physics-based modeling are *simulation* and *control*. Simulation refers to the algorithms that emulate the laws of physics, while control refers to the algorithms steering, in our case, the human character.

[1]Since physical simulators tend to approximate the physics rather than exactly model them these are sometimes called *force laws* rather than laws of physics [138].

[2]For more in-depth explanations, the author recommends the excellent SIGGRAPH course of Witkin and Baraff [138], the tutorial of Liu and Jain [73] and the thesis of Stępień [119].

6.1.1 Rigid Body Simulation

Both the Bullet [28] simulator used in Paper IV and the Tiny Differentiable Simulator (TDS) [49] employed in Paper V are rigid body simulators. That implies that they approximate the human body by a set of non-deformable geometric primitives (see §6.2). The simulators compute the motion of the rigid bodies given external forces (e.g., gravity), body joint torques (approximating muscle activations), and contact forces from collisions (e.g., a foot touching the ground).

Simulation can be viewed as an initial value problem of an ordinary differential equation (ODE) describing the dynamics of the rigid body simulator. The state update at time t is defined as $\dot{\boldsymbol{s}}(t)$ and depends on the dynamics of the simulator F and the state of rigid bodies $\boldsymbol{s}(t)$ as given by

$$\dot{\boldsymbol{s}}(t) = F(\boldsymbol{s}(t), t). \tag{1.36}$$

In the classical formulation, the state of a rigid body is its position $\boldsymbol{x}(t)$, orientation $\boldsymbol{q}(t)$, linear momentum $\boldsymbol{p}(t)$, and angular momentum $\boldsymbol{l}(t)$. The position and orientation are defined with respect to a world coordinate frame. Orientation is usually represented with unit quaternions as they are compact, do not suffer from gimbal lock, and do not have problems with numerical drift common with rotation matrices [138]. The derivative of the position and orientation may be written as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{v}(t) \tag{1.37}$$

$$\dot{\boldsymbol{q}}(t) = \frac{1}{2}\boldsymbol{\omega}(t)\boldsymbol{q}(t) \tag{1.38}$$

where $\boldsymbol{\omega}(t)\boldsymbol{q}(t)$ is shorthand for multiplication between the two quaternions $[0, \boldsymbol{\omega}(t)]$ and $\boldsymbol{q}(t)$[1]. The linear velocity of a body with mass m is given by $\boldsymbol{v}(t) = \frac{\boldsymbol{p}(t)}{m}$ and its linear acceleration by

$$\dot{\boldsymbol{v}}(t) = \frac{\dot{\boldsymbol{p}}(t)}{m} \tag{1.39}$$

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{f}(t) \tag{1.40}$$

[1] See Witkin and Baraff [138, Appendix B] for a derivation of the quaternion update formula.

where $\boldsymbol{f}$ is the force acting on the center of mass of body[1]. The relationship between the angular momentum and angular velocity $\boldsymbol{\omega}$ is more complicated as this depends on the mass and how it is distributed in the body. The inertia tensor $\boldsymbol{I}$ defines the scaling between the angular momentum and velocity as

$$\boldsymbol{l}(t) = \boldsymbol{I}(t)\boldsymbol{\omega}(t) \tag{1.41}$$

$$\dot{\boldsymbol{l}}(t) = \boldsymbol{\tau}(t) \tag{1.42}$$

where $\boldsymbol{\tau}$ is the torque on the body. For simple geometric primitives, the inertial tensor can be evaluated symbolically, while for complex shapes, the integrals over the mass distribution would be precomputed before the simulation begins. In summary, the state $\boldsymbol{s}(t)$ for a rigid body can be expressed as

$$\boldsymbol{s}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{q}(t) \\ \boldsymbol{p}(t) \\ \boldsymbol{l}(t) \end{bmatrix} \tag{1.43}$$

with the state update $\dot{\boldsymbol{s}}(t)$ given by

$$\dot{\boldsymbol{s}}(t) = \frac{d}{dt}\begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{q}(t) \\ \boldsymbol{p}(t) \\ \boldsymbol{l}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}(t) \\ \frac{1}{2}\boldsymbol{\omega}(t)\boldsymbol{q}(t) \\ \boldsymbol{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} = \begin{bmatrix} \frac{\boldsymbol{p}(t)}{m} \\ \boldsymbol{I}(t)^{-1}\boldsymbol{l}(t)\boldsymbol{q}(t) \\ \boldsymbol{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix}. \tag{1.44}$$

Constraints. Given (1.44) and an ODE solver, we could now simulate a system of *unconstrained* rigid bodies. However, for our purposes, we need to be able to introduce constraints on the system. These constraints might enforce non-penetration between two bodies or act as joints connecting multiple bodies. There exist multiple ways to implement constraints. One approach is through the introduction of energy functions that penalize unwanted behavior. For example, we could connect two bodies by a damped spring according to Hook's law

$$\boldsymbol{f}_a = -\left[k_s(|\boldsymbol{d}| - r) + k_d\frac{\dot{\boldsymbol{d}} \times \boldsymbol{d}}{|\boldsymbol{d}|}\right]\frac{\boldsymbol{d}}{|\boldsymbol{d}|} \tag{1.45}$$

[1]This is analogous to the relationship $\boldsymbol{a} = \boldsymbol{f}/m$ in particle simulation.

where $\boldsymbol{d} = \boldsymbol{x}_a - \boldsymbol{x}_b$ is the distance between the two bodies, r is the rest length of the spring k_s and k_d are the spring and damping constants, respectively. A spring gives rise to two opposite directed forces $\boldsymbol{f}_a = -\boldsymbol{f}_b$ pulling the bodies to distance r of each other. These forces are summed along with other forces acting on the bodies and, unless strong enough, might not cancel competing forces resulting in violated constraints.

Another method for modeling constraints is through *constraint forces*. In the case of a spring-based constraint, we maintain the constraint by measuring the illegal displacement and generating a proportional and opposite force. This force is generated *after* the constraint has been violated, and it competes against all other forces acting on the body. A better approach would be to directly cancel out forces acting against the constraint as they are generated. However, while doing so, we must respect the law of conservation of energy. This is accomplished by converting the body's acceleration into "legal" acceleration that does not violate our constraint[1]. These constraints are commonly used when implementing non-penetration constraints, e.g., to generate ground reaction forces.

Simulation Steps. Generally, each time step of the simulation can be summarized into the following key steps:

1) Forward kinematics and forward dynamics. In the first step, the forces and torques acting on the bodies are computed. This gives the world position of the bodies as well as unconstrained updates.

2) Collision detection. Given the world position of the bodies, the simulator runs collision detection to identify bodies in contact. This is a costly operation requiring, at worst, $O(N^2)$ operations where N is the number of bodies in the system. However, in practice, many methods exist for caching collisions and checking only for collisions between bodies that are close to each other.

3) Solving constraints. Given the detected contacts, the simulator computes the forces required to satisfy non-penetration constraints. Additionally, other constraints may be included in this step.

4) Integration. The constrained velocities and accelerations are integrated using an ODE solver to compute the state at the next time step of the simulation.

[1]Using *the principle of virtual work* [42] the constraint forces are directed such that they do not add nor remove energy from the system's kinematic energy while keeping the body's state valid. See the lecture on constrained dynamics by Witkin and Baraff [138] for a more in-depth explanation.

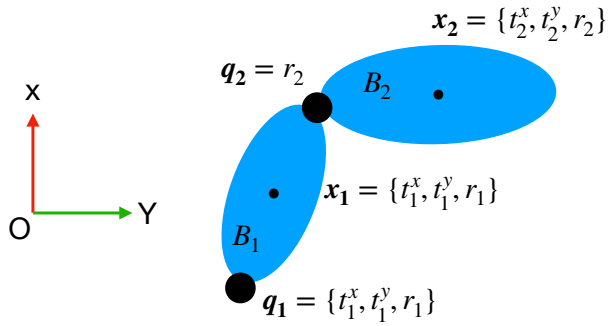


Figure 1.8: Example of two 2d rigid bodies connected by a 1-DoF hinge joint. In maximal coordinates, we would represent the state of each body B_i using $\boldsymbol{x}_i$, which contains the center of mass position and orientation. In reduced or generalized coordinates, we represent the state as $\boldsymbol{q}_i$, the rotation, and translation relative to its parent. In this example $|\boldsymbol{x}| = 6$ and $|\boldsymbol{q}| = 4$, though for large bodies, this difference becomes even more significant. Note that the reduced coordinate representation implicitly enforces the constraint between the bodies as it does not support configurations where they are separated.

6.1.2 Algorithms

The above description of rigid body simulation presented the classical Newton-Euler approach to physics-based modeling [138] in order to build intuition. However, state-of-the-art physics engines have evolved beyond this formulation. Unfortunately, today's algorithms trade intuitive approaches for more compact state representations, speed, and fidelity. Below is a summary of the approach of Bullet and TDS.

Featherstone's Rigid Body Formulation. Representing the human body as rigid bodies connected by springs is possible, but it may lead to difficulties when tuning the spring constants. Improperly tuned springs may cause oscillation or limbs drifting apart [11]. Another approach is to use a *generalized-coordinate* formulation[1]. That is, rather than connecting a rigid bodies through constraints; it may be parameterized such that only valid configurations are possible. For example, a human body with M body parts may be represented in two ways. In *maximal coordinates*, the body's configuration is represented by the Cartesian coordinate for each body part (that is $\boldsymbol{x}_{\text{maximal}} \in \mathbb{R}^{3\times M}$). In generalized coordinates, the system's configuration is formulated as $\boldsymbol{x} \in \mathbb{R}^N$ where N is the total number of rotational degrees of freedom in the kinematic skeleton. This reduces the parameterization size and limits the body to only valid configurations, see the example in Fig. 1.8.

Both Bullet and Tiny Differentiable Simulator represents the state as the relative joint rotations along the kinematic tree of the multibody model. In particular, they rely on the formulation by Featherstone [37]. According to Featherstone's formulation, the state of a simulated body is expressed by

[1] Also called *reduced* coordinates.

$$\boldsymbol{s}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix}, \tag{1.46}$$

where $\boldsymbol{q}(t)$ and $\dot{\boldsymbol{q}}(t)$ are vectors of joint rotations and joint velocities, respectively[1]. This compact representation comes at the cost of more complicated algorithms since each body's world position and orientation must be recursively computed starting from the root joint. Computing contact forces and applying external forces (such as gravity) also becomes more involved as these forces – usually expressed in the world coordinate system – must be transformed into accelerations in the joint space.

Featherstone's formulation builds on Lagrangian dynamics, which studies the system's energies, unlike the Newton-Euler formulation (in §6.1.1), which studies the forces acting on the system. The canonical equation of motion [37] is defined as

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{1.47}$$

where $\boldsymbol{\tau}$ is a vector of forces[2] applied in the joints, $\ddot{\boldsymbol{q}}$ are the joint accelerations, $\boldsymbol{M}$ is a matrix of inertia terms dependent on the current body configuration $\boldsymbol{q}$, and $\boldsymbol{c}$ is a vector of force terms accounting for all forces acting on the system other than $\boldsymbol{\tau}$, such as gravity. $\boldsymbol{\tau}$ and $\ddot{\boldsymbol{q}}$ are the variables of the equation system, and $\boldsymbol{H}$ and $\boldsymbol{c}$ are the coefficients. Thus, we control our simulated human by supplying the simulator with $\boldsymbol{\tau}$ from which the forward dynamics algorithm solves for the resulting (potentially unconstrained) joint accelerations $\ddot{\boldsymbol{q}}$. Below is a brief description of the algorithms featured in Bullet and TDS.

Recursive Newton-Euler Algorithm (RNEA). An inverse dynamics algorithm that computes the torques $\boldsymbol{\tau}$ given accelerations $\ddot{\boldsymbol{q}}$. RNEA [37, 76] is used to compute $\boldsymbol{c}$ in (1.47).

Composite-Rigid-Body Algorithm (CRBA). An efficient approach to computing the $\boldsymbol{M}$ given a body configuration $\boldsymbol{q}$ is the composite-rigid-body algorithm [37, 129]. It recursively computes the inertia of each connected rigid body in $O(N^2)$, where N is the number of bodies.

Articulated Body Algorithm (ABA). Introduced in Featherstone [36], the articulated body algorithm is a forward dynamics algorithm for computing joint accelerations $\ddot{\boldsymbol{q}}$ given torques $\boldsymbol{\tau}$ in $O(N)$, where N is the number of joints in the system. ABA used by TDS.

Articulated Islands Algorithm (AIA). A forward dynamics algorithm intended improvement on ABA by combining it with the constraint-solving mechanism of the sequential

[1] Note the overloaded notation. In Featherstone's formulation, $\boldsymbol{q}(t)$ denotes the joint rotations in the kinematic skeleton, while in the maximal coordinate formulation (see (1.43)), it denotes the rigid body's orientation.

[2] The term forces is used as the joints may be either rotational or translational.

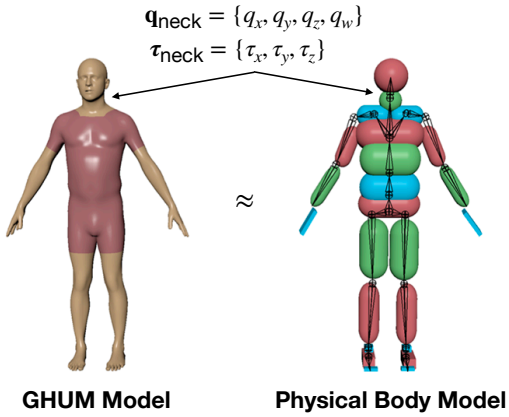


Figure 1.9: The physical body model used in Paper IV-V. The model is actuated by torque motors in the joints and has the same kinematic tree as the GHUM [143] model. The pose $\boldsymbol{q}$ is represented as the concatenation of all joint rotations. The torque vector $\boldsymbol{\tau}$ is the concatenation of all joint torque vectors.

impulse algorithm [22]. We empirically found that Bullet, which uses AIA, could support a larger simulation step size and was generally more stable than TDS.

6.2 Physical Body Model

The methods in Paper IV-V aim to support physics-based reconstruction of various dynamic motions, including complex full-body contacts. Therefore, our physical model must represent the entire body and approximate the subject's size to accurately track the motion in 3d. As our physical simulators require that the simulated subject consists of geometric primitives, we convert the subject's GHUM mesh into a rigid body approximation, see Fig. 1.9. We optimize the size parameters of the geometric primitives to approximate the volume of the subject's GHUM mesh using a loss similar to Al Borno et al. [3].

We compute the physical body's inertia tensors by estimating each body part's mass and computing the inertia tensors using each geometric primitive's analytical formula. The mass is estimated by first regressing the total mass of the subject from the volume of their GHUM mesh. We train a neural network regressor on the CAESAR [100] dataset containing body scans and the subjects' weight. The total body mass is then distributed among the body parts according to an average weight distribution from the medical literature.

The global position and pose of the physical body model is expressed through the pose vector

$$\boldsymbol{q} = [q_0, q_1, \ldots, q_J] \,, \tag{1.48}$$

where $\boldsymbol{q}_{0:3}$ contains the global position of the root joint in Cartesian coordinates, $\boldsymbol{q}_{3:7}$ contains the global orientation of the root joint, and $\boldsymbol{q}_{7:J}$ contains the concatenated quaternion body joint angles. To control the physical body we provide the physical simulator with the torque vector

$$\boldsymbol{\tau} = [\tau_0, \tau_1, \ldots, \tau_K] \,, \tag{1.49}$$

where $\boldsymbol{\tau}_{0:3}$ corresponds to the linear force and $\boldsymbol{\tau}_{3:6}$ corresponds to the torque acting on the root joint. We set $\boldsymbol{\tau}_{0:6} = 0$ to disable non-physical root residual forces, that is, external forces acting directly on the root joint. $\boldsymbol{\tau}_{6:K}$ contains the concatenated 3d torque vectors corresponding to the body joints in $\boldsymbol{q}_{7:J}$, see Fig. 1.9. Given 16 3d body joints we have $J = 71$ and $K = 54$.

While our physical body model presents a good first-order approximation, many possible improvements exist. For example, our model uses static torque limits, while Jiang et al. [57] synthesize body motions using learned pose-dependent torque limits. Using pose-dependent torque limits provides a useful physical prior, especially when visual evidence is lacking due to, e.g., blurred frames. Biomechanics researchers use OpenSim [113] and detailed muscle models to perform, for example, gait analysis [104]. However, this type of simulation is computationally more expensive than joint-actuated rigid body simulation. Lee et al. [68] present a reinforcement learning model for controlling a muscle-activated character in a rigid body simulator. However, their model only simulates the lower body and only learns to generate a specific motion. Using a mesh-based deformable body model would likely improve realism, especially for grasping, as noted by Jain and Liu [56]. However, simulating meshes is computationally expensive compared to rigid body simulation, and our work currently emphasizes body motion rather than grasping.

6.3 Trajectory Optimization

In Paper IV-V, we seek to reproduce a motion from a video in physical simulation to produce a physically constrained estimate of said motion. We first estimate the per-frame poses using a 3d human pose estimation network. Next, we need to recover the time-varying torques used to control the physical body model. Similar to Al Borno et al. [2], we recover the control signal through *trajectory optimization*[1] explained in the following section.

Given the forward simulation function D

[1]Also referred to as optimization with spacetime constraints [139].

$$\boldsymbol{s}_{t+1} = D(\boldsymbol{s}_t, \boldsymbol{\tau}_t) \,, \tag{1.50}$$

which takes the current state of the body $\boldsymbol{s}_t$ at time t together with a joint torque vector $\boldsymbol{\tau}_t$ and produces the state of the body at the next time step $t+1$. Controlling the body for T time steps from an initial state of $\boldsymbol{s}_0$ would result in the following sequence

$$\boldsymbol{s}_0, \boldsymbol{\tau}_0 \xrightarrow{D} \boldsymbol{s}_1, \boldsymbol{\tau}_1 \xrightarrow{D} \ldots \xrightarrow{D} \boldsymbol{s}_{T-1}, \boldsymbol{\tau}_{T-1} \xrightarrow{D} \boldsymbol{s}_T \,, \tag{1.51}$$

where $\boldsymbol{\tau}_{0:T-1} = \{\tau_0, \ldots, \tau_{T-1}\}$ is the time-varying torque vector controlling the body. Assuming we have a loss function $L(\boldsymbol{s}_t, \hat{\boldsymbol{s}}_t)$ that computes the error between the state of the physical body $\boldsymbol{s}_t$ and the desired state $\hat{\boldsymbol{s}}_t$, we can compute the loss for each time step as

$$\begin{array}{ccccccccc} \boldsymbol{s}_0 & , \boldsymbol{\tau}_0 \xrightarrow{D} & \boldsymbol{s}_1 & , \boldsymbol{\tau}_1 \xrightarrow{D} \ldots \xrightarrow{D} & \boldsymbol{s}_{T-1} & , \boldsymbol{\tau}_{T-1} \xrightarrow{D} & \boldsymbol{s}_T \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ L(\boldsymbol{s}_0, \hat{\boldsymbol{s}}_0) & & L(\boldsymbol{s}_1, \hat{\boldsymbol{s}}_1) & & L(\boldsymbol{s}_{T-1}, \hat{\boldsymbol{s}}_{T-1}) & & L(\boldsymbol{s}_T, \hat{\boldsymbol{s}}_T) \end{array} \tag{1.52}$$

with the total loss of the trajectory given by the sum

$$L(\boldsymbol{s}_{0:T}, \hat{\boldsymbol{s}}_{0:T}) = \sum_{t=0}^{T} L(\boldsymbol{s}_t, \hat{\boldsymbol{s}}_t) \,. \tag{1.53}$$

As the state $\boldsymbol{s}_t$ depends on the torque $\boldsymbol{\tau}_{0:t-1}$ and the initial state $\boldsymbol{s}_0$, we can rewrite (1.53) to define the goal of trajectory optimization as

$$\min_{\boldsymbol{\tau}_{0:T-1}} L(\boldsymbol{\tau}_{0:T-1}; \boldsymbol{s}_0, \hat{\boldsymbol{s}}_{0:T}) \,, \tag{1.54}$$

that is, finding the time-varying control vectors $\boldsymbol{\tau}_{0:T-1}$ that minimizes the reconstruction loss. Our work estimates the reference trajectory, $\hat{\boldsymbol{s}}_{0:T}$, which we aim to imitate from the video sequence using a neural network predictor. After optimization, the solution $\boldsymbol{\tau}^*_{0:T-1}$ will produce $\boldsymbol{s}_{0:T}$, a motion sequence close to $\hat{\boldsymbol{s}}_{0:T}$ but constrained by our dynamics D.

If (1.54) is differentiable, that is, if D and L are differentiable, then we may attempt to minimize the loss using gradient-based methods such as steepest descent or BFGS [39]. Otherwise, we must resort to a gradient-free method, such as CMA-ES [44].

6.3.1 PD Joint Control

The formulation in (1.54) has some practical issues. For one, physical simulators usually run at a high simulation frequency to ensure stability (anywhere between 100-1000 Hz). Optimizing control for one second of video at, e.g., 1000 Hz would result in the vast search space of $\boldsymbol{\tau} \in \mathbb{R}^{1000\times 54}$. Changing the control signal at 1000 Hz may also introduce jitter, hurting the reconstruction quality and making the motion look unnatural. Finally, while our simulation runs at high frequency, the video's visual evidence is usually captured at 25-50 frames per second. Thus, for most of our time steps, we have no visual evidence to imitate. Furthermore, if we optimize over time-varying joint torques, it is unclear how we should select the initial guess $\boldsymbol{\tau}^0$. One approach would be inverse dynamics, however, in the presence of contacts, this is non-trivial.

We follow Al Borno et al. [2] to address these issues and infer the torque vectors using a proportional-derivative (PD) controller. That is, rather than optimizing time-varying torque vectors, we optimize time-varying control signals in the form of desired joint angles and use PD controllers to compute the actual torque values. For a 1-DoF joint, we compute the torque as

$$\tau_t = k_p(\hat{q}_t - q_t) - k_d\dot{q}_t \;, \tag{1.55}$$

where $\hat{q}_t$ is the controller's target, q_t is the current joint angle, $\dot{q}_t$ is the current joint angle velocity, and k_p and k_d are constants of the controller. In our approach, we tune k_p and k_d manually while other methods [116, 150] predict them using neural networks. (1.55) can be thought of as a damped spring that generates torque that pulls the current joint angle q_t towards the target joint angle $\hat{q}_t$.

Rewriting our trajectory loss in (1.54) using PD control in (1.55) gives

$$\min_{\hat{\boldsymbol{q}}_{0:T}} L(\hat{\boldsymbol{q}}_{0:T}; \boldsymbol{s}_0, \hat{\boldsymbol{s}}_{0:T}) \;, \tag{1.56}$$

where we optimize the physical motion with respect to the time-varying vectors of PD control targets $\hat{\boldsymbol{q}}_{0:T}$. Note that the control targets $\hat{\boldsymbol{q}}_{0:T}$ are not the same as the actual joint angles $\boldsymbol{q}_{0:T}$ of the physical motion. The former are used to infer torque vectors using the PD rule in (1.55), while the latter are the *realized* poses of the physical motion. During optimization, we set the initial guess of the control targets $\hat{\boldsymbol{q}}^0_{0:T} = \boldsymbol{q}^{kin}_{0:T}$, where $\boldsymbol{q}^{kin}_{0:T}$ are the per-frame body joint angles estimated from video by a neural network predictor.

To decrease the optimization search space, we may downsample the control signal by varying $\hat{\boldsymbol{q}}_t$ at the frequency of the video's frame rate. When computing joint torques for sim-

ulation time steps in between two video frames, we use the next frame's control target, for example:

$$\tau_{t'} = k_p(\hat{q}_{t_2} - q_{t'}) - k_d \dot{q}_{t'} \ , \tag{1.57}$$

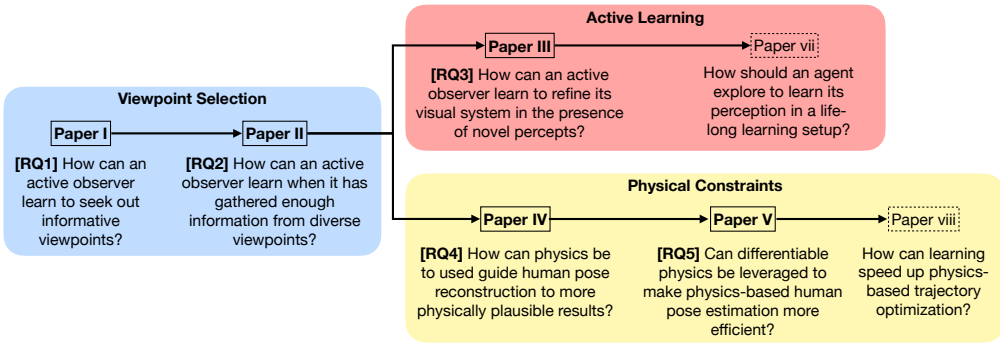
for $t_1 < t' < t_2$, where t_1 and t_2 are two consecutive frames of the video and t' is an intermediate control step.

6.4 Differentiable Physics

This thesis refers to differentiable physics as physics formulations where the forward simulation function D in (1.50) is differentiable. In that case, we may minimize the loss function in (1.56) using a gradient-based method. These methods tend to converge faster and suffer less from the curse of dimensionality than their gradient-free counterparts.

The development of differentiable physics simulators is an active area of research, with significant progress made in recent years [21, 40, 49, 53, 67, 79, 102, 103, 134, 144]. The different differentiable physics simulators differ both in their theoretical foundations as well as their practical implementations. For example, a great variety exists in how they compute contact forces, represent the rigid body state, compute gradients, and what type of collision geometries they support. They also differ in practical matters, such as in what programming language they are implemented, if they support automatic vectorization, and how efficient they are in terms of memory and computation.

Despite significant progress, differentiable simulators tend to be less sophisticated than their non-differentiable counterparts. Partly because rewriting simulation algorithms in auto differentiation frameworks or with analytical gradients is challenging. Hu et al. [53] found that the time-of-impact must be accurately computed. Otherwise, the gradients during contacts might be incorrect. Zhong et al. [158] further investigated the gradients of differentiable physics simulators and found that, depending on implementation, there were significant errors in the gradients during contacts. Metz et al. [83] concluded that computing gradients through multiple simulation steps (applications of D in (1.50)) had similar issues as backpropagation through recurrent neural networks. Depending on the system's dynamics, gradients could vanish or explode [15]. In Paper V, we use the Tiny Differentiable Simulator [49] (TDS), which uses a similar physics formulation as Bullet [28], that is, Featherstone's reduced-coordinates algorithms [37]. Gradients are computed using the automatic differentiation framework CppAD [13]. Similar to Xu et al. [144], we empirically found that, despite the difficulties mentioned above, by optimizing over small time windows ($\sim$1s), gradients were stable enough for optimization using BFGS [39].



Paper I	Domes to Drones: Self-Supervised Active Triangulation for 3D Human Pose Reconstruction
Paper II	Deep Reinforcement Learning for Active Human Pose Estimation
Paper III	Embodied Visual Active Learning for Semantic Segmentation
Paper IV	Trajectory Optimization for Physics-Based Reconstruction of 3d Human Pose from Monocular Video
Paper V	Differentiable Dynamics for Articulated 3d Human Motion Reconstruction
Paper vii	Embodied Learning for Lifelong Visual Perception
Paper viii	Transformer-Based Learned Optimization

Figure 1.10: Overview of the publications in the thesis (in bold), their main research questions, and their follow-up work not included in the thesis (in roman).

7 Conclusion and Outlook

The work of this thesis can roughly be divided into three separate categories (see Fig. 1.10), each focusing on different contributions to visual perception. This section discusses the conclusions, follow-up work, avenues for future work, and limitations of the work in this thesis.

Paper I–III study the question posed in RQ1, and each presents different learning-based approaches to seeking out informative viewpoints. For example, in active human pose estimation (studied in Paper I–II), the agent must learn to seek out occlusion-free viewpoints that complement previously seen views to reconstruct the subjects. Using a multi-camera dataset, we could train an active observer on real images of humans. The interesting next step would be to adapt this policy to control real drones in a motion capture system. Naturally, this would pose non-trivial challenges in terms of, for example, the sim-to-real gap and the delay between viewpoint selection drone movement. To that aim, Fan et al. [34] extended the active human pose estimation methodology for multiple agents and proposed a method for synchronization and communication between the agents. For the single drone case, Arzati and Arzanpour [10] presented a reinforcement learning-based agent similar to *ACTOR* for viewpoint selection for skin cancer detection using drone-based dermoscopy.

Regarding RQ2, Paper II studies how an active observer may learn to determine when it has gathered enough information. The Pose-DRL model learns to not only select viewpoints but to *stop* when it believes additional views will not reduce the reconstruction error. Our

experiments show that, for our problem, selecting a small set of informative views is not only efficient but also crucial, as fusing all views will include poor estimates that negatively impact the final reconstruction. This could be mitigated by improving the simple method for fusing poses. Rather than taking the median of several pose estimates, a neural network-based pose fusion function could be learned inspired by how Iskakov et al. [55] learned a network for pose triangulation.

Paper III presents a method for how an active observer may learn to refine its visual system (see RQ3). Unlike in classical active learning, the embodied agent must explore and seek out novel percepts. Defining novelty is, of course, a non-trivial exercise with many possible approaches, for example, through a self-supervised prediction error [95]. Our method implicitly learns to seek out novel views but integrating an explicit novelty estimate could perhaps further improve the agent's performance, as would reconstructing the environment into a 3d semantic map using the depth information associated with each view [24]. By using a 3d reconstructed map of the environment, semantic labels could be propagated much more efficiently between widely different views compared to the current approach of propagating labels using optical flow.

Paper III studies a "tabula rasa agent" training its perceptual model from scratch. Another setting of interest would be refining a pre-trained perceptual system deployed to a new environment, for example, a household robot trained in one house and then deployed in another. This problem is studied in the follow-up work of Nilsson et al. [91]. Another exciting line of work is adapting embodied visual active learning to human pose estimation. One possible approach could be to have multiple drones equipped with human pose estimators, requesting annotations if their predictions (captured from different viewpoints) are inconsistent. That would signify that the human pose estimator of at least one of the drones produces incorrect results.

This thesis proposed integrating physical constraints in the pose reconstruction process via a physical simulator to answer RQ4. The results presented in Paper IV were promising, but there are many avenues for future work. While integrating physics allows for a richer output with torque and force estimates, it also requires additional information about the subject and the scene. The torque and force estimates' accuracy depends on the body's estimated physical properties. Our proposed method roughly estimates the subject's mass distribution. A more fine-grained estimation could improve the accuracy of our model's physical outputs. While plausible, the joint torque limits used in our model are generous, and more biological limits, which in practice are pose-dependent, would increase the realism of the model [57]. Furthermore, modeling the body forces as individual muscle activations rather than the torques applied to the joints [68] would also increase physical realism. Finally, a more accurate representation of the subject would feature a deformable body that more closely resembles the human body, perhaps based on surface estimation [5, 132]. However, soft body simulation is traditionally more computationally expensive than rigid body

simulation. Hence, soft body simulation would further slow down the already costly reconstruction process. Another area for improvement when developing physics-based human pose estimation methods is the lack of real-world ground-truth data. This stems from the difficulty of measuring ground-truth values for physical quantities such as contact forces and body torques. As such, the estimates produced by most methods should only be considered rough estimates until their accuracy can be thoroughly evaluated.

While our proposed solution support interactions with objects (for example, sitting on a chair), it currently requires manual modeling of all objects in the scene. This limits what types of videos may be reconstructed without significant manual effort. Integrating methods [29, 147] for automatic 3d scene reconstruction would be a significant first step toward physically reconstructing subjects in more complex scenes. In the same vein, the current limitation of a static camera could be lifted by estimating the camera movement using structure from motion techniques [111]. Finally, the physics-based system should ideally reconstruct both object and person interactions. Thus supporting the simultaneous reconstruction of multiple people would be of great interest.

The thesis addressed RQ5 in Paper V by demonstrating how a differentiable physics simulator can be applied to physics-based 3d human pose estimation. This proved magnitudes faster than our gradient-free optimization technique while achieving similar results on our test sets. However, differentiable simulators are still the subject of active research, with many competing implementations. For example, the Tiny Differentiable Simulator used in our paper does not support check-pointing the computational graph[1] [103] nor pruning contacts based on distance. Thus, the memory usage is significant and linear with the optimization window size and quadratic in the number of contact points.

Using a differentiable simulator enables gradient-based system identification. Prior work has attempted system identification for simple structures using a differentiable renderer coupled with a differentiable simulator [50, 88]. Using a similar methodology to identify the body mass distribution and properties (such as friction and elasticity constants) of objects in the scene could increase the fidelity of the reconstruction and the estimated quantities.

Another avenue of future work is to employ machine learning-based methods for control together with the differentiable simulator. While capable of learning dynamic motions such as backflips [96], motion controllers trained using reinforcement typically are slow to train and tend not to generalize to motions different from those observed during training. However, Xu et al. [144] recently demonstrated how motion controllers may be trained magnitudes faster using reinforcement learning and the gradients from a differentiable simulator. Furthermore, there is active research into skill-reuse for learned motion controllers [99]. In

[1] Essentially computing parts of the graph as needed rather than storing the entire graph in memory at all times.

the future, a physics-based reconstruction system might be able to quickly reconstruct common motions with learned motion controllers and only resort to trajectory optimization in the case of especially difficult motions. Once reconstructed, a difficult motion could be added as training examples for the learned motion controller.

Integrating a differentiable physics simulator as a component during training of 3d human pose estimation networks could also be fruitful. For example, one could imagine a physics loss in addition to the standard pose estimation losses. The physics loss would steer the network toward predicting more physically plausible motions.

In addition, internal components of the physical simulator could be learned and specialized to specific tasks to speed up inference. One such example was presented by Fussell et al. [41]. They learn a neural network that can emulate a rigid body simulator for a human body. Another example is Heiden et al. [49], which replaces parts of the physics engine with learned components to bridge the sim-to-real gap by correcting artifacts caused by the simplifications of the rigid body formulation. Physics-based reconstruction could be made more efficient and more accurate using these approaches.

Finally, the benefits of research into active perception and physics-based inference are not limited to 3d human pose estimation. Constructing intelligent agents that learn how to acquire information, adapt to novel environments, and consider the effects of the physical world when reasoning are three fundamentally desirable traits. This thesis provides some insights, but the underlying research questions are open-ended and difficult to answer definitively. In the end, the author hopes that the effort put into this thesis produced a stepping stone that other researchers may tread on along the long journey toward general artificial intelligence.

8 Author Contributions

Co-authors are abbreviated as follows: Mykhaylo Andriluka (MA), Erwin Coumans (EC), Erik Gärtner (EG), David Nilsson (DN), Aleksis Pirinen (AP), Cristian Sminchisescu (CS), and Hongyi Xu (HX).

Paper I: Domes to Drones: Self-Supervised Active Triangulation for 3D Human Pose Reconstruction

The project was conceived by CS, EG, and AP. The method was developed by EG and AP with continuous feedback from CS. EG and AP jointly implemented the system and performed experiments. Finally, EG and AP wrote the manuscript with revisions from CS. Overall, EG and AP contributed equally to the project.

Paper II: Deep Reinforcement Learning for Active Human Pose Estimation

CS conceived the main idea, refined by EG and AP. EG and AP developed the method aided by CS. EG and AP implemented the experimental setup and performed all experiments together. EG and AP wrote the manuscript with significant contributions from CS. Overall, EG and AP contributed equally to the project.

Paper III: Embodied Visual Active Learning for Semantic Segmentation

DN and CS conceived the core idea of the project. The method was developed by DN and CS with continuous feedback from EG and AP. DN implemented most of the code with substantial contributions from EG and AP. Experiments were carried out mainly by DN and, to a lesser extent, by EG and AP. EG contributed with compute infrastructure software and support. Finally, DN, AP, and EG wrote the manuscript with feedback from CS.

Paper IV: Trajectory Optimization for Physics-Based Reconstruction of 3d Human Pose from Monocular Video

CS conceived the initial project idea with input from EG, MA, and HX. The method was developed by EG and MA with continuous feedback from CS and HX. EG and MA wrote most of the code, while HX wrote the initial version of the kinematics pipeline. EG and MA carried out experiments together. EG, MA, HX, and CS wrote and revised the paper.

Paper V: Differentiable Dynamics for Articulated 3d Human Motion Reconstruction

EG, MA, and CS conceived the project. EG and MA developed the method. EC provided technical expertise regarding the differentiable physics simulator and physical simulation in general. EG wrote the vast majority of the code with help from MA. EG formulated and performed most experiments with assistance from MA. EG wrote the manuscript with revisions from MA and CS.

References

[1] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):44–58, 2005. 10

[2] Mazen Al Borno, Martin De Lasa, and Aaron Hertzmann. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics*, 19(8):1405–1414, 2012. 34, 36

[3] Mazen Al Borno, Ludovic Righetti, Michael J Black, Scott L Delp, Eugene Fiume, and Javier Romero. Robust physics-based motion retargeting with realistic body shapes. In *Computer Graphics Forum*, volume 37, pages 81–92. Wiley Online Library, 2018. 33

[4] Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imghum: Implicit generative models of 3d human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5461–5470, 2021. 10

[5] Thiemo Alldieck, Mihai Zanfir, and Cristian Sminchisescu. Photorealistic monocular 3d reconstruction of humans wearing clothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1506–1515, 2022. 39

[6] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021. 23

[7] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014. 11

[8] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988. 22

[9] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005. 10

[10] Mojtaba Ahangar Arzati and Siamak Arzanpour. Viewpoint selection for dermdrone using deep reinforcement learning. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 544–553. IEEE, 2021. 38

[11] David Baraff. Linear-time dynamics using lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, 1996. 31

[12] Eduard Gabriel Bazavan, Andrei Zanfir, Mihai Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Hspace: Synthetic parametric humans animated in complex environments. *arXiv preprint arXiv:2112.12867*, 2021. 11

[13] Bradley M. Bell. Cppad: A c++ algorithmic differentiation package, 2021. 37

[14] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957. 14

[15] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994. 37

[16] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38(6):1–11, 2019. 26

[17] Marcus A Brubaker and David J Fleet. The kneed walker for human pose tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 24

[18] Marcus A Brubaker, Leonid Sigal, and David J Fleet. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2389–2396. IEEE, 2009. 24

[19] Marcus A Brubaker, David J Fleet, and Aaron Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International journal of computer vision*, 87(1):140–155, 2010. 24

[20] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 9, 12

[21] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019. 37

[22] Erin Catto et al. Fast and simple physics using sequential impulses. In *Proceedings of game developer conference*, 2006. 33

[23] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 23

[24] Devendra Singh Chaplot, Murtaza Dalal, Saurabh Gupta, Jitendra Malik, and Russ R Salakhutdinov. Seal: Self-supervised embodied active learning using exploration and 3d consistency. *Advances in Neural Information Processing Systems*, 34:13086–13098, 2021. 39

[25] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, 2020. 11

[26] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994. 22

[27] Cl Connolly. The determination of next best views. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 432–435. IEEE, 1985. 21

[28] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. 28, 37

[29] Manuel Dahnert, Ji Hou, Matthias Nießner, and Angela Dai. Panoptic 3d scene reconstruction from a single rgb image. *Advances in Neural Information Processing Systems*, 34:8282–8293, 2021. 40

[30] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2018. 22

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 10

[32] Sai Kumar Dwivedi, Nikos Athanasiou, Muhammed Kocabas, and Michael J Black. Learning to regress bodies from images using differentiable semantic rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11250–11259, 2021. 10

[33] Petros Faloutsos, Michiel Van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 251–260, 2001. 25

[34] Zhen Fan, Xiu Li, and Yipeng Li. Multi-agent deep reinforcement learning for online 3d human poses estimation. *Remote Sensing*, 13(19):3995, 2021. 38

[35] Anthony C Fang and Nancy S Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)*, 22(3):417–426, 2003. 26

[36] Roy Featherstone. The calculation of robot dynamics using articulated-body inertias. *The international journal of robotics research*, 2(1):13–30, 1983. 32

[37] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014. 25, 31, 32, 37

[38] Mihai Fieraru, Mihai Zanfir, Teodor Szente, Eduard Bazavan, Vlad Olaru, and Cristian Sminchisescu. Remips: Physically consistent 3d reconstruction of multiple interacting people under weak supervision. *Advances in Neural Information Processing Systems*, 34:19385–19397, 2021. 10, 27

[39] Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, NY, USA, 1987. 35, 37

[40] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021. 37

[41] Levi Fussell, Kevin Bergamin, and Daniel Holden. Supertrack: Motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021. 41

[42] Herbert Goldstein, Charles Poole, and John Safko. Classical mechanics, 2002. 30

[43] Héctor H González-Banos and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11): 829–848, 2002. 21

[44] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*, pages 75–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32494-2. doi: 10.1007/3-540-32494-1_4. URL https://doi.org/10.1007/3-540-32494-1_4. 35

[45] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 13

[46] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997. 12, 13

[47] Mohamed Hassan, Duygu Ceylan, Ruben Villegas, Jun Saito, Jimei Yang, Yi Zhou, and Michael J Black. Stochastic scene-aware motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11374–11384, 2021. 11

[48] Mohamed Hassan, Partha Ghosh, Joachim Tesch, Dimitrios Tzionas, and Michael J Black. Populating 3d scenes by learning human-scene interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14708–14718, 2021. 11

[49] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL `https://github.com/google-research/tiny-differentiable-simulator`. 28, 37, 41

[50] Eric Heiden, Ziang Liu, Vibhav Vineet, Erwin Coumans, and Gaurav S Sukhatme. Inferring articulated rigid body dynamics from rgbd video. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022. 40

[51] Cherie Ho, Andrew Jong, Harry Freeman, Rohan Rao, Rogerio Bonatti, and Sebastian Scherer. 3d human reconstruction in the wild with collaborative aerial cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5263–5269. IEEE, 2021. 21

[52] David T Hoffmann, Dimitrios Tzionas, Michael J Black, and Siyu Tang. Learning to train with synthetic humans. In *German conference on pattern recognition*, pages 609–623. Springer, 2019. 11

[53] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019. 37

[54] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7): 1325–1339, 2013. 11

[55] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7718–7727, 2019. 39

[56] Sumit Jain and C Karen Liu. Controlling physics-based characters using soft contacts. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–10, 2011. 34

[57] Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)*, 38(4):1–12, 2019. 26, 34, 39

[58] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015. 11, 12, 21

[59] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020. 23

[60] Atul Kanaujia, Cristian Sminchisescu, and Dimitris Metaxas. Semi-supervised hierarchical models for 3d human pose reconstruction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 10

[61] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018. 11

[62] Sena Kiciroglu, Helge Rhodin, Sudipta N Sinha, Mathieu Salzmann, and Pascal Fua. Activemocap: Optimized viewpoint selection for active human motion capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 103–112, 2020. 21

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 11

[64] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5253–5263, 2020. 12

[65] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 23

[66] A Krizhevsky, I Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. 2012 advances in neural information processing systems (nips). *Neural Information Processing Systems Foundation, La Jolla, CA*, 2012. 10

[67] Quentin Le Lidec, Igor Kalevatykh, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6(2):3413–3420, 2021. 37

[68] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38 (4):1–13, 2019. 34, 39

[69] David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995. 22

[70] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13401–13412, 2021. 12

[71] Zongmian Li, Jiri Sedlar, Justin Carpentier, Ivan Laptev, Nicolas Mansard, and Josef Sivic. Estimating 3d motion and forces of person-object interactions from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8640–8649, 2019. 24

[72] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 9, 11

[73] C Karen Liu and Sumit Jain. A short tutorial on multibody dynamics. *Georgia Institute of Technology, School of Interactive Computing, Tech. Rep. GIT-GVU-15-01-1*, 8, 2012. 27

[74] C Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (TOG)*, 24 (3):1071–1081, 2005. 26

[75] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 10

[76] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. 1980. 32

[77] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J Black. Learning to dress 3d people in generative clothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6469–6478, 2020. 10

[78] Qianli Ma, Shunsuke Saito, Jinlong Yang, Siyu Tang, and Michael J Black. Scale: Modeling clothed humans with a surface codec of articulated local elements. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16082–16093, 2021. 10

[79] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC). 37

[80] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings*

of the IEEE/CVF international conference on computer vision, pages 5442–5451, 2019. 11

[81] Miguel Mendoza, J Irving Vasquez-Gomez, Hind Taud, L Enrique Sucar, and Carolina Reta. Supervised learning of the next-best-view for 3d object reconstruction. *Pattern Recognition Letters*, 133:224–231, 2020. 21

[82] Dimitris Metaxas and Demetri Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993. 24

[83] Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021. 37

[84] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021. 23

[85] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 16

[86] Igor Mordatch, Jack M Wang, Emanuel Todorov, and Vladlen Koltun. Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 32(6):1–8, 2013. 26

[87] Lea Muller, Ahmed AA Osman, Siyu Tang, Chun-Hao P Huang, and Michael J Black. On self-contact and human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9990–9999, 2021. 10

[88] J Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2020. 40

[89] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 36(4):1–10, 2017. 21

[90] Tobias Nägeli, Samuel Oberholzer, Silvan Plüss, Javier Alonso-Mora, and Otmar Hilliges. Flycon: real-time environment-independent multi-view human pose estimation with aerial vehicles. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. 21

[91] David Nilsson, Aleksis Pirinen, Erik Gärtner, and Cristian Sminchisescu. Embodied learning for lifelong visual perception. *arXiv preprint arXiv:2112.14084*, 2021. 39

[92] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. 26

[93] Ahmed A A Osman, Timo Bolkart, and Michael J. Black. STAR: A sparse trained articulated human body regressor. In *European Conference on Computer Vision (ECCV)*, pages 598–613, 2020. URL `https://star.is.tue.mpg.de`. 10

[94] Priyanka Patel, Chun-Hao P Huang, Joachim Tesch, David T Hoffmann, Shashank Tripathi, and Michael J Black. Agora: Avatars in geography optimized for regression analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13468–13478, 2021. 11

[95] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017. 39

[96] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018. 26, 40

[97] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)*, 37(6):1–14, 2018. 26

[98] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021. 26

[99] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022. 26, 40

[100] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 2017. 33

[101] Zoran Popović and Andrew Witkin. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20, 1999. 26

[102] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. Scalable differentiable physics for learning and control. In *International Conference on Machine Learning*, pages 7847–7856. PMLR, 2020. 37

[103] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pages 8661–8671. PMLR, 2021. 37, 40

[104] Apoorva Rajagopal, Christopher L Dembia, Matthew S DeMers, Denny D Delp, Jennifer L Hicks, and Scott L Delp. Full-body musculoskeletal model for muscle-driven simulation of human gait. *IEEE transactions on biomedical engineering*, 63(10): 2068–2079, 2016. 34

[105] Deva Ramanan. Learning to parse images of articulated bodies. *Advances in neural information processing systems*, 19, 2006. 10

[106] Davis Rempe, Leonidas J Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. Contact and human dynamics from monocular video. In *European conference on computer vision*, pages 71–87. Springer, 2020. 24, 25

[107] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021. 22

[108] Nitin Saini, Eric Price, Rahul Tallamraju, Raffi Enficiaud, Roman Ludwig, Igor Martinovic, Aamir Ahmad, and Michael J Black. Markerless outdoor human motion capture using multiple autonomous micro aerial vehicles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 823–832, 2019. 21

[109] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019. 9, 10

[110] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019. 23

[111] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 40

[112] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 19

[113] Ajay Seth, Jennifer L Hicks, Thomas K Uchida, Ayman Habib, Christopher L Dembia, James J Dunne, Carmichael F Ong, Matthew S DeMers, Apoorva Rajagopal,

Matthew Millard, et al. Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS computational biology*, 14(7):e1006223, 2018. 34

[114] Burr Settles. Active learning literature survey. 2009. 22

[115] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics (ToG)*, 39(6):1–16, 2020. 24, 25

[116] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics (ToG)*, 40(4):1–15, 2021. 25, 26, 36

[117] Leonid Sigal. Human pose estimation. In *Computer Vision: A Reference Guide*, pages 573–592. Springer, 2021. 9

[118] Leonid Sigal, Michael Isard, Benjamin Sigelman, and Michael Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. *Advances in neural information processing systems*, 16, 2003. 10

[119] Jakub Stępień. *Physics-Based Animation of Articulated Rigid Body Systems for Virtual Environments*. PhD thesis, Ph. D. Dissertation. Silesian University of Technology, 2013. 27

[120] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 23

[121] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 14, 18

[122] Rahul Tallamraju, Eric Price, Roman Ludwig, Kamalakar Karlapalem, Heinrich H Bülthoff, Michael J Black, and Aamir Ahmad. Active perception based formation control for multiple aerial vehicles. *IEEE Robotics and Automation Letters*, 4(4):4491–4498, 2019. 21

[123] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012. 26

[124] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014. 10

[125] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of 28th British Machine Vision Conference*, pages 1–13, 2017. 11

[126] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 109–117, 2017. 11

[127] Timo Von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 601–617, 2018. 11

[128] Marek Vondrak, Leonid Sigal, and Odest Chadwicke Jenkins. Physical simulation for probabilistic motion tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 24

[129] Michael W Walker and David E Orin. Efficient dynamic computer simulation of robotic mechanisms. 1982. 32

[130] Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012. 26

[131] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. Deep 3d human pose estimation: A review. *Computer Vision and Image Understanding*, 210:103225, 2021. 9

[132] Shaofei Wang, Marko Mihajlovic, Qianli Ma, Andreas Geiger, and Siyu Tang. Metaavatar: Learning animatable clothed human models from few depth images. *Advances in Neural Information Processing Systems*, 34:2810–2822, 2021. 39

[133] Xiaolin Wei and Jinxiang Chai. Videomocap: Modeling physically realistic human motion from monocular video sequences. In *ACM SIGGRAPH 2010 papers*, pages 1–10. 2010. 24

[134] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *arXiv preprint arXiv:2103.16021*, 2021. 37

[135] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2019. 22

[136] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. 17

[137] Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018. 25

[138] Andrew Witkin and David Baraff. Physically based modeling: Principles and practice: Differential equation basics. *Course Note A SIGGRAPH*, 97, 1997. 27, 28, 30, 31

[139] Andrew Witkin and Michael Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988. 34

[140] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4):33–1, 2020. 26

[141] Yan Wu, Jiahao Wang, Yan Zhang, Siwei Zhang, Otmar Hilliges, Fisher Yu, and Siyu Tang. Saga: Stochastic whole-body grasping with contact. *arXiv preprint arXiv:2112.10103*, 2021. 10, 11

[142] Kevin Xie, Tingwu Wang, Umar Iqbal, Yunrong Guo, Sanja Fidler, and Florian Shkurti. Physics-based human motion estimation and synthesis from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11532–11541, 2021. 25

[143] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Ghum & ghuml: Generative 3d human shape and articulated pose models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6184–6193, 2020. 9, 10, 33

[144] Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022. 37, 40

[145] Lan Xu, Yebin Liu, Wei Cheng, Kaiwen Guo, Guyue Zhou, Qionghai Dai, and Lu Fang. Flycap: Markerless motion capture using multiple autonomous flying cameras. *IEEE transactions on visualization and computer graphics*, 24(8):2284–2297, 2017. 21

[146] Jianwei Yang, Zhile Ren, Mingze Xu, Xinlei Chen, David J Crandall, Devi Parikh, and Dhruv Batra. Embodied amodal recognition: Learning to move to perceive objects. In *ICCV*, pages 2040–2050, 2019. 21, 22

[147] Hongwei Yi, Chun-Hao P Huang, Dimitrios Tzionas, Muhammed Kocabas, Mohamed Hassan, Siyu Tang, Justus Thies, and Michael J Black. Human-aware object placement for visual environment reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3959–3970, 2022. 11, 27, 40

[148] Ri Yu, Hwangpil Park, and Jehee Lee. Human dynamics from monocular video with dynamic camera movements. *ACM Transactions on Graphics (TOG)*, 40(6): 1–14, 2021. 26

[149] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. *Advances in Neural Information Processing Systems*, 33:21763–21774, 2020. 25

[150] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7159–7169, 2021. 25, 26, 36

[151] Andrei Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes-the importance of multiple scene constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2148–2157, 2018. 11, 27

[152] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *European Conference on Computer Vision*, pages 465–481. Springer, 2020. 12

[153] Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Neural descent for visual 3d human pose and shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14484–14493, 2021. 10, 11

[154] Mihai Zanfir, Andrei Zanfir, Eduard Gabriel Bazavan, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Thundr: Transformer-based 3d human reconstruction with markers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12971–12980, 2021. 9

[155] Petrissa Zell, Bastian Wandt, and Bodo Rosenhahn. Joint 3d human motion capture and physical analysis from monocular videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 17–26, 2017. 24

[156] Jason Y. Zhang, Sam Pepose, Hanbyul Joo, Deva Ramanan, Jitendra Malik, and Angjoo Kanazawa. Perceiving 3d human-object spatial arrangements from a single image in the wild. In *European Conference on Computer Vision (ECCV)*, 2020. 27

[157] Yan Zhang and Siyu Tang. The wanderings of odysseus in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20481–20491, 2022. 11

[158] Yaofeng Desmond Zhong, Jiequn Han, and Georgia Olympia Brikis. Differentiable physics simulations with contacts: Do they have correct gradients wrt position, velocity and control? In *ICML 2022 2nd AI for Science Workshop*. 37

[159] Xiaowei Zhou, Sikang Liu, Georgios Pavlakos, Vijay Kumar, and Kostas Daniilidis. Human motion capture using a drone. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2027–2033. IEEE, 2018. 21

[160] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017. 22

[161] Yuliang Zou, Jimei Yang, Duygu Ceylan, Jianming Zhang, Federico Perazzi, and Jia-Bin Huang. Reducing footskate in human motion reconstruction with ground contact constraints. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 459–468, 2020. 27

Image Credits

Front cover. The author partially generated this image with DALL·E 2, OpenAI's large-scale image-generation model. The image is a composition of multiple outputs from DALL·E 2, together with manual edits by the author. The printing of the image is in accordance with the usage policy: *"Subject to the Content Policy and Terms, you own the images you create with DALL·E, including the right to reprint, sell, and merchandise – regardless of whether an image was generated through a free or paid credit"*[1].

Fig. 1.2. All photos are used with permission under the *Unsplash license*[2].

- Photo a by Kelsey Chance, `https://unsplash.com/photos/CutTQTt2HyI`.
- Photo b by Hannah Skelly, `https://unsplash.com/photos/4aJ9VluXAgo`.
- Photo c by Peggy Anke, `https://unsplash.com/photos/V_-0E6zIWjU`.
- Photo d by jurien huggins, `https://unsplash.com/photos/drIebcFZqpE`.
- Photo e by Luke Stackpoole, `https://unsplash.com/photos/gxof7su_KJk`.

[1] `https://help.openai.com/en/articles/6425277-can-i-sell-images-i-create-with-dall-e`

[2] `https://unsplash.com/license`

Part II

Scientific Publications

Paper I

Aleksis Pirinen*, **Erik Gärtner***, Cristian Sminchisescu
Domes to Drones: Self-Supervised Active Triangulation for 3D Human Pose Reconstruction
Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2019

Domes to Drones: Self-Supervised Active Triangulation for 3D Human Pose Reconstruction

Aleksis Pirinen[1] **Erik Gärtner**[1*]
Cristian Sminchisescu[1,2]

[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]Google Research

Abstract

Existing state-of-the-art estimation systems can detect 2d poses of multiple people in images quite reliably. In contrast, 3d pose estimation from a single image is ill-posed due to occlusion and depth ambiguities. Assuming access to multiple cameras, or given an *active* system able to position itself to observe the scene from multiple viewpoints, reconstructing 3d pose from 2d measurements becomes well-posed within the framework of standard multi-view geometry. Less clear is what is an informative set of viewpoints for accurate 3d reconstruction, particularly in complex scenes, where people are occluded by others or by scene objects. In order to address the view selection problem in a principled way, we here introduce *ACTOR*, an *active triangulation agent for 3d human pose reconstruction*. Our fully trainable agent consists of a 2d pose estimation network (any of which would work) and a deep reinforcement learning-based policy for camera viewpoint selection. The policy predicts observation viewpoints, the number of which varies adaptively depending on scene content, and the associated images are fed to an underlying pose estimator. Importantly, training the view selection policy requires *no annotations* – given a pre-trained 2d pose estimator, ACTOR is trained in a self-supervised manner. In extensive evaluations on complex multi-people

*Denotes equal contribution, order determined by coin flip.

scenes filmed in a Panoptic dome, under multiple viewpoints, we compare our active triangulation agent to strong multi-view baselines, and show that ACTOR produces significantly more accurate 3d pose reconstructions. We also provide a proof-of-concept experiment indicating the potential of connecting our view selection policy to a physical drone observer.

1 Introduction

Estimating 2d and 3d human pose from *given* images or video is an active research area, with deep learning playing a prominent role in most of today's state-of-the-art pose and shape estimation models [2, 5, 25, 26, 27, 31, 38]. Monocular 3d pose estimation is however ill-posed [30] due to depth ambiguities, and these cannot always be resolved by priors or by increasing a feed-forward model's predictive power. Given access to multiple cameras, or given an *active* observer which can capture images from multiple viewpoints, reconstructing 3d pose from 2d estimates however becomes tractable within the framework of standard multi-view geometry. An active setup for triangulating 2d estimates also addresses many common practical issues, such as partial observability due to occlusion, either self-induced or due to other people or objects.

Given sufficiently many viewpoints, 3d pose reconstructions from 2d estimates can be made robust and accurate, and such results have even been used as (pseudo)ground-truth [21, 37]. While inferring 3d reconstructions from tens or hundreds of viewpoints works in carefully constructed setups, it is not always practical or desirable to rely on so many cameras. In this work we take a different approach, introducing *ACTOR*, an *active triangulation agent for obtaining 3d human pose reconstructions*. ACTOR consists of a 2d pose (human body joints) estimation network (any of which could be used) and a deep reinforcement learning-based policy for observer (i.e. camera location and pose) prediction, within a fully trainable system. Instead of operating exhaustively over all cameras, ACTOR is able to select a much smaller set of cameras yet still produces accurate 3d pose reconstructions. Our proposed methodology is implemented in the Panoptic multi-view framework [21], where the scene can be observed in time-freeze, from a dense set of viewpoints, and over time, providing a proxy for an active observer. In evaluations using Panoptic we show that our system learns to select camera locations that yield more accurate 3d pose reconstructions compared to strong multi-view baselines. We also provide a proof-of-concept experiment indicating the potential of connecting ACTOR to a physical drone observer. Training our policy for view selection requires no 2d or 3d pose annotations – given a pre-trained 2d pose estimator, ACTOR can be trained in a self-supervised manner.

Related Work. In addition to recent literature focusing on extracting 3d human representations from a single image or video [3, 25, 26, 27, 28, 31, 33, 38, 39], a parallel line

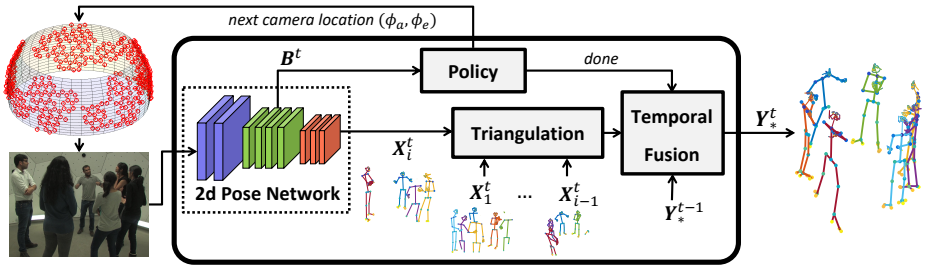


Figure 1.11: Overview of *ACTOR*, our active triangulation agent for 3d human pose reconstruction. A random view is initially given and the image is fed to a 2d body joint predictor yielding estimates for all visible people ($\boldsymbol{X}_i^t$) and the core of the agent's state ($\boldsymbol{B}^t$). The policy network predicts camera locations until all joints have been triangulated, then switches to the next active-view at time $t+1$. The predicted camera location is encoded via spherical angles relative to the agent's location on the viewing sphere, and the closest camera is selected. When the agent is done it outputs $\boldsymbol{Y}_\star^t$, the final 3d reconstructions of all people in the scene, which are obtained by a combination of spatial fusion (triangulation of 2d poses $\boldsymbol{X}_1^t, \ldots, \boldsymbol{X}_k^t$) and temporal fusion with $\boldsymbol{Y}_\star^{t-1}$ on a per-joint basis. As described in §3.2, we train the viewpoint selection policy using self-supervision.

of work concentrates on lifting 2d estimates to 3d. [6] present an unsupervised approach for recovering 3d human pose from 2d estimates in single images. This is achieved by a self-consistency loss based on a lift-reproject-lift process, relying on a network that discriminates between real and fake 2d poses in the reprojection step. Related ideas based on an adversarial framework [11] are also pursued in [9]. A self-supervised learning methodology for monocular 3d human pose estimation is described in [23]. During training, the system leverages multi-view 2d pose estimates and epipolar geometry to obtain 3d pose estimates, which are then used to train the monocular 3d pose prediction system. These weakly-supervised methods for monocular 3d pose estimation eliminate the need for expensive 3d ground-truth annotations but tend to not be as accurate as their fully-supervised counterparts.

Multi-view frameworks can, on the other hand, rely on triangulation in order to obtain accurate 3d pose reconstructions given 2d estimates. In contrast to methods performing exhaustive fusion over all cameras, ACTOR actively selects a smaller subset of viewpoints over which to triangulate. Our approach can be considered as a generalization of next-best-view (NBV) selection, and is superficially similar to other NBV-works [14, 15, 17, 18, 20, 32, 36]. Differently from them, the number of viewpoints explored by our agent varies adaptively based on the complexity of the scene. Also, NBV approaches typically decide the next view by greedily and locally evaluating some hand-crafted utility function exhaustively over a set of candidates – we instead frame the task as a deep RL problem where the policy is trained to maximize an explicit *global objective*, searching over entire sequences of viewpoints, and by triangulating as many joints as possible. In a broader sense, ACTOR relates to work on active agents trained to perform various tasks in 3d environments [1, 7, 8, 12, 35, 40]. We are not aware of any prior work that tackles the

problem of active triangulation in multi-view setups.

2 Human Pose Reconstruction from Active Triangulation

We here describe the terminology and concepts of 3d human pose reconstruction from active triangulation. The proposed framework is applicable to any number of people as we aim for a system able to actively reconstruct *all* people in the scene, the number of which may vary. We study the active triangulation problem in the CMU Panoptic multi-camera framework [21] since its data consists of real videos of people and allows for reproducible experiments. The subjects are filmed by densely positioned time-synchronized HD-cameras as they perform movements ranging from basic pose demonstrations to different social interactions. Panoptic offers 2d and 3d joint annotations, but as we will show no such annotations are required for training our viewpoint selection system. See Fig. I.11 for an overview of our active 3d human pose reconstruction model.

Terminology. Triangulation of 3d pose reconstructions from 2d estimates requires observing the targets from several cameras, each capturing an image v_i^t (referred to as a *view* or *viewpoint*) indexed by time-step t and camera i. The set $\{v_1^t, \dots, v_N^t\}$ of all views in a time-step t is called a *time-freeze*. A subset of these is an *active-view*, $\mathcal{V}^t = \{v_1^t, \dots, v_k^t\}$, which contains k cameras selected (by some agent or heuristic) from the time-freeze at time t. A sequence of temporally contiguous active-views is referred to as an *active-sequence*, $\mathcal{S}^{1:T} = \{\mathcal{V}^1, \mathcal{V}^2, \dots, \mathcal{V}^T\}$, where T is its length. Unless the context requires both indices we will omit the time super-script t to simplify notation, which implies that all elements belong to the same timestep. The set of all predicted 2d pose estimates corresponding to a view v_i is denoted $\boldsymbol{X}_i = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_M] \in \mathbb{R}^{30 \times M}$, where $\boldsymbol{x}$ is a single 2d pose estimate, based on detecting 15 human body joints, and M is the number of people observed from that viewpoint.

Task description. *Active triangulation for 3d human pose reconstruction* is the task of producing active-views with corresponding accurate fused 3d pose reconstructions for all people present, $\boldsymbol{Y}_\star = [\boldsymbol{y}_{1\star}, \dots, \boldsymbol{y}_{M\star}]$, given 2d pose estimates $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ associated with the active-view. These active-views then form an active-sequence of accurate 3d pose reconstructions. As it is challenging to select appropriate viewpoints for satisfactory triangulation, especially in crowded scenes where people are often occluding each other, the task is considered completed once each individual's joint has been observed from at least two different viewpoints (the minimum requirement for performing a triangulation), or after a given exploration budget is exceeded.

Matching and triangulating people. The active triangulation system must tackle the problems of tracking and identifying people across various views and through time. The agent

receives appearance models[1] for the different people at the beginning of an active-sequence. For each view, the agent compares the people detected by the 2d pose estimator with the given appearance models and matches them across space and time using the Hungarian algorithm. To reconstruct 3d poses from 2d estimates associated with the selected viewpoints, we compute triangulation between each pair of viewpoints [16, 24] and perform per-body-joint fusion (averaging) of the associated 3d reconstructions. More sophisticated triangulation methods would be possible; here we selected pairwise averaging due to computational efficiency which is important during training.

3 Active Triangulation Agent

We now introduce our active triangulation agent, ACTOR, and describe its state representation and action space in §3.1. In §3.2 we describe the *annotation-free* reward signal for training ACTOR to efficiently triangulate the joints of all people.

In the first active-view $\mathcal{V}^1$, the agent is given an initial random view v_1^1. It then predicts camera locations $v_2^1, \ldots, v_k^1$ until the active-view is completed. An active-view is considered completed once the agent has triangulated the joints of all people within the time-freeze, or after a given exploration budget has been exceeded. The 2d pose estimator is computed for images collected at every visited viewpoint v_i^t, yielding estimates $\boldsymbol{X}_i^t$ for all visible people. Camera locations are specified by the relative azimuth and elevation angles (jointly referred to as *spherical angles*) on the viewing sphere.

Once the agent has triangulated the joints of all people within a time-freeze, it continues to the next active-view $\mathcal{V}^{t+1}$. At this time the triangulated 3d pose reconstructions $\boldsymbol{Y}^t$ are temporally fused with the reconstructions $\boldsymbol{Y}_\star^{t-1}$ from the previous active-view, $\boldsymbol{Y}_\star^t = f(\boldsymbol{Y}_\star^{t-1}, \boldsymbol{Y}^t)$. As the 2d pose estimator we use in this work is accurate, we have opted for a straightforward temporal fusion. We define $I = I_{\text{tri}} \cup I_{\text{miss}}$, where I indexes all joints, I_{tri} indexes the successfully triangulated joints in the current time-step, and I_{miss} indexes joints missed in the current time-step. Then we set $\boldsymbol{Y}_\star^t[I_{\text{tri}}] = \boldsymbol{Y}^t[I_{\text{tri}}]$ for the joints that were successfully triangulated in the current time-step, and $\boldsymbol{Y}_\star^t[I_{\text{miss}}] = \boldsymbol{Y}_\star^{t-1}[I_{\text{miss}}]$. Hence we temporally propagate from the previous time-step only those joint reconstructions that were missed in the current time-step. The initial viewpoint v_1^{t+1} for $\mathcal{V}^{t+1}$ is set to the final viewpoint v_k^t of $\mathcal{V}^t$, i.e. $v_1^{t+1} = v_k^t$. The process repeats until the end of the active-sequence. Fig. 1.11 shows a schematic overview of ACTOR.

[1] Instance-sensitive features generated using a VGG-19 based [29] siamese instance classifier, trained with a contrastive loss to differentiate people on the training set.

3.1 State-Action Representation

In this section, while describing the state and action representations, we will assume that the agent acts in a single time-freeze. This allows us to simplify notation and index steps within the active-view by t. The state is represented as a tuple $S^t = (\boldsymbol{B}^t, \boldsymbol{C}^t, \boldsymbol{u}^t)$, where $\boldsymbol{B}^t \in \mathbb{R}^{H \times W \times C}$ is the deep feature map from the 2d pose estimator. $\boldsymbol{C}^t \in \mathbb{N}^{w \times h \times 2}$ is a *camera history*, which encodes[1] the previously visited cameras on the rig. It also contains a representation of the distribution of cameras on the rig. The auxiliary array $\boldsymbol{u}^t \in \mathbb{R}^{17}$ contains the number of actions taken, the number of people detected, as well as a binary vector indicating which joints have been triangulated for all people.

A deep stochastic policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{c}^t|S^t)$ parametrized by $\boldsymbol{\theta}$ is used to predict the next camera location $\boldsymbol{c}^t = (\phi_a^t, \phi_e^t)$, were (ϕ_a^t, ϕ_e^t) is the azimuth-elevation angle pair encoding the camera location. To estimate the camera location probability density, the base feature map $\boldsymbol{B}^t$ is processed through two convolutional blocks. The output of the second convolutional block is concatenated with $\boldsymbol{C}^t$ and $\boldsymbol{u}^t$ and fed to the policy head, consisting of 3 fully-connected layers with $\tanh$ activations.

As the policy predicts spherical angles, we choose to sample these from the periodical von Mises distribution. We use individual distributions in the azimuth and elevation directions. The probability density function for the azimuth angle is given by

$$\pi_{\boldsymbol{\theta}}\left(\phi_a^t|S^t\right) = \frac{1}{2\pi I_0(m_a)} \exp\{m_a \cos(\phi_a^t - \tilde{\phi}_a(\boldsymbol{w}_a^\top \boldsymbol{z}_a^t + b_a))\} \tag{1.58}$$

where the zeroth-order Bessel function I_0 normalizes (1.58) to a probability distribution on the unit circle. Here $\tilde{\phi}^a$ is the mean of the distribution (parameterized by the deep network), m_a is the precision parameter,[2] and $\boldsymbol{w}_a$ and b_a are trainable weights and bias, respectively. The second to last layer of the policy head outputs $\boldsymbol{z}_a^t$. For the azimuth prediction, the support is the full circle. Therefore we set

$$\tilde{\phi}_a(\boldsymbol{w}_a^\top \boldsymbol{z}_a^t + b_a) = \pi \tanh(\boldsymbol{w}_a^\top \boldsymbol{z}_a^t + b_a) \tag{1.59}$$

The probability density for the elevation prediction has the same form (1.58) as the azimuth. As there are no cameras below the ground-plane of the rig, nor cameras directly above the people (cf. Fig. 1.11), we limit the elevation angle range to $[-\kappa, \kappa]$, where $\kappa = \pi/6$. Thus the mean elevation angle becomes

$$\tilde{\phi}_e(\boldsymbol{w}_e^\top \boldsymbol{z}_e^t + b_e) = \kappa \tanh(\boldsymbol{w}_e^\top \boldsymbol{z}_e^t + b_e) \tag{1.60}$$

[1] The camera history consists of w bins in the azimuth direction and h bins in the elevation direction. It is agent-centered, i.e. relative to the agent's current viewpoint. We set $w = 9$ and $h = 5$.

[2] The precision parameters m_a and m_e are treated as constants, but we anneal them over training as the policy becomes better at predicting camera locations.

3.2 Reward Signal for Self-Supervised Active Triangulation

As explained in §3, ACTOR predicts camera locations until the individual body joints of all people have been detected from at least two different views (minimum requirement for 3d triangulation) or after a given exploration budget B is exceeded; we set $B = 10$ during training. We use the indicator variable d_t to denote whether or not the agent has triangulated all joints ($d_t = 1$ if all joints have been triangulated). We want to encourage the agent to fulfill the task while selecting as few camera locations as possible, which gives rise to the reward design in (1.61) below. Note that our reward is *not* based on ground-truth pose annotations – it relies solely on automatic 2d pose (body joint) detections.

$$r^t = \begin{cases} -\beta/M, & \text{if } d_t = 0, t < B \text{ and camera not already visited} \\ -\beta/M - \epsilon, & \text{if } d_t = 0, t < B \text{ and camera already visited} \\ 1, & \text{if } d_t = 1, t \leq B \\ \tau_{min}, & \text{if } d_t = 0, t = B \end{cases} \tag{1.61}$$

The first and second rows of (1.61) reflect intermediate rewards, where the agent receives a penalty ϵ (we set $\epsilon = 2.5$) if it predicts a previous camera location. To encourage efficiency the agent also receives a time-step penalty β for not yet having completed the triangulation (β is set to 0.2). This penalty is normalized by the number of people M for scaling purposes, as we expect more cameras be required to triangulate multiple people. The third and fourth rows represent rewards the agent obtains at the end of the active-view. It receives $+1$ if it triangulates the joints of all M persons within its exploration budget B. The fourth row defines the reward if the agent fails to triangulate some joints within the exploration budget. It then receives the minimum fraction of covered joints for any person, τ_{min}. Policy gradients are used to learn ACTOR's policy parameters, where we maximize expected cumulative reward on the training set with the objective $J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s} \sim \pi_{\boldsymbol{\theta}}} \left[\sum_{t=1}^{|\boldsymbol{s}|} r^t \right]$, where $\boldsymbol{s}$ denotes state-action trajectories. This objective function is approximated using REINFORCE [34].

4 Experiments

Dataset. We consider both multi-people scenes (named *Mafia* and *Ultimatum* in Panoptic) and single-people ones (*Pose*). The scenes with multiple people are expected to be particularly challenging for the agent, as occlusions are common. Panoptic data comes as 30 FPS time-synchronized videos. To make the size more manageable and increase movement between frames we downsample the data to 2 FPS. We use the HD cameras, of which there are about 30 per scene, since they provide better image quality than VGA and are sufficiently dense, yet spread apart far enough to make each viewpoint unique. We select 20

Table 1.1: Mean 3d reconstruction error (mm/joint) for ACTOR and baselines on the Panoptic test sets. *Multi* denotes multi-people data (union of *Mafia* and *Ultimatum*); *single* is the single-person *Pose* split. We show total errors which include translation errors (top) and hip-aligned errors (bottom). Columns indicate the number of cameras inspected, ranging from 2 to 10. We also show results for auto-mode, where camera location selection terminates when the joints of all people have been triangulated, but using 10 cameras at most. For this column we also show the average number of cameras inspected in parentheses. ACTOR outperforms both the heuristic baselines on all types of scenes. The advantage of a trained system is most pronounced for complex multi-people scenes where selecting informative viewpoints is important. ACTOR-ob and ACTOR-ntf denote ablated versions of our agent, cf. §4.2.

Model	Data	Auto	2	3	4	5	6	7	8	9	10
ACTOR	multi	125.6 (8.84)	502.4	281.5	201.0	168.4	151.6	141.2	132.1	126.1	122.1
		96.2 (8.84)	247.2	179.3	146.4	131.1	118.5	111.6	101.9	95.2	92.3
	single	74.6 (4.28)	172.1	107.5	81.9	71.2	67.1	64.9	63.3	62.1	61.3
		60.5 (4.28)	151.3	92.8	68.9	59.4	55.6	53.2	51.3	49.9	49.0
ACTOR-ob	multi	148.9 (8.79)	555.2	372.9	276.4	217.4	185.2	166.6	154.0	146.1	142.5
		108.1 (8.79)	299.6	305.7	231.2	182.4	155.9	131.9	119.4	112.3	109.3
	single	80.2 (4.58)	187.3	122.6	95.1	80.6	72.4	68.9	67.4	67.0	66.8
		67.3 (4.58)	159.7	104.4	77.7	64.2	56.6	53.3	52.3	51.8	51.6
ACTOR-ntf	multi	138.9 (8.84)	925.7	565.2	353.1	242.8	196.5	172.2	154.8	143.7	136.6
		102.0 (8.84)	334.4	258.0	198.4	159.1	138.4	124.5	112.0	102.9	98.3
	single	75.9 (4.28)	274.0	151.4	99.6	79.3	71.8	67.9	65.5	63.9	62.7
		61.6 (4.28)	228.1	132.4	85.3	66.9	59.8	55.9	53.3	51.5	50.3
Random	multi	142.7 (9.34)	570.1	469.9	316.1	259.9	269.3	238.5	220.2	198.8	188.3
		125.9 (9.34)	347.3	406.4	350.1	278.0	263.0	218.8	196.2	179.5	160.0
	single	82.6 (4.90)	203.6	139.4	107.2	89.9	81.1	75.1	71.0	67.9	65.8
		68.7 (4.90)	178.0	125.7	93.8	76.4	67.6	61.3	56.8	53.4	51.0
Max-Azim	multi	132.0 (9.01)	479.3	375.8	288.4	226.0	195.7	170.2	149.2	137.7	128.6
		102.7 (9.01)	259.4	282.1	235.0	200.0	196.8	158.2	131.3	114.1	103.7
	single	75.5 (4.41)	185.7	119.5	88.0	79.5	73.7	68.8	64.5	63.2	62.1
		63.61 (4.41)	161.2	106.3	76.5	67.7	61.0	56.3	52.0	50.0	48.5
Oracle	multi	94.5 (6.67)	254.4	147.6	113.1	98.2	90.3	86.4	84.1	82.8	81.9
		74.0 (6.67)	163.1	110.3	89.2	78.8	72.8	69.0	66.4	64.5	63.0
	single	54.0 (2.97)	123.0	60.2	49.2	45.3	43.6	42.8	42.3	42.2	42.4
		48.1 (2.97)	108.2	54.5	43.3	39.5	37.5	36.2	35.2	34.6	34.2

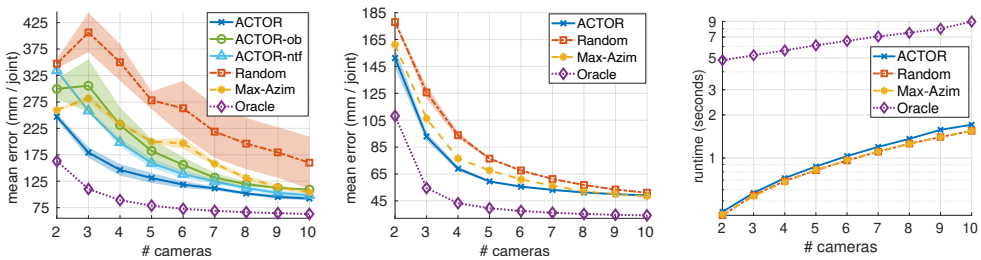


Figure 1.12: Column 1-2: Mean 3d reconstruction error per joint vs number of cameras on the test sets (means and standard errors over 5 seeds). Column 1: Multi-people data. Column 2: Single-people data. ACTOR decreases errors faster than baselines, particularly for multi-people data with occlusions. The oracle uses 3d ground-truth and is shown as gold standard. ACTOR also outperforms the ablated variants ACTOR-ob and ACTOR-ntf, cf. §4.2. Ablated models are not plotted in the single-people setting to avoid visual clutter – see Table 1.1 for these results. Column 3: Runtime (log-scale) per active-view vs number of cameras for a 3-people scene. The oracle computes errors using 3d ground-truth for all views and persons to select its next camera, making it very slow.

scenes (343k images) which are split randomly into training, validation and test sets with 10, 4, and 6 scenes, respectively. There is no overlap of scenes between the three sets, which

forces the agent to learn a fairly general policy.

Implementation details. ACTOR is implemented on top of the OpenPose 2d pose estimation system [5], though any 2d pose predictor would work. As described in §3, temporal fusion of 3d reconstructions across active-views ensures that missed joints are instead covered by the associated estimates from an earlier point in time. In case there is no previous estimate for a missing joint, it is set to the average of the successfully triangulated ones (to be able to compute errors). We use per-joint median averaging for fusing 3d pose reconstructions across views and temporal steps.

Training. We train the policy network with batches consisting of experiences from 5 active-sequences, each of length 10. Adam [22] is used for parameter updates. We normalize cumulative rewards for each episode to zero mean and unit variance over each batch to reduce variance in the policy updates. The exploration budget B (maximum trajectory length) is set to 10 camera locations per active-view, including the initial camera. The policy is trained for 75k episodes with learning rate initially set to `5e-7`, then halved after 720k steps and again after 1440k steps. The precision parameters (m_a, m_e) of the von Mises distributions are linearly annealed from $(1, 10)$ to $(25, 50)$ during training, which makes the camera prediction increasingly deterministic as the training progresses.

Baselines. We evaluate ACTOR against several multi-view baselines. They use the same 2d pose estimator, matching algorithm, triangulation method and temporal fusion. All methods receive the same initial random camera at the start of an active-sequence. We compare to the following baselines: i) *Random:* Selects random cameras (it never selects the same camera twice); *Max-Azim:* The first three views are selected at 90, 180 and 270 degrees azimuth relative to the initial view, so the four first views are at 90 degrees azimuth from each other. The subsequent four views are also selected at 90 degrees azimuth from each other, but at a 45 degree azimuth offset relative to the first four views. At each azimuth, it samples a random elevation angle. The last 2 cameras are selected randomly, and we ensure each camera is different. This baseline produces a wide coverage of the viewing sphere without the need to know in advance how many cameras will be selected; iii) *Oracle:* Before selecting one camera, this computes the improvement in 3d pose reconstruction error associated with all available cameras. It then selects the camera that maximally decreases the error. In addition to cheating when it selects views, the oracle is also impractically slow since it exhaustively computes errors for all cameras in each step. Thus it is only shown as a gold standard.

4.1 Main Results

Our ACTOR agent is compared to the baselines on the Panoptic test sets on active-sequences consisting of 10 active-views. We train ACTOR with 5 different random network initializa-

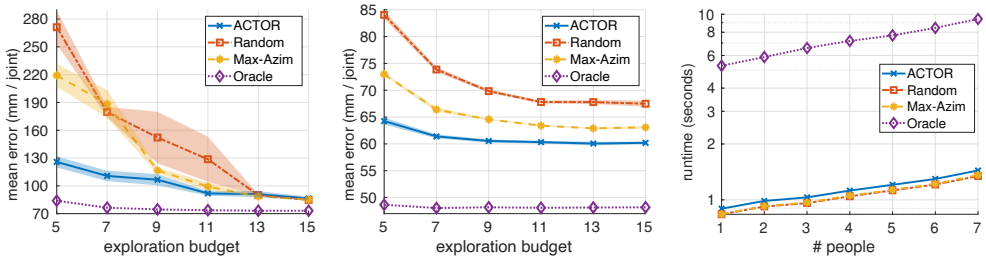


Figure 1.13: Column 1-2: Mean 3d reconstruction error per joint vs exploration budget B (maximum number of cameras) on the test sets. As mentioned in §3.2, ACTOR was trained solely at budget $B = 10$. Column 1: Multi-people data. Column 2: Single-people data. The relative gain to the baselines is higher at smaller exploration budgets (e.g. 93 mm/joint improvement over *Max-Azim* on multi-people data at $B = 5$), where the system quickly needs to select cameras triangulating the joints of all people. The accuracy curves are flatter for single-people data, as in general the systems need fewer cameras to triangulate the joints of a single person – the models hence tend to stop before their budget is exhausted. Column 3: Runtime (log-scale) when varying the number of people in a scene while keeping the the number of selected cameras constant at 6 per active-view.

tions and report average results with standard errors of the means (we early stop training for each network initialization based on errors on the validation set). For the non-deterministic heuristic baselines (*Random* and *Max-Azim* – the oracle is deterministic) we report results across 5 seeds, including standard errors of the means. In Table 1.1 we report 3d pose reconstruction errors for auto stopping and for a fixed number of views. ACTOR is more accurate and uses fewer cameras on average, compared to the heuristic baselines. Fig. 1.12 shows 3d pose reconstruction error versus number of views. ACTOR significantly outperforms the heuristic baselines, especially for complex multi-person scenes (e.g. 103 and 78 mm/joint improvements over *Max-Azim* at 3 and 6 cameras, respectively). Multi-people scenes are more difficult to analyze due to occlusions and thus require intelligent viewpoint selection – one clearly sees the advantages of a learned system in such scenarios.

Fig. 1.13 shows how the exploration budget B (max number of views) affects 3d reconstruction error. At smaller budgets ACTOR's improvements over the heuristic baselines are even larger, which shows that our trained system is significantly more efficient at finding good views over which to triangulate the body joints. Runtimes versus number of cameras are shown in column 3 of Fig. 1.12 and versus number of people in column 3 of Fig. 1.13. OpenPose runs at about 0.134 seconds per image, while the policy network inference has an overhead of 0.005 seconds per action, which is negligible compared to the 2d pose estimator. For visualizations[1] of ACTOR operating in various scenes, see Fig. 1.15.

[1]In this case we equip ACTOR with an OpenPose system that estimates detailed faces, hands and feet. We do not refine the pre-trained ACTOR model that was trained using the standard OpenPose estimator.

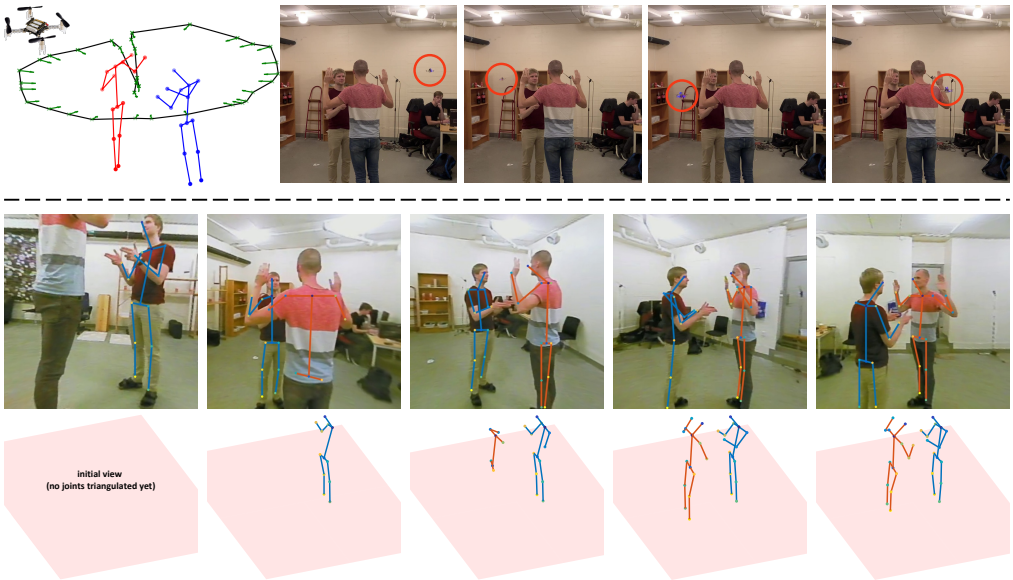


Figure 1.14: From domes to drones. Proof-of-concept experiment illustrating that ACTOR can be connected to an active drone observer to reconstruct 3d poses from informative viewpoints. Above the dashed line to the left we show the drone's loop (the sharp peak is due to take-off and landing), with sampled camera locations as green arrows. We also show the 3d pose reconstructions obtained by triangulating from *all* 33 sampled camera locations. The 9-by-9 cm Crazyflie drone used is shown in the very top left corner; it can be used safely due to its small size and weight. Sample locations of the drone are also shown above the line (drone locations are highlighted with red circles in images). Below the line we show views seen by ACTOR and aggregated 3d pose reconstructions. After observing 5 viewpoints, the two bodies are fully 3d reconstructed, with an average 2d reprojection error of 11.5 pixels (averaged over all 33 cameras), significantly better than the exhaustively triangulated reconstructions to the left, with an average reprojection error of 35.4 pixels.

4.2 Ablation Studies

In this section we study how ACTOR is affected by i) removing all state features except the deep feature blob $\boldsymbol{B}^t$ (ACTOR-ob; *ob* stands for *only blob*), and ii) using no temporal fusion of 3d pose reconstructions (ACTOR-ntf). Similarly as for the main ACTOR model, ACTOR-ob is trained over 5 different network initializations with individual early stopping on the validation set (ACTOR-ntf uses the same parameters as ACTOR but without temporal fusion during inference). The results are shown in Table 1.1 and Fig. 1.12. The full ACTOR agent outperforms the ablated variants for all data splits. For multi-people data, ACTOR drastically outperforms ACTOR-ob, which indicates the need of representing earlier visited cameras ($\boldsymbol{C}^t$) as part of the state space. For single-people data, ACTOR-ob is almost as good as ACTOR, but this data is very simple and occlusion-free and does not require too sophisticated camera selection. Finally, the full agent outperforms ACTOR-ntf when operating using few cameras, which makes sense as there is a big risk of the system missing to triangulate some joints, in which case a backup from earlier active-views may help.

4.3 From Domes to Drones

The dense Panoptic multi-camera dome provides an idealization in which we can generate controllable and reproducible experiments. It is also useful for training ACTOR, as we do not actually have to move a camera around. However, in many practical scenarios one does not have access to a multi-view setup and may instead have to resort to a single but moving camera. One such scenario is that of a drone circling a set of people, and aiming to reconstruct their 3d poses.

To test ACTOR's drone-controlling capacity, we captured three small scenes where a drone circles around two people performing various poses. We then fine-tuned ACTOR with learning rate `1e-6` for 3k episodes (15 minutes) on two scenes, keeping all other hyperparameters the same, and ran the model on the third scene. In Fig. 1.14, ACTOR selects 5 different views to reconstruct the targets. It should be noted that the setting of this drone experiment differs drastically from that of Panoptic. For example, the drone's camera quality is worse (VGA rather than HD), and the loop generated by the drone has a much smaller radius than Panoptic's viewing sphere (about 1.5 meters for the drone versus about 3 meters for Panoptic), so there are fewer views where e.g. the feet are visible. In future work we plan to more tightly integrate ACTOR in the loop, so as to direct the drone to observe targets from informative views.

5 Conclusions

We have presented *ACTOR*, a deep RL-based agent to actively reconstruct 3d poses from 2d estimates via triangulation. Training the viewpoint selection policy requires no annotations and only uses an off-the-shelf 2d human pose estimator for self-supervision. We evaluated the model in complex scenarios with multiple interacting people and showed that by intelligently selecting informative views the agent outperforms strong multi-view baselines in both speed and accuracy. We also provided proof-of-concept results which indicate that ACTOR can be used in single-camera settings, e.g. to control a physical drone observer.

Acknowledgments: This work was supported by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535 and PCCF-2016-0180, the EU Horizon 2020 Grant DE-ENIGMA, Swedish Foundation for Strategic Research (SSF) Smart Systems Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We would also like to thank Patrik Persson for support with the drone experiments.

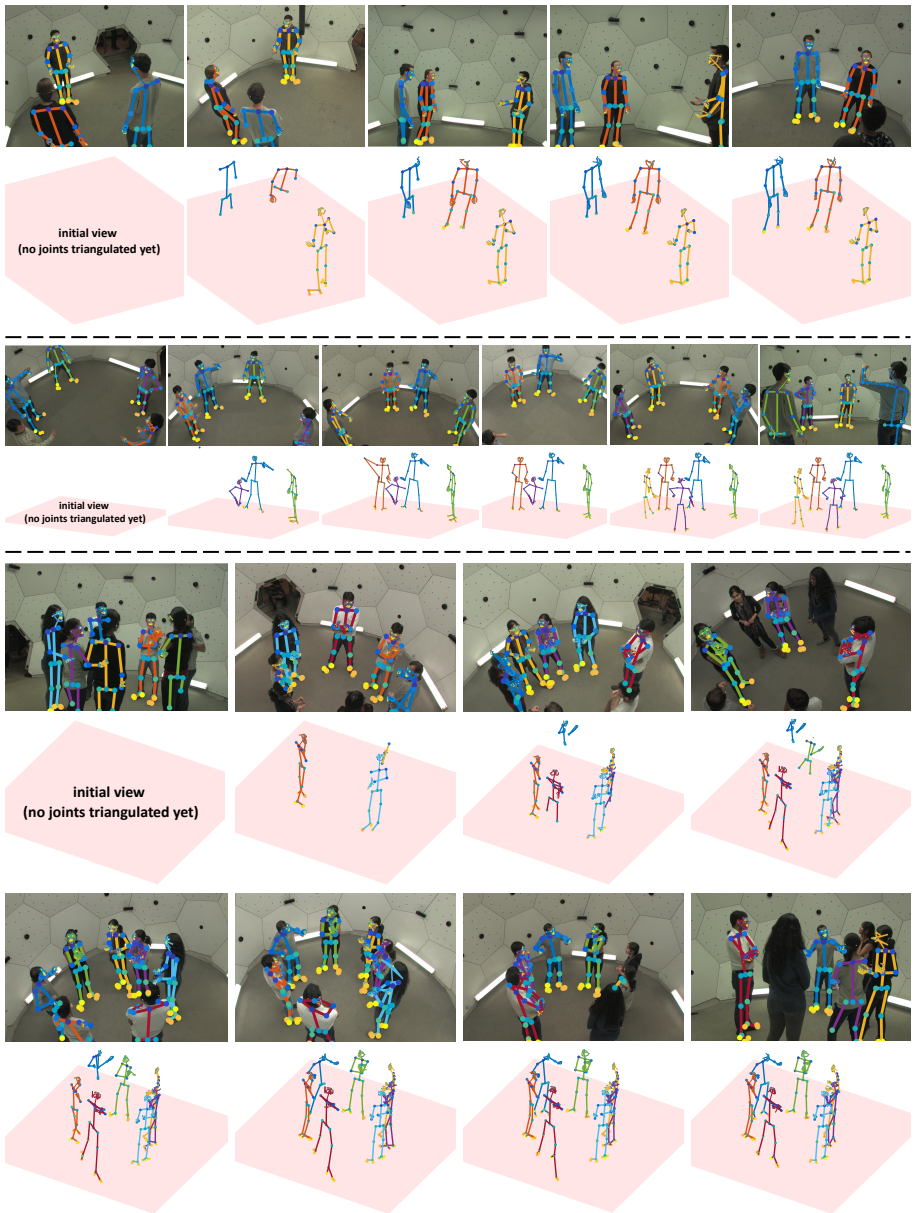


Figure 1.15: ACTOR operating in three different multi-people scenes (examples delimited by dashed lines). Visualizations are shown for initial active-views and thus have no propagated 3d pose estimates from earlier time steps. Each example shows the views selected by ACTOR, including 2d pose estimates (first view randomly given). Below these we show aggregated 3d pose reconstructions. Top: 3-person scene. One of the persons is reconstructed already at the 2nd view; all of them are reconstructed at the 5th view. The mean 3d reconstruction error decreases from 268 to 51 mm/joint between the 2nd and last view. Middle: 5-person scene, where 3d pose reconstructions improve over the 6 views. The error decreases from 296 to 68 mm/joint. Bottom: 6-person scene, where people stand quite close to each other, which makes it difficult to triangulate all joints due to occlusions. ACTOR observes the scene from 8 diverse views, and the error decreases from 342 to 69 mm/joint.

References

[1] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Košecká, and Alexander C Berg. A dataset for developing and benchmarking active vision. In *ICRA*, pages 1378–1385. IEEE, 2017. 67

[2] A Arnab, C Doersch, and A Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *CVPR*, 2019. 66

[3] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it SMPL: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, 2016. 66

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *NIPS*, pages 737–744, 1994. 83

[5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *CVPR*, 2017. 66, 73, 82

[6] Ching-Hang Chen, Ambrish Tyagi, Amit Agrawal, Dylan Drover, Rohith MV, Stefan Stojanov, and James M Rehg. Unsupervised 3d pose estimation with geometric self-supervision. In *CVPR*, pages 5714–5724, 2019. 67

[7] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, volume 5, page 6, 2018. 67

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 67

[9] Dylan Drover, Ching-Hang Chen, Amit Agrawal, Ambrish Tyagi, Cong Phuoc Huynh, et al. Can 3d pose be learned from 2d projections alone? In *ECCV*, pages 78–94. Springer, 2018. 67

[10] Erik Gärtner. Hyperdock. `https://github.com/ErikGartner/Hyperdock`, 2019. 82

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 67

[12] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *CVPR*, pages 4089–4098, 2018. 67

[13] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, pages 1735–1742. IEEE, 2006. 83

[14] Xiaoguang Han, Zhaoxuan Zhang, Dong Du, Mingdai Yang, Jingming Yu, Pan Pan, Xin Yang, Ligang Liu, Zixiang Xiong, and Shuguang Cui. Deep reinforcement learning of volume-guided progressive view inpainting for 3d point scene completion from a single depth image. In *CVPR*, pages 234–243, 2019. 67

[15] Sebastian Haner and Anders Heyden. Covariance propagation and next best view planning for 3d reconstruction. In *ECCV*, pages 545–556. Springer, 2012. 67

[16] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997. doi: 10.1006/cviu.1997.0547. 69

[17] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *ECCV*, pages 489–505. Springer, 2016. 67

[18] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, 2018. 67

[19] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 82

[20] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, pages 3813–3822, 2016. 67

[21] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, 2015. 66, 68, 84

[22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 73

[23] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Self-supervised learning of 3d human pose using multi-view geometry. In *CVPR*, 2019. 67

[24] Manolis Lourakis. Stereo triangulation. https://www.mathworks.com/matlabcentral/fileexchange/67383-stereo-triangulation, Nov 2018. Retrieved May 22, 2019. 69

[25] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. 66

[26] Alin-Ionut Popa, Mihai Zanfir, and Cristian Sminchisescu. Deep multitask architecture for integrated 2d and 3d human sensing. In *CVPR*, 2017. 66

[27] Helge Rhodin, Nadia Robertini, Dan Casas, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. General automatic human shape and motion capture using volumetric contour cues. In *ECCV*, 2016. 66

[28] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net: Localization-classification-regression for human pose. In *CVPR*, 2017. 66

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 69, 83

[30] C. Sminchisescu and B. Triggs. Building Roadmaps of Minima and Transitions in Visual Models. *International Journal of Computer Vision*, 61(1), 2005. 66

[31] Bugra Tekin, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Learning to fuse 2d and 3d image cues for monocular body pose estimation. In *ICCV*, pages 3941–3950, 2017. 66

[32] J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. Volumetric next-best-view planning for 3d object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 11(10):159, 2014. 67

[33] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, pages 4724–4732, 2016. 66

[34] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992. 71

[35] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, pages 9068–9079, 2018. 67

[36] Bo Xiong and Kristen Grauman. Snap angle prediction for 360 panoramas. In *ECCV*, pages 3–18, 2018. 67

[37] Zhixuan Yu, Jae Shin Yoon, Prashanth Venkatesh, Jaesik Park, Jihun Yu, and Hyun Soo Park. Humbi 1.0: Human multiview behavioral imaging dataset. *arXiv preprint arXiv:1812.00281*, 2018. 66

[38] Andrei Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes–the importance of multiple scene constraints. In *CVPR*, pages 2148–2157, 2018. 66

[39] Andrei Zanfir, Elisabeta Marinoiu, Mihai Zanfir, Alin-Ionut Popa, and Cristian Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *NeurIPS*, pages 8410–8419, 2018. 66

[40] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017. 67

Supplementary Material

This supplementary provides more insight into our ACTOR model and experimental setup. Section §A describes the details of the network architecture, implementation, and hyperparameters. §B elaborates on how we match 2d pose estimates in space and time using instance features. In §C we provide 2d reprojection errors onto 2d OpenPose [5] estimates on the Panoptic test splits. Finally, §D describes further dataset details.

A Model Architecture

See Fig. A.16 for a full description of the ACTOR network architecture. ACTOR was implemented in Caffe[19] and MATLAB. We used an open-source TensorFlow implementation of OpenPose[1]. All code and pre-trained weights have been made publicly available.[2]

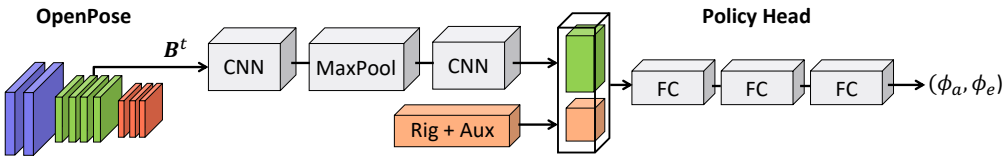


Figure A.16: ACTOR policy architecture. A multi-people 2d pose estimation system (here OpenPose, but any similar deep system would would work) processes an input image. The deep feature maps $\boldsymbol{B}^t$ produced by OpenPose (`conv4_4_CPM`) is fed into the ACTOR policy network and is processed by two convolutional layers with ReLU-activations. The first and second convolutional layers both have 3×3 kernels with stride 1. Their output dimensions are $8 \times 39 \times 21$ and $4 \times 18 \times 9$, respectively. The max pooling layer has a 2×2 kernel with stride 2. The output from the second convolutional layer is then concatenated with agent-centric camera rig information about previously visited cameras relative to current position (*Rig*), and auxiliary information such as the number of joints triangulated and number of people detected in the view (*Aux*). The flattened and concatenated data is then fed into three fully-connected layers with tanh-activations with 1024, 512 and 2 neurons respectively. The final output is scaled by two constants to produce radial angles on the viewing sphere.

A.1 Hyperparamters

Hyperparameter search was performed using two powerful workstations equipped with several NVIDIA Titan V100 GPU:s. Training a single model for 40k episodes took about 32 hours using one GPU and to speed up results while searching for optimal hyperparameters we trained several model configurations in parallell using Hyperdock[10]. The most important parameters for training ACTOR were learning rate, precision of the the von Mises (m_a, m_e) and the annealing rate of the precision. See Table A.2 for a summary of the values tested for these hyperparameters. In total we trained around 10 different versions of the

[1] https://gist.github.com/alesolano/b073d8ec9603246f766f9f15d002f4f4
[2] https://github.com/ErikGartner/actor

final model with varying hyperparameters and evaluated each of them on the validation set. Finally, the best model was evaluated on the test dataset and retrained with four additional random seeds to measure the model's sensitivity to the random seed (the model is not very sensitive as indicated in Fig. 2, main paper).

Table A.2: The values tested for the most important hyperparameters when training ACTOR. The final and best values are highlighted in bold. For the von Mises precisions, the arrow indicates linear annealing performed during training (e.g. from $(m_a, m_e) = (1, 10)$ to $(m_a, m_e) = (25, 50)$ for the best configuration).

Hyperparameter	Attempted values
Learning Rate	{1e-7, **5e-7**, 1e-6, 5e-6}
von Mises precision	{**(1, 10)** $\rightarrow$ **(25, 50)**, (10, 50) $\rightarrow$ (20, 100), (10,50) $\rightarrow$ (100, 500)}

B Matching Multiple People

ACTOR reconstructs multiple people in both space and time from 2d pose estimates. In order to track and match these estimates we compute instance sensitive features. These deep features can then be stably matched to each other using the Hungarian algorithm, where the L2 distance is used to compute the matching cost.

We trained an instance classifier structured as a siamese network[4] using a contrastive loss [13] that aims to produce 50-dimensional features for each person that can be used to distinguish individuals. As input the instance classifier takes VGG-19[29] features from the bounding box of the 2d pose estimate. The instance classifier is trained for 40k iterations on the training set with a mini-batch size of 16 where half contains positive pairs and the other half contains negative pairs. The training examples are sampled randomly in both space and time yielding a robust classifier. Lastly, the instance classifier is fine-tuned for 2k iterations on each scene, creating scene-specific versions of the classifier that are slightly adapted to the environment of those scenes. This tuning is performed outside the range of the active-sequence in which the agent operates.

At the start of an active-sequence the agents is given an appearance model for each target it should reconstruct. These appearance models are averages of K different instance features computed for each target in the scene but from time-freezes that are *not part of the current active-sequence*. We denote the i:th instance feature for the l:th person by $\boldsymbol{u}_i^l$, with $i = 1, \ldots, K$. In practice we use $K = 10$. Then we set as appearance model $\boldsymbol{m}^l$ to:

$$\boldsymbol{m}^l = \text{median}(\boldsymbol{u}_1^l, \ldots, \boldsymbol{u}_K^l) \tag{62}$$

For each camera location we compute the distance between the instance features of each detected person to all appearance models in that scene. This gives us a cost matrix whose

elements are $c^{j,l} = \|\boldsymbol{u}^j - \boldsymbol{m}^l\|_2^2$, i.e., the cost to match detection j to person l. Given this matrix we assign detections according to the Hungarian algorithm. Since there might be false detections by the 2d pose estimator and not all people are visible from every camera location we filter out matches with a cost larger than a threshold $\mathcal{C}$, such that all matches satisfy $c^{j,l} \leq \mathcal{C}$ (we set $\mathcal{C} = 0.5$).

If a person is never detected in an active-view, and if it does not have a previous temporal backup to use as 3d pose reconstruction (cf. §3 and the implementation details in §4 of the main paper), we set each joint estimate to the ground-truth center hip location. Obviously, this estimate is implausible and highly inaccurate – it is used only to compute average errors (not including such an estimate when computing average errors would be another option, but this would not penalize missed persons).

C Reprojection Errors onto OpenPose 2d Estimates

The 3d ground-truth in Panoptic is generated from exhaustive triangulation of 2d pose estimates [21], but those 2d pose estimates are not from OpenPose. Thus it is relevant to also look at reprojection errors onto the OpenPose 2d estimates, since these errors are not affected by any potential incorrect bias in the 3d ground-truth. Such reprojection errors are shown in Fig. D.17. We note that ACTOR is more accurate relative to the oracle in this metric. For single-people data the agent even converges close to the oracle, while the oracle is still slightly better for multi-people data due to its more difficult nature with occlusions. ACTOR yields lower reprojection errors than the heuristic baselines, with an exception at 2 cameras for multi-people data where *Max-Azim* is more accurate. Note however that ACTOR was not trained to produce accurate estimates at any fixed number of cameras, but rather to quickly triangulate all joints. Despite this, we outperform the baselines in the vast majority of cases.

D Additional Dataset Insights

Table D.3 shows the size and split of the Panoptic dataset [21] into train, validation and test sets. The data was created using scripts that downsampled from 30 FPS to 2 FPS to increase movement between frames.

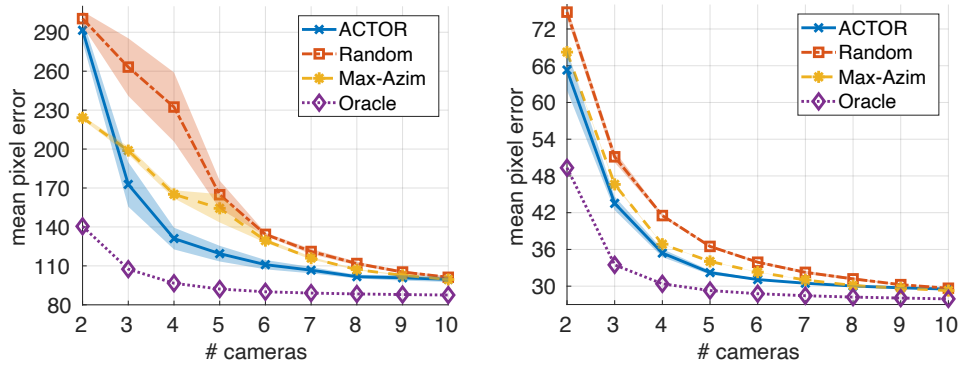


Figure D.17: Mean 2d reprojection errors per joint relative to OpenPose 2d estimates vs number of cameras on the test sets. Left: Multi-people data. Right: Single-people data. ACTOR reduces the 2d reprojection error faster than the heuristic baselines, particularly for multi-people data. Single-person scenes are easier to reconstruct, especially when using many cameras – also note that all models converge close to the error of the oracle in this case.

Table D.3: The number of images in our dataset categorized by scene type and subset type (training, validation, and testing). Note that *Mafia* and *Ultimatum* are complex multi-people scenes and that they account for more than half of the dataset.

	Train	Val	Test	All
Mafia	53,100	27,900	33,728	114,728
Ultimatum	27,960	4,340	55,825	88,125
Pose	51,079	29,672	59,288	140,039
All	132,139	61,912	148,841	342,892

Paper II

Erik Gärtner*, Aleksis Pirinen*, Cristian Sminchisescu
Deep Reinforcement Learning for Active Human Pose Estimation
Association for the Advancement of Artificial Intelligence (AAAI), New York, USA, 2020

Deep Reinforcement Learning for Active Human Pose Estimation

Erik Gärtner[1] **Aleksis Pirinen**[1*]
Cristian Sminchisescu[1,2,3]

[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]Institute of Mathematics of the Romanian Academy
[3]Google Research

Abstract

Most 3d human pose estimation methods assume that input – be it images of a scene collected from one or several viewpoints, or from a video – is given. Consequently, they focus on estimates leveraging prior knowledge and measurement by fusing information spatially and/or temporally, whenever available. In this paper we address the problem of an active observer with freedom to move and explore the scene spatially – in 'time-freeze' mode – and/or temporally, by selecting informative viewpoints that improve its estimation accuracy. Towards this end, we introduce *Pose-DRL*, a fully trainable deep reinforcement learning-based active pose estimation architecture which learns to select appropriate views, in space and time, to feed an underlying monocular pose estimator. We evaluate our model using single- and multi-target estimators with strong result in both settings. Our system further learns automatic stopping conditions in time and transition functions to the next temporal processing step in videos. In extensive experiments with the Panoptic multi-view setup, and for complex scenes containing multiple people, we show that our model learns to select viewpoints that yield significantly more accurate pose estimates compared to strong multi-view baselines.

*Denotes equal contribution, order determined by coin flip.

1 Introduction

Existing human pose estimation models, be them designed for 2d or 3d reconstruction, assume that viewpoint selection is outside the control of the estimation agent. This problem is usually solved by a human, either once and for all, or by moving around and tracking the elements of interest in the scene. Consequently, the work is split between *sufficiency* (e.g. instrumenting the space with as many cameras as possible in motion capture setups), *minimalism* (work with as little as possible, ideally a single view, as given), or *pragmatism* (use whatever is available, e.g. a stereo system and lidar in a self-driving car). While each of these scenarios and their underlying methodologies make practical or conceptual sense in their context of applicability, none covers the case of an active observer moving in the scene in order to reduce uncertainty, with emphasis on trading accuracy and computational complexity. There are good reasons for this, as experimenting with an active system faces the difficulty of linking perception and action in the real world, or may have to resort on simulation, which can however lack visual appearance and motion realism, especially for complex articulated and deformable structures such as people.

In this work we consider 3d human pose estimation from the perspective of an active observer, and operate with an idealization that allows us to distill the active vision concepts, develop new methodology, and test it on real image data. We work with a Panoptic massive camera grid [14], where we can both observe the scene in time-freeze, from a dense variety of viewpoints, and process the scene temporally, thus being able to emulate a moving observer. An active setup for 3d human pose estimation addresses the incomplete body pose observability in any monocular image due to depth ambiguities or occlusions (self-induced or produced by other people or objects). It also enables adaptation with respect to any potential limitations of the associated monocular pose estimation system, by sequentially selecting views that when combined yield accurate pose predictions.

In this context we introduce *Pose-DRL*, a deep reinforcement learning (RL) based active pose estimation architecture operating in a dense camera rig, which learns to select appropriate viewpoints to feed an underlying monocular pose predictor. Moreover, our model learns when to stop viewpoint exploration in time-freeze, or continue to the next temporal step when processing video. In evaluations using Panoptic we show that our system learns to select sets of views yielding more accurate pose estimates compared to strong multi-view baselines. The results not only show the advantage of intelligent viewpoint selection, but also that often 'less is more', as fusing too many possibly incorrect viewpoint estimates leads to inferior results.

As our model consists of a deep RL-based active vision module on top of a task module, it can be easily adjusted for other visual routines in the context of a multi-camera setup by simply replacing the task module and retraining the active vision component, or refin-

ing them jointly in case of access and compatibility. We show encouraging results using different pose estimators and task settings.

2 Related Work

Extracting 2d and 3d human representations from *given* images or video is a vast research area, recently fueled by progress in keypoint detection [20, 31], semantic body parts segmentation [25], 3d human body models [18], and 3d motion capture data [10, 30]. Deep learning plays a key role in most human pose and shape estimation pipelines [3, 16, 19, 21, 25, 27, 28, 36], sometimes in connection with non-linear refinement [3, 36]. Systems integrating detailed face, body and hand models have also been proposed [15]. Even so, the monocular 3d case is challenging due to depth ambiguities which motivated the use of additional ordering constraints during training [22].

In addition to recent literature for static pipelines, the community has recently seen an increased interest for active vision tasks, including RL-based visual navigation [2, 8, 33, 39]. In [2] a real-world dataset of sampled indoor locations along multiple viewing directions is introduced. An RL-agent is trained to navigate to views in which a given instance detector is accurate, similar in spirit to what we do, but in a different context and task.

A joint gripping and viewing policy is introduced in [5], also related to us in seeking policies that choose occlusion-free views. The authors of [6] introduce an active view selection system and jointly learn a geometry-aware model for constructing a 3d feature tensor, which is fused together from views predicted by a policy network. In contrast to us, their policy predicts one of 8 adjacent discrete camera locations, they do not consider moving objects, their model does not automatically stop view selection, and they do not use real data. In [11, 34], active view selection is considered for panoramic completion and panorama projection, respectively. Differently from us, their view selection policies operate on discretized spheres and do not learn automatic stopping conditions. An approach for active multi-view object recognition is proposed in [13], where pairs of images in a view trajectory are sequentially fed to a CNN for recognition and for next best view prediction. Optimization is done over discretized movements and pre-set trajectory lengths, in contrast to us.

Most related to us is [24], who also consider active view selection in the context of human pose estimation. However, they work with 2d joint detectors and learn to actively triangulate those into 3d pose reconstructions. Thus we face different challenges – while [24] only require each joint to be visible in two views for triangulation, our model has to consider which views yield accurate fused estimates. Furthermore, their model does not learn a stopping action that trades accuracy for speed, and they do not study both the single-target and multi-target cases, as we do in this paper.

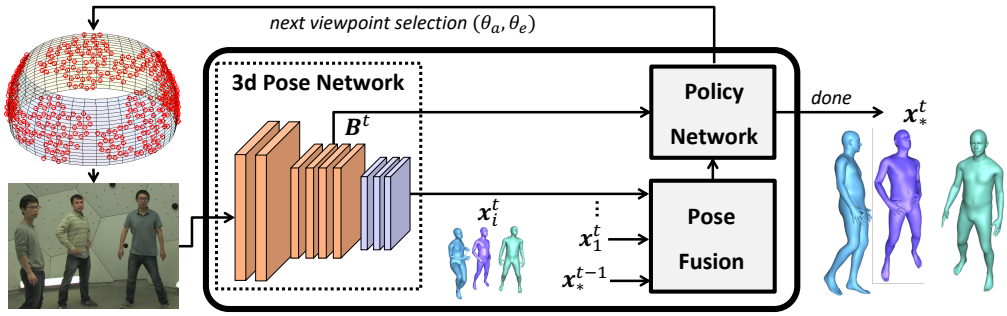


Figure 2.1: Overview of our Pose-DRL agent for active human pose estimation. The agent initially observes the scene from a randomly given camera on the rig. In each visited viewpoint, the associated image is processed by a 3d pose estimation network, producing the base state $\boldsymbol{B}^t$ of the agent and pose estimate(s) $\boldsymbol{x}_i^t$. The pose estimate is fused together with estimates from previous viewpoints $\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{i-1}^t$ and the previous temporal step $\boldsymbol{x}_*^{t-1}$. Both the current and fused estimate are fed as additional features to the agent. At each step the policy network outputs the next viewpoint, until it decides it is done and continues to next active-view at time $t+1$. The viewpoint selection action predicts spherical angles relative to the agent's current location on the camera rig, and the closest camera associated with the predicted angles is visited next. When the agent is done it outputs $\boldsymbol{x}_*^t$, the per-joint fusion of the individual pose estimates seen during the current active-view and the fused pose estimate from the previous active-view, cf. (2.2). Pose-DRL can be used either to reconstruct a target person, or to reconstruct all people in the scene. The underlying pose estimator is exchangeable – we show strong results using two different ones in §5.1.

Aside from active vision applications in real or simulated environments, reinforcement learning has also been successfully applied to other vision tasks, e.g. object detection [4, 23], object tracking [35, 38] and visual question answering [7].

3 Active Human Pose Estimation

In this section we describe our active human pose estimation framework, arguing it is a good proxy for a set of problems where an agent has to actively explore to understand the scene and integrate task relevant information. For example, a single view may only contain parts of the human body (or be absent of the person altogether) and the agent needs to find a better view to capture the person's pose. Pose estimators are often trained on a limited set of viewing angles and yield lower performance for others. Our setup forces the agent to also take any estimation limitations into account when selecting multiple views. In particular, we show in §5.1 that learning to find good views and fusing them is more important than relying on a large number of random ones, or the full available set, as standard – see also Fig. 2.4.

Concepts in the following sections will, for simplicity, be described assuming the model is estimating the pose of a single *target person* (though scenes may contain multiple people occluding the target). The setting in which *all* people are reconstructed simultaneously is described in §4.4.

3.1 Active Pose Estimation Setup

We idealize our active pose estimation setup using CMU's Panoptic installation [14] as it captures real video data of scenes with multiple people and cameras densely covering the viewing sphere. This allows us to simulate an active agent observing the scene from multiple views, without the complexity of actually moving a camera. It also enables controllable and reproducible experiments. The videos are captured in a large spherical dome fitted with synchronized HD cameras.[1] Inside the dome several human actors perform a range of movements, with 2d and 3d joint annotations available. The dataset is divided into a number of *scenes*, video recordings from all synchronized cameras capturing different actors and types of movements, ranging from simple pose demonstrations to intricate social games.

Terminology. We call a *time-freeze* $\{v_1^t, \ldots, v_N^t\}$ the collection of views from all N time-synchronized cameras at time t, with v_i^t the image (referred to as *view* or *viewpoint*) taken by camera i at time t. A subset of a time-freeze is an *active-view* $\mathrm{V}t = \{v_1^t, \ldots, v_k^t\}$ containing k selected views from the time-freeze. A temporally contiguous sequence of active-views is referred to as an *active-sequence*, $\mathcal{S}^{1:T} = \{\mathrm{V}1, \mathrm{V}2, \ldots, \mathrm{V}T\}$. We will often omit the time superfix t unless the context is unclear; most concepts will be explained at the level of time-freezes. The image corresponding to a view v_i can be fed to a pose predictor to produce a pose estimate $\boldsymbol{x}_i \in \mathbb{R}^{45}$ ($15\times$ 3d joints).

Task definition. We define the task of *active pose estimation* at each time step as selecting views from a time-freeze to generate an active-view. The objective is to produce an accurate fused estimate $\boldsymbol{x}_\star$ from pose predictions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ associated with the active-view (k may vary between active-views). The deep pose estimation network is computationally demanding and therefore working with non-maximal sets of views decreases processing time. Moreover, it improves estimates by ignoring obstructed views, or those a given pose predictor cannot accurately handle. The goal of the full active pose estimation task is to produce accurate fused pose estimates over the full sequence, i.e., to produce an active-sequence with accurate corresponding fused pose estimates.

3.2 Detection and Matching of Multiple People

To solve active human pose estimation the model must address the problems of detecting, tracking, and distinguishing people in a scene. It must also be robust to variations in appearance since people are observed over time and from different views. We use Faster R-CNN [26] for detecting people. At the start of an active-sequence the agent is given ap-

[1] There are about 30 cameras per scene. The HD cameras provide better image quality than VGA and are sufficiently dense, yet spread apart far enough to make each viewpoint unique.

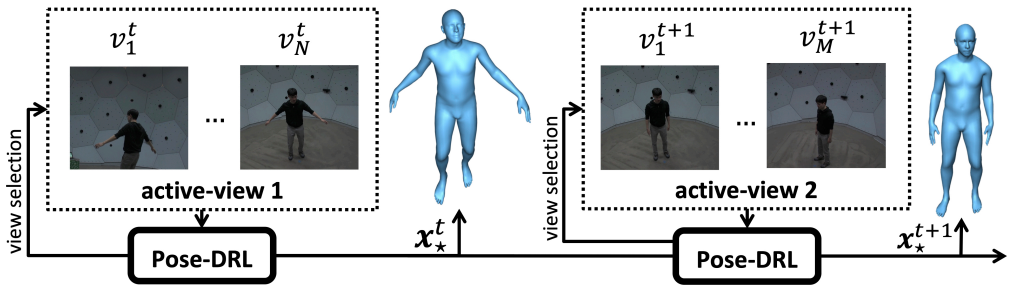


Figure 2.2: Illustration of how Pose-DRL operates on an active-sequence, here shown for a single-person scenario. Fused pose estimates are fed to subsequent active-views within the active-sequence, both as additional state representation for action selection, and for fusing poses temporally.

pearance models, consisting of instance-sensitive features for each person. For each visited view, the agent computes instance features for all detected persons, comparing them with the given appearance models to identify the different people.

Obtaining appearance models. A generic instance classifier, implemented as a VGG-19 based [29] siamese network, is trained for 40k iterations on the training set with a contrastive loss to distinguish between different persons. Each mini-batch (of size 16) consists of randomly sampled pairs of ground-truth crops of people in the training set. We ensure that the training is balanced by sampling pairs of person crops such that the probability of the two crops containing the same person is the same as that of containing two different persons. The people crops are sampled uniformly across scenes, spatially and temporally, yielding a robust instance classifier.

Once the instance classifier has been trained, we fine-tune it for 2k iterations for each scene and then use it to construct appearance models at the beginning of an active-sequence. For each person, we sample L instance features from time-freezes from the same scene, but outside of the time span of the current active-sequence to limit overfitting. Denote by $\boldsymbol{u}_i^l$, the i:th instance feature for the l:th person, with $i = 1, \ldots, L$. Then we set as appearance model:

$$\boldsymbol{m}^l = \text{median}(\boldsymbol{u}_1^l, \ldots, \boldsymbol{u}_L^l) \tag{2.1}$$

We set $L = 10$ to obtain a diverse set of instance features for each person, yielding a robust appearance model.

Stable matching of detections. In each visited viewpoint during an active-sequence, the agent computes instance features for all detected persons, comparing them with the given appearance models to identify the different people. To ensure a stable matching, we use the Hungarian algorithm. Specifically, the cost $c^{j,l}$ of matching the j:th detection with instance feature $\boldsymbol{u}^j$ in the current viewpoint to the appearance model $\boldsymbol{m}^l$ of the l:th person is

$c^{j,l} = \|\boldsymbol{u}^j - \boldsymbol{m}^l\|_2^2$. Since the target person may not be visible in all viewpoints throughout the active-sequence, we specify a *cost threshold*, $\mathcal{C} = 0.5$, such that if the assignment cost $c^{j,l}$ of the target is above it (i.e. $c^{j,l} > \mathcal{C}$), we consider the person to not be visible in the view. In that case the associated pose is not fused into the final estimate.

4 Deep Reinforcement Learning Model

We now introduce our Pose-DRL agent for solving the active human pose estimation task and first explain the agent's state representation and actions, then define the reward signal for training an agent which selects views that yield an accurate fused pose estimate while keeping down processing time.

4.1 Overview of the Pose-DRL Agent

The agent is initiated at a randomly selected view v_1^1 in the first active-view V1 of an active-sequence $\mathcal{S}^{1:T}$. Within the current active-view Vt, the agent issues *viewpoint selection* actions to progressively select a sequence of views $v_2^t, \ldots, v_k^t$, the number of which may vary between active-views. At each view v_i^t the underlying pose estimator predicts the pose $\boldsymbol{x}_i^t$. As seen in Fig. 2.1 the cameras are approximately located on a partial sphere, so a viewpoint can be specified by the azimuth and elevation angles (referred to as *spherical angles*). Thus for viewpoint selection the Pose-DRL agent predicts spherical angles relative to its current location and selects the camera closest to those angles.

Once the agent is done exploring viewpoints associated to a particular time-freeze it issues the *continue* action and switches to the next active-view V$t+1$, at which time the collection of individual pose estimates $\boldsymbol{x}_i^t$ from the different viewpoints are fused together with the estimate $\boldsymbol{x}_\star^{t-1}$ from the previous active-view V$t-1$:

$$\boldsymbol{x}_\star^t = f(\boldsymbol{x}_\star^{t-1}, \boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_k^t) \tag{2.2}$$

Including the previous time step estimate $\boldsymbol{x}_\star^{t-1}$ in the pose fusion as in (2.2) often improves results (see §5.2). After returning the fused pose estimate $\boldsymbol{x}_\star^t$ for the current active-view, the agent continues to the next active-view V$t+1$. The initial view v_1^{t+1} for V$t+1$ is set to the final view v_k^t of $\mathcal{V}^t$, i.e., $v_1^{t+1} = v_k^t$. The process repeats until the end of the active-sequence. Fig. 2.1 and 2.2 show model overviews for active-views and active-sequences, respectively.

Table 2.1: Reconstruction error (mm/joint) for Pose-DRL and baselines on active-sequences on the selected Panoptic test splits. Results are shown both for the setting where the agent decides the number of views (auto) *and* when using a fixed number of views. In the latter case, the number of views is set to the closest integer corresponding to the average in auto mode, rounded up. The baselines are also evaluated at this preset number of views. The average number of views are shown in parentheses. Pose-DRL models which automatically select the number of views outperform the heuristic baselines and fixed Pose-DRL models on all data splits, despite fusing estimates from fewer views on average. Left: Single-target mode (S), using DMHS as pose estimator. The agent significantly outperforms the baselines (e.g. 35 mm/joint improvement over *Max-Azim* on multi-people data *Maf + Ult*). Right: Multi-target mode (M), using MubyNet as pose estimator. MubyNet is a more recent and accurate estimator, so the average errors are typically lower than the DMHS-counterparts. Automatic termination is useful in the multi-target setting as well, although it does not provide as drastic gains as in the single-target setup.

Model	# Views	Maf	Ult	Pose	Maf + Ult	All
Pose-DRL-S	auto	130.3 (4.6)	135.4 (3.4)	135.3 (3.7)	134.2 (3.8)	135.0 (3.7)
	fixed	144.7 (5.0)	157.5 (4.0)	135.1 (4.0)	155.5 (4.0)	140.4 (4.0)
Rand-S	fixed	160.2 (5.0)	178.3 (4.0)	145.7 (4.0)	175.6 (4.0)	157.1 (4.0)
Max-Azim-S	fixed	156.3 (5.0)	171.4 (4.0)	139.9 (4.0)	169.4 (4.0)	150.3 (4.0)
Oracle-S	fixed	103.4 (5.0)	108.9 (4.0)	106.5 (4.0)	108.5 (4.0)	105.4 (4.0)

Model	# Views	Maf	Ult	Pose	Maf + Ult	All
Pose-DRL-M	auto	114.8 (7.5)	116.4 (6.6)	104.6 (2.1)	115.9 (6.8)	110.7 (4.5)
	fixed	114.8 (8.0)	118.0 (7.0)	106.7 (3.0)	117.6 (7.0)	112.8 (5.0)
Rand-M	fixed	128.8 (8.0)	134.9 (7.0)	115.9 (3.0)	131.4 (7.0)	126.0 (5.0)
Max-Azim-M	fixed	123.5 (8.0)	131.2 (7.0)	116.3 (3.0)	131.6 (7.0)	126.4 (5.0)
Oracle-M	fixed	98.6 (8.0)	102.4 (7.0)	90.2 (3.0)	101.6 (7.0)	92.6 (5.0)

4.2 State-Action Representation

To simplify notation, we here describe how the agent operates in a given time-freeze, and in this context will use t to index actions within the active-view, as opposed to temporal structures. The state at step t is the tuple $S^t = (\boldsymbol{B}^t, \boldsymbol{X}^t, \boldsymbol{C}^t, \boldsymbol{u}^t)$. Here $\boldsymbol{B}^t \in \mathbb{R}^{H \times W \times C}$ is a deep feature map associated with the underlying 3d pose estimation architecture. $\boldsymbol{X}^t = \{\boldsymbol{x}_t, \tilde{\boldsymbol{x}}, \boldsymbol{x}_\star^{hist}\}$ where $\boldsymbol{x}_t$ is the current individual pose estimate, $\tilde{\boldsymbol{x}} = f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ is the current partially fused pose estimate, and $\boldsymbol{x}_\star^{hist}$ is a history of fused predictions from 4 previous active-views. The matrix $\boldsymbol{C}^t \in \mathbb{N}^{w \times h \times 2}$ consists of an *angle canvas*, a discretized encoding[1] of the previously visited regions on the camera rig, as well as a similar encoding of the camera distribution over the rig. Finally, $\boldsymbol{u}^t \in \mathbb{R}^2$ is an auxiliary vector holding the number of actions taken and the number of people detected.

For action selection we use a deep stochastic policy $\pi_{\boldsymbol{\theta}}(A^t|S^t)$ parametrized by $\boldsymbol{w}$ which predicts the action $A^t = \{\theta_a^t, \theta_e^t, c^t\}$. Here (θ_a^t, θ_e^t) is the azimuth-elevation angle pair, jointly referred to as *viewpoint selection*, and c^t is a Bernoulli variable indicating whether to continue to the next active-view (occurring if $c^t = 1$), referred to as the *continue* action. To produce action probabilities the base feature map $\boldsymbol{B}^t$ is fed through two convolutional blocks which are shared between the viewpoint selection and continue actions. The output of the second convolutional block is then concatenated with $\boldsymbol{X}^t$, $\boldsymbol{C}^t$ and $\boldsymbol{u}^t$ and fed to viewpoint selection- and continue-branches with individual parameters. Both action branches consist of 3 fully-connected layers with $\tanh$ activations. The probability of issu-

[1]The camera sphere is discretized into w bins in the azimuth direction and h bins in the elevation direction, appropriately wrapped to account for periodicity. We set $w = 9$ and $h = 5$.

ing the continue action is computed using a sigmoid layer:

$$\pi_{\boldsymbol{\theta}}(c^t = 1|S^t) = \sigma\left[\boldsymbol{w}_c^\top \boldsymbol{z}_c^t + b_c\right] \tag{2.3}$$

where $\boldsymbol{w}_c$ and b_c are trainable weights and bias, and $\boldsymbol{z}_c^t$ is the output from the penultimate fully-connected layer of the *continue* action branch.

Due to the periodic nature of the viewpoint prediction task we rely on von Mises distributions for sampling the spherical angles. We use individual distributions for the azimuth and elevation angles. The probability density function for the azimuth is given by:

$$\pi_{\boldsymbol{\theta}}\left(\theta_a^t|S^t\right) = \frac{1}{2\pi I_0(m_a)} \exp\{m_a \cos(\theta_a^t - \tilde{\theta}_a(\boldsymbol{w}_a^\top \boldsymbol{z}_a^t + b_a))\} \tag{2.4}$$

where I_0 is the zeroth-order Bessel function, normalizing (2.4) to a proper probability distribution over the unit circle $[-\pi, \pi]$. Here $\tilde{\theta}^a$ is the mean of the distribution (parametrized by the neural network), m_a is the precision parameter,[1] $\boldsymbol{w}_a$ and b_a are trainable weights and bias, respectively, and $\boldsymbol{z}_a^t$ comes from the penultimate fully-connected layer of the viewpoint selection action branch. The support for the azimuth angle should be on a full circle $[-\pi, \pi]$, and hence we set

$$\tilde{\theta}_a(\boldsymbol{w}_a^\top \boldsymbol{z}_a^t + b_a) = \pi \tanh(\boldsymbol{w}_a^\top \boldsymbol{z}_a^t + b_a) \tag{2.5}$$

The probability density function for the elevation angle has the same form (2.4) as that for the azimuth. However, as seen in Fig. 2.1, the range of elevation angles is more limited than for the azimuth angles. We denote this range $[-\kappa, \kappa]$ and the mean elevation angle thus becomes[2]

$$\tilde{\theta}_e(\boldsymbol{w}_e^\top \boldsymbol{z}_e^t + b_e) = \kappa \tanh(\boldsymbol{w}_e^\top \boldsymbol{z}_e^t + b_e) \tag{2.6}$$

In practice, when sampling elevation angles from the von Mises, we reject samples outside the range $[-\kappa, \kappa]$.

4.3 Reward Signal for Policy Gradient Objective

The agent should strike a balance between choosing sufficiently many cameras so the resulting 3d pose estimate is as accurate as possible, while ensuring that not too many cameras are visited, to save processing time. As described earlier, the two types of actions are *viewpoint selection* and *continue*. We will next cover the reward functions for them.

[1] We treat the precision parameters as constants but increase them over training to focus the policy on high-reward viewpoints.

[2] With notation analogous to that of the azimuth angle, cf. (2.5).

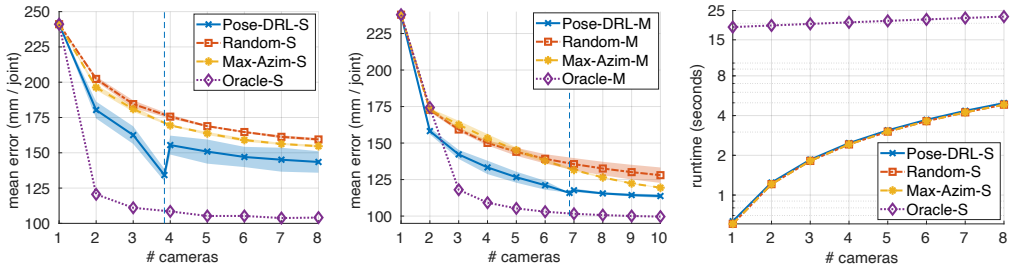


Figure 2.3: How the number of views affects pose estimation error and runtimes of Pose-DRL and baselines on multi-people data (union of *Mafia* and *Ultimatum* test sets). We show mean and 95% confidence intervals over 5 seeds. Left: Reconstructing a single target person. Estimation error reduces with added viewpoints, and the agent consistently outperforms the non-oracle baselines. The automatic *continue* action (dashed line at 3.8 views on average) yields significantly lower reconstruction errors than any fixed viewpoint schemes. Hence the auto-model clearly provides the best speed-accuracy trade-off. Middle: Simultaneously reconstructing all persons. The agent outperforms the heuristic baselines in this setting too. Adaptively determining when to continue to the next active-view (6.8 views on average) yields better results than fusing from 7 cameras all the time. The gain is not as pronounced as in the single-target case, since more viewpoints typically leads to increased estimation accuracy for some of the persons. Right: Runtime of the Pose-DRL agent and baselines vs. number of views (log scale). The oracle always needs to evaluate the deep pose estimation system and detector for all cameras due to its need to sort from best to worst, independently of the number of viewpoints, which explains its high runtime. Our agent is as fast as the heuristic baselines.

Viewpoint selection reward. At the end of an active-view we give a reward which is inversely proportional to the ratio between the final and initial reconstruction errors within the active-view. We also give a penalty $\epsilon = 2.5$ each time the agent goes to an already visited viewpoint. Thus the viewpoint selection reward is:

$$r_v^t = \begin{cases} 0, & \text{if } c^t = 0 \text{ and view not visited} \\ -\epsilon, & \text{if } c^t = 0 \text{ and view visited before} \\ 1 - \frac{\varepsilon^k}{\varepsilon^1}, & \text{if } c^t = 1 \end{cases} \tag{2.7}$$

where k is the number of views visited prior to the agent issuing the *continue* action ($c^t = 1$), ε^1 is the reconstruction error associated with the initial viewpoint, and ε^k denotes the final reconstruction error, i.e. $\varepsilon^k = \|\boldsymbol{x}_\star - \boldsymbol{x}_{\text{gt}}\|_2^2$. Here $\boldsymbol{x}_\star$ is the final fused pose estimate for the active-view, cf. (2.2), and $\boldsymbol{x}_{\text{gt}}$ is the ground-truth 3d pose for the time-freeze.

Continue action reward. The *continue* action has two purposes: *(i)* ensure that not too many viewpoints are visited to reduce computation time, and *(ii)* stop before suboptimal viewpoints are explored, which could happen if the agent is forced to visit a preset number of viewpoints. Therefore, the *continue* action reward is:

$$r_c^t = \begin{cases} 1 - \frac{\min_{j \in \{t+1,\ldots,k\}} \varepsilon^j}{\varepsilon^t} - \tau, & \text{if } c^t = 0 \\ 1 - \frac{\varepsilon^k}{\varepsilon^1}, & \text{if } c^t = 1 \end{cases} \tag{2.8}$$

At each step that the agent decides *not* to continue to the next active-view ($c^t = 0$), the agent is rewarded relative to the ratio between the error at the best future stopping point within the active-view (with lowest reconstruction error) and the error at the current step. If in the future the agent selects viewpoints that yield lower reconstruction error the agent is rewarded, and vice verse if the best future error is higher. In addition, the agent gets a penalty τ at each step, which acts as an *improvement threshold*, causing the reward to become negative unless the ratio is above the specified threshold τ. This encourages the agent not to visit many viewpoints in the current active-view unless the improvement is above the given threshold. On the validation set we found $\tau = 0.07$ to provide a good balance.

Policy gradient objective. We train the Pose-DRL network in a policy gradient framework, maximizing expected cumulative reward on the training set with objective

$$J(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{s}\sim\pi_{\boldsymbol{\theta}}}\left[\sum_{t=1}^{|\boldsymbol{s}|} r^t\right] \tag{2.9}$$

where $\boldsymbol{s}$ denotes state-action trajectories, and the reward signal $r^t = r_v^t + r_c^t$, cf. (2.7) - (2.8). We approximate the gradient of the objective (2.9) using REINFORCE [32].

4.4 Active Pose Estimation of Multiple People

So far we have explained the Pose-DRL system that estimates the pose of a target person, assuming it is equipped with a detection-based single person estimator. This system can in principle estimate multiple people by generating active-sequences for each person individually. However, to find a *single* active-sequence that reconstructs *all* persons, one can equip Pose-DRL with an image-level multi-people estimator instead. In that case, the state representation is modified to use the image level feature blob from the multi-people estimator ($\boldsymbol{B}_t$ in Fig. 2.1). The reward signal used when learning to reconstruct all people is identical to (2.7) - (2.8), except that the rewards are averaged over the individual pose estimates. Thus Pose-DRL is very adaptable in that the underlying pose estimator can easily be changed.

5 Experiments

Dataset. We use diverse scenes for demonstrating and comparing our active pose estimation system, considering complex scenes with multiple people (*Mafia*, *Ultimatum*) as well as single person ones (*Pose*). The motions range from basic poses to various social games. Panoptic provides data as 30 FPS-videos which we sample to 2 FPS, making the data more manageable in size. It also increases the change in pose between consecutive frames.

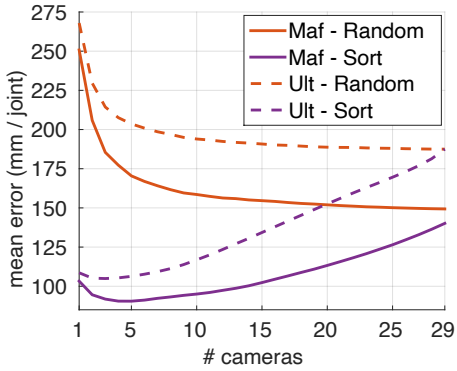


Figure 2.4: Per-joint pose reconstruction error for the monocular human pose estimation architecture DMHS vs. number of viewpoints, both when randomly choosing viewpoints, and when using an sorting strategy which selects viewpoints in ascending order of individual reconstruction error (note that this requires ground-truth). Results shown for multi-people data (*Mafia*, *Ultimatum*) on the CMU Panoptic dataset. For a good viewpoint selection policy such as *Sort*, estimation accuracy only improves when adding a few extra cameras, but then begins to deteriorate, indicating the need to adaptively terminate viewpoint selection early enough.

The data we use consists of the same 20 scenes as in [24]. The scenes are randomly split into training, validation and test sets with 10, 4 and 6 scenes, respectively. Since we split the data over the scenes, the agent needs to learn a general look-around-policy which adapts to various circumstances (scenarios and people differ between scenes). All model selection is performed exclusively on the training and validation sets; final evaluations are performed on the test set. The data consists of 343k images, of which 140k are single-person and 203k are multi-people scenes.

Implementation details. We attach Pose-DRL on top of the DMHS monocular pose estimation system [25]. In the multi-people setting described in §4.4 we instead use MubyNet [37]. Both estimators were trained on Human3.6M [10]. To avoid overfitting we do not to fine-tune these on Panoptic, and instead emphasize how Pose-DRL can select good views with respect to the underlying estimation system (but joint training is possible). We use an *identical set of hyperparameters* when using DMHS and MubyNet, except the improvement threshold τ, which is -0.07 for DMHS and -0.04 for MubyNet, which shows that Pose-DRL is robust with respect to the pose estimator used. We use median averaging for fusing poses, cf. (2.2).

Training. We use 5 active-sequences, each consisting of length 10, to approximate the policy gradient, and update the policy parameters using Adam [17]. As standard, to reduce variance we normalize cumulative rewards for each episode to zero mean and unit variance over the batch. The maximum trajectory length is set to 8 views including the initial one (10 in the multi-target mode, as it may require more views to reconstruct all people). The *view-*

point selection and *continue* actions are trained jointly for 80k episodes. The learning rate is initially set to `5e-7` and is halved at 720k and 1440k agent steps. We linearly increase the precision parameters m_a and m_e of the von Mises distributions from $(1, 10)$ to $(25, 50)$ in training, making the viewpoint selection increasingly focused on high-rewarding regions as training proceeds.

Baselines. To evaluate our active human pose estimation system we compare it to several baselines, similar to [24]. For fair comparisons, the baselines use the same pose estimator, detector and matching approach. All methods obtain the same initial random view as the agent at the start of the active-sequence. We design the following baselines: i) *Random:* Selects k different random views; ii) *Max-Azim:* Selects k different views equidistantly with respect to the azimuth angle. At each azimuth angle it selects a random elevation angle; iii) *Oracle:* Selects as next viewpoint the one that minimizes the fused 3d pose reconstruction when combined with pose estimates from all viewpoints observed so far (averaged over all people in the multi-target setting). This baseline cheats by extensively using ground-truth information, and thus it shown as a lower bound with respect to reconstruction error. In addition to cheating during viewpoint selection, the oracle is also impractically slow since it requires computing pose estimates for *all* available viewpoints and exhaustively computing errors for all cameras in each step.

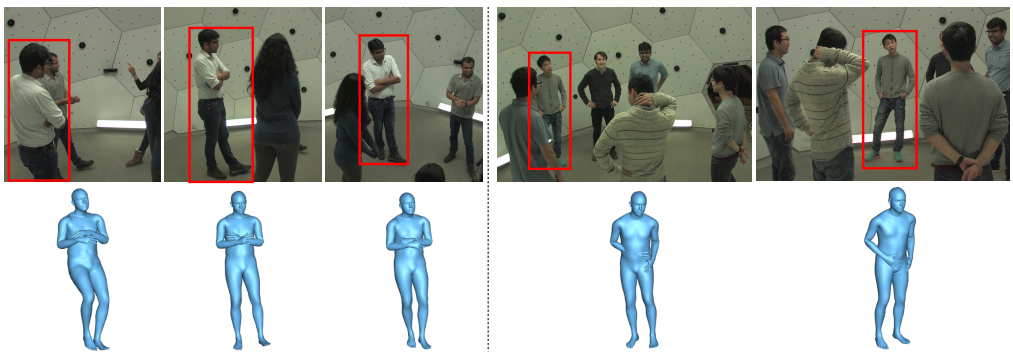


Figure 2.5: Visualizations of Pose-DRL reconstructing a given target person (red bounding box). Left: A *Mafia* test scene. The target is viewed from behind and is partially visible in the first view, producing the poor first estimate. As the agent moves to the next view, the person becomes more clearly visible, significantly improving the estimate. The last view from the front further increases accuracy. The agent decides to terminate after three views with error decreasing from 200.1 to 120.9 mm/joint. Right: An *Ultimatum* test scene where the agent only requires two viewpoints prior to automatically continuing to the next active-view. The target person is only partially visible in the initial viewpoint, and the right arm that is not visible results in a non-plausible configuration in the associated estimate. As the agent moves to the next viewpoint the person becomes fully visible, and the final fused estimate is both physically plausible and accurate. The reconstruction error reduces from 160 to 104 mm/joint.

5.1 Quantitative Results

We report results both for the Pose-DRL agent that tracks and reconstructs a single target person (possibly in crowded scenes) and for the Pose-DRL model which actively estimates poses for all persons in the scene, cf. §4.4. Pose-DRL is trained over 5 different random initializations of the policy network, and we report average results. In each case, training the model 80k steps gave best results on the validation set, so we use that. Also, for the heuristic baselines we report average results over 5 seeds (the oracle is deterministic).

Our agent is compared to the baselines on the Panoptic test set on active-sequences consisting of 10 active-views. Table 2.1 presents reconstruction errors. Fig. 2.3 shows how the the number of selected views affects accuracy and runtimes. For visualizations[1] of Pose-DRL, see Fig. 2.5 - 2.7.

Single-target estimation. It is clear from Table 2.1 (left) and Fig. 2.3 (left) that Pose-DRL outperforms the heuristic baselines, which is particularly pronounced for multi-people data. In such scenes, the view selection process is more delicate, as it requires avoiding cameras where the target is occluded. We note that the automatically stopping agent yields by far the most accurate estimates, which shows that it is capable of continuing to the next active-view when it is likely that the current one does not provide any more good views. Thus it is often better to fuse a few accurate estimates than including a larger set of poorer ones.

Multi-target estimation. From Table 2.1 (right) and Fig. 2.3 (middle) we see that the agent outperforms the heuristic baselines as in the case with a single target. Automatic view selection termination does not yield as big improvements in accuracy as in the single-target case. In the single-target setting the agent stops early to avoid occluded and bad views, but when reconstructing all people there is more reason to keep selecting additional views to find some views which provide reasonable estimates for each person. This also explains the decreased gaps between the various methods – there may be many sets of cameras which together provide a fairly similar result when averaged over all people in the scene. A future improvement could include selective fusing a subset of estimates in each view. Running in auto mode still yields more accurate estimates than fixed schemes which use a larger number of views.

Runtimes. The runtimes[2] for Pose-DRL and baselines are shown in Fig. 2.3. DMHS and Faster R-CNN require 0.50 and 0.11 seconds per viewpoint, respectively, which constitutes the bulk of the processing time. The policy network has an overhead of about 0.01 seconds per action, negligible in relation to the pose estimation system.

[1] We use SMPL [18] for the 3d shape models.

[2] Shown for DMHS-based systems. Using MubyNet (which requires 1.01 seconds per image) gives runtime curves which look qualitatively similar.

Table 2.2: Ablations on the test sets, showing the effect of removing certain components of the DMHS-based Pose-DRL system. Results (errors, mm/joint) are for models that select a fixed number of views (shown in parentheses), where the number of views are the same as in Table 2.1. Providing more information than the base feature map $\boldsymbol{B}^t$ is crucial for crowded scenes with multiple people (*Maf*, *Ult*), as is including previous pose estimates in the current pose fusion.

Model	Settings	Maf	Ult	Pose
Pose-DRL	full model	144.7 (5)	157.5 (4)	135.1 (4)
	$\boldsymbol{B}^t$ only	153.5 (5)	166.9 (4)	134.4 (4)
	reset	152.5 (5)	160.8 (4)	133.4 (4)

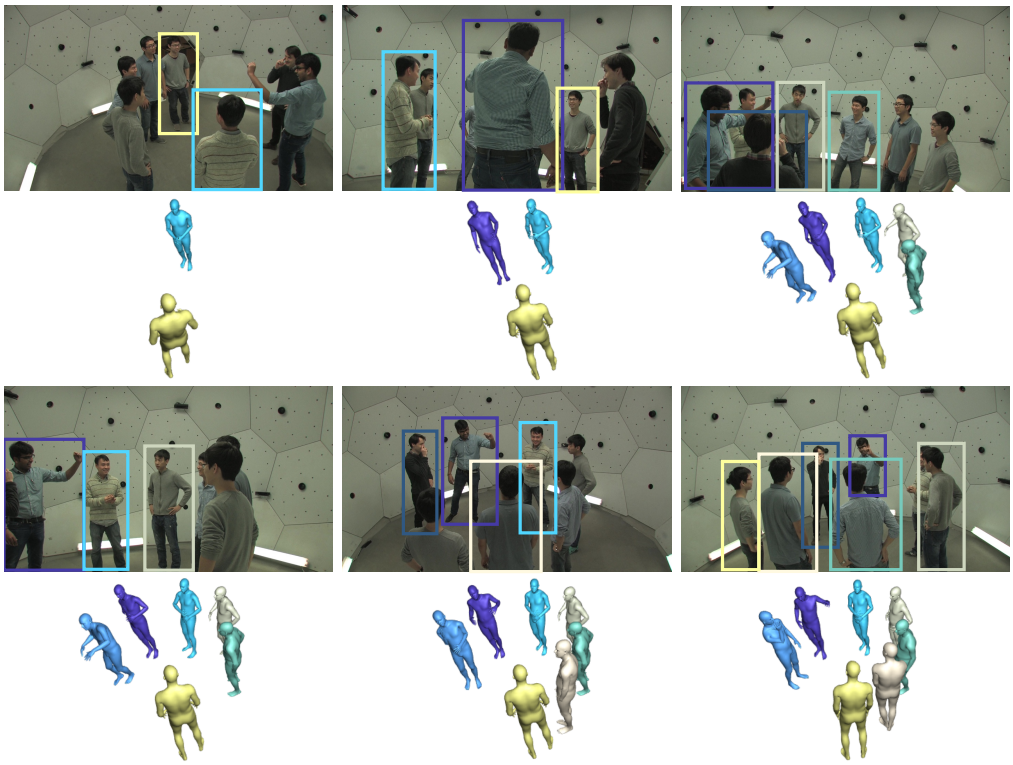


Figure 2.6: Visualization of how Pose-DRL performs multi-target pose estimation for an *Ultimatum* test scene. In this example the agent sees six viewpoints prior to automatically continuing to the next active-view. The mean error decreases from 358.9 to 114.6 mm/joint. Only two people are detected in the initial viewpoint, but the number of people detected increases as the agent inspects more views. Also, the estimates of already detected people improve as they get fused from multiple viewpoints.

5.2 Ablation Studies

In this section we compare the full agent to versions lacking parts of the model: i) providing only the base feature map $\boldsymbol{B}^t$, and ii) not propagating the fused reconstruction $\boldsymbol{x}_\star^t$ to the next active-view (*reset*), cf. (2.2). The results are given in Table 2.2, and show that the full

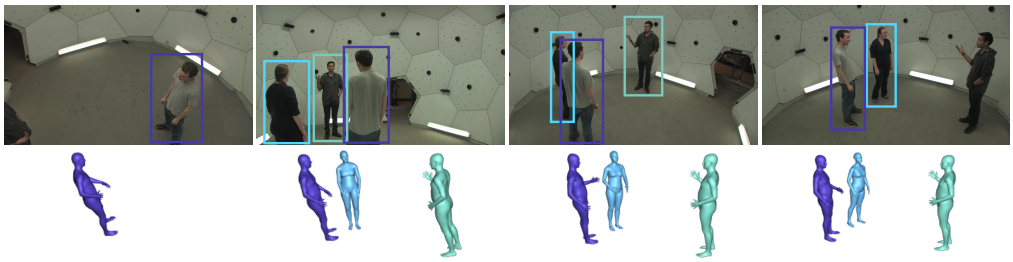


Figure 2.7: Visualization of how Pose-DRL performs multi-target pose estimation an *Ultimatum* validation scene. The agent chooses four viewpoints prior to automatically continuing to the next active-view. The mean error decreases from 334.8 to 100.9 mm/joint. Only one of the persons is visible in the initial viewpoint, and from a poor angle. This produces the first, incorrectly tilted pose estimate, but the estimate improves as the agent inspects more viewpoints. The two remaining people are successfully reconstructed in subsequent viewpoints.

model outperforms the stripped-down versions for multi-people data (*Mafia*, *Ultimatum*), while simpler single-people data in *Pose* is not sensitive to removing some parts of the model. There is significantly more room for intelligent decision making for complex multi-people data, where the model has to avoid occlusions, and thus it requires a stronger state description and fusion approach. In contrast, selecting views in single-people scenes is less fragile to the particular camera choices as there is no risk of choosing views where the target is occluded.

6 Conclusions

In this paper we have presented *Pose-DRL*, a fully trainable deep reinforcement-learning based active vision model for human pose estimation. The agent has the freedom to move and explore the scene spatially and temporally, by selecting informative views that improve its accuracy. The model learns automatic stopping conditions for each moment in time, and transition functions to the next temporal processing step in video. We showed in extensive experiments – designed around the dense Panoptic multi-camera setup, and for complex scenes with multiple people – that Pose-DRL produces accurate estimates, and that our agent is robust with respect to the underlying pose estimator used. Moreover, the results show that our model learns to select an adaptively selected number of informative views which result in considerably more accurate pose estimates compared to strong multi-view baselines.

Practical developments of our methodology would include e.g. real-time intelligent processing of multi-camera video feeds or controlling a drone observer. In the latter case the model would further benefit from being extended to account for physical constraints, e.g. a single camera and limited speed. Our paper is a key step since it presents fundamental methodology required for future applied research.

Acknowledgments: This work was supported by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535 and PCCF-2016-0180, the EU Horizon 2020 Grant DE-ENIGMA, Swedish Foundation for Strategic Research (SSF) Smart Systems Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Finally, we would like to thank Alin Popa, Andrei Zanfir, Mihai Zanfir and Elisabeta Oneata for helpful discussions and support.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 110

[2] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Košecká, and Alexander C Berg. A dataset for developing and benchmarking active vision. In *ICRA*, pages 1378–1385. IEEE, 2017. 91

[3] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it SMPL: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, 2016. 91

[4] J.C. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 92

[5] Ricson Cheng, Arpit Agarwal, and Katerina Fragkiadaki. Reinforcement learning of active vision for manipulating objects under occlusions. In *CoRL*, pages 422–431, 2018. 91

[6] Ricson Cheng, Ziyan Wang, and Katerina Fragkiadaki. Geometry-aware recurrent neural networks for active visual recognition. In *NeurIPS*, pages 5081–5091, 2018. 91

[7] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *CVPR*, pages 2951–2960, 2017. 92

[8] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, volume 5, page 6, 2018. 91

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 110

[10] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (7):1325–1339, jul 2014. 91, 100

[11] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, 2018. 91

[12] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 110

[13] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, pages 3813–3822, 2016. 91

[14] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, 2015. 90, 93, 112

[15] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *CVPR*, 2018. 91

[16] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 91

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 100

[18] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *SIGGRAPH*, 34 (6):248:1–16, 2015. 91, 102, 112

[19] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. 91

[20] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *CVPR*, 2017. 91

[21] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *CVPR*, 2017. 91

[22] Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Ordinal depth supervision for 3D human pose estimation. In *CVPR*, 2018. 91

[23] Aleksis Pirinen and Cristian Sminchisescu. Deep reinforcement learning of region proposal networks for object detection. *CVPR*, 2018. 92

[24] Aleksis Pirinen, Erik Gärtner, and Cristian Sminchisescu. Domes to drones: Self-supervised active triangulation for 3d human pose reconstruction. In *NeurIPS*, pages 3907–3917, 2019. 91, 100, 101

[25] Alin-Ionut Popa, Mihai Zanfir, and Cristian Sminchisescu. Deep multitask architecture for integrated 2d and 3d human sensing. In *CVPR*, 2017. 91, 100, 110

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 93, 110

[27] Helge Rhodin, Nadia Robertini, Dan Casas, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. General automatic human shape and motion capture using volumetric contour cues. In *ECCV*, 2016. 91

[28] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net: Localization-classification-regression for human pose. In *CVPR*, 2017. 91

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 94

[30] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018. 91

[31] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, pages 4724–4732, 2016. 91

[32] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992. 99

[33] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, pages 9068–9079, 2018. 91

[34] Bo Xiong and Kristen Grauman. Snap angle prediction for 360 panoramas. In *ECCV*, pages 3–18, 2018. 91

[35] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-driven visual object tracking with deep reinforcement learning. *IEEE transactions on neural networks and learning systems*, 29(6):2239–2252, 2018. 92

[36] Andrei Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes–the importance of multiple scene constraints. In *CVPR*, pages 2148–2157, 2018. 91

[37] Andrei Zanfir, Elisabeta Marinoiu, Mihai Zanfir, Alin-Ionut Popa, and Cristian Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *NeurIPS*, pages 8410–8419, 2018. 100, 110

[38] Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017. 92

[39] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017. 91

Supplementary Material

In this supplemental we provide additional insights into our Pose-DRL model. Details of the network architecture are provided in §A. Further model insights and dataset details are provided in §B. A description of how we handle missed detections or failed matchings are given in §C. Finally, additional visualizations are shown in §D.

A Model Architecture

See Fig. A.8 for a description of the Pose-DRL architecture. The underlying pose estimation networks, DMHS [25] and MubyNet [37], as well as our agent were implemented in Caffe [12] and MATLAB. For the Faster R-CNN detector [26] we used a publicly available Tensorflow [1] implementation,[1] with ResNet-101 [9] as base feature extractor.

Table A.3: Pose-DRL agent's selection statistics of good / bad viewpoints on the test set splits. The agent consistently chooses a high percentage of good cameras while avoiding bad cameras. Note that randomly choosing cameras would result in always having 10% chosen among the 10% best cameras, and similar for the 10% worst cameras.

	10% best	10% worst	Rest
Mafia	52 %	2%	46%
Ultimatum	67%	1%	32%
Pose	24%	2%	74%
All	43%	2%	55%

Table A.4: Breakdown of the subset of the Panoptic dataset used in this work for the training, validation and test splits, respectively. In each cell is shown the number of images. The fourth row shows the total number of images in the train, val and test splits (summed over *Mafia*, *Ultimatum* and *Pose*). The fourth column shows the total number of images for *Mafia*, *Ultimatum* and *Pose* (summed over the train, val and test splits). The bottom-right cell shows the total number of images in the entire used dataset.

	Train	Val	Test	All
Mafia	53,100	27,900	33,728	114,728
Ultimatum	27,960	4,340	55,825	88,125
Pose	51,079	29,672	59,288	140,039
All	132,139	61,912	148,841	342,892

[1] https://github.com/smallcorgi/Faster-RCNN_TF

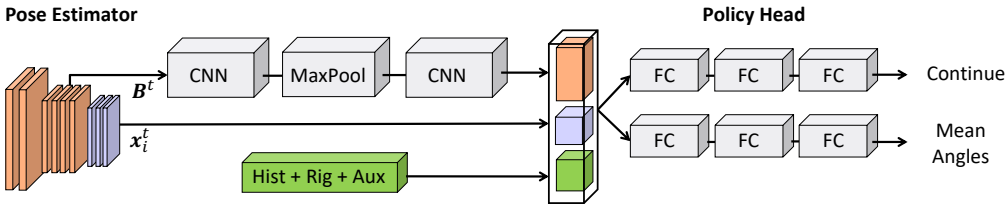


Figure A.8: Pose-DRL network architecture. The pose estimator is shown to the left (we have shown results for two different pose estimators, DMHS and MubyNet, but any other moncoular pose estimator would work). When using the single person pose estimator DMHS, the input is a bounding box containing the target person, and its convolutional feature map $\boldsymbol{B}_t$ forms the base state of the agent. For the multi-person estimator MubyNet, the full image is instead fed as input and the associated feature map $\boldsymbol{B}_t$ is used as the base state. Next, $\boldsymbol{B}_t$ is processed by two convolutional layers with ReLU-activations (first conv: 3×3 kernel, stride 1, output dimension $21 \times 21 \times 8$; max pool: 2×2 kernel, stride 2, output dimension $11 \times 11 \times 8$; second conv: 3×3 kernel, stride 1, output dimension $9 \times 9 \times 4$). It is then concatenated with the pose prediction information $\boldsymbol{x}_i^t$ for the current active-view, a history of the last 4 fused pose estimates from previous active-views (*Hist*), camera rig information (*Rig*), containing both a description of the camera rig as well as the agent's current and previously visited viewpoints within the rig, and auxiliary information (*Aux*) with the number of actions taken and number of people detected. Note that pose information is used in the single-target mode only; for the multi-person setting we omit pose information in the state space as there may be a variable number of persons per scene. However, in this setting the agent instead has access to image level information. See more about the state space in §4.2 in the main paper. The concatenated state is subsequently fed to the two action branches: the *continue* action branch (top) and the *viewpoint selection* action branch (bottom). Both branches use tanh-activations for the hidden fully-connected (FC) layers. For the *continue* action branch, the output is turned into a *continue* probability through a sigmoid-layer, cf. (2) in the main paper. For the *viewpoint selection* action branch, the azimuth and elevation mean angles are produced by a scaled tanh-layer, cf. equations (4) and (5) in the main paper. In the *continue* action branch the three FC-layers have 512, 512, and 1 output neurons each respectively, while the *viewpoint selection* action branch's three FC-layers have 1024, 512, and 2 output neurons, respectively.

B Additional Insights and Details

More about runtimes. All experiments reported in this supplementary material and in the main paper were performed using an Ubuntu workstation using a single Titan V100. Training the Pose-DRL policy from scratch took about 70 hours after having pre-computed all DMHS / MubyNet features, Faster R-CNN bounding boxes and instance features. When presenting the runtimes (see Figure 2 in the main paper) we include the time needed to compute these detections and features.

Quality of selected viewpoints. To obtain further insights into which cameras the agent is selecting on average, we tracked how often the agent selects good vs bad viewpoints (for the DMHS-based model). Specifically, for each selected camera in the various test set splits, we sorted it into being in either the 10% best or worst cameras based on associated individual reconstruction error. The results are shown in Table A.3. It can be seen that the agent typically selects among the best while avoiding the worst viewpoints. The viewpoint errors are more uniform for the single-people *Pose* scenes, since there are no viewpoints where the target is occluded, hence the camera selection statistics are also more uniform for *Pose*.

Further dataset insights. In Table A.4 we show how we randomly split the Panoptic dataset [14] into train, test, and validation sets.

C Handling Missed Detections or Matchings

For an overview of how we detect and match multiple people, refer to §3.2 in the main paper. In this section we describe what happens in case some persons are not detected or matched. For the detection-based DMHS-version of Pose-DRL, if in a viewpoint there are no detections, or if no detection has a matching cost below the threshold $\mathcal{C}$, the underlying pose estimator is computed on the entire input image to obtain a base state descriptor $\boldsymbol{B}_t$ for decision making (no associated pose is fused in this case).

It is possible that one or several persons are not detected in a single viewpoint in an active-view. In this case the pose estimate is set to the fused estimate from the previous active-view as a backup. In case a previous estimate also does not exist (could happen e.g. in the initial active-view of an active-sequence), to be able to compute a reconstruction error we set a placeholder pose estimate where each joint is equal to the center hip location of the ground-truth. Naturally, this is an extremely poor and implausible estimate, but it is used only to be able to compute an error (another option would be to not include such an estimate when computing average errors, but that would not penalize the fact that a person was never detected and reconstructed).

D Additional Visualizations of Pose-DRL

In Fig. D.9 - D.10 we show two additional visualizations of how Pose-DRL performs single-target pose estimation in active-views from the Panoptic [14] test set we have used in this work. We use SMPL [18] for the 3d shape models (both here and in the main paper), and use per-joint median averaging for fusing poses. As it is referenced in the visualizations, we show the equation for a partially fused pose (for the first j steps) within an active-view[1] below:

$$\tilde{\boldsymbol{x}} = f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i) \tag{10}$$

[1]For active-sequence processing, the agent also fuses temporally by adding the previous fused estimate; see eq. (1) in the main paper

D.1 Using Pose-DRL the Wild

The dense CMU Panoptic studio provides a powerful environment for training and evaluating our proposed model, however it is also interesting to test the model's applicability in the the real world. To this end we captured data with an off-the-shelf smartphone and used internal sensors to estimate the camera pose matrix for each image. This simple process of walking around subjects while they stand still emulates the *time-freeze* setup in Panoptic and allows us to test our model in the real world. Please note that neither the 3d pose estimation network nor the policy was re-trained; only the instance detector was refined to produce accurate appearance models for the detected people. See Fig. D.11 for resulting visualizations. Please note we obtained consent from the people shown.

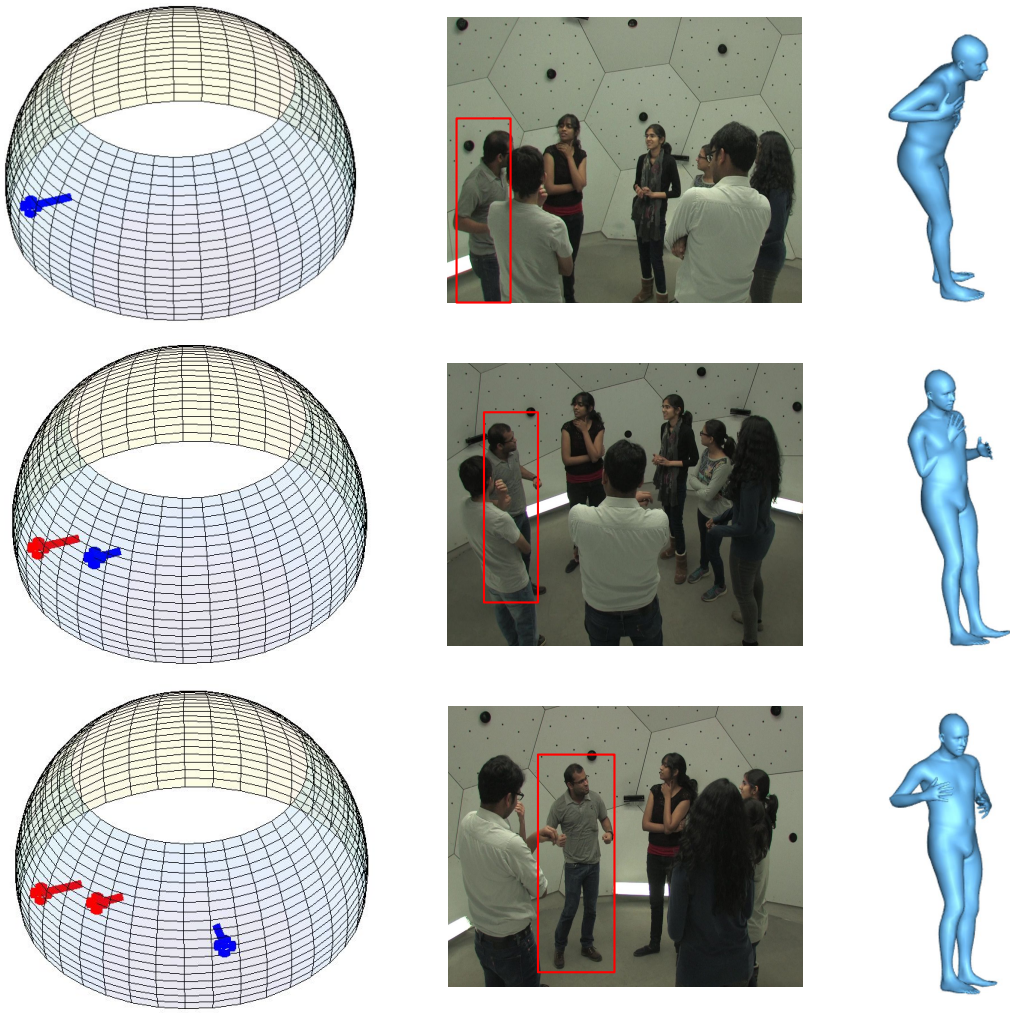


Figure D.9: Visualization of how Pose-DRL performs single-target reconstruction on an active-view (set of viewpoints for a time-freeze) for a *Mafia* test scene. In this case the agent sees three viewpoints prior to automatically continuing to the next active-view. The reconstruction error reduces from 168 to 107 mm/joint. Left: Viewpoints seen by the agent, where blue marks the current viewpoint (camera) and red marks previous viewpoints. Note that the initial camera was given randomly. Middle: Input images associated to the viewpoints, also showing the detection bounding box of the target person in red – detections for the other people are left out to avoid visual clutter. Right: SMPL visualizations of the partially fused poses, cf. (10). The target person is only partially visible in the initial viewpoint, and the associated pose estimate is inaccurate with the reconstruction incorrectly tilting forward. As the agent visits more viewpoints, the stance of the reconstruction becomes straighter and more correct. The person is fully visible in the final viewpoint, and the associated final fused estimate is accurate and plausible.

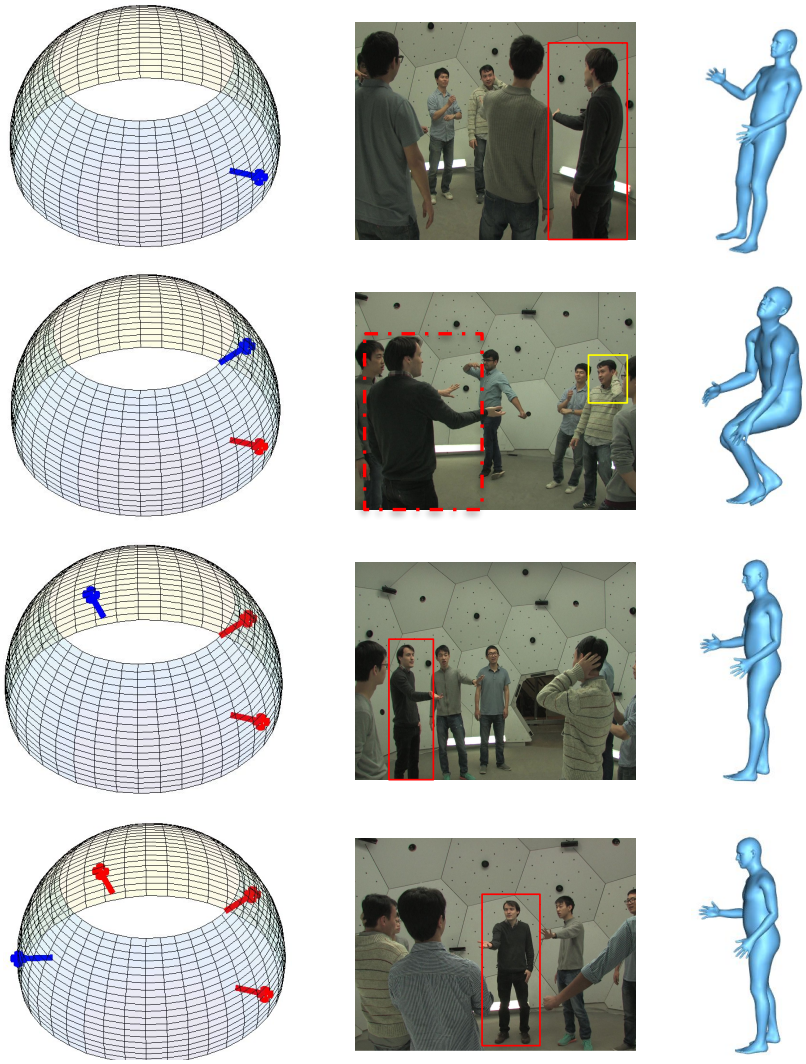


Figure D.10: Visualization of how Pose-DRL performs single-target reconstruction on an active-view (set of viewpoints for a time-freeze) for an *Ultimatum* test scene, where in this case the detection and matching is incorrect for the second viewpoint. Left: Viewpoints seen by the agent, where blue marks the current viewpoint (camera) and red marks previous viewpoints. Note that the initial camera was given randomly. Middle: Input images associated to the viewpoints, also showing the detection bounding box of the target person in red – detections for the other people are left out to avoid visual clutter. In the second viewpoint with the incorrect detection and matching, the target person is indicated with a dashed red bounding box, and the incorrect detection used in the pose fusion is shown in yellow. Right: SMPL visualizations of the partially fused poses, cf. (10). The target person is viewed from a suboptimal direction in the first viewpoint, causing the associated pose estimate to be incorrectly tilted. As the agent moves to the next viewpoint to get a better view of the person, the underlying detection and matching system suggests an incorrect detection to feed the pose estimator, which causes the fused estimate to deteriorate severely. However, the agent is able to remedy this by selecting two more good and diverse viewpoints where the target is clearly visible, yielding a considerably better fused pose estimate. In this example the agent sees four viewpoints prior to automatically continuing to the next active-view. The reconstruction error reduces from 149 to 119 mm/joint.

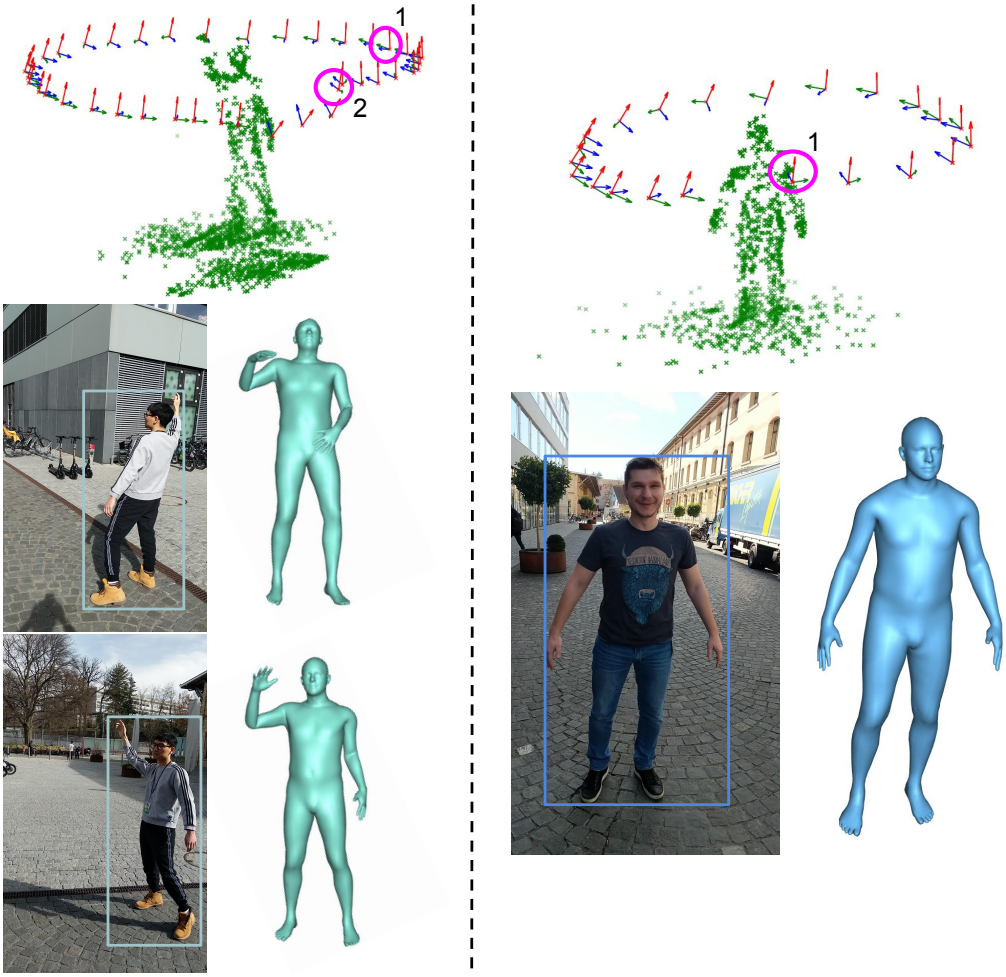


Figure D.11: People standing in various poses, captured with a smartphone camera from different viewpoints. Note that this data is significantly different from that obtained from Panoptic, with more challenging outdoor lighting conditions, human-imposed errors from holding and directing the smartphone camera, etcetera. We show two visualization of how Pose-DRL operates in different scenarios. Pose-DRL was *not* re-trained on this data; we use the same model weights as for producing the results in the main paper. In each scenario we also show the 3d configuration of the scene, as well as which viewpoints are selected by the agent and in which order (pink circles). Left: In this example the agent sees two views before terminating viewpoint selection. The initial randomly given viewpoint produces a pose estimate where the arms are not accurate, which is corrected for in the second and final viewpoint. Right: The agent receives a very good initial viewpoint and decides to terminate viewpoint selection immediately, producing an accurate pose estimate. See §D.1 for more details about these visualizations.

Paper III

David Nilsson, Aleksis Pirinen, **Erik Gärtner**, Cristian Sminchisescu
Embodied Visual Active Learning for Semantic Segmentation
Association for the Advancement of Artificial Intelligence (AAAI), Virtual conference, 2021

Embodied Visual Active Learning for Semantic Segmentation

David Nilsson[1,2] **Aleksis Pirinen**[1] **Erik Gärtner**[1,2*]
Cristian Sminchisescu[1,2]

[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]Google Research

Abstract

We study the task of *embodied visual active learning*, where an agent is set to explore a 3d environment with the goal to acquire visual scene understanding by actively selecting views for which to request annotation. While accurate on some benchmarks, today's deep visual recognition pipelines tend to not generalize well in certain real-world scenarios, or for unusual viewpoints. Robotic perception, in turn, requires the capability to refine the recognition capabilities for the conditions where the mobile system operates, including cluttered indoor environments or poor illumination. This motivates the proposed task, where an agent is placed in a novel environment with the objective of improving its visual recognition capability. To study embodied visual active learning, we develop a battery of agents – both learnt and pre-specified – and with different levels of knowledge of the environment. The agents are equipped with a semantic segmentation network and seek to acquire informative views, move and explore in order to propagate annotations in the neighbourhood of those views, then refine the underlying segmentation network by online retraining. The trainable method uses deep reinforcement learning with a reward function that balances two competing objectives: *task performance, represented as visual recognition accuracy*, which requires exploring the environment, and the necessary *amount of annotated data* requested

*Work was partially performed during a Google internship.

during active exploration. We extensively evaluate the proposed models using the photorealistic Matterport3D simulator and show that a fully learnt method outperforms comparable pre-specified counterparts, even when requesting fewer annotations.

1 Introduction

Imagine a household robot in a home it has never been before and equipped with a visual sensing module to perceive its environment and localize objects. If the robot fails to recognize some objects, or to adapt to changes in the environment, over time, it may not be able to properly perform its tasks. Much of the recent success of visual perception has been achieved by deep CNNs, e.g. in image classification [15, 23, 41], semantic segmentation [6, 25] and object detection [34, 35]. Such systems may however be challenged by unusual viewpoints or domains, as noted e.g. by Ammirato et al. [2] and Yang et al. [53]. Moreover, a mobile household robot should ideally operate with lightweight, re-trainable and task-specific perception models, rather than large and comprehensive ones, which could be demanding computationally and not tailored to the needs of a specific house.

In practice, even in closed but large environments, developing robust scene understanding by exhaustive approaches may be difficult, as looking everywhere requires an excessive amount of annotation labor. All views are however not equally informative, as a view containing many diverse objects is likely more useful than one covering a single semantic class, e.g. a wall. This suggests that in learning visual perception one does not have to label exhaustively. As new, potentially difficult arrangements appear in an evolving environment, it would be useful to identify those automatically, based on the task and demand, rather than programmatically, by periodically re-training a complete model. Moreover, the agent could make the most out of its embodiment by propagating a given ground truth annotation using motion – as measured by the perceived optical flow – in that neighborhood. The agent can then self-train, online, for increased performance. The key questions are how should one explore the environment, how to select the most informative views to annotate, and how to make the most out of them. We analyze these questions in an *embodied visual active learning* framework, illustrated in Fig. 3.1.

To ground the embodied visual active learning task, in this work we measure visual perception ability as semantic segmentation accuracy. The agent is equipped with a semantic segmentation system and must move around and request annotations in order to refine it. After exploring the scene the agent should be able to accurately segment all views in the explored area. This requires an exploration policy covering different objects from diverse viewpoints and selecting sufficiently many annotations to train the perception model. The agent can also propagate annotations to different nearby viewpoints using optical flow and

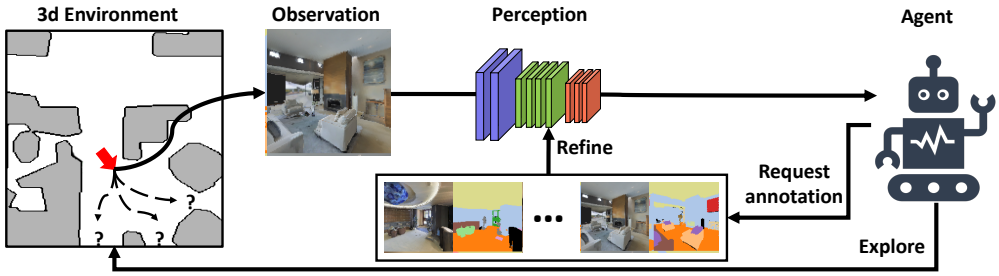


Figure 3.1: Embodied visual active learning. An agent in a 3d environment must explore and occasionally request annotation in order to efficiently refine its visual perception. The navigation component makes this task significantly more complex than traditional active learning, where the data pool over which the agent queries annotations, either in the form of image collections or pre-recorded video streams, is static and given.

then self-train. We develop a battery of methods, ranging from pre-specified ones to a fully trainable deep reinforcement learning-based agent, which we evaluate extensively in the photorealistic Matterport3D environment [4].

In summary, our main contributions are:

- We study the task of *embodied visual active learning*, where an agent should explore a 3d environment to acquire visual scene understanding by actively selecting views for which to request annotation. The agent then propagates information by moving in the neighborhood of those views and self-trains;
- In our setup, visual learning and exploration can inform and guide one another since the recognition system is selectively and gradually refined during exploration, instead of being trained at the end of a trajectory on a full set of densely annotated views;
- We develop a variety of methods, both learnt and pre-specified, to tackle our task in the context of semantic segmentation;
- We perform extensive evaluation in a photorealistic 3d environment and show that a fully learnt method outperforms comparable pre-specified ones.

2 Related Work

The embodied visual active learning setup leverages several computer vision and machine learning concepts, such as embodied navigation, active learning and active vision. There is substantial recent literature on embodied agents navigating in real or simulated 3d environments, especially given the recent emergence of large-scale simulators [9, 22, 36, 37, 49].

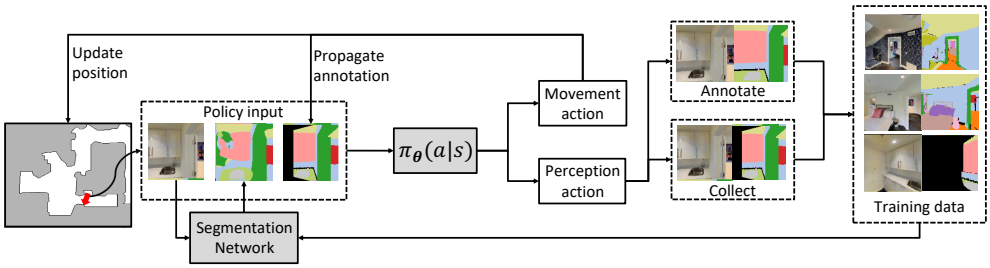


Figure 3.2: **Embodied visual active learning for semantic segmentation.** A first-person agent is placed in a room and a deep network predicts the semantic segmentation of the agent's view. Based on the view and its segmentation, the agent can either select a *movement action* to change position and viewpoint, or select a *perception action* (`Annotate` or `Collect`). `Annotate` adds the current view and its ground truth segmentation to the pool of training data for the segmentation network, while `Collect` is a cheaper version (no additional supervision required) where the current view and the last annotated view – propagated to the agent's current position using optical flow – is added to the training set. The propagated annotation is also a policy input for the learnt agent in §3.3. After a perception action, the segmentation network is refined on the current training set. The embodied visual active learning process is considered successful if, after selecting a limited number of `Annotate` actions or an exploration budget is exhausted, the segmentation network can accurately segment any other view in the environment where the agent operates. Note that the map (left) is *not* provided as input to the learnt agent in §3.3.

We here briefly review variants of embodied learning. In Embodied Question Answering [8, 47, 54], an agent is given a question, e.g. "What color is the car?". The agent must typically explore the environment quite extensively in order to be able to answer. Mousavian et al. [28], Zhu et al. [57] task the agent with reaching a target view using as few steps as possible. The agent receives the current view and the target as inputs in each step. In point-goal navigation [14, 27, 37, 38] the agent is given coordinates of a target to reach using visual information and ego-motion. In visual exploration [5, 7, 10, 32, 33, 55] the task is to explore an unknown environment as quickly as possible, by covering the whole scene area. In Ammirato et al. [2], Yang et al. [53] an agent is tasked to navigate an environment to increase the accuracy of a pre-trained recognition model, e.g. by moving around occluded objects. This is in contrast to our work where the goal is to collect views for *training* a perception model. Whereas in Ammirato et al. [2], Yang et al. [53] the agent is spawned close to the target object, we cannot make such assumptions, as our task is not only to accurately recognize a single object or view, but to do so for *all* views in the potentially large area explored by the agent.

There are relations to curiosity-driven learning [30, 52], in that we also seek an agent which visits novel views (states). In Pathak et al. [30], exploration is aided by giving rewards based on the prediction error of a self-supervised inverse-dynamics model. This is a task-independent exploration strategy useful to search 2d or 3d environments during training. In our setup, exploration is task-specific in that it is aimed specifically at refining a visual recognition system in a novel environment. Moreover, we use semi-dense rewards for both visual learning and for exploration. Hence we are not operating using sparse rewards where curiosity approaches often outperform other methods.

Our work is also related to Pot et al. [31], Song et al. [42], Wang et al. [46], Zhong et al. [56]. Differently from us, Song et al. [42] uses hand-crafted annotation and exploration strategies, aiming to label all voxels in a 3d reconstruction by selecting a subset of frames covering all voxels. This is a form of exhaustive annotation and a visual perception system is not trained. Hence the system can only analyze objects in annotated voxels. In our setup the agent is instead tasked with both exploration and the selection of views to annotate, and we learn a perception module aiming to generalize to unseen views. In contrast to us, Pot et al. [31], Wang et al. [46], Zhong et al. [56] do not consider an agent choosing where to move in the environment, nor which parts to label. Instead, they use all views seen when following a pre-specified path for training a visual recognition system. Pot et al. [31] use an object detector obtained by self-supervised learning and clustering. Wang et al. [46], Zhong et al. [56] use constraints from SLAM to improve a given segmentation model. This approach could in principle complement our label propagation, and is orthogonal to our main proposals.

Next-best-view (NBV) prediction [13, 17, 18, 20, 43, 50] is superficially similar to our task. In Jayaraman and Grauman [18] an agent is trained to reveal parts of a panorama and a model is built to complete all views of the panorama. Our setup allows free movement in an environment, hence it features a navigation component which makes our task more comprehensive. While NBV typically integrates information from all predicted views, our task requires the adaptive selection of only a subset of the views encountered during the agent's navigation trajectory.

Active learning [11, 12, 26, 29, 40, 48] can be seen as the static version of our setup, as it considers approaches for learning what parts of a larger *pre-existing* and *static* training set should be fed into the training procedure, and in what order. We instead consider the active learning problem in an embodied setup, where an agent can move and actively select views for which to request annotation. Embodiment makes it possible to use motion to propagate annotations, hence effectively generate new ones at no additional annotation cost. In essence, our work lays groundwork towards marrying the active vision and the active learning paradigms.

3 Embodied Visual Active Learning

Embodied visual active learning is an interplay between a first-person agent, a 3d environment and a trainable perception module. See Fig. 3.1 for a high-level abstraction and Fig. 3.2 for details of the particular task considered in this paper. The perception module processes images (views) observed by the agent in the environment. The agent can request annotations for views in order to refine the perception module. It should ideally request very few annotations as these are costly. The agent can also generate more annotations for

free by neighborhood exploration using label propagation, such that when trained on that data the perception module becomes increasingly more accurate in the explored environment. To assess how successful an agent is on the task, we test how accurate the perception module is on multiple random viewpoints selected uniformly in the area explored by the agent.

Task overview. The agent begins each episode randomly positioned and rotated in a 3d environment, with a randomly initialized semantic segmentation network. The ground truth segmentation mask for the first view is given for the initial training of the segmentation network. The agent can choose *movement actions* (`MoveForward`, `MoveLeft`, `MoveRight`, `RotateLeft`, `RotateRight` with 25 cm movements and 15 degree rotations), or *perception actions* (`Annotate`, `Collect`). If the agent moves or rotates, the ground truth mask is propagated using optical flow. At any time, the agent may choose to insert the propagated annotation into its training set with the `Collect` action, or to ask for a new ground truth mask with the `Annotate` action. After an `Annotate` action the propagated annotation mask is re-initialized to the ground truth annotation. After each perception action, the segmentation network $\mathcal{S}$ is refined on the training set, which is expanded with the new data point.

The agent's performance is evaluated at the end of the episode. The goal is to maximize the mIoU and mean accuracy of the segmentation network on the views in the area explored by the agent. Specifically, a set of *reference views* are randomly sampled within a disc of radius r centered at the starting location, and the segmentation network is evaluated on these. Hence to perform well the agent is required to explore its surroundings, and it should refine its perception module in regions of high uncertainty.

3.1 Methods for the Proposed Task

We develop several methods to evaluate and study the embodied visual active learning task. All methods except the RL-agent issue the `Collect` action when 30% of the propagated labels are unknown and `Annotate` when 85% are unknown. The intuition is that the prespecified methods should request annotation when most pixels are unlabeled. The specific hyperparameters of all models were set based on a validation set.

Random. Uniformly selects random movement actions. This baseline is thus a lower bound in terms of embodied exploration for this task.

Rotate. Continually rotates left. This method is useful in comparing with trainable agents that move and explore, i.e. to monitor what improvements can be expected from embodiment.

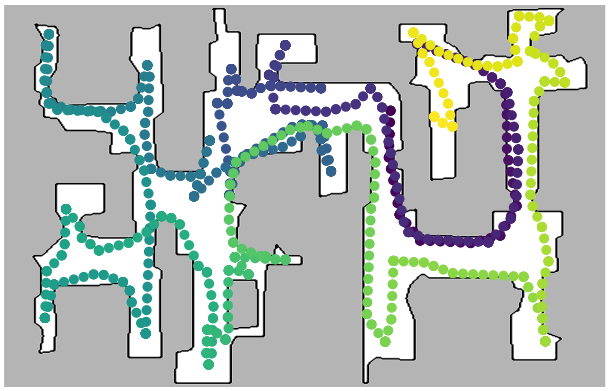


Figure 3.3: An example of a space filling curve in a Matterport3D floor plan. Methods based on the space filler assume complete spatial knowledge of the environment.

Bounce. Explores by walking straight forward until it hits a wall, then samples a new random direction and moves forward until it collides with a new wall, and so on. This agent quickly explores the environment.

Frontier exploration. This method builds a map, online, by using using depth and motion from the simulator [51]. All pixels with depth within a 4m threshold are back-projected in 3d and then classified as either obstacles or navigable, based on height relative to the ground plane. This agent is confined to move within the reference view radius r, which is a choice to its advantage[1] as annotated views will more likely be similar to reference views that reside within that same radius.

Space filler. Follows a shortest space filling curve within the reference view radius r, and as r increases the entire environment is explored. This baseline makes strong and somewhat less general (or depending on the application, altogether unrealistic) assumptions in order to create a path: knowing the floor plan in advance, as well as which locations are reachable from the start. It also only moves within the reference view radius, and knows the shortest geodesic paths to take on the curve. Hence, this method can be considered an upper bound for other methods. The space filling curve is computed by placing a grid of nodes onto the floor plan (1m resolution, using a sampling and reachability heuristic), and then finding the shortest path around it with an approximate traveling salesman solver. Fig. 3.3 shows a space filling curve in a Matterport3D floor plan.

RL-agent. This fully trainable method we develop jointly learns exploration and perception actions in a reinforcement learning framework. See the full description in §3.3.

[1] This ensures it is evaluated under ideal conditions in contrast to the RL-agent in §3.3.

3.2 Semantic Segmentation Network

Each method uses the same FCN-inspired deep network [25] for semantic segmentation. The network consists of 3 blocks of convolutional layers, each containing 3 convolutional layers with kernels of size 3×3. The first convolutional layer in each block uses a stride of 2, which halves the resolution. For each block the number of channels doubles, using 64, 128 and 256 channels respectively. Multiple predictions are made using the final convolutional layers of each block. The multi-scale predictions are resized to the original image resolution using bilinear interpolation and are finally summed up, resulting in the final segmentation estimate. Note that we have deliberately chosen to make the network small so that it can be efficiently refined on new data.

At the beginning of each episode, the parameters are initialized randomly, and we train the network on the very first view, for which we always supply the ground truth segmentation. Each time `Annotate` or `Collect` is selected, we refine the network. Mini-batches of size 8, which always include the latest added labeled image, are used in training. We use random cropping and scaling for data augmentation. The network is refined either until it has trained for $1,000$ iterations or until the accuracy of a mini-batch exceeds 95%. We use a standard cross-entropy loss averaged over all pixels. The segmentation network is trained using stochastic gradient descent with learning rate 0.01, weight decay 10^{-5} and momentum 0.9. To propagate semantic labels, we compute optical flow between consecutive viewpoints using PWC-Net [44]. The optical flow is computed bidirectionally and only pixels where the difference between the forward and backward displacements is less than 2 pixels are propagated [45]. We found that labels were reliably tracked over several frames when using 2 pixels as a threshold.

3.3 Reinforcement Learning Agent

To present the reinforcement-learning agent for our task, we begin with an explanation of the state-action representation and policy network, followed by the reward structure and finally policy training.

Actions, states and policy. The agent is represented as a deep stochastic policy $\pi_{\boldsymbol{\theta}}(a_t|s_t)$ that samples an action a_t in state s_t at time t. The actions are `MoveForward`, `MoveLeft`, `MoveRight`, `RotateLeft`, `RotateRight`, `Annotate` and `Collect`. The full state is $s_t = \{\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t, \boldsymbol{F}_t\}$ where $\boldsymbol{I}_t \in \mathbb{R}^{127\times127\times3}$ is the image, $\boldsymbol{S}_t = \mathcal{S}_t(\boldsymbol{I}_t) \in \mathbb{R}^{127\times127\times3}$ is the semantic segmentation mask predicted by the deep network $\mathcal{S}_t$ (this network is refined over an episode; t indexes the network parameters at time t), $\boldsymbol{P}_t \in \mathbb{R}^{127\times127\times3}$ is the propagated annotation, and $\boldsymbol{F}_t \in \mathbb{R}^{7\times7\times2048}$ is a deep representation of $\boldsymbol{I}_t$ (a ResNet-50 backbone feature map).

The policy consists of a *base processor*, a *recurrent module* and a *policy head*. The base processor consists of two learnable components: ϕ_{img} and ϕ_{res}. The 4-layer convolutional network ϕ_{img} takes as input the depth-wise concatenated triplet $\{\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t\}$, producing $\phi_{img}(\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t) \in \mathbb{R}^{512}$. Similarly, the 2-layer convolutional network ϕ_{res} yields an embedding $\phi_{res}(\boldsymbol{F}_t) \in \mathbb{R}^{512}$ of the ResNet features $\boldsymbol{F}_t$. An LSTM [16] with 256 cells constitutes the recurrent module, which takes as input $\phi_{img}(\boldsymbol{I}_t, \boldsymbol{S}_t, \boldsymbol{P}_t)$ and $\phi_{res}(\boldsymbol{F}_t)$. The input has length 1024. The hidden LSTM state is fed to the policy head, consisting of a fully-connected layer followed by a 7-way softmax which produces action probabilities.

Rewards. In training, the main reward is related to the mIoU improvement of the final segmentation network $\mathcal{S}_T$ over the initial $\mathcal{S}_0$ on a reference set $\mathcal{R}$. The set $\mathcal{R}$ is constructed at the beginning of each episode by randomly selecting views within a geodesic distance r from the agent's starting location, and contains views with corresponding ground truth semantic segmentation masks. At the end of an episode of length T, the underlying perception module is evaluated on $\mathcal{R}$. Specifically, after an episode (with T steps), the agent receives as final reward:

$$R_T = \text{mIoU}(\mathcal{S}_T, \mathcal{R}) - \text{mIoU}(\mathcal{S}_0, \mathcal{R}) \tag{3.1}$$

To obtain a denser signal, tightly coupled with the final objective, we also give a reward proportional to the improvement of $\mathcal{S}$ on the reference set $\mathcal{R}$ after each `Annotate` (*ann*) and `Collect` (*col*) action:

$$R_t^{ann} = \text{mIoU}(\mathcal{S}_t, \mathcal{R}) - \text{mIoU}(\mathcal{S}_{t-1}, \mathcal{R}) - \epsilon^{ann} \tag{3.2}$$

$$R_t^{col} = \text{mIoU}(\mathcal{S}_t, \mathcal{R}) - \text{mIoU}(\mathcal{S}_{t-1}, \mathcal{R}) \tag{3.3}$$

To ensure the agent does not request costly annotations too frequently, each `Annotate` action is penalized with a negative reward $-\epsilon^{ann}$ (we set $\epsilon^{ann} = 0.01$), as seen in (3.2). Such a penalty is *not* given for the free `Collect` action. Moreover, the dataset we use has 40 different semantic classes, but some are very rare and apply only to small objects, and some might not even be present in certain houses. We address this imbalance by computing the mIoU using only the 10 largest classes, ranked by the number of pixels in the set of reference views for the current episode.

While the rewards (3.1) - (3.3) should implicitly encourage the agent to explore the environment in order to request annotations for distinct, informative views, we empirically found useful to include an additional explicit exploration reward. Denote by $\{\boldsymbol{x}_i\}_{i=1}^{t-1} = \{(x_i, y_i)\}_{i=1}^{t-1}$ the positions the agent has visited up to time $t-1$ in its current episode, and let $\boldsymbol{x}_t = (x_t, y_t)$ denote its current position. We define the exploration (*exp*) reward based on a kernel density estimate of the agent's visited locations:

$$R_t^{exp} = a - bp_t(\boldsymbol{x}_t) := a - \frac{b}{t-1}\sum_{i=1}^{t-1} k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{3.4}$$

where a and b are hyperparameters (both set to 0.003). Here $p_t(\boldsymbol{x}_t)$ is a Gaussian kernel estimate of the density with bandwidth 0.3m. It is large for previously visited positions and small for unvisited positions, thereby encouraging the agent's expansion towards new places in the environment. The exploration reward is only given for movement actions. Note that the pose $\boldsymbol{x}_i$ is only used to compute the reward R_t^{exp} and is not available to the policy via the state space.

Policy training. The policy network is trained using PPO [39] based on the RLlib reinforcement learning package [24], as well as OpenAI Gym [3]. For optimization we use Adam [21] with batch size 512, learning rate 10^{-4} and discount rate 0.99. During training, each episode consists of 256 actions. The agent is trained for 4k episodes, which totals 1024k steps.

Our system is implemented in TensorFlow [1], and it takes about 3 days to train an agent using 4 Nvidia Titan X GPUs. An episode of length 256 took on average about 3 minutes using a single GPU, and during training we used 4 workers with one GPU each, collecting rollouts independently. The runtime per episode varies depending on how frequently the agent decides to annotate, as training the segmentation network is the bottleneck and accounts for approximately 90% of the run-time. We used optical flow from the simulator to speed up policy training. For evaluation, the RL-agent and all other methods use PWC-Net to compute optical flow. The ResNet-50 feature extractor is pre-trained on ImageNet [19] with weights frozen during policy training.

4 Experiments

In this section we provide empirical evaluations of various methods. The primary metrics are mIoU and segmentation accuracy but we emphasize that we test the exploration and annotation selection capability of *policies* – the mIoU and accuracy measure how well agents explore in order to refine their perception. Differently from accuracy, the mIoU does not become overly high by simply segmenting large background regions (such as walls), hence it is more representative of overall semantic segmentation quality.

Experimental setup. We evaluate the methods on the Matterport3D dataset [4] using the embodied agent framework Habitat [37]. This setup allows the agent to freely explore photorealistic 3d models of large houses, that have ground truth annotations for 40 diverse semantic classes. Hence it is a suitable environment for evaluation. To assess the generalization capability of the RL-agent we train and test it in *different* houses. We use the same 61, 11 and 18 houses for training, validation and testing as Chang et al. [4]. The RL-agent and all pre-specified methods except the space filler are *comparable* in terms of assumptions, cf. §3.1. The space filler assumes full spatial knowledge of the environment (ground truth

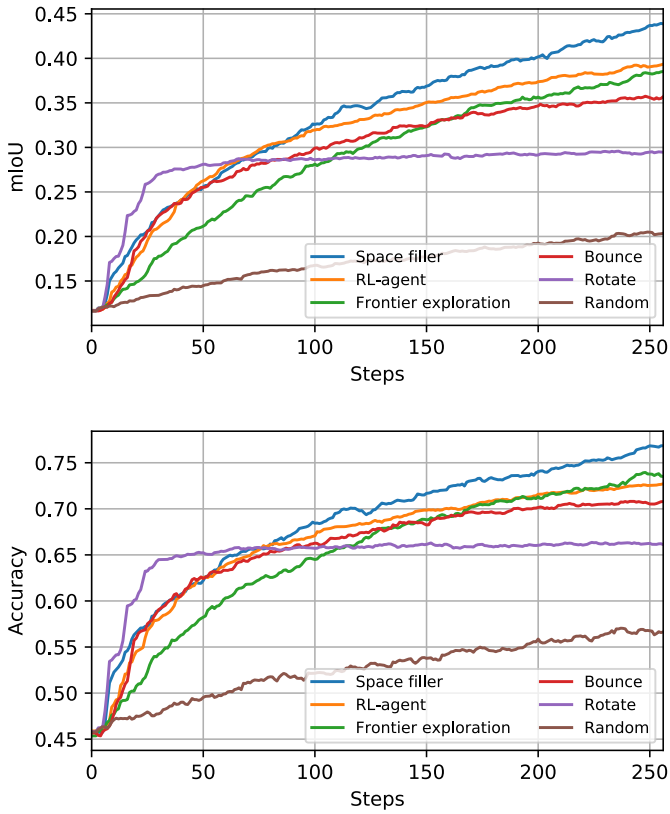


Figure 3.4: Mean segmentation accuracy and mIoU versus number of actions (steps), evaluated on the Matterport3D test scenes. The RL-agent was trained on 256-step episodes. This agent fairly quickly outperforms all other comparable pre-specified agents. *Rotate* is strong initially since it quickly gathers many annotations in a 360 degree arc, but is eventually outperformed by most other methods that move around in the houses. Frontier exploration yields similar accuracy as the RL-agent after about 170 steps, but uses significantly more annotations (cf. Table 3.1) and assumes perfect pose and depth information. The space filler, which assumes full knowledge of the environment, yields the best results after about 100 steps.

map) and hence has inherent advantages over the other methods.

During RL-agent training we randomly sample starting positions and rotations from the training houses at the start of each episode. An episode ends after 256 actions. Hyperparameters of the learnt and pre-specified agents are tuned on the validation set. For validation and testing we use 3 and 4 starting positions per scene, respectively, so each agent is tested for a total of 33 episodes in validation and 72 episodes in testing. The reported metrics are the mean over all these runs. All methods are evaluated on the same starting positions in the same houses. The reference views used to evaluate the semantic segmentation performance are obtained by sampling 32 random views within a 5 m geodesic distance of the agent's starting position at the beginning of each episode. In training the reference views are

sampled randomly. During validation and testing, for fairness, we sample the same views for a given starting position when we test different agents. Note that there is no overlap between reference views during policy training and testing, since training, validation and testing houses are non-overlapping.

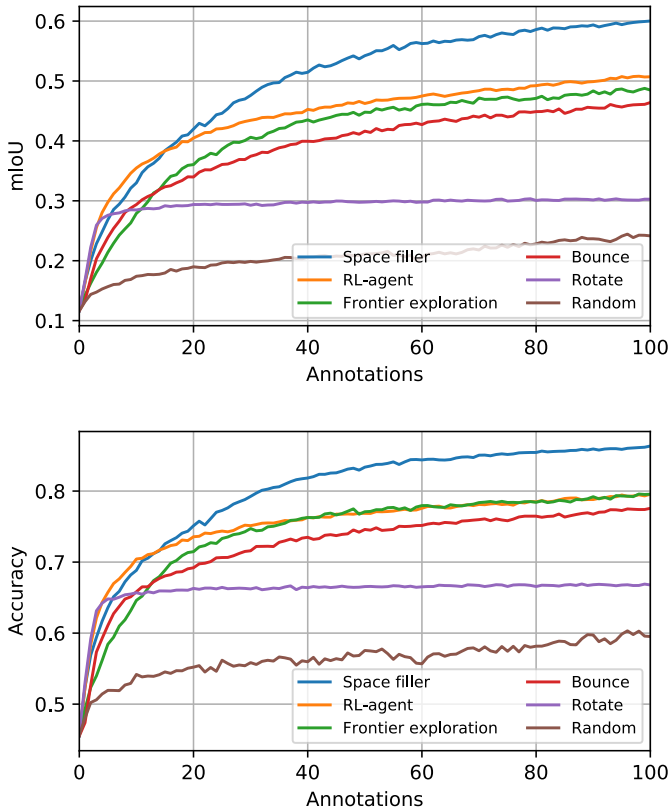


Figure 3.5: Mean segmentation accuracy and mIoU for a varying number of requested annotations evaluated on the Matterport3D test scenes. The RL-agent outperforms all comparable pre-specified methods (although frontier exploration matches it in accuracy after about 40 annotations), indicating that it has learnt an exploration policy which generalizes to novel scenes. The space filler, as expected, outperforms the RL-agent, except for less than 15 annotations. Thus the RL-agent is best before and around its training regime, where on average annotates 16.7 times per episode, cf. Table 3.1.

Recall that the RL-agent's policy parameters are denoted by $\boldsymbol{\theta}$. Let $\boldsymbol{\theta}_{seg}$ denote the parameters of the underlying semantic segmentation network, in order to clarify when we reset, freeze and refine $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{seg}$, respectively. For RL-training, we refine $\boldsymbol{\theta}$ during policy estimation in the training houses. When we evaluate the policy on the validation or test houses we freeze $\boldsymbol{\theta}$ and only use the policy for inference. The parameters of the segmentation network $\boldsymbol{\theta}_{seg}$ are always reset at the beginning of an episode, regardless of which house we deploy the agent in, and regardless of whether the policy network is training or not. During an episode, we refine $\boldsymbol{\theta}_{seg}$ exactly when the agent selects the `Annotate` or

Table 3.1: Comparison of different agents for a fixed episode length of 256 actions on the Matterport3D test scenes. The RL-agent gets higher mIoU using far fewer annotations than comparable pre-specified methods, implying that the RL-agent's policy selects more informative views to annotate.

Method	mIoU	Acc	# Ann	# Coll
Space filler	0.439	0.769	24.7	23.9
RL-agent	0.394	0.727	16.7	15.2
Frontier exploration	0.385	0.735	24.2	21.6
Bounce	0.357	0.708	29.6	26.0
Rotate	0.295	0.661	34.3	32.7
Random	0.204	0.566	29.1	19.5

`Collect` actions (this applies also to all the other methods described in §3.1). Thus annotated views in an episode are used to refine $\boldsymbol{\theta}_{seg}$ in that episode only, and are not used in any other episodes.

4.1 Main Results

We measure the performances of the agents in two settings: (a) with unlimited annotations but limited total actions (max 256, as during RL-training), or (b) for a limited annotation budget (max 100) but unlimited total actions. All methods were tuned on the validation set in a setup similar to (a) with 256 step episodes. Note however that the number of annotations can differ for different methods in a 256 step episode. The setup (b) is used to assess how the different methods compare for a fix number of annotations.

Fixed episode length. Table 3.1 and Fig. 3.4 show results on the test scenes for episodes of length 256. The RL-agent outperforms the comparable pre-specified methods in mIoU and accuracy, although frontier exploration – which uses perfect pose and depth information, and is idealized to always move within the reference view radius – yields similar accuracy after about 170 steps. The RL-agent uses much fewer annotations than other methods, hence those annotated views are more informative. The space filler, which assumes perfect knowledge of the map, outperforms the RL-agent but uses significantly more annotations. Note that the *Rotate* baseline saturates, supporting the intuition that an agent has to move around in order to increase performance in complex environments.

Fixed annotation budget. In Table 3.2 and Fig. 3.5 we show test results when the annotation budget is limited to 100 images per episode. As expected, the space filler yields the best results, although the RL-agent gets comparable performance when using up to 15 annotations. The RL-agent outperforms comparable pre-specified methods in mIoU and accuracy. Frontier exploration obtains similar accuracy. We also see that the episodes of the RL-agent are longer.

Qualitative examples. Fig. 3.6 shows examples of views that the RL-agent choose to an-

Table 3.2: Comparison of different agents for a fixed budget of 100 annotations on Matterport3D test scenes. The RL-agent gets a higher mIoU than comparable pre-specified agents, despite not being trained in this setting.

Method	mIoU	Acc	# Steps	# Coll
Space filler	0.600	0.863	1048	91
RL-agent	0.507	0.796	1541	94
Frontier exploration	0.485	0.796	998	84
Bounce	0.464	0.776	861	87
Rotate	0.303	0.668	752	96
Random	0.242	0.595	910	64

notate. The agent explores large parts of the space and the annotated views are diverse, both in their spatial locations and in the types of semantic classes they contain. Fig. 3.7 shows how the segmentation network's performance on two reference views improves during an episode. The two views are initially poorly segmented, but as the agent explores and acquires annotations for novel views, the accuracy on the reference views increases.

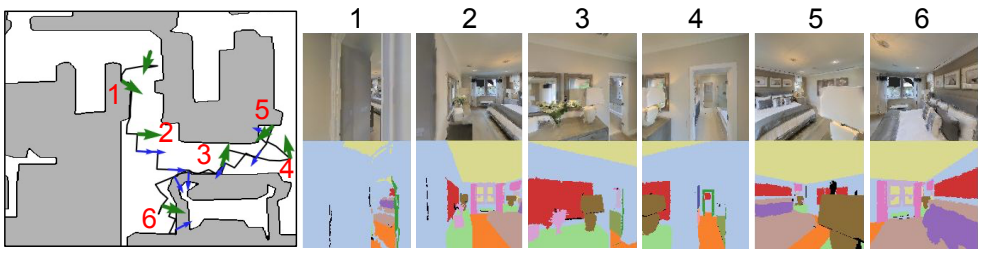


Figure 3.6: The first six requested annotations by the RL-agent in a room from the test set. Left: Map showing the agent's trajectory and the six first requested annotations (green arrows). The initially given annotation is not indicated with a number. Blue arrows indicate `Collect` actions. Right: For each annotation (numbered 1 - 6) the figures show the image seen by the agent and the ground truth received when the agent requested annotations. As can be seen, the agent quickly explores the room and requests annotations containing diverse semantic classes.

4.2 Ablation Studies of the RL-agent

Ablation results of the RL-agent on the validation set are in Table 3.3. We compare to the following versions: i) Policy without visual features ϕ_{img}; ii) Policy without ResNet features ϕ_{res}; iii) No additional exploration reward (3.4), i.e. $R_t^{exp} = 0$; iv) No `Collect` action and $\boldsymbol{P}_t$ is not an input to ϕ_{img}; and v) Only exploration trained, using the heuristic strategy for annotations. We trained the ablated models for 4,000 episodes as for the full model.

Both the validation accuracy and mIoU are higher for the full RL-model compared to all ablated variants, justifying design choices. The model not relying on propagating annotations and using the `Collect` action performs somewhat worse than the full model despite a comparable amount of annotations. The learnt annotation strategy yields higher mIoU

Table 3.3: Ablation study of different RL-based model variants for 256-step episodes on the validation set. The full RL-agent outperforms all ablated models at a comparable or lower number of requested annotations.

Variant	mIoU	Acc	# Ann	# Coll
Full model	0.427	0.732	16.4	16.4
No collect nor $\boldsymbol{P}_t$	0.415	0.727	17.9	0.0
Only exploration	0.411	0.727	16.1	14.4
$R_t^{exp} = 0$	0.401	0.719	17.7	47.4
No ϕ_{img}	0.378	0.696	14.3	3.8
No ResNet	0.375	0.705	23.3	0.3

and accuracy compared to the heuristic one, at comparable number of annotations. The exploration reward is important in encouraging the agent to navigate to unvisited positions – without it performance is worse, despite a comparable number of annotations. The agent trained without the exploration reward uses an excessive number of `Collect` actions, so this agent often stands still instead of moving. Finally, omitting either visual or ResNet features from the policy significantly harms accuracy for the resulting recognition system.

4.3 Analysis of Annotation Strategies

In this section we examine how different annotation strategies affect the task performance on the validation set for the space filler and bounce methods. Specifically, the annotation strategies are:

- **Threshold perception.** This is the variant evaluated in §4.1, i.e. it issues the `Collect` action when 30% of the propagated labels are unknown and `Annotate` when 85% are unknown.
- **Learnt perception.** We train a simplified RL-agent where the movement actions are restricted to follow the exploration trajectory of the baseline method (space filler and bounce, respectively). This model has 3 actions: move along the baseline exploration path, `Annotate` and `Collect`. All other training settings are identical to the full RL-agent.
- **Random perception.** In each step, this variant follows the baseline exploration trajectory with 80% probability, while annotating views and collecting propagated labels with 10% probability each.

As can be seen in Table 3.4, the best results for the space filler are obtained by using the threshold strategy, which also annotates slightly less frequently than other variants. Using learnt perception actions yields better results compared to random perception actions, and takes slightly fewer annotations per episode. Similar results carry over to the bounce

Table 3.4: Results for different model variants of the space filler method. We report the mean on the validation scenes. The threshold perception strategy – which is the one used in the main evaluations in §4.1 – yields the best results.

Variant	mIoU	Acc	# Ann	# Coll
Threshold perception	0.472	0.770	20.8	19.9
Learnt perception	0.454	0.755	22.8	37.4
Random perception	0.446	0.747	24.2	24.4

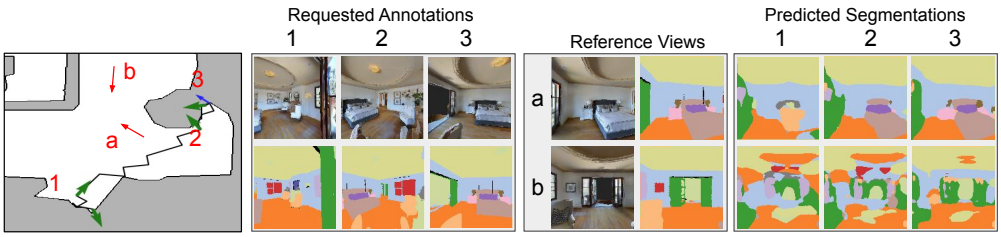


Figure 3.7: Example of the RL-agent's viewpoint selection and how its perception improves over time. We show results of two reference views after the first three annotations of the RL-agent. Left: Agent's movement path is drawn in black on the map. The annotations (green arrows) are numbered 1 - 3, and the associated views are shown immediately right of the map (the initially given annotation is not shown). Red arrows labeled *a* - *b* indicate the reference views. Right: Reference views and ground truth masks, followed by predicted segmentation after one, two and three annotations. Notice clear segmentation improvements as the agent requests more annotations. Specifically, note how reference view *a* improves drastically with annotation *2* as the bed is visible in that view, and with annotation *3* where the drawer is seen. Also note how segmentation improves for reference view *b* after the door is seen in annotation *3*.

method in Table 3.5, i.e. the best results are again obtained by the threshold variant. The model with a learnt annotation strategy fails to converge to anything better than heuristic perception strategies. In fact, it converges to selecting `Collect` almost 40% of the time, which indicates a lack of movement for this variant.

In Table 3.3 we saw that a learnt exploration method with a heuristic annotation strategy yields worse results than a fully learnt model. Conversely, the results from Table 3.4 and Table 3.5 show that a heuristic exploration method using a learnt annotation strategy yields worse results than an entirely heuristic model. Together these results indicate that it is necessary to learn how to annotate and explore jointly to provide the best results, given comparable environment knowledge.

4.4 Pre-training the Segmentation Network

Recall that our semantic segmentation network is randomly initialized at the beginning of each episode. In this section we evaluate the effect of instead pre-training the segmentation network[1] on the 61 training houses using about 20,000 random views. In Table 3.6 we

[1]In this pre-training experiment, we use the same architecture and hyperparameters for the segmentation network as when it is trained and deployed in the embodied visual active learning task.

Table 3.5: Results for different model variants of the bounce method. We report the mean on the validation scenes. The threshold perception strategy – which is the one used in the main evaluations in §4.1 – yields the best results, but also uses the largest amount of annotations on average.

Variant	mIoU	Acc	# Ann	# Coll
Threshold perception	0.388	0.706	27.4	24.5
Learnt perception	0.375	0.699	14.6	98.8
Random perception	0.379	0.698	25.9	24.6

compare using this pre-trained segmentation network as initialization for the RL-agent with the case of random initialization. We also show results when not further fine-tuning the pre-trained segmentation network, i.e. when not performing any embodied visual active learning.

The weak result obtained when not fine-tuning (first row) indicates significant appearance differences between the houses. This is further suggested by the fact that the RL-agent gets a surprisingly modest boost from pre-training the segmentation network (third row vs second row). Note the different number of annotated views used here – the agent without pre-training uses only 16.4 views on average, while the other uses about $20,000 + 14.4$ annotated views, if we count all the images used for pre-training. Due to relatively marginal gains for a large number of annotated images, we decided to evaluate all agents without pre-training the segmentation network.

Table 3.6: Results for different training regimes for the semantic segmentation network. A pre-trained segmentation network generalizes poorly to unseen environments (first row), and there is relatively little gain for the RL-agent by having a pre-trained segmentation network (third row). Note that pre-training uses over 1000x more annotations compared to performing embodied active visual learning from scratch.

Variant	mIoU	Acc	# Ann	# Coll
Pre-train, no RL	0.208	0.549	20k	0.0
No pre-train, RL	0.427	0.732	16.4	16.4
Pre-train, RL	0.461	0.780	20k + 14.4	13.3

5 Conclusions

In this paper we have explored the *embodied visual active learning* task for semantic segmentation and developed a diverse set of methods, both pre-designed and learning-based, in order to address it. The agents can explore a 3d environment and improve the accuracy of their semantic segmentation networks by requesting annotations for informative viewpoints, propagating annotations via optical flow at no additional cost by moving in the neighborhood of those views, and self-training. We have introduced multiple baselines as well as a more sophisticated fully learnt model, each exposing different assumptions and knowledge of the environment. Through extensive experiments in the photorealistic Matterport3D environment we have thoroughly investigated the various methods and shown

that the fully learning-based method outperforms comparable non-learnt approaches, both in terms of accuracy and mIoU, while relying on fewer annotations.

Acknowledgments: This work was supported in part by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535 and PCCF-2016-0180, the EU Horizon 2020 Grant DE-ENIGMA, Swedish Foundation for Strategic Research (SSF) Smart Systems Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 128

[2] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Košecká, and Alexander C Berg. A dataset for developing and benchmarking active vision. In *ICRA*, pages 1378–1385. IEEE, 2017. 120, 122

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 128

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision*, 2017. 121, 128

[5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 122

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2017. 120

[7] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *ICLR*, 2019. 122

[8] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, volume 5, page 6, 2018. 122

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, pages 1–16, 2017. 121

[10] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, June 2019. 122

[11] Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, 2017. 123

[12] D. Feng, X. Wei, L. Rosenbaum, A. Maki, and K. Dietmayer. Deep active learning for efficient training of a lidar 3d object detector. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 667–674, 2019. 123

[13] Erik Gärtner, Aleksis Pirinen, and Cristian Sminchisescu. Deep reinforcement learning for active human pose estimation. In *AAAI*, pages 10835–10844, 2020. 123

[14] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, pages 2616–2625, 2017. 122

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 120

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 127

[17] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *ECCV*, pages 489–505. Springer, 2016. 123

[18] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, 2018. 123

[19] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848. 128

[20] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, pages 3813–3822, 2016. 123

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 128

[22] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*, 2017. 121

[23] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 120

[24] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed

reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062, 2018. 128

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 120, 126, 142

[26] Edwin Lughofer. Single-pass active learning with conflict and ignorance. *Evolving Systems*, 3(4):251–271, 2012. 123

[27] Dmytro Mishkin, Alexey Dosovitskiy, and Vladlen Koltun. Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*, 2019. 122

[28] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *ICRA*, pages 8846–8852. IEEE, 2019. 122

[29] Alejandro Pardo, Mengmeng Xu, Ali Thabet, Pablo Arbelaez, and Bernard Ghanem. Baod: Budget-aware object detection. *arXiv preprint arXiv:1904.05443*, 2019. 123

[30] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, volume 2017, 2017. 122

[31] Etienne Pot, Alexander Toshev, and Jana Kosecka. Self-supervisory signals for object discovery and detection. *arXiv preprint arXiv:1806.03370*, 2018. 123

[32] William Qi, Ravi Teja Mullapudi, Saurabh Gupta, and Deva Ramanan. Learning to move with affordance maps. In *ICLR*, 2020. 122

[33] Santhosh K Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. An exploration of embodied visual exploration. *arXiv preprint arXiv:2001.02192*, 2020. 122

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 120

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015. 120

[36] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 121

[37] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, pages 9339–9347, 2019. 121, 122, 128

[38] Alexander Sax, Bradley Emi, Amir R. Zamir, Leonidas J. Guibas, Silvio Savarese, and Jitendra Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. In *CoRL*, 2019. 122

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 128

[40] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009. 123

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 120

[42] Shuran Song, Linguang Zhang, and Jianxiong Xiao. Robot in a room: Toward perfect object recognition in closed environments. *CoRR, abs/1507.02703*, 2015. 123

[43] Shuran Song, Andy Zeng, Angel X Chang, Manolis Savva, Silvio Savarese, and Thomas Funkhouser. Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. In *CVPR*, pages 3847–3856, 2018. 123

[44] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018. 126

[45] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, pages 438–451. Springer, 2010. 126

[46] Kai Wang, Yimin Lin, Luowei Wang, Liming Han, Minjie Hua, Xiang Wang, Shiguo Lian, and Bill Huang. A unified framework for mutual improvement of slam and semantic segmentation. In *ICRA*, pages 5224–5230. IEEE, 2019. 123

[47] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied Question Answering in Photorealistic Environments with Point Cloud Perception. In *CVPR*, 2019. 122

[48] Mark Woodward and Chelsea Finn. Active one-shot learning. *arXiv preprint arXiv:1702.06559*, 2017. 123

[49] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *CVPR*. IEEE, 2018. 121

[50] Bo Xiong and Kristen Grauman. Snap angle prediction for 360 panoramas. In *ECCV*, pages 3–18, 2018. 123

[51] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, pages 146–151. IEEE, 1997. 125

[52] Hsuan-Kung Yang, Po-Han Chiang, Kuan-Wei Ho, Min-Fong Hong, and Chun-Yi Lee. Never forget: Balancing exploration and exploitation via learning optical flow. *arXiv preprint arXiv:1901.08486*, 2019. 122

[53] Jianwei Yang, Zhile Ren, Mingze Xu, Xinlei Chen, David J Crandall, Devi Parikh, and Dhruv Batra. Embodied amodal recognition: Learning to move to perceive objects. In *ICCV*, pages 2040–2050, 2019. 120, 122

[54] Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L. Berg, and Dhruv Batra. Multi-target embodied question answering. In *CVPR*, 2019. 122

[55] Lintao Zheng, Chenyang Zhu, Jiazhao Zhang, Hang Zhao, Hui Huang, Matthias Niessner, and Kai Xu. Active scene understanding via online semantic reconstruction. In *Computer Graphics Forum*, pages 103–114. Wiley Online Library, 2019. 122

[56] Fangwei Zhong, Sheng Wang, Ziqi Zhang, and Yizhou Wang. Detect-slam: Making object detection and slam mutually beneficial. In *WACV*, pages 1001–1010. IEEE, 2018. 123

[57] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017. 122

Supplementary Material

A Introduction

In this supplementary material we provide additional insights into our proposed models for the embodied active visual learning task. Details of the semantic segmentation network are given in §B, and of the RL-agent's policy network in §C. In §D we provide an algorithmic description of how the RL-agent operates within our task. In §E we compare with additional variants / hyperparameter configurations of the bounce method and conclude that the variants used in the main paper provide the best results in terms of mIoU.

B Semantic Segmentation Network

Fig. E.8 contains a schematic overview of the semantic segmentation network. We deliberately made it small so that it would be very quick to refine it with new data. It consists of 3 blocks of convolutional layers, each containing 3 convolutional layers with kernels of size 3×3. The first convolutional layer in each block uses a stride of 2, which halves the resolution. For each block the number of channels doubles. Multiple predictions are made using the final convolutional layers of each block. The multi-scale predictions are resized to the original image resolution using bilinear interpolation and are finally summed up, resulting in the final segmentation. The network resembles FCN [25] by predicting the semantic segmentation at multiple scales. In training we use a standard cross-entropy loss averaged over all pixels. The segmentation network is trained using stochastic gradient descent with learning rate 0.01, weight decay 10^{-5} and momentum 0.9.

C Policy Network of the RL-Agent

See Fig. E.9 for an overview of the policy network. The policy consists of two branches, where the first processes the image and segmentation inputs, and the second processes the extracted deep features.

D Pseudo Code

The full procedure of our RL-model for embodied visual active learning for semantic segmentation is described in Algorithm E.1. It describes among other things how the states and segmentation network are updated during an episode.

Table D.7: Results of different model variants of the bounce method. We report the mean on the validation scenes. Threshold perception is used in the main paper.

Variant	mIoU	Acc	# Ann	# Coll
Threshold perception	0.388	0.706	27.4	24.5
Learned perception	0.375	0.699	14.6	98.8
Random perception	0.379	0.698	25.9	24.6
Version 1	0.353	0.677	11.4	0.0
Version 2	0.372	0.695	9.8	9.6
Version 3	0.381	0.701	20.0	10.4

E Variants of Bounce

We tried multiple perception strategies of the bounce baseline on the validation set and present in Table D.7 the results of 3 different versions (in addition to those perception strategies described in §4.3):

- **Version 1.** Recall that the bounce method samples a random rotation after bouncing in a wall, and then begins moving in that direction. This version annotates prior to walking in a new direction (it issues no `Collect` actions).
- **Version 2.** Issues `Annotate` after rotating towards a new direction, and issues `Collect` four steps (1 m) after that.
- **Version 3.** Issues `Annotate` after rotating towards a new direction, and issues `Annotate` and `Collect` with 10% probability each when walking forward.

We see that the third version – with more frequent annotations and collects compared to versions 1 and 2 – obtains the best performance in terms of accuracy and mIoU on the validation set for 256-step episodes. However, it does not outperform the threshold strategy.

Algorithm E.1 Procedural code for the RL-agent in the embodied visual active learning for semantic segmentation task.

Initialize parameters of the segmentation network $\mathcal{S}$
Initialize location (x_1, y_1) and rotation ϕ_1 randomly and let $\boldsymbol{x}_1 = (x_1, y_1, \phi_1)$
Extract image $\boldsymbol{I}_1$ and receive associated annotation mask $\boldsymbol{A}_1$ at $\boldsymbol{x}_1$; initialize training set $\mathcal{D} = \{(\boldsymbol{I}_1, \boldsymbol{A}_1)\}$
Perform initial training of $\mathcal{S}$ on $\mathcal{D}$
Initialize propagated annotation $\boldsymbol{P}_1 = \boldsymbol{A}_1$
Compute segmentation $\boldsymbol{S}_1 = \mathcal{S}(\boldsymbol{I}_1)$ and deep features $\boldsymbol{F}_1$
Initialize agent state $s_1 = (\boldsymbol{I}_1, \boldsymbol{S}_1, \boldsymbol{P}_1, \boldsymbol{F}_1)$
for $t = 1, \ldots, T$ **do**
    Sample action $a_t \sim \pi_{\boldsymbol{\theta}}(\cdot|s_t)$
    if $a_t \in \{$`MoveForward`, `MoveLeft`, `MoveRight`, `RotateLeft`, `RotateRight`$\}$ **then**
        Set $\boldsymbol{x}_{t+1}$ according to movement
        Propagate annotation $\boldsymbol{P}_{t+1} = \text{flow}(\boldsymbol{P}_t)$
    else
        Set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$
        Set $\boldsymbol{P}_{t+1} = \boldsymbol{P}_t$
    end if
    Obtain view $\boldsymbol{I}_{t+1}$ associated to $\boldsymbol{x}_{t+1}$
    if $a_t =$ `Annotate` **then**
        Obtain annotation mask $\boldsymbol{A}_{t+1}$ at $\boldsymbol{x}_{t+1}$
        Update training set $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{I}_{t+1}, \boldsymbol{A}_{t+1})\}$
        Refine $\mathcal{S}$ on $\mathcal{D}$
        Reset propagated annotation $\boldsymbol{P}_{t+1} = \boldsymbol{A}_{t+1}$
    else if $a_t =$ `Collect` **then**
        Update training set $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{I}_{t+1}, \boldsymbol{P}_{t+1})\}$
        Refine $\mathcal{S}$ on $\mathcal{D}$
    end if
    Compute segmentation $\boldsymbol{S}_{t+1} = \mathcal{S}(\boldsymbol{I}_{t+1})$ and deep features $\boldsymbol{F}_{t+1}$
    Update agent state $s_{t+1} = (\boldsymbol{I}_{t+1}, \boldsymbol{S}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{F}_{t+1})$
end for
return $\mathcal{S}_\star$ (*trained segmentation network)*

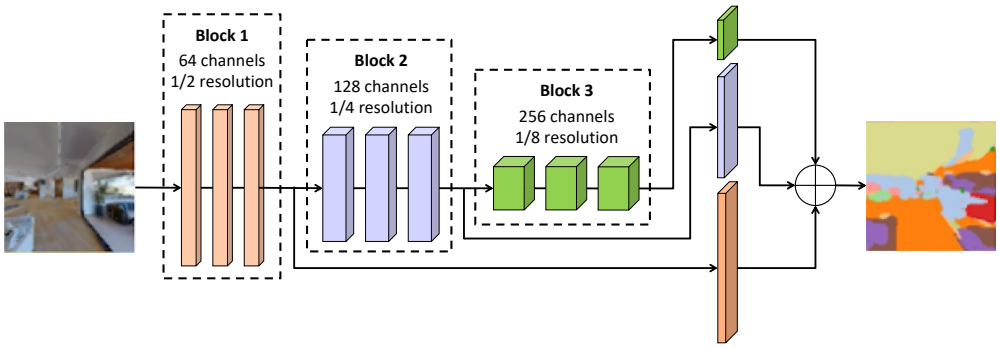


Figure E.8: Architecture of the deep network we use for semantic segmentation. The input image is processed sequentially through 3 blocks, each containing 3 convolutional layers. The first convolutional layer in each block uses a stride of 2, which halves the resolution for each block, and at the same time we double the number of channels. The semantic segmentation is predicted for multiple resolutions and are summed together to predict the semantic segmentation.

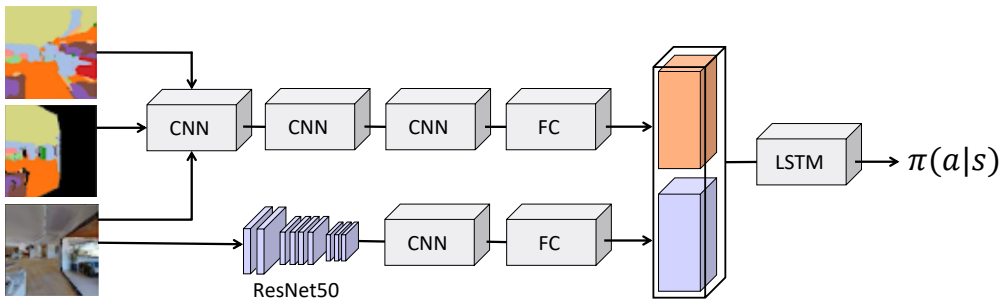


Figure E.9: The policy network architecture for the RL-agent. The network has three inputs: the current RGB image $\boldsymbol{I} \in \mathbb{R}^{127\times127\times3}$ (bottom), the current segmentation prediction $\boldsymbol{S} \in \mathbb{R}^{127\times127\times3}$ (top), and the current optical flow propagated ground truth segmentation $\boldsymbol{P} \in \mathbb{R}^{127\times127\times3}$ (middle). All three inputs are stacked depthwise and then processed by three convolutional layers and a fully connected layer (this processing subnetwork is called ϕ_{img} in §3.3 of the main paper). The first layer has 32 filters, kernel size 8×8, and stride 4. The second layer has 64 filters, kernel size 4×4, and stride 2. The third layer has 64 filters, kernel size 3×3, and stride 1. Finally, the fully connected layer has 512 outputs. In addition, the RGB image is passed through an image feature extractor (ResNet-50), called ϕ_{res} with output $\boldsymbol{F}_t$ in the main paper. The deep features $\boldsymbol{F}_t$ are subsequently passed through a convolutional layer with 128 filters, kernel size 2×2 and stride 2. Finally, these features are processed by a fully connected layer with 512 outputs. These two input branches are then concatenated and fed to an LSTM with 256 cells. The hidden state of the LSTM is finally passed to a softmax layer to produce the action distribution.

Paper IV

Erik Gärtner, Mykhaylo Andriluka, Hongyi Xu, Cristian Sminchisescu
Trajectory Optimization for Physics-Based Reconstruction of 3d Human Pose from Monocular Video
Computer Vision and Pattern Recognition (CVPR), New Orleans, USA, 2022

Trajectory Optimization for Physics-Based Reconstruction of 3d Human Pose from Monocular Video

Erik Gärtner[1,2] **Mykhaylo Andriluka**[1] **Hongyi Xu**[1]
Cristian Sminchisescu[1]

[1]Google Research
[2]Lund University

Abstract

We focus on the task of estimating a physically plausible articulated human motion from monocular video. Existing approaches that do not consider physics often produce temporally inconsistent output with motion artifacts, while state-of-the-art physics-based approaches have either been shown to work only in controlled laboratory conditions or consider simplified body-ground contact limited to feet. This paper explores how these shortcomings can be addressed by directly incorporating a fully-featured physics engine into the pose estimation process. Given an uncontrolled, real-world scene as input, our approach estimates the ground-plane location and the dimensions of the physical body model. It then recovers the physical motion by performing trajectory optimization. The advantage of our formulation is that it readily generalizes to a variety of scenes that might have diverse ground properties and supports any form of self-contact and contact between the articulated body and scene geometry. We show that our approach achieves competitive results with respect to existing physics-based methods on the Human3.6M benchmark [13], while being directly applicable without re-training to more complex dynamic motions from the AIST benchmark [36] and to uncontrolled internet videos.

1 Introduction

In this paper, we address the challenge of reconstructing physically plausible articulated 3d human motion from monocular video aiming to complement the recent methods [15, 16, 23, 42, 42, 49] that achieve increasingly more accurate 3d pose estimation results in terms of standard joint accuracy metrics, but still often produce reconstructions that are visually unnatural.

Our primary mechanism to achieve physical plausibility is to incorporate laws of physics into the pose estimation process. This naturally allows us to impose a variety of desirable properties on the estimated articulated motion, such as temporal consistency and balance in the presence of gravity. Perhaps one of the key challenges in using physics for pose estimation is the inherent complexity of adequately modeling the diverse physical phenomena that arise due to interactions of people with the scene. In the recent literature [29, 30, 31, 43] it is common to keep the physics model simple to enable efficient inference. For example, most of the recent approaches opt for using simplified contact models (considering foot contact only), ignore potential effects due to interaction with objects other than the ground-plane, and do not model more subtle physical effects such as sliding and rolling friction, or surfaces with varying degrees of softness. Clearly there are many real-world scenarios where leveraging a more feature-complete physical model is necessary. We explore physics-based articulated pose estimation using feature-complete physical simulation as a building block to address this shortcoming. The advantage of such an approach is that it allows our method to be readily applicable to a variety of motions and scenarios that have not previously been tackled in the literature (see fig. 4.1 and 4.2). Specifically, in contrast to [29, 30, 31, 43] our approach can reconstruct motions with any type of contact between the body and the ground plane (see fig. 4.1). Our approach can also model interaction with obstacles and supporting surfaces such as furniture and allows for varying the stiffness and damping of the ground-plane to represent special cases such as trampoline floor (see fig. 4.2). We rely on the Bullet [7] engine, which was previously used for simulating human motion in [25]. However, none of our implementation details are engine-specific, so we envision that the quality of our results might continue to improve with further development in physical simulation.

The main contribution of this paper is to experimentally evaluate the use of trajectory optimization for physics-based articulated motion estimation on laboratory and real-world data using a generic physics engine as a building block. We demonstrate that combining a feature-complete physics engine and trajectory optimization can reach competitive or better accuracy than state-of-the-art methods while being applicable to a large variety of scenes and motion types. Furthermore, to the best of our knowledge, we are the first to apply physics-based reconstruction to complex real-world motions such as the ones shown in fig. 4.1 and 4.2. As a second contribution, we generate technical insights such as demonstrating

that we can reach excellent alignment of estimated physical motion with 2d input images by automatically adapting the 3d model to the person in the image, and employing appropriate 2d alignment losses. This is in contrast to related work [29, 30, 31, 43] that typically does not report 2d alignment error and qualitatively may not achieve good 2d alignment of the physical model with the image. We also contribute to the understanding of the use of the

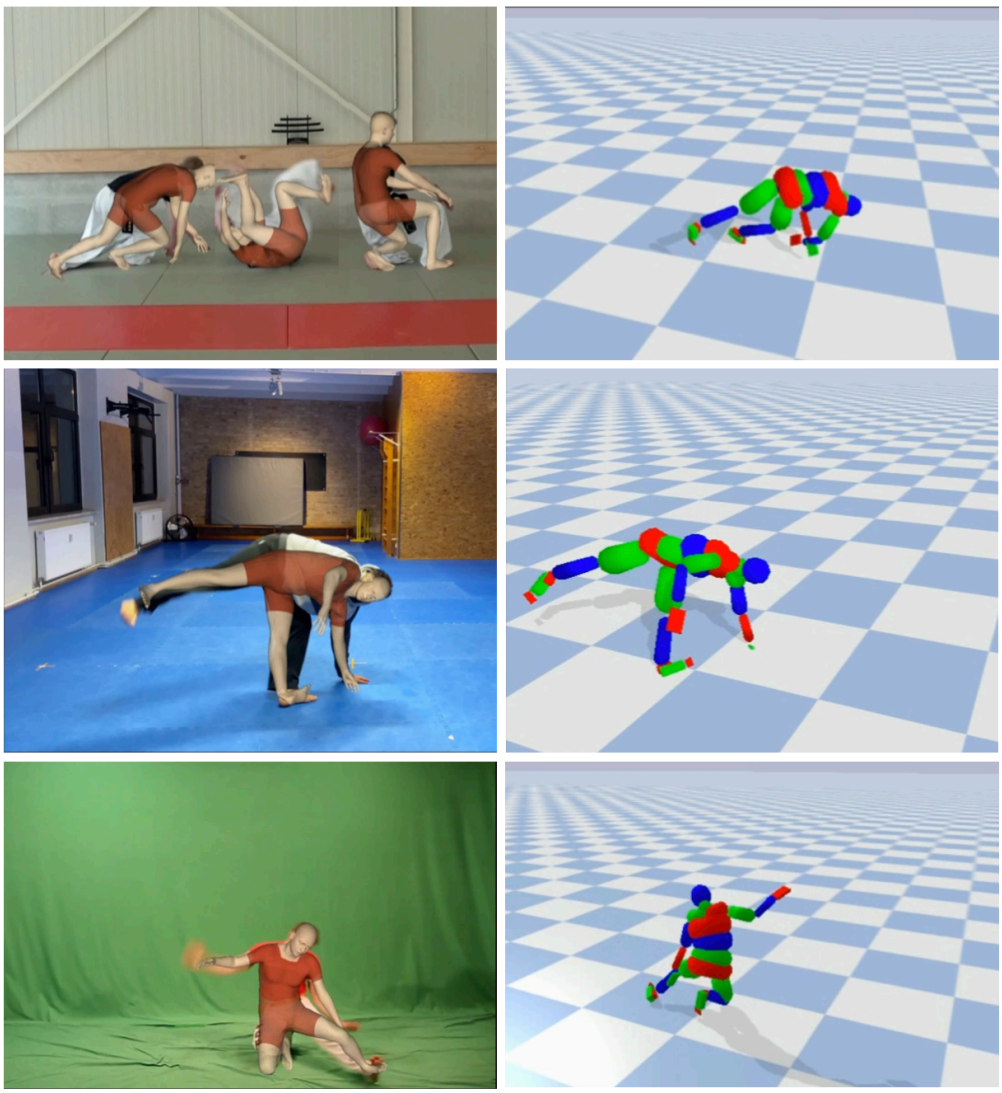


Figure 4.1: Example results of our approach on internet videos of dynamic motions. Note that our model can reconstruct physically plausible articulated 3d motion even in the presence of complex contact with the ground: full body contact (top row), feet and hands (middle), and feet and knee contacts (bottom).

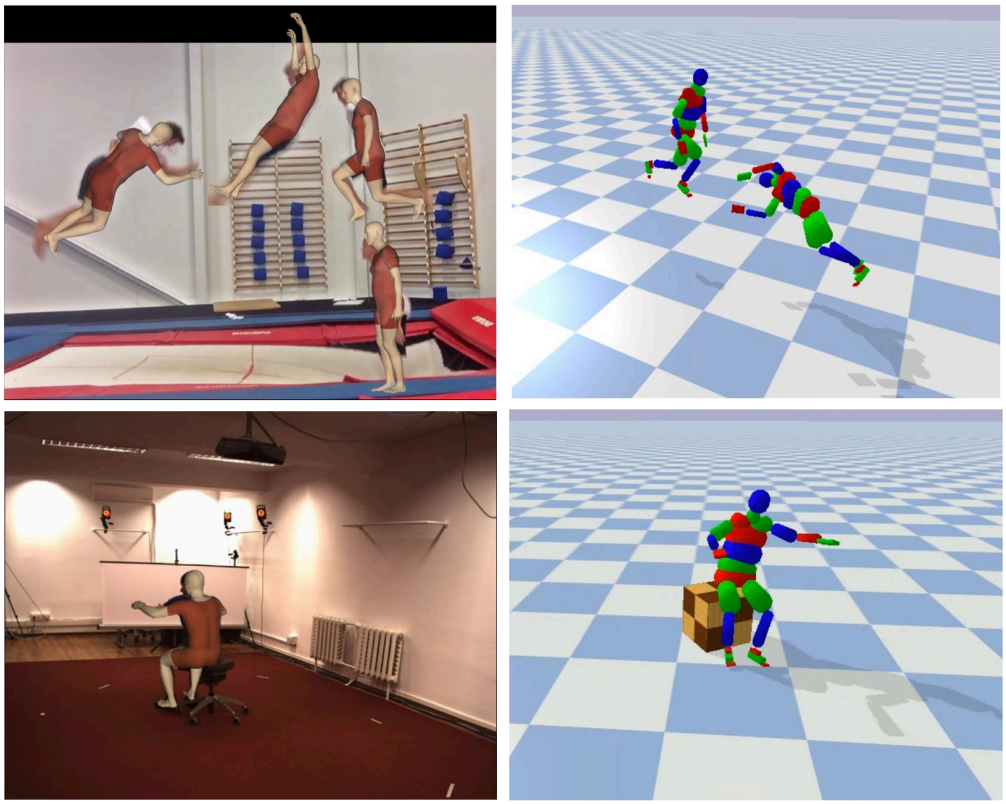


Figure 4.2: Examples results of our approach for scene with soft ground (top) and interaction with a chair (bottom).

residual root force control [45]. Such residual root force has been hypothesized as essential to bridge the simulation-to-reality gap and compensate for inaccuracies in the physical model. We experimentally demonstrate that the use of physically unrealistic residual force control might not be necessary, even in cases of complex and dynamic motions.

2 Related work

In the following, we first discuss recent literature on 3d human pose estimation that does not incorporate physical reasoning. We then review the related work on physics-based human modeling and compare our approach to other physics-based 3d pose estimation approaches.

3d pose estimation without physics. State-of-the-art methods are highly effective in estimating 2d and 3d people poses in images [5, 15, 47], and recent work has been able to

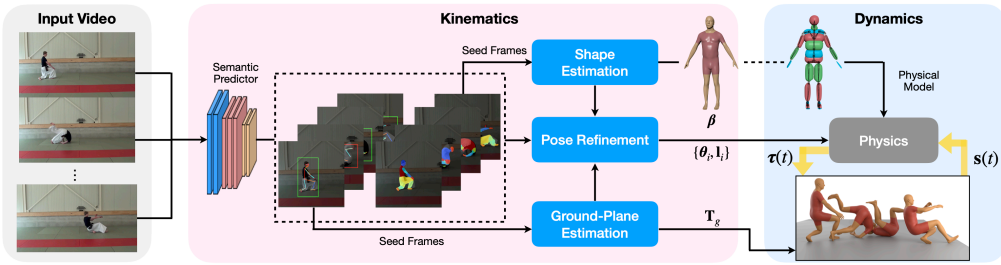


Figure 4.3: Overview. Given a monocular video of a human motion, we estimate the parameters of a physical human model and motor control trajectories $\boldsymbol{\tau}(t)$ such that the physically simulated human motion aligns with the video. We first use an inference network that predicts 2d landmarks $\mathbf{l}_i$ and body semantic segmentation masks from the video frames. From n seed frames we estimate a time-consistent human shape $\boldsymbol{\beta}$ and the ground-plane location $\mathbf{T}_g$. These are then kept fixed during a per-frame pose refinement step which provides the 3d kinematic initialization $\{\boldsymbol{\theta}_i\}$ to the physics optimization. The dynamics stage creates a physical model that mirrors the statistical shape model with appropriate shape and mass. Our dynamics optimization improves 3d motion estimation taking into account 3d kinematics, 2d landmarks and physical constraints. We refer to §3 for details.

extend this progress to 3d pose estimation in video [16, 23, 42]. The key elements driving the performance of these methods is the ability to estimate data-driven priors on articulated 3d poses [16, 48] and learn sophisticated CNN-based representations from large corpora of annotated training images [13, 14, 21, 37]. As such, these methods perform very well on common poses but are still challenged by rare poses. Occlusions, difficult imaging conditions, and dynamic motions (e.g. athletics) remain a challenge as these are highly diverse and hard to represent in the training set. As pointed out in [29], even for common poses state-of-the-art methods still often generate reconstructions prone to artifacts such as floating, footskating, and non-physical leaning. We aim to complement the statistical models used in the state-of-the-art approaches by incorporating laws of physics into the inference process and thus adding a component that is universally applicable to any human motion regardless of the statistics of the training or test set.

In parallel with recent progress in pose estimation, we now have accurate statistical shape and pose models [3, 20, 44]. These body models are typically estimated from thousands of scans of people and can generate shape deformations for a given pose. In this paper, we take advantage of these improvements and use a statistical body shape model [44] to define the dimensions of our physical model and derive the mass from the volume of the body parts.

Physics-based human motion modeling. Human motion modeling has been a subject of active research in computer graphics [2, 17], robotics [8] and reinforcement learning [11, 25, 41] literature. With a few exceptions, most of the models in these domains have been constructed and evaluated using the motion capture data [2]. Some work such as [26] use images as input, aiming to train motion controllers for a simulated character capable of performing the observed motion under various perturbations. That work focuses on

Table 4.1: Comparison of recent physics-based articulated pose estimation approaches. "Contact model" indicates what contact points between body and ground are considered, "Residual force" indicates if the physical model allows application of additional external force to move the person (see [45]), "Body model" specifies if approach adapts the physical model to person in the video, and "Real-world videos" specifies if approach has also been evaluated on real-world videos or only on videos captured in laboratory conditions.

	Contact model	Real-time	Physics implementation	Residual force	Body model	Real-world videos
Li *et al.* [19]	body joints	no	custom	no	fixed	yes
Rempe *et al.* [29]	feet	no	custom	no	fixed	yes
PhysCap [30]	feet	yes	custom	yes	fixed	yes
Shimada *et al.* [31]	feet	yes	custom	yes	fixed	yes
SimPoE [46]	full body	yes	MuJoCo [35]	yes	adapt.	no
Xie *et al.* [43]	feet	no	custom	no	adapt.	no
DiffPhy [9]	full body	no	TDS [12]	no	adapt.	yes
Ours	full body	no	Bullet [7]	no	adapt.	yes

training motion controllers for a fixed character, whereas our focus is on estimating the motion of the subject observed in the image. Furthermore, the character's size, shape, and mass are independent of the observed subject. [17] propose a realistic human model that directly represents muscle activations and a method to learn control policies for it. [40] generate motions for a variety of character sizes and learn control policies that adapt to each size. [17, 40] and similar results in the graphics literature do not demonstrate this for characters observed in real images and do not deal with challenges of jointly estimating physical motion and coping with ambiguity in image measurements or the 2d to 3d lifting process [33].

Physics-based 3d pose estimation. Physics-based human pose estimation has a long tradition in computer vision [4, 22, 38]. Early works such as [38] already incorporated physical simulation as prior for 3d pose tracking but only considered simple motions such as walking and mostly evaluated in the multi-view setting in the controlled laboratory conditions. We list some of the properties of the recent works in tab. 4.1. [19] demonstrate joint physics-based estimation of human motion and interaction with various tool-like objects. [29] proposes a formulation that simplifies physics-based reasoning to feet and torso only, and infers positions of other body parts through inverse kinematics, whereas [19] jointly model all body parts and also include forces due to interaction with an object. [30, 31] use a specialized physics-based formulation that solves for ground-reaction forces given pre-detected foot contacts and kinematic estimates. In contrast, we do not assume that contacts can be detected a-priori, and in our approach, we estimate these as part of the physical inference. Hence we are not limited to predefined types of contact as [19, 29, 30, 31] or their accurate a-priori estimates. We show that we quantitatively improve over [29, 30], and qualitatively show how we can address more difficult in-the-wild internet videos of activities such as somersaults and sports, which would be difficult to reconstruct using previous methods. Our work is conceptually similar to SimPoE [46] in that both works use physics simulation. In contrast to SimPoE, we introduce a complete pipeline that is applicable to real-world videos, whereas SimPoE has been tested only in laboratory conditions and re-

quires a calibrated camera. Furthermore, since SimPoE relies on reinforcement learning to train dataset-specific neural network models to control the simulated body, it is not clear how well SimPoE would generalize to variable motions present in real-world videos. One clear advantage of the SimPoE approach is its fast execution at test time, which comes at the cost of lengthy pre-training. Our approach is related to the approach of [43] which also estimates 3d human motion by minimizing an objective function that incorporates physics constraints. Perhaps the most significant differences to [43] are that (1) we use the full-featured physics model whereas they consider simplified physical model, (2) their model considers physics-based loss, but the output is not required to correspond to actual physical motion, and (3) they do not discuss performance of the approach on real-world data. The advantage of [43] is that they define a differentiable model that can be readily optimized with gradient descent. Finally, the concurrent work [9] tackles physics-based human pose reconstruction by minimizing a loss using a differentiable physics simulator given estimated kinematics.

3 Our approach

We present an overview of our approach in Fig. 4.3. Given monocular video as input, we first reconstruct the initial kinematic 3d pose trajectory using a kinematic approach of [49] and use it to estimate body shape and the position of the ground plane relative to the camera. Subsequently, we instantiate a physical person model with body dimensions and weight that match the estimated body shape. Next, we formulate an objective function that measures the similarity between the motion of the physical model and image measurements and includes regularization terms that encourage plausible human poses and penalize jittery motions. Finally, we reconstruct the physical motion by minimizing this objective function with respect to the joint torque trajectories. To realize the physical motion, we rely on the implementation of rigid body dynamics available in Bullet [7].

3.1 Body model and control

We model the human body as rigid geometric primitives connected by joints. Our model consists of 26 capsules and has 16 3d body joints for a total of 48 degrees of freedom. We rely on a statistical model of human shape [44] to instantiate our model for a variety of human body types. To that end, given the 3d mesh representing the body shape, we estimate dimensions of the geometric primitives to approximate the mesh following the approach of [2]. We then compute the mass and inertia of each primitive based on its volume and estimate the mass based on an anatomical weight distribution [28] from the statistical human shape dataset CAESAR [27].

We do not model body muscle explicitly and instead actuate the model by directly applying the torque at the body joints. We denote the vector of torques applied at time t as $\boldsymbol{\tau}_t$, the angular position, and velocity of each joint at time t as $\mathbf{q}_t$ and $\dot{\mathbf{q}}_t$, and the set of 3d Cartesian coordinates of each joint at time t as $\mathbf{x}_t$. Similarly to [24], we control the motion of the physical model by introducing a sequence of control targets $\hat{\mathbf{q}}_{1:T} = \{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_t\}$ which are used to derive the torques via a control loop. The body motion in our model is then specified by the initial body state $\mathbf{s}_0 = (\mathbf{q}_0, \dot{\mathbf{q}}_0)$, the world geometry $\mathbf{G}$ specifying the position and orientation of the ground plane, the control trajectory for each joint $\hat{\mathbf{q}}_{1:T}$ and the corresponding control rule. We assume the initial acceleration to be $\mathbf{0}$. To implement the control loop we rely on the articulated islands algorithm[1] (AIA) [34] that incorporates motor control targets as constraints in the linear complementarity problem (LCP) (*cf.* (6.3) a, b in [34]) alongside contact constraints. AIA enables stable simulation already at 100 Hz compared to 1000-2000 Hz for PD control used in [2, 9, 24].

3.2 Physics-based articulated motion estimation

Our approach to the task of physical motion estimation is generally similar to other trajectory and spacetime optimization approaches in the literature [1, 2, 39]. We perform optimization over a sequence of overlapping temporal windows, initializing the start of each subsequent window with the preceding state in the previous window. To reduce the dimensionality of the search space, we use cubic B-spline interpolation to represent the control target $\hat{\mathbf{q}}_{1:T}$ and perform optimization over the spline coefficients [6]. Given the objective function L introduced in §3.3 we aim to find the optimal motion by minimizing L with respect to the spline coefficients of the control trajectory $\hat{\mathbf{q}}_{1:T}$. We initialize the control trajectory with the kinematic estimates of the body joints (see §3.4). The initial state is initialized from the corresponding kinematic estimate. We use the finite difference computed on the kinematic motion to estimate the initial velocity. As in [1, 2] we minimize the objective function with the evolutionary optimization approach CMA-ES [10] since our simulation environment does not support differentiation with respect to the dynamics variables. We generally observe convergence with CMA-ES after 2000 iterations per window with 100 samples per iteration. The inference takes $20 - 30$ minutes when evaluating 100 samples in parallel.

3.3 Objective functions

We use a composite objective function given by a weighted combination of several components.

[1] "POSITION_CONTROL" mode in Bullet.

3d pose. To encourage reconstructed physical motion to be close to the estimated kinematic 3d poses $\mathbf{q}^{\mathrm{k}}_{1:T}$ we use the following objective functions

$$L_{COM}(\hat{\mathbf{q}}_{1:T}) = \sum_t (\|\mathbf{c}_t - \mathbf{c}^{\mathrm{k}}_t\|_2^2 + \|\dot{\mathbf{c}}_t - \dot{\mathbf{c}}^{\mathrm{k}}_t\|_2^2) \tag{4.1}$$

$$L_{pose} = \sum_t \sum_{j \in \mathbf{J}} \arccos(|\langle \mathbf{q}_{tj}, \mathbf{q}^{\mathrm{k}}_{tj} \rangle|) \tag{4.2}$$

where $\mathbf{c}_t$ and $\mathbf{c}^{\mathrm{k}}_t$ denote the position of the center of mass at time t in the reconstructed motion and kinematic estimate. L_{pose} measures the angle between observed joint angles and their kinematic estimates and the summation (4.2) is over the set J of all body joints including the base joint which defines the global orientation of the body.

2d re-projection. To encourage alignment of 3d motion with image observations, we use a set of $N = 28$ landmark points that include the main body joints, eyes, ears, nose, fingers, and endpoints of the feet. Let $\mathbf{l}_t$ denote the positions of 3d landmarks on the human body at time t, $\mathbf{C}$ be the camera projection matrix that maps world points into the image via perspective projection, $\mathbf{l}^d_t$ be the vector of landmark detections by the CNN-detector, and $\mathbf{s}_t$ the corresponding detection score vector. The 2d landmark re-projection loss is then defined as

$$L_{2d} = \sum_t \sum_n \mathbf{s}_{tn} \|\mathbf{C}\mathbf{l}_{tn} - \mathbf{l}^d_{tn}\|_2. \tag{4.3}$$

See §3.4 for details on estimating the 2d landmarks.

Regularization. We include several regularizers into our objective function. Firstly, we use the normalizing flow prior on human poses introduced in [48] which penalize unnatural poses. The loss is given by

$$L_{nf} = \sum_t \|\mathbf{z}(\mathbf{q}_t)\|_2, \tag{4.4}$$

where $\mathbf{z}(\mathbf{q}_t)$ is the latent code corresponding to the body pose $\mathbf{q}_t$. To discourage jittery motions we a add total variation loss on the acceleration of joints

$$L_{TV} = \frac{1}{J} \sum_t \sum_j \|\ddot{\mathbf{x}}_{tj} - \ddot{\mathbf{x}}_{t-1,j}\|_1 \tag{4.5}$$

Finally, we include a L_{lim} term that adds exponential penalty on deviations from anthropomorphic joint limits. The overall objective L used in physics-based motion estimation is given by the weighted sum of (4.1- 4.5) and of the term L_{lim}. See the supplemental material for details.

Table 4.2: Ablation of kinematics improvements on HUND on a validation subset of Human3.6M. *+S* indicates time-consistent body shape, *+O* indicates additional non-linear optimization, *+G* using ground-plane constraints, and *+T* temporal smoothness constraints.

Model	MPJPE-G	MPJPE	MPJPE-PA
HUND [49]	239	116	72
+ S	233	110	71
+ SO	178	85	62
+ SO + G	148	84	63
+ SO + T	186	85	61
+ SO + GT	135	80	58

3.4 Kinematic 3d pose and shape estimation

In this section, we describe our approach to extracting 2d and 3d evidence from the input video sequence.

Body shape. Given the input sequence, we proceed first to extract initial per-frame kinematic estimates of the 3d pose and shape using HUND [49]. As part of its optimization pipeline HUND also recovers the camera intrinsics $\mathbf{c}$ and estimates the positions of 2d landmarks, which we use in the 2d re-projection objective in (4.3). HUND is designed to work on single images, so our initial shape and pose estimates are not temporally consistent. Therefore, to improve the quality of kinematic 3d pose initialization, we extend HUND to pose estimation in video. We evaluate the additional steps introduced in this section in the experiments shown in tab. 4.2 using a validation set of 20 sequences from Human3.6M dataset. In our adaptation, we do not re-train the HUND neural network predictor and instead, directly minimize the HUND loss functions with BFGS. As a first step, we re-estimate the shape jointly over multiple video frames. To keep optimization tractable, we first jointly estimate shape and pose over a subset of $n = 5$ seed frames and then re-estimate the pose in all video frames keeping the updated shape fixed. The seed frames are selected by the highest average 2d keypoint confidence score. We refer to the HUND approach with re-estimated shape as *HUND+S* and to our approach where we subsequently also re-estimate the pose as *HUND+SO*. In tab. 4.2 we show results for both variants. Note that *HUND+SO* improves considerably compared to the original HUND results.

Ground plane. We define the location of the ground plane by the homogeneous transformation $\mathbf{T}_g$ that maps from the HUND coordinates to the canonical coordinate system in which the ground plane is passing through the origin, and its normal is given by the "y" axis. Let $\mathbf{M}^t$ be a subset of points on the body mesh at frame t. The signed distance from the mesh points to the ground plane is given by $D(\mathbf{M}^t) = \mathbf{T}_g\mathbf{M}^t\mathbf{e}_y$, where $\mathbf{e}_y = [0, 1, 0, 0]^T$ is the unit vector of the "y" axis in homogeneous coordinates. To estimate the transforma-

tion $\mathbf{T}_g$ we introduce an objective function

$$L_{gp}(\mathbf{T}_g, \mathbf{M}) = \sum_t \| \min(\delta, L_k(D(\mathbf{M}^t)))\|_2, \tag{4.6}$$

where $L_k(D^t)$ corresponds to the smallest $k = 20$ signed distances in D^t. This objective favors $\mathbf{T}_g$ that places body mesh in contact with the ground without making preference for a specific contact points. This objective is also robust to cases when person is in the air by clipping the distance at δ, which we set to 0.2m in the experiments in this paper. We recover $\mathbf{T}_g$ by minimizing

$$\begin{aligned} L_{gp}(\mathbf{T}_g) =& L_{gp}(\mathbf{T}_g, \mathbf{M}_l) + L_{gp}(\mathbf{T}_g, \mathbf{M}_r) \\ &+ 2L_{gp}(\mathbf{T}_g, \mathbf{M}_b), \end{aligned} \tag{4.7}$$

where $\mathbf{M}_l$, $\mathbf{M}_r$ and $\mathbf{M}_b$ are the meshes of the left foot, right foot and whole body respectively. This biases the ground plane to have contact with the feet, but is still robust to cases when person is jumping or touching the ground with other body parts (e.g. as in the case of a somersault).

3d pose. In the final step, we re-estimate the poses in all frames using the estimated shape and ground plane while adding the temporal consistency objective

$$L_{temp} = \sum_t \|\mathbf{M}^t - \mathbf{M}^{t-1}\|_2 + \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\|_2, \tag{4.8}$$

where $\mathbf{M}^t$ is a body mesh and $\boldsymbol{\theta}_t$ is a HUND body pose vector in frame t. To enforce ground plane constraints we use (4.6), but now keep $\mathbf{T}_g$ fixed and optimize with respect to body pose. In the experiments in tab. 4.2 we refer to the variant of our approach that uses temporal constraints in (4.8) as *HUND+SO+T* and to the full kinematic optimization that uses both temporal and ground plane constraints as *HUND+SO+GT*. Tab. 4.2 demonstrates that both temporal and ground-truth constraints considerably improve the accuracy of kinematic 3d pose estimation. Even so, the results of our best variant *HUND+SO+GT* still contain artifacts such as motion jitter and footskating, which are substantially reduced by the dynamical model (see tab. 4.3).

4 Experimental results

Datasets. We evaluate our method on three human motion datasets: Human3.6M [13], HumanEva-I [32] and AIST [36]. In addition, we qualitatively evaluate on our own "in-the-wild" internet videos. To compare different variants of our approach in tab. 4.2 and tab. 4.3 we use a validation set composed of 20 short 100-frame sequences from the Human3.6M dataset. We use the same subset of full-length sequences as proposed in [43] for

the main evaluation in tab. 4.4. We use a preprocessed version of the AIST dataset [36] from [18] which contains pseudo 3d body pose ground-truth obtained through multi-view reconstruction. For our experiments, we select a subset of fifteen videos featuring diverse dances of single subjects. For the evaluation on HumanEva-I, we follow the protocol defined in [29] and evaluate on the walking motions from the validation split of the dataset using images from the first camera. We assume known camera extrinsic parameters in the Human3.6M experiments and estimate them for other datasets. In order to speed up the computation of the long sequences of Human3.6M in Table 4.4 we compute all temporal windows in parallel and join them together in post-processing.

We report results using mean global per-joint position error (mm) overall joints (MPJPE-G), as well as translation aligned (MPJPE) and Procrustes aligned (MPJPE-PA) error metrics. Note that to score on the MPJPE-G metric an approach should be able to both estimate the articulated pose and correctly track the global position of the person in world coordinates. In addition to standard evaluation metrics, we implement the foot skate and floating metrics similar to those introduced in [29] but detect contacts using a threshold rather than through contact annotation. Finally, we report image alignment (MPJPE-2d) and 3d joint velocity error in m/s. See supplementary for further details.

Table 4.3: Ablation experiments of the dynamics model on a validation set of 20 sequences from the Human3.6M dataset.

Model	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d	Velocity	Footskate (%)	Float (%)
HUND+SO	178	85	62	12	1.3	25	40
HUND+SO + Dynamics	167	87	62	12	0.45	7	1
HUND+SO+GT	135	80	58	12	0.58	64	0
HUND+SO+GT + Dynamics	**132**	80	**57**	**11**	**0.27**	8	0
HUND+SO+GT + Dynamics							
w/o 2d re-projection, (4.3)	154	104	68	17	0.32	-	-
w/o 3d joints, (4.2)	134	84	60	11	0.27	-	-
w/o COM, (4.1)	149	81	57	11	0.31	-	-
w/o COM and 3d joints, (4.1, 4.2)	151	85	59	11	0.33	-	-
w/o pose prior, (4.4)	138	80	57	11	0.24	-	-

Analysis of model components. In tab. 4.3 we present ablation results of our approach. Our full dynamical model uses kinematic inputs obtained with *HUND+SO+GT* introduced in §3.4 and is denoted as *HUND+SO+GT + Dynamics*. Our dynamical model performs comparably or slightly better compared to *HUND+SO+GT* on joint localization metrics (e.g. MPJPE-G improves slightly from 135 to 132 mm) but greatly reduces motion artifacts. The percentage of frames with footskate is reduced from 64 to 8 and error in velocity from 0.58 to 0.27 m/s. We also evaluate a dynamic model based on a simpler kinematic variant *HUND+SO* that does not incorporate ground-plane and temporal constraints when re-estimating poses from video. For *HUND+SO*, the inference with dynamics similarly improves perceptual metrics considerably. Note that *HUND+SO* produces output that suffers from both footskating (25% of frames) and floating (40% of frames). Adding ground-plane constraints in (*cf.* (4.6)) removes floating artifacts in *HUND+SO+GT*, but

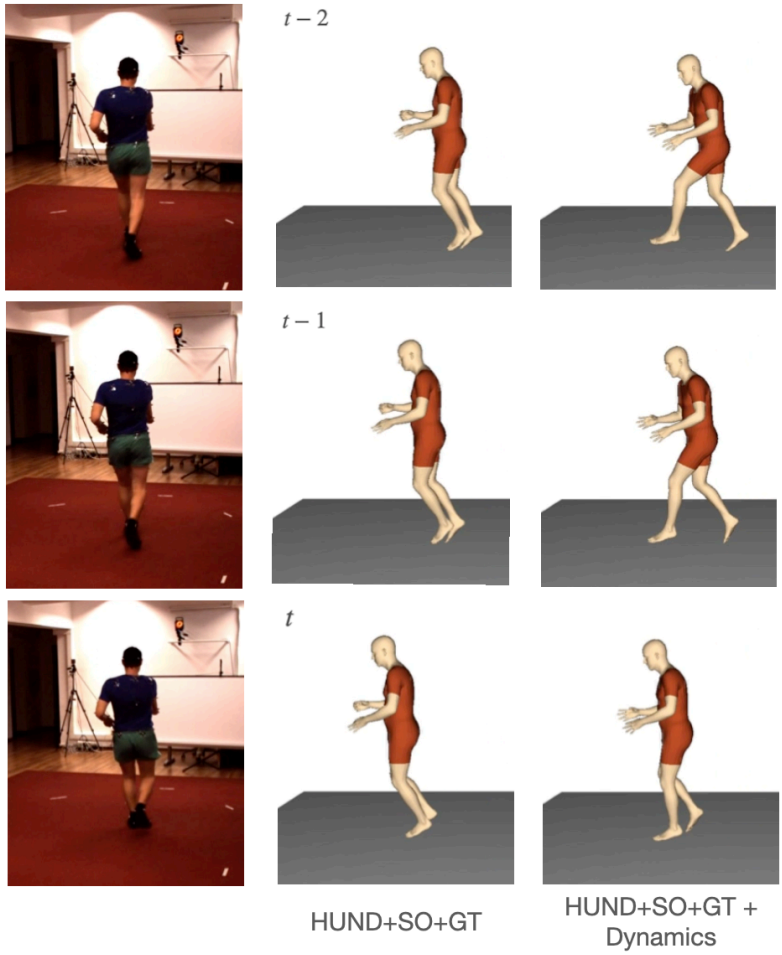


Figure 4.4: Qualitative results on the Human3.6M dataset. Note how the dynamical model (right) recovers plausible locomotion.

the output still suffers from footskating (64% of the frames). Dynamical inference helps to substantially reduce both types of artifacts both for *HUND+SO* and *HUND+SO+GT*. In fig. 4.4 we show example output of *HUND+SO+GT* + *Dynamics* and compare it to *HUND+SO+GT* which it uses for initialization. Note that for *HUND+SO+GT* the person in the output appears to move forward by floating in the air, whereas our dynamics approach infers plausible 3d poses consistent with the subject's global motion. In the bottom part of tab. 4.3 we report results for our full model *HUND+SO+GT* + *Dynamics* while ablating components of the objective function (*cf.* §3.3). We observe that all components of the objective function contribute to the overall accuracy. The most important com-

Table 4.4: Quantitative results of our models compared to prior work on Human3.6M [13], HumanEva-I [32] and a subset of AIST [18, 36].

Dataset	Model	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d	Velocity	Footskate (%)
Human3.6M	VIBE [16]	208	69	44	16	0.32	27
	PhysCap [30]	-	97	65	-	-	-
	SimPoE [46]	-	**57**	**42**	-	-	
	Shimada *et al.* [31]	-	77	58	-	-	-
	Xie *et al.* [43] (Kinematics)	-	74	-	-	-	-
	Xie *et al.* [43] (Dynamics)	-	68	-	-	-	-
	Ours: HUND+SO+GT	145	83	56	14	0.46	48
	Ours: HUND+SO+GT + Dynamics	**143**	84	56	**13**	**0.24**	4
HumanEva-I	Rempe et al. [29] (Kinematics)	408	-	-	-	-	-
	Rempe et al. [29] (Dynamics)	422	-	-	-	-	-
	Ours: HUND+SO+GT	208	**90**	76	14	0.51	40
	Ours: HUND+SO+GT + Dynamics	**196**	91	74	14	**0.27**	4
AIST	Ours: HUND+SO+GT	156	**107**	**67**	**10**	0.59	51
	Ours: HUND+SO+GT + Dynamics	**154**	113	69	13	**0.41**	4

ponents are the 2d re-projection (*cf.* (4.3)) and difference in COM position (*cf.* (4.1)). Without these, the MPJPE-G increases from 132 to 154 and 151 mm, respectively. Excluding the 3d joints component leads to only a small loss of accuracy from 132 to 134 mm.

Comparison to state-of-the-art. In tab. 4.4 we present the results of our full model on the Human3.6M, HumanEva-I, and AIST datasets. We compare to VIBE [16] using the publicly available implementation by the authors and use the evaluation results of other approaches as reported in the original publications. Since VIBE generates only root-relative pose estimates, we use a similar technique as proposed in PhysCap [30] and estimate the global position and orientation by minimizing the 2d joint reprojection error. On the Human3.6M benchmark, our approach improves over VIBE and our own *HUND+SO+GT* in terms of joint accuracy and perceptual metrics. Compared to VIBE, the MPJPE-G improves from 208 to 143 mm, MPJPE-2d improves from 16 to 13 px, and the percentage of footskating frames are reduced from 27% to 4%. Interestingly our approach achieves the best MPJPE-PA overall physics-based approaches except the pretrained SimPoE, but reaches somewhat higher MPJPE compared to [31] and fairly recent work of [43] (82 mm vs 68 mm for [43] and 77 mm for [31]). Note that [43] start with a stronger kinematic baseline (74 mm MPJPE) and that the performance of other approaches might improve as well given such better kinematic initialization. Furthermore, our dynamics approach improves over the results of [29] on HumanEva-I and achieves significantly better MPJPE-G compared to *HUND+SO+GT*. On the AIST dataset, dynamics similarly improves in terms of MPJPE-G, footskating, and velocity compared to our kinematic initialization.

Results on real-world internet video. We show example results of our approach on the AIST dataset [36] in fig. 4.5 and on the real-world internet videos in fig. 4.1, 4.2 and 4.6. To obtain the results with a soft floor shown in fig. 4.2 we manually modify the stiffness and damping floor parameters to mimic the trampoline behavior. The sequence with the

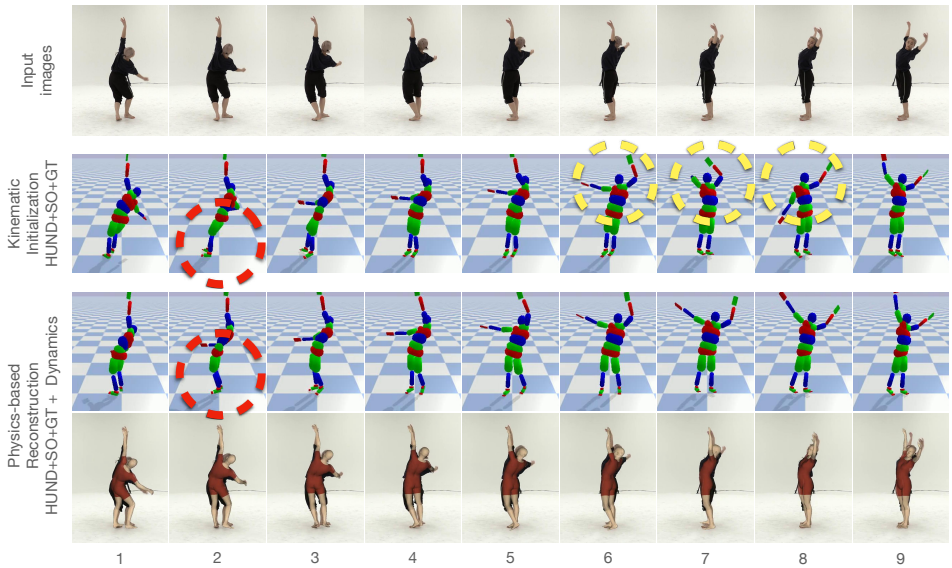


Figure 4.5: Example result on AIST [36]. The kinematic initialization produces poses that are unstable in the presence of gravity (red circle) or poses that are temporally inconsistent (yellow circles). Our physics-based approach corrects both errors.

chair from the Human3.6M dataset shown in Fig. 4.2 (bottom) is generated by manually adding a chair to the scene since our approach does not perform reasoning about scene objects.

In Fig. 4.5 we qualitatively compare the output of our full system with physics to our best kinematic approach *HUND+SO+GT*. We strongly encourage the reader to watch the video in supplemental material[1] to appreciate the differences between the two approaches and to see the qualitative comparison to VIBE [16]. We observe that our physics approach is often able to correct out-of-balance poses produced by *HUND+SO+GT* (e.g. second frame in fig. 4.5) and substantially improves temporal coherence of the reconstruction. Note that typically both *HUND+SO+GT* and our physics-based approach produce outputs that match 2d observations, but the physics-based approach estimates 3d pose more accurately. For example, in the first sequence in fig. 4.6 the physics-based model infers the pose that enables the person to jump in subsequent frames, whereas *HUND+SO+GT* places the left leg at an angle that would make the jump impossible. Note that the output of the physics-based approach can deviate significantly from the kinematic initialization (fig. 4.7 and second example in fig. 4.6. This is particularly prominent in the fig. 4.7 where we show example result on a difficult sequence where 2d keypoint estimation fails to localize the legs in several frames due to occlusion by the clothing. Note that in this example our full model

[1]See tiny.cc/traj-opt.

with dynamics is able to generate reasonable sequence of 3d poses despite multiple failures in the kinematic initialization.

Failure cases of our approach. We show a few characteristic examples of the failure cases of our approach in fig. 4.8. Note that our physics-based reconstruction depends on the kinematic 3d pose estimation for initialization and also uses it in one of the components of the loss (*cf.* eq. 4.2). Therefore our physics-based approach is likely to fail when kinematic reconstruction is grossly incorrect (see fig. 4.8 (b)) or when it fails to estimate position of the limb important to maintain the overall pose (see fig. 4.8 (a)). Our physics-based model might also fail when the estimate of the ground-plane with respect to the camera is inaccurate. Note how in fig. 4.8 (c) the kinematic estimate positions the standing person at an angle to the true ground-plane normal vector (red arrow). As a result in this example the physics-based reconstruction tilts the person at the torso to maintain stable pose given the incorrect gravity vector (see the two bottom rows in fig. 4.8 (c)).

5 Conclusion

In this paper, we have proposed a physics-based approach to 3d articulated video reconstruction of humans. By closely combining kinematic and dynamic constraints within an optimization process that is contact, mass, and inertia aware, with values informed by body shape estimates, we are able to improve the physical plausibility and reduce reconstruction artifacts compared to purely kinematic approaches. One of the primary goals of our work has been to demonstrate the advantages of incorporating an expressive physics model into the 3d pose estimation pipeline. Clearly, such a model makes inference more involved compared to specialized physics-based approaches such as [30, 43], but with the added benefit of being more capable and general.

Ethical considerations. This work aims to improve the quality of human pose reconstruction through the inclusion of physical constraints. We believe that the level of detail in our physical model limits its applications in tasks such as person identification or surveillance. The same limitation also prevents its use in the generation of e.g. deepfakes, particularly as the model lacks a photorealistic appearance. We believe our model is inclusive towards and supports a variety of different body shapes and sizes. While we do not study this in the paper, we consider it important future work.

Acknowledgements. We would like to thank Erwin Coumans for his help with the project, as well as the supportive anonymous reviewers for their insightful comments.

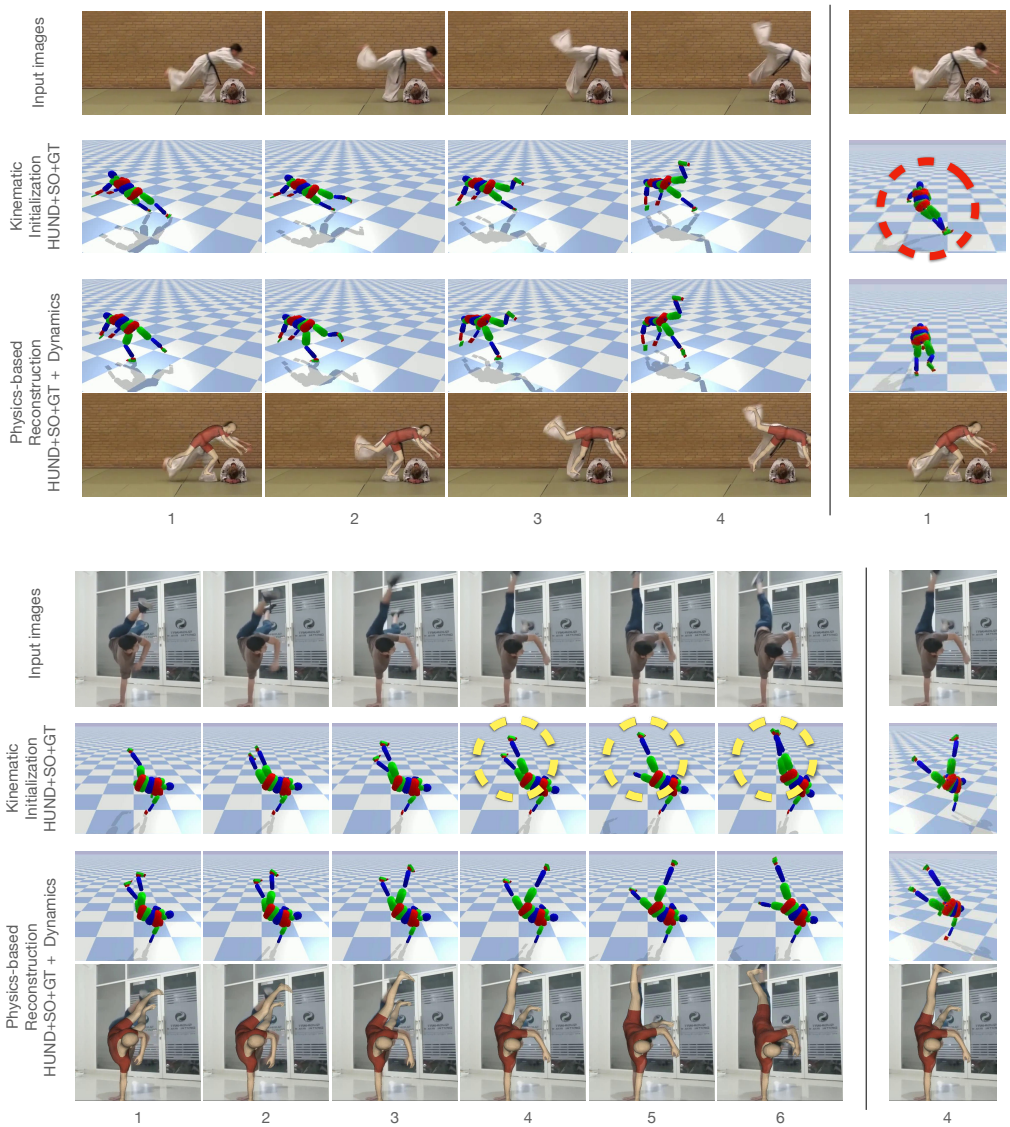


Figure 4.6: Example results on real-world videos. In the top row sequence, the kinematic initialization incorrectly places the left foot before the jump. We highlight the mistake by showing the scene from another viewpoint (red circle). The kinematic initialization also fails to produce temporally consistent poses in the example in the bottom row (yellow circles). Our physics-based inference corrects both errors and generates a more plausible motion. See tiny.cc/traj-opt for more results.

Figure 4.7: Example results on a difficult real-world video in which the legs of the person are occluded by the clothing. Note that 2d keypoints on the legs are incorrectly localized in multiple consecutive frames due to severe occlusion (second row) which results in poor 3d pose estimation by the kinematic model (third row). Interestingly our full model with dynamic is able to recover from errors in the kinematic initialization and generates reasonable sequence of 3d body poses (fourth row).

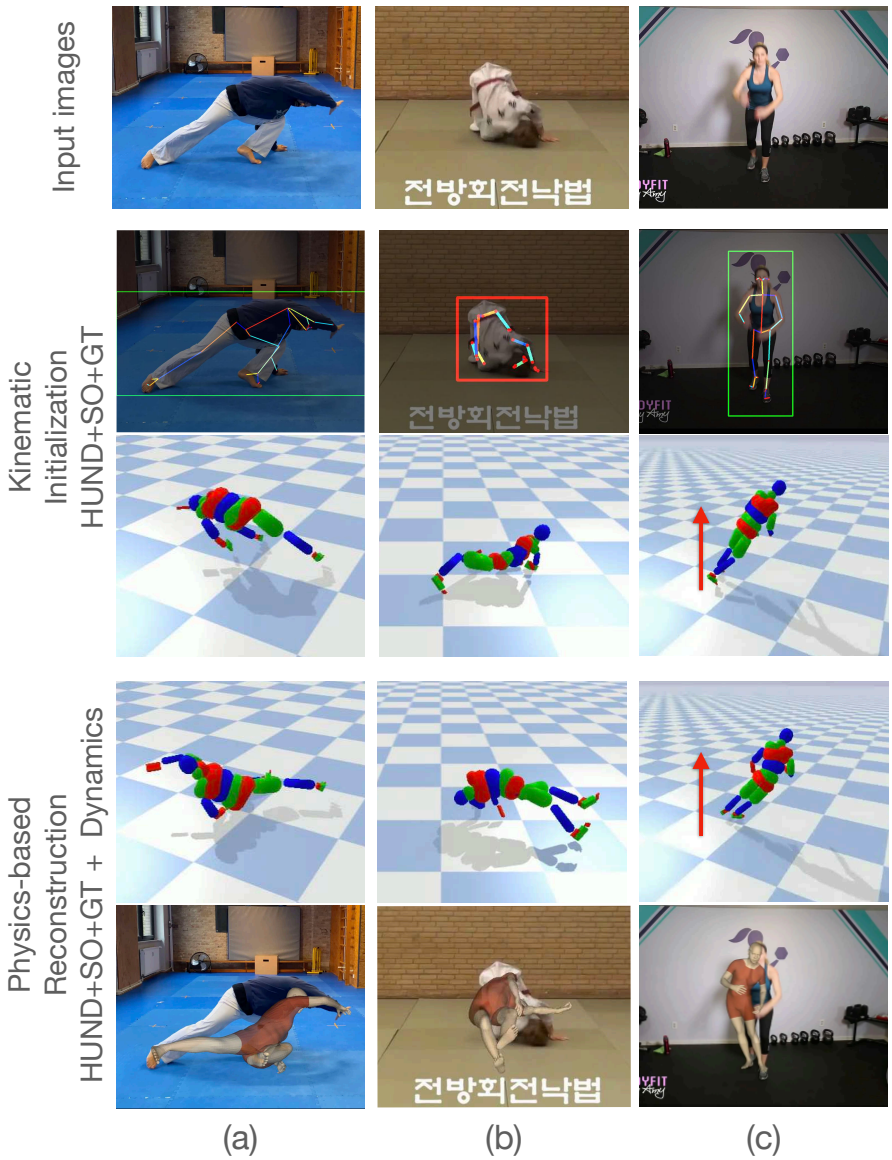


Figure 4.8: Examples of the characteristic failure cases of our approach on the real-world videos. Note that physics-based modeling introduces additional coupling between positions of the body limbs. While this is typically seen as an advantage, it also means that failure to estimate one limb correctly can propagate to other body limbs. For example in (a) our approach failed to correctly estimate position of the left arm which is used to support the body. As a result the overall 3d pose is worse for the dynamics (forth row) compared to the kinematic initialization (third row). Our physics-based reconstruction might also fail due to poor kinematics initialization (b) or due to failure to correctly estimate the orientation of the ground plane relative to the camera (c).

References

[1] Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann. Trajectory optimization for full-body movements with complex contacts. In *IEEE transactions on visualization and computer graphics*, volume 19, pages 1405–14, 08 2013. doi: 10.1109/TVCG.2012.325. 156

[2] Mazen Al Borno, Ludovic Righetti, Michael J. Black, Scott L. Delp, Eugene Fiume, and Javier Romero. Robust Physics-based Motion Retargeting with Realistic Body Shapes. In *Computer Graphics Forum*, 2018. 153, 155, 156, 173

[3] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005. 153

[4] M. A. Brubaker, L. Sigal, and D. J. Fleet. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2389–2396, 2009. doi: 10.1109/ICCV.2009.5459407. 154

[5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *CVPR*, 2017. 152

[6] Michael F. Cohen. Interactive spacetime control for animation. In *SIGGRAPH*, 1992. 156

[7] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019. 150, 154, 155

[8] M. Da Silva, Y. Abe, and J. Popović. Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum*, 27(2):371–380, 2008. 153

[9] Erik Gärtner, Mykhaylo Andriluka, Erwin Coumans, and Cristian Sminchisescu. Differentiable dynamics for articulated 3d human motion reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 154, 155, 156

[10] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*, pages 75–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32494-2. doi: 10.1007/3-540-32494-1_4. URL `https://doi.org/10.1007/3-540-32494-1_4`. 156, 177

[11] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL http://arxiv.org/abs/1707.02286. 153

[12] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://github.com/google-research/tiny-differentiable-simulator. 154

[13] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (7):1325–1339, jul 2014. 149, 153, 159, 162, 175, 176

[14] H. Joo, T. Simon, and Y. Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8320–8329, 2018. doi: 10.1109/CVPR.2018.00868. 153

[15] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 150, 152

[16] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. 150, 153, 162, 163

[17] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics*, 38:1–13, 07 2019. doi: 10.1145/3306346.3322972. 153, 154

[18] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Learn to dance with aist++: Music conditioned 3d dance generation, 2021. 160, 162, 176

[19] Zongmian Li, Jiri Sedlar, Justin Carpentier, Ivan Laptev, Nicolas Mansard, and Josef Sivic. Estimating 3d motion and forces of person-object interactions from monocular video. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 154

[20] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015. 153

[21] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, October 2019. 153

[22] Dimitris Metaxas. *Physics-Based Vision*. Kluwer Academic Publishing, 1997. 154

[23] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 150, 153

[24] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311. 156, 174

[25] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. In *SIGGRAPH*, 2018. 150, 153

[26] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37 (6), November 2018. 153

[27] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 2017. 155

[28] Stanley Plagenhoef, F Gaynor Evans, and Thomas Abdelnour. Anatomical data for analyzing human motion. *Research quarterly for exercise and sport*, 54(2):169–178, 1983. 155

[29] Davis Rempe, Leonidas J. Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. Contact and human dynamics from monocular video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 150, 151, 153, 154, 160, 162, 174, 175

[30] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics*, 39(6), dec 2020. 150, 151, 154, 162, 164, 175, 178

[31] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics*, 40(4), aug 2021. 150, 151, 154, 162

[32] L. Sigal, A. Balan, and M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1):4–27, March 2010. 159, 162, 175

[33] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3d human tracking. In *CVPR*, 2003. 154

[34] Jakub Stepien. *Physics-Based Animation of Articulated Rigid Body Systems for Virtual Environments*. PhD thesis, 10 2013. 156

[35] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109. 154

[36] Shuhei Tsuchida, Satoru Fukayama, Masahiro Hamasaki, and Masataka Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 501–510, Delft, Netherlands, November 2019. 149, 159, 160, 162, 163, 175, 176

[37] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision (ECCV)*, sep 2018. 153

[38] M. Vondrak, L. Sigal, and O. C. Jenkins. Physical simulation for probabilistic motion tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587580. 154

[39] Andrew Witkin and Michael Kass. Spacetime constraints. In *SIGGRAPH*, 1988. 156

[40] Jungdam Won and Jehee Lee. Learning body shape variation in physics-based characters. *ACM Trans. Graph.*, 2019. 154

[41] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39(4), 2020. URL `https://doi.org/10.1145/3386569.3392381`. 153

[42] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 150, 153

[43] Kevin Xie, Tingwu Wang, Umar Iqbal, Yunrong Guo, Sanja Fidler, and Florian Shkurti. Physics-based human motion estimation and synthesis from videos. In *Int. Conf. Comput. Vis.*, 2021. 150, 151, 154, 155, 159, 162, 164, 175

[44] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. GHUM & GHUML: Generative 3d human shape and articulated pose models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6184–6193, 2020. 153, 155, 173, 174

[45] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In *Advances in Neural Information Processing Systems*, 2020. 152, 154

[46] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 154, 162

[47] Andrei Zanfir, Elisabeta Marinoiu, Mihai Zanfir, Alin-Ionut Popa, and Cristian Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *NeurIPS*, pages 8410–8419, 2018. 152

[48] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, Bill Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. *arXiv preprint arXiv:2003.10350*, 2020. 153, 157

[49] Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Neural descent for visual 3d human pose and shape. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 150, 155, 158

Supplementary Material

This supplementary material provides further details on our methodology and the data we used. §A presents details on our physical human body model, §B provides details regarding our simulation parameters, §C presents our physics metrics, in §D we present the datasets used in our experiments, §E provides details about our method's hyperparameters, and lastly §F summarizes our computational setup. When referring to equations or material in the main paper we will denote this by *(mp)*. Finally, please see our supplemental video for qualitative results of our method at tiny.cc/traj-opt.

A Physical Body Model

Given a GHUM [44] body mesh $\mathbf{M}(\boldsymbol{\beta}, \boldsymbol{\theta}_0)$ associated with the shape parameters $\boldsymbol{\beta}$ and the rest pose $\boldsymbol{\theta}_0$, we build a simulation-ready rigid multibody human model that best approximates the mesh with a set of parameterized geometric primitives (*cf.* Fig. A.9). The hands and feet are approximated with boxes whereas the rest of the body links are approximated with capsules. The primitives are connected and articulated with the GHUM body joints.

Inspired by [2], we optimize the primitive parameters by minimizing

$$\begin{aligned} L(\boldsymbol{\psi}) = &\sum_{b \in \mathbf{B}} \sum_{\mathbf{v}_g \in \mathbf{M}_b} \min_{\mathbf{v}_p \in \hat{\mathbf{M}}_b} ||\mathbf{v}_g - \mathbf{v}_p|| + \\ &+ \sum_{b \in \mathbf{B}} \sum_{\mathbf{v}_p \in \hat{\mathbf{M}}_b} \min_{\mathbf{v}_g \in \mathbf{M}_b} ||\mathbf{v}_p - \mathbf{v}_g||, \end{aligned} \tag{9}$$

where $\boldsymbol{\psi}$ are the size parameters for the primitives, i.e. length and radius for the capsules, and depth, height and width for the boxes. The loss penalizes the bi-directional distances between pairs of nearest points on the GHUM mesh $\mathbf{M}_b$ and surface of the primitive geometry $\hat{\mathbf{M}}_b$ associated with the body link b.

Furthermore, we learn a nonlinear regressor $\boldsymbol{\psi}(\boldsymbol{\beta})$ with an MLP that performs fast shape approximation at run time. The regressor consists of two 256-dimensional fully connected layers, and is trained with 50K shapes generated with Gaussian sampling of the latent shape space $\boldsymbol{\beta}$ together with the paired optimal primitive parameters using (9).

Our physical model share an identical skeleton topology with GHUM but does not model the face and finger joints, due to the focused interest on the body dynamics in this work. Extending with finger joints, however, would enable simulation of hand-object interactions which would be interesting, but we leave this for future work. We note that there is a

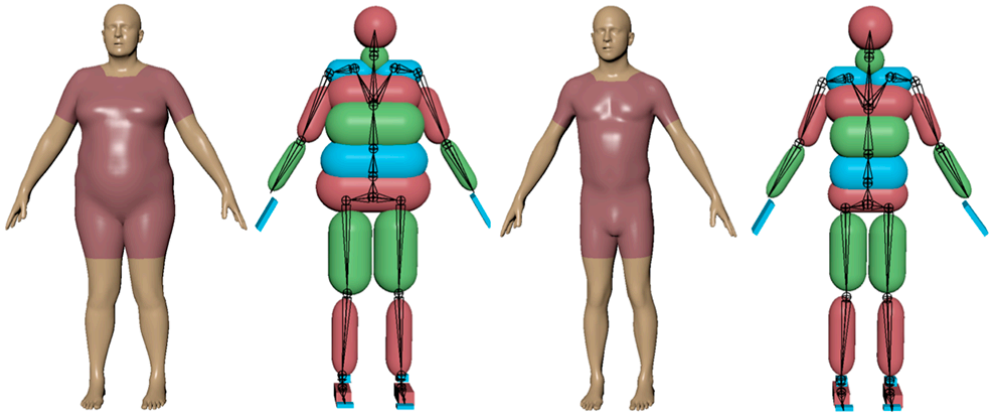


Figure A.9: The physical body model's shape and mass parameters are based on an associated GHUM [44] mesh.

bijective mapping for the shared 16 body joints between our model and GHUM, which allows for fast conversion between the physical and stastical representation.

B Simulation Details

We run the Bullet simulation at 200 Hz, with friction coefficient $\mu = 0.9$ and gravitational acceleration constant 9.8 m/s^2. The PD-controllers controlling each torque motor is tuned with position gain $k_p = 4.0$, velocity gain $k_d = 0.3$, and torque limits similar to those presented in [24].

C Additional Metrics

In addition to the standard 2d and 3d joint position error metrics, we evaluate our reconstructions using physical plausibility metrics similar to those proposed in [29]. Since the authors were unable to share their code we implement our own versions the metrics which doesn't require foot-ground contact annotations. A foot contact is defined as at least $N = 10$ vertices of a foot mesh being in contact with the ground plane. We set the contact threshold to $d = 0.005$ m for kinematics. To account for the modeling error when approximating the foot with a box primitive we set the contact threshold for dynamics to $d = -0.015$ m.

Footskate. The percentage of frames in a sequence where either foot joint moves more than 2 cm between two adjacent frames while the corresponding foot was in contact with the ground-plane.

Table A.5: Weights of the objective function described in §3.3 *(mp)*and (1) for our three main datasets: Human3.6M [13], AIST [36], and HumanEva-I [32]. "Grid" specifies the values evaluated while selecting hyperparameter values. Note that we did not exhaustively explore all combination.

Weight	H36M	AIST	HumanEva-I	Grid
w_{COM}	15.0	15.0	15.0	{1, 2, 5, 10, 15, 25 }
w_{pose}	0.5	0.5	0.5	{0.1, 0.5, 1, 2 }
w_{2d}	4.0	4.0	4.0	{1, 2, 4, 8, 10 }
w_{nf}	1.0	1.0	1.0	{0.001, 0.1, 1, 10}
w_{TV}	1.0	1.0	1.0	{0.1, 1, 10}
w_{lim}	1.0	1.0	1.0	{0.1, 1, 10}

Float. The percentage of frames in a sequence where at least one of the feet was not in contact but was within 2 cm of the ground-plane. This metric captures the common issue of reconstructions floating above the ground while not penalizing correctly reconstructed motion of e.g. jumps.

Velocity. The mean error between the 3d joint velocities in the ground-truth data and the joint velocity in the reconstruction. High error velocity indicates that the estimated motion doesn't smoothly follow the trajectory of the true motion. We define the velocity error as

$$e_v = \frac{1}{N}\sum_{i=1}^{N}\sum_{k\in K}|\dot{\tilde{\mathbf{x}}}_k^i - \dot{\mathbf{x}}_k^i|, \tag{10}$$

where $\dot{\tilde{\mathbf{x}}}_k^i$ is the magnitude of the ground-truth 3d joint velocity vector (in m/s) for joint k at frame i and where $\dot{\mathbf{x}}_k^i$ denotes the reconstructed joint. We estimate the velocity using finite differences from 3d joint positions and use first frame translation aligned joint estimates (as in MPJPE-G).

D Datasets

Human3.6M. We use two subsets for our experiments on Human3.6M [13]. When we compare our method to state-of-the-art methods we use a dataset split similar to the one used in [43]. See Table B.8 for the complete lists of sequences we use. Similarly to [30, 43], we down sample the sequences from 50 FPS to 25 FPS.

When perform ablations of our model we a smaller subset where we select 20 4-sec sequences from the test split of Human3.6M dataset (subjects 9 and 11). We selected sequences that show various dynamic motions such as walking dog, running and phoning (with large motion range), to sitting and purchasing (with occluded body parts). For each sequence, we randomly selected one of the four cameras. We list the sequences in Table B.6.

HumanEva-I. We evaluate our method on the subset of HumanEva-I walking sequences [32] as selected by [29], see Table D.9.

Table B.6: The subset of Human3.6M used in the ablation experiments. Note that the data was downsampled from 50 to 25 FPS.

Sequence	Subject	Camera Id	Frames
Phoning	S11	55011271	400-599
Posing_1	S11	58860488	400-599
Purchases	S11	60457274	400-599
SittingDown_1	S11	54138969	400-599
Smoking_1	S11	54138969	400-599
TakingPhoto_1	S11	54138969	400-599
Waiting_1	S11	58860488	400-599
WalkDog	S11	58860488	400-599
WalkTogether	S11	55011271	400-599
Walking_1	S11	55011271	400-599
Greeting_1	S9	54138969	400-599
Phoning_1	S9	54138969	400-599
Purchases	S9	60457274	400-599
SittingDown	S9	55011271	400-599
Smoking	S9	60457274	400-599
TakingPhoto	S9	60457274	400-599
Waiting	S9	60457274	400-599
WalkDog_1	S9	54138969	400-599
WalkTogether_1	S9	55011271	400-599
Walking	S9	58860488	400-599

AIST. We select four second video sequences from the public dataset [18, 36], showing fast and complex dancing motions, picked randomly from one of the 10 cameras. We list our selected sequences in Table B.7.

"In-the-wild" internet videos. We perform qualitative evaluation of our model on videos of dynamic motions rarely found in laboratory captured datasets. These videos were made available on the internet under a CC-BY license which grants the express permission to be used for any purpose. Note that we only used the videos to perform qualitative analysis of our approach – the videos will not be redistributed as a dataset.

D.1 Human Data Usage

This work relies on recorded videos of humans. Our main evaluation is performed on two standard human pose benchmarks: Human3.6M[1] [13] and AIST[2] [36]. These datasets have been approved for research purposes according to their respective websites. Both datasets contain recordings of actors in laboratory settings. To complement this, we perform qualitative evaluation on videos released on the internet under creative commons licenses.

[1] http://vision.imar.ro/human3.6m/
[2] https://aistdancedb.ongaaccel.jp/

Table B.7: Sequences used for evaluation on AIST.

Sequence	Frames
gBR_sBM_c06_d06_mBR4_ch06	1-120
gBR_sBM_c07_d06_mBR4_ch02	1-120
gBR_sBM_c08_d05_mBR1_ch01	1-120
gBR_sFM_c03_d04_mBR0_ch01	1-120
gJB_sBM_c02_d09_mJB3_ch10	1-120
gKR_sBM_c09_d30_mKR5_ch05	1-120
gLH_sBM_c04_d18_mLH5_ch07	1-120
gLH_sBM_c07_d18_mLH4_ch03	1-120
gLH_sBM_c09_d17_mLH1_ch02	1-120
gLH_sFM_c03_d18_mLH0_ch15	1-120
gLO_sBM_c05_d14_mLO4_ch07	1-120
gLO_sBM_c07_d15_mLO4_ch09	1-120
gLO_sFM_c02_d15_mLO4_ch21	1-120
gMH_sBM_c01_d24_mMH3_ch02	1-120
gMH_sBM_c05_d24_mMH4_ch07	1-120

E Hyperparameters

The most important hyperparameters are the weights of the weighted objected function described in §3.3 *(mp)*. Where combined loss function is given by

$$\begin{aligned} L = & w_{COM} L_{COM} + w_{pose} L_{pose} \\ & + w_{2d} L_{2d} + w_{nf} L_{nf} + w_{TV} L_{TV} \\ & + w_{lim} L_{lim}. \end{aligned} \tag{11}$$

We tuned the weights on sequences from the training splits. The goal was to scale the different components such that they have roughly equal magnitudes while minimizing the MPJPE-G error. See Table A.5 for details regarding the search grid and the chosen parameter values.

F Computational Resources

For running small experiments we used a desktop workstation equipped with an "Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz" CPU, 128 GB system memory and two NVIDIA Titan Xp GPUs. We ran kinematics in the cloud using instances with a V100 GPU, 48 GB of memory and 8 vCPUs. In the dynamics experiments, we used instances with 100 vCPUs and 256 GB of memory for the CMA-ES [10] optimization. Optimizing a window of 1 second of video takes roughly 20 min using a 100 vCPUs instance.

Table B.8: The evaluation subset of Human3.6M used in the main evaluation. The subset is similar to the one used in [30]. We downsampled the data from 50 FPS to 25 FPS.

Sequence	Subject	Camera Id
S11	Directions_1	60457274
S11	Discussion_1	60457274
S11	Greeting_1	60457274
S11	Posing_1	60457274
S11	Purchases_1	60457274
S11	TakingPhoto_1	60457274
S11	Waiting_1	60457274
S11	WalkDog_1	60457274
S11	WalkTogether_1	60457274
S11	Walking_1	60457274
S9	Directions_1	60457274
S9	Discussion_1	60457274
S9	Greeting_1	60457274
S9	Posing_1	60457274
S9	Purchases_1	60457274
S9	TakingPhoto_1	60457274
S9	Waiting_1	60457274
S9	WalkDog_1	60457274
S9	WalkTogether_1	60457274
S9	Walking_1	60457274

Table D.9: Sequences used for evaluation on HumanEva-I.

Sequence	Subject	Camera Id	Frames
Walking	S1	C1	1-561
Walking	S2	C1	1-438
Walking	S3	C1	1-490

Paper V

Erik Gärtner, Mykhaylo Andriluka, Erwin Coumans, Cristian Sminchisescu
Differentiable Dynamics for Articulated 3d Human Motion Reconstruction
Computer Vision and Pattern Recognition (CVPR), New Orleans, USA, 2022

Differentiable Dynamics for Articulated 3d Human Motion Reconstruction

Erik Gärtner[1,2] **Mykhaylo Andriluka**[1] **Erwin Coumans**[1]
Cristian Sminchisescu[1]

[1]Google Research
[2]Lund University

Abstract

We introduce DiffPhy, a differentiable physics-based model for articulated 3d human motion reconstruction from video. Applications of physics-based reasoning in human motion analysis have so far been limited, both by the complexity of constructing adequate physical models of articulated human motion, and by the formidable challenges of performing stable and efficient inference with physics in the loop. We jointly address such modeling and inference challenges by proposing an approach that combines a physically plausible body representation with anatomical joint limits, a differentiable physics simulator, and optimization techniques that ensure good performance and robustness to suboptimal local optima. In contrast to several recent methods [39, 41, 56], our approach readily supports full-body contact including interactions with objects in the scene. Most importantly, our model connects end-to-end with images, thus supporting direct gradient-based physics optimization by means of image-based loss functions. We validate the model by demonstrating that it can accurately reconstruct physically plausible 3d human motion from monocular video, both on public benchmarks with available 3d ground-truth, and on videos from the internet.

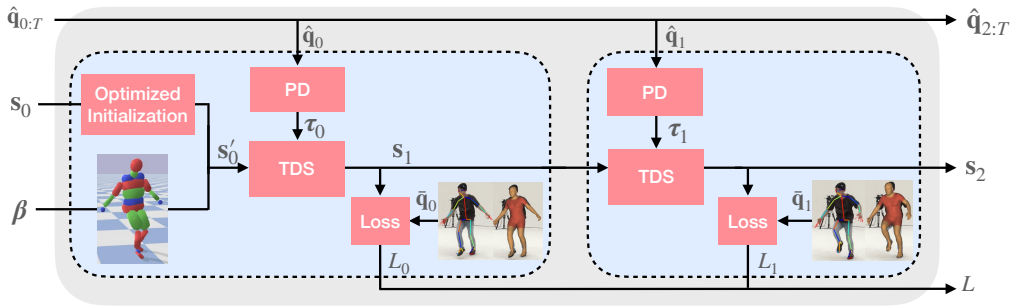


Figure 5.1: Overview of DiffPhy. Given kinematic estimates (described in §3.1) of a subject's body shape β, the body's initial pose and velocity $\mathbf{s}_0$, and time-varying 3d poses $\bar{\mathbf{q}}_{0:T}$ with detected 2d keypoints, our model reconstructs the motion in physical simulation, by minimizing a differentiable loss L (see §3.5). DiffPhy optimizes the control trajectory $\hat{\mathbf{q}}_{0:T}$ containing joint angle targets to PD controllers (*cf.* (5.4)). In turn, the PD controllers compute a torque vector $\boldsymbol{\tau}$, which actuates motors in the joints of the simulated body. DiffPhy integrates a full-featured differentiable simulator, TDS [17] (described in §3.2), that supports complex contacts. Each subject is represented by means of a personalised physical model (see §3.3). In addition, we optimize the initial state (see §3.6), which makes DiffPhy robust to low quality initial estimates. The outputs are 3d pose estimates that align with visual evidence and respect physical constraints.

1 Introduction

We seek to contribute to the development of physics-based methodology as one of the building blocks in constructing accurate and robust 3d visual human sensing systems. Incorporating the laws of physics into the visual reasoning process is appealing as it promotes the plausibility of estimated motion and facilitates more efficient use of training examples [9]. We focus on articulated human motion as an epitome of a real-world prediction task that is both well studied and challenging. Existing state-of-the-art approaches demonstrate relatively high accuracy in terms of joint position estimation metrics [23, 24, 55, 63]. However, predictions can sometimes be physically implausible, even for simple motions such as walking and running. For instance, estimates can include unreasonably abrupt transitions in world space, or artifacts such as foot skating or non-equilibrium states [39, 41]. Many methods are typically trained on large motion capture datasets and encounter difficulties when tested on motions not well represented in those training sets. Arguably, imposing some form of physics-based generally valid prior on the articulated motion estimates should greatly improve the plausibility of results.

However, physics-based reasoning comes at the cost of substantial modeling and inference complexity. Typically, physics-based articulated estimation methods rely on rigid body dynamics (RBD) [10, 45], a formulation that introduces many auxiliary variables corresponding to forces acting at the body joints at each time step. Moreover, physical contact results in non-smooth effects where small changes to model parameters might result in substantially different motions. Therefore inferring physics variables given the inherent uncertainty in monocular video, and under contact discontinuities, becomes significantly

difficult, algorithmically and computationally. Despite such challenges, a number of recent methods successfully apply physics-based constraints for articulated human motion estimation [2, 39, 41, 60]. One possibility to cope with modeling complexity, explored in recent work, is to simplify the physics and model contacts only between the body and the feet [39, 41, 56]. Others use auxiliary external forces applied at the body to compensate for modeling error [41, 60].

In this paper, we aim to broaden the methodology for physics-based articulated human motion estimation. Specifically, we demonstrate that we can successfully leverage recent progress in differentiable simulation [17, 19, 53] in order to incorporate physics-based constraints into the articulated 3d human motion reconstruction. Our approach, *DiffPhy*, relies on gradient-based optimization, connects end-to-end with images, and does not require simplifying assumptions on contacts or the introduction of external non-physical residual forces.

2 Related work

Kinematics-based 3d Human Pose Estimation. The problem of monocular 3d pose estimation is usually addressed through end-to-end [30, 31, 61], or two-stage [8, 18] models where neural networks are used to predict 3d joint positions. This is an ill-posed problem due to depth ambiguities and occlusion. The networks are usually trained on vast pose datasets [21, 22, 29, 51] which usually supports good performance on poses previously observed during training. Several methods [24, 62, 63] directly regress the parameters of statistical body models [27, 57] (rather than 3d joint positions), including the subject's body shape as well as kinematic pose. The methods mentioned above take a purely visual inference approach to the problem and do not consider physics-based constraints. As observed by [39], this may cause artifacts such as jitter, ground-penetration, foot sliding, or unnatural leaning [41].

Physics-based 3d Human Pose Estimation. Recent work [15, 28, 39, 41, 42, 43, 56, 60] aims to increase realism, by using physics to regularize reconstruction. This aims to enforce physical constraints such as proper contact and dynamic coherence. In [39] motion is reconstructed through optimization, but the method only accounts for collisions between the feet and the ground. Such simplifications are recurring in current approaches and limit the types of motions that can be reconstructed. In contrast, in this work, we use a full-featured physical simulator which supports contacts between all objects in the scene. PhysCap [41] is a real-time optimization-based approach, where feet contact is pre-detected based on a neural network. During the physics-based inference, contacts are considered fixed and therefore cannot be corrected or improved. Moreover, following [59] the method uses non-physical "residual forces" which improve 3d joint reconstruction metrics at the cost

of altered physical plausibility. Since we aim to increase the physicality of reconstructed motions, we avoid using any residual forces. [60] follows on [34, 54] to learn a neural network that estimates torques to drive a model in the full-featured physical simulator MuJoCo [48]. However, MuJoCo is non-differentiable, hence the need to resort to expensive training using numerical gradients in a reinforcement learning setting. The method is trained for millions of steps using 3d ground-truth labels from a motion capture dataset, but the method's ability to generalize to in-the-wild is not demonstrated. Similarly to [39], the method assumes a known ground plane, whereas *DiffPhy* estimates it. [43] integrates a simplified physics approach, dubbed "physionical", into a neural network that estimates joint torques and ground-reaction forces. Similarly to [41] they detect foot contact using a neural network predictor rather than by means of physical simulation. Most recently, [56] introduced a method relying on a simplified physical formulation that makes it possible to refine 3d pose estimates well enough to train motion synthesis models based on that output. However, the method assumes a known ground plane, models only foot contact, and implements a simplified physical body scaled solely based on the estimated bone length rather than shape estimates. Finally, in our concurrent work [15], we perform physics-based human pose reconstruction of complex motions through trajectory optimization based on CMA-ES [16] in the non-differentiable simulator Bullet [7]. This general approach uses a mature and full-featured simulator which, while capable, is slow due to costly black-box optimization. The method does not optimize the initial state of the body (see §3.6) together with the joint control variables, being more vulnerable to unfavorable initialisation. In summary, this work takes the novel approach of tightly integrating physics into the reconstruction process through a full-featured *differentiable* physics model. As a result, DiffPhy supports complex full-body contacts, connects pixels-to-physics using end-to-end differentiable losses, supports personalised body models, does not resort to residual forces, and is robust to poor initialization. See Tab. 5.1 for an overview of physics-based methods.

It is worth mentioning that, aside from physical simulation, there exist many other approaches to grounding the human pose estimates using e.g., inertial estimates from IMUs [58], scene constraints [5, 64], and motion priors [40].

Differentiable Physics for Human Modeling. Physical simulation is a mature area with several established simulation engines available [7, 25, 48]. These engines implement forward simulation but do not facilitate the computation of derivatives necessary for efficient gradient-based optimization. These simulators are well-suited for training with gradient-free methods such as reinforcement-learning or evolutionary algorithms and have been used for gradient-free optimization of human motion models [2, 35, 60]. More recently differentiable physics simulators have emerged [6, 14, 17, 38, 53]. Applying these to human motion reconstruction is difficult due to noisy gradients [19, 33], and a non-convex objective function. We present a methodology using gradient-based local search with stochastic global optimization enabling the first use of a full-featured differentiable physics model [17]

Table 5.1: Feature comparison against other physics-based methods. *Body* compares the type of physical body representation where "adapt" means individually constructed based on shape estimate, *Cont.* column compares what type of contacts are supported, *DP* whether the method uses a differentiable physical formulation, *Training* if the physical inference requires training, $\mathbf{T}_g$ compares if the ground plane is estimated (as opposed to assumed known), and *No RF* if the method avoid non-physical residual forces. Only our method does not require any additional training and uses a full-featured differentiable physics formulation.

Method	Body	Cont.	DP	Trained	$\mathbf{T}_g$	No RF
Rempe *et al.* [39]	Fixed	Feet	✗	Contacts	✗	✓
PhysCap [41]	Fixed	Feet	✓	Contacts	✓	✗
SimPoE [60]	Adapt	Full	✗	Yes	✗	✗
Shimada *et al.* [43]	Fixed	Feet	✓	Yes	✓	✗
Xie *et al.* [56]	Fixed	Feet	✓	No	✗	✓
Dynamics [15]	Adapt	Full	✗	Prior	✓	✓
DiffPhy	Adapt	Full	✓	No	✓	✓

for human pose reconstruction from video. Furthermore, we show that our approach is magnitudes faster than a purely sampling-based approach.

3 Methodology

This section presents our approach to reconstructing 3d human shape and motion from video with differentiable physics in the loop. Given a monocular video of a human subject, we use a kinematic neural network to estimate 2d body keypoints, the body shape, and 3d body poses. Since estimating 3d pose from monocular video is ill-posed, due to e.g. depth-ambiguities and occlusion [44], the kinematic 3d reconstructions may suffer from self-penetration, inconsistent translation, jitters, floating above the ground, and non-physical leaning [39, 41]. We, therefore, reconstruct the motion in physical simulation, by jointly accounting for both visual evidence and the constraints of physical simulation (e.g. collisions, gravity, and Newton's laws of motion). See Fig. 5.1 for an overview of our approach.

3.1 Kinematic Initialization

Given a sequence of monocular images $\{I_i\}$, we assume a pinhole camera with intrinsics $\mathbf{i} = [f_x, f_y, c_x, c_y]$ and constant camera extrinsics. We obtain the visual evidence used in our optimization objectives following the procedure introduced in [15]. This relies on HUND [62], a 3d pose estimator that produces per-frame 2d keypoints $\bar{\mathbf{x}}_i$ with confidence scores $\mathbf{c}_i$, 3d body poses $\boldsymbol{\theta}_i$, and 3d body shape $\boldsymbol{\beta}_i$, where $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are the GHUM [57] posing and shape parameters, respectively.

Since HUND is a per-frame estimator, a temporally consistent shape is recovered by select-

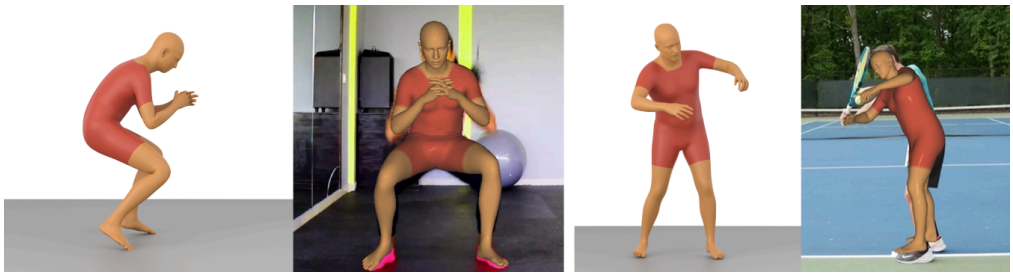


Figure 5.2: Qualitative results on two in-the-wild sequences. Sports and dynamic activities are rarely found in motion capture datasets.

ing the $N = 5$ highest-scoring frames according to keypoint confidences. For these frames, HUND image losses [62] are minimized using BFGS under the additional constraint of a constant shape, $\boldsymbol{\beta}$, across all frames. In addition, [15] introduces a final round of optimization where poses are updated under the time-consistent body shape and a temporal smoothness loss to reduce jittering.

Finally, as the ground plane location is not assumed to be known and HUND produces estimates in camera space $\mathbf{k}$, we estimate the global transform $\mathbf{T}_g \in \mathbb{R}^{3\times4}$ for the physical scene, with gravity along the y axis, as well as the ground plane at $y = 0$. This is achieved by minimizing

$$L_g(\mathbf{T}_g) = \sum_i^N \| \min\big(\delta, \mathbf{L}_y\big(\mathbf{T}_g[\mathbf{M}(\boldsymbol{\beta}, \boldsymbol{\theta}_i), 1]\big)\big)\|^2, \tag{5.1}$$

where $\mathbf{L}_y$ is an operator that extracts the $k = 20$ smallest signed distances from the mesh vertices $\mathbf{M}(\boldsymbol{\beta}, \boldsymbol{\theta}_i)$ after the global transformation. This assumes the body is in ground plane contact for most of the sequence. To allow for frames where the subject is not in contact with the ground, we clip the maximum shortest distance to the ground to $\delta = 20$ cm.

3.2 Differentiable Physics Simulation Model

We implement our models in the framework of the "Tiny Differentiable Simulator" (TDS) [17]. This formulates rigid-body dynamics for articulated bodies in terms of reduced coordinates. Elements in the vector $\mathbf{q}$ represent the position of each joint, and elements in the vector $\dot{\mathbf{q}}$ represent joint space velocities, based on revolute and spherical joints. Given the state of the body $\vec{s}_t = (\mathbf{q}_t, \dot{\mathbf{q}}_t)$ at time t, as well as the vector of joint torques $\boldsymbol{\tau}_t$, and external forces $\mathbf{f}_t$, the computation shown in Fig. 5.3 produces a new body state $\vec{s}_{t+\delta t}$ corresponding to the rigid multi-body dynamics with contacts. To that end, we first run forward kinematics to compute world space positions and velocities, as well as forward dynamics to compute

unconstrained acceleration obtained without taking contacts into account. The forward dynamics computes the acceleration by solving the equation of motion for the kinematic tree given by

$$\boldsymbol{\tau}_t = \mathbf{H}(\mathbf{q}_t)\ddot{\mathbf{q}}_t + \mathbf{C}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{f}_t^x) \tag{5.2}$$

where $\mathbf{H}(\mathbf{q})$ is the joint-space inertia matrix, $\mathbf{C}$ the a joint space bias force and $\mathbf{f}^x$ is the vector of external forces. The forward dynamics is computed by propagation-based Articulated-Body Algorithm (ABA) [11] that traverses the kinematic chain of the body three times in order to compute quantities necessary to finally obtain the acceleration of each rigid component of the body[1]. The joint-space inertia matrix is computed using the Composite Rigid Body Algorithm (CRBA) [11].

Unconstrained accelerations $\ddot{\mathbf{q}}^u_{t+\delta t}$ are then used to compute unconstrained velocities, which in conjunction with the output of the forward kinematics $\mathbf{x}_{t+\delta t}$ are used to update the contact points between the articulated body and the environment. Contact points with positive (separating) distance are classified as inactive, while contact points with zero or negative distance are active. Active contacts generate a repulsive impulse that needs to be taken into account when computing the new body state. To that end, the forward dynamics computation is phrased as a linear complementarity problem (LCP) at the velocity level [46, 47]

$$\begin{aligned} \mathbf{J}_c\mathbf{H}^{-1}\mathbf{J}_c^\top\mathbf{p} + \mathbf{J}_c\dot{\mathbf{x}} &= \mathbf{v} \\ \mathbf{v} &= [\mathbf{v}_u, \mathbf{v}_b] \\ \text{s.t.} \qquad \mathbf{v}_u^\top\mathbf{p}_u = 0 \qquad \mathbf{v}_u \geq 0 \qquad \mathbf{p}_u \geq 0 \qquad \mathbf{v}_b &= 0 \end{aligned} \tag{5.3}$$

where $\mathbf{J}_c$ is a contact Jacobian for the positions of contact points computed in the previous step, $\mathbf{p}$ is the vector of reaction impulses, and $\mathbf{v}$ is the vector of relative velocities. The indices u and b indicate the unilateral and bilateral portion of constraints, respectively. The LCP problem in (5.3) is then iteratively solved with a projected Gauss-Seidel method following the formulation in [47], by relying on a per-contact LCP [20]. The final step of the computation is to obtain joint positions $\mathbf{q}_{t+\delta t}$ from joint velocities using semi-implicit Euler integration.

3.3 Physical Human Body Modeling

In the physical simulation, we model the human body as rigid geometric primitives connected by joints. The model is comprised of 16 joints with a total of 48 degrees of freedom, joining together 26 capsules that represent the various body parts (*cf.* Fig. 5.1). The shape

[1] See tab. 7.1 in Featherstone [11] for the Articulated-Body Algorithm.

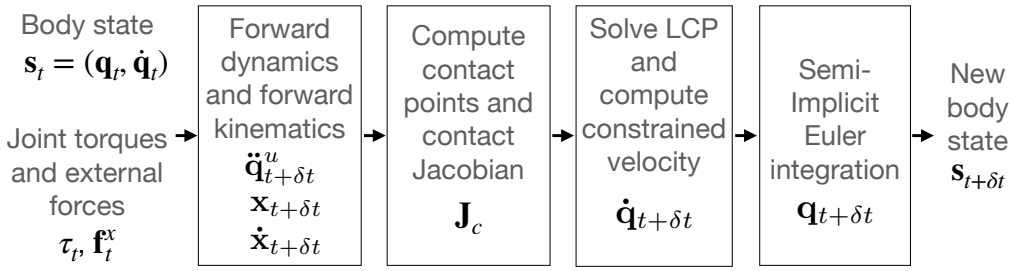


Figure 5.3: Overview of the simulation step of the physics model that updates the current state S_t to a new state after time step δ_t. For each computational block we include the output quantities used in the subsequent block.

and mass of the model is automatically adapted for various body shapes by relying on a statistical body model [57]. Given the 3d mesh $\mathbf{M}(\boldsymbol{\beta}, \emptyset)$ corresponding to a shape estimate $\boldsymbol{\beta}$ in rest pose, we infer the dimensions of the geometric primitives following the approach of [2]. The process is entirely automatic and yields individualized physical models for each subject. As a physical model requires mass, we first estimate the total mass of the body based on a human shape dataset [36] then distribute the weight according to an anatomical distribution [37]. Finally, the inertia of each primitive is computed based on its mass and dimensions.

DiffPhy reconstructs a motion in simulation by actuating torque motors in the joints of the body. Following prior work [1] we optimize over control targets to proportional-derivative (PD) controllers rather than over the torques directly. We define the body's angular joint positions as $\mathbf{q}_t$, and joint velocities as $\dot{\mathbf{q}}_t$, the associated 3d Cartesian coordinates of the joints as $\mathbf{x}_t$ for the time step t. Given a set of joint targets $\hat{\mathbf{q}}_{1:T} = \{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_t\}$ the PD controllers infer the joint torques as

$$\boldsymbol{\tau}_t = k_p(\hat{\mathbf{q}}_t - \mathbf{q}_t) - k_d\dot{\mathbf{q}}_t, \tag{5.4}$$

where k_p and k_d are gain parameters of PD controllers. We may then specify a motion of length T as the initial state $\mathbf{s}_0 = (\mathbf{q}_0, \dot{\mathbf{q}}_0)$, the world geometry $\mathbf{G}$ defining the position and orientation of the ground plane, and a target trajectory for the joints $\hat{\mathbf{q}}_{1:T}$. Given the loss presented in (5.5) we reconstruct the motion by minimizing $L = L(\mathbf{s}_0, \mathbf{G}, \hat{\mathbf{q}}_{1:T})$ with respect to $\hat{\mathbf{q}}_{1:T}$.

3.4 Gradient-Based Optimization

Given our loss function $L = L(\mathbf{S}_0, \mathbf{G}, \hat{\mathbf{q}}_{1:T})$ we can use any gradient-based optimization method to minimize the loss with respect to $\hat{\mathbf{q}}_{1:T}$. Since the loss function is non-convex, convergence to suboptimal local minima is possible. Therefore, a global optimization combined with a local gradient-based search is expected to outperform a purely local

Table 5.2: Comparison of optimization strategies on our Human3.6M validation set. BFGS and Basin-BFGS both use gradients, while CMA-ES is a gradient-free approach. Note that Basin-Hopping together with BFGS (Basin-BFGS) improves the performance of BFGS by combining it with stochastic global optimization. Using only a purely sampling-based approach (CMA-ES) requires magnitudes more function evaluations while still not finding better optima for our loss. BFGS was given a sufficiently large evaluation budget to converge.

Method	# eval	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d
CMA-ES	80k	206.7	125.7	77.4	16.9
BFGS	122	160.1	100.1	68.9	15.5
Basin-BFGS	509	144.9	84.6	61.1	12.6

method. One such method is the global stochastic optimization *Basin-Hopping* [52]. It uses a two-stage approach, which alternates between performing gradient-based local search and stochastic global search. Based on an initial candidate, it first performs a local search. It then randomly perturbs the local minimum, performs a local search again on the new candidate, and then either accepts or rejects the new solution based on the Metropolis criterion [32]. In our model, we use BFGS [13] for local optimization.

3.5 Optimization Objectives

Reconstructing a motion sequence amounts to finding the control trajectory $\hat{\mathbf{q}}_{1:T}$ that minimizes the reconstruction loss L under the constraints of the simulation dynamics. In this work, we formulate L as a weighted combination of loss functions

$$L = w_r L_r + w_j L_j + w_i L_i + w_l L_l, \tag{5.5}$$

with the weights $w_r = 10.0$, $w_j = 0.1$, $w_i = 0.01$, and $w_l = 0.01$. The root position loss L_r measures errors between the 3d position of the simulated pelvis root joint $\mathbf{x}_t^{\text{root}}$ and the kinematically estimated position $\bar{\mathbf{x}}_t^{\text{root}}$

$$L_r(\hat{\mathbf{q}}_{1:T}) = \frac{1}{T}\sum_{t}^{T} \|\bar{\mathbf{x}}_t^{\text{root}} - \mathbf{x}_t^{\text{root}}\|^2 \tag{5.6}$$

at time t where T is the total length of the sequence. L_j computes the rotational distance between the kinematic pose estimate and the simulated body's pose

$$L_j(\hat{\mathbf{q}}_{1:T}) = \frac{1}{TK}\sum_{t}^{T}\sum_{k}^{K} \arccos(|\mathbf{q}_t^k \cdot \bar{\mathbf{q}}_t^k|), \tag{5.7}$$

where $\bar{\mathbf{q}}_t^k$ and $\mathbf{q}_t^k$ are rotations expressed as quaternions for joint k at time t for kinematics and the simulated character respectively. Note the difference between $\hat{\mathbf{q}}$ and $\mathbf{q}$, where the

former are the PD control targets and the latter are the joint angles of the simulated model (*cf.* (5.4)). L_i computes the 2d projection loss

$$L_i(\hat{\mathbf{q}}_{1:T}) = \frac{1}{TK}\sum_{t}^{T}\sum_{k}^{K} \mathbf{c}_t^k \|\bar{\mathbf{x}}_t^k - \Pi(\mathbf{x}_t^k, \mathbf{i})\|^2, \tag{5.8}$$

where $\Pi(\mathbf{x}_t^k, \mathbf{i})$ is the perspective operator projecting the simulated model's joint $\mathbf{x}_t^k$ onto the image with camera intrinsics $\mathbf{i}$ weighted by the keypoint detection confidence score c_t^k. Finally, L_l is a regularizer that penalizes joints outside of human anatomical limits as present in the statistical body model [57]

$$\begin{aligned} L_l(\hat{\mathbf{q}}_{1:T}) = \frac{1}{TK}\sum_{t}^{T}\sum_{k}^{K} \| \max(z_{\text{lower}}^k - \mathbf{q}_t^k, 0) \\ + \max(\mathbf{q}_t^k - z_{\text{upper}}^k, 0)\|^2, \end{aligned} \tag{5.9}$$

where z_{upper}^k and z_{lower}^k are upper and lower bounds for joint k respectively.

Note that in the above definitions, the positions of body joints angles $\mathbf{q}_t^k$ and 3d joint positions $\mathbf{x}_t^k$ are dependent on the control trajectory up until time t, as part of the physics formulation introduced in §3.2.

3.6 Optimized Initialization

We initialize the pose $\mathbf{q}_0$ in the first time step of the simulation to the kinematically estimated pose $\bar{\mathbf{q}}_0$ and estimate the velocity $\dot{\mathbf{q}}_0$ using finite differences between the first two kinematic poses $\{\bar{\mathbf{q}}_0, \bar{\mathbf{q}}_1\}$. However, if the initial kinematic pose estimate is poor, this might lead to a low quality starting pose from which the simulation cannot recover. Similarly, jitters in the kinematic poses may cause a significant error in the estimated initial velocity. We address these issues by including the initial pose and velocity as variables to optimize. We experimentally validate how such a relatively straightforward approach significantly impacts the results.

4 Experiments

Datasets. We quantitatively evaluate DiffPhy on the Human3.6M [21], and a subset of the AIST [49] pose datasets. The former contains a diverse set of motions from a motion capture laboratory, whereas the latter contains dance videos with triangulated 3d joints as pseudo-ground-truth. As only DiffPhy and SimPoe [60] supports full-body contacts

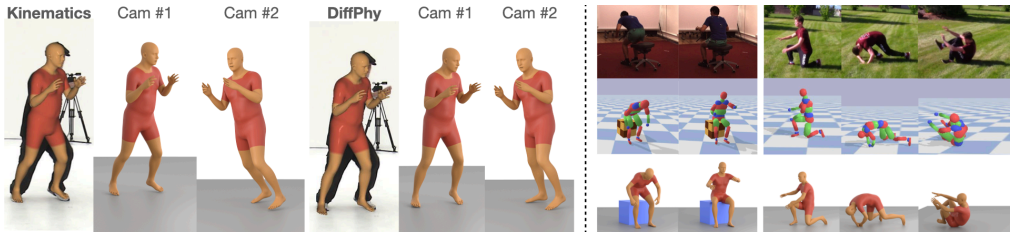


Figure 5.4: Qualitative examples on the AIST dataset (left) and of complex contacts (right). The AIST example shows that both kinematics and DiffPhy projects well into the image. However, when rendered from another viewpoint (cam #2) it becomes clear that kinematics exhibits unrealistic leaning while the physical constraints corrects the pose to keep the body in balance. See tiny.cc/diffphy for more.

(*cf.* Tab. 5.1), PhysCap [41] proposed evaluating on a subset of the Human3.6M. This protocol eliminated all sequences requiring more than foot-floor contacts. Hence to allow for comparison, we use this subset in Tab. 5.3, but note that our method is more general and supports contacts for all body parts. For ablations, we use 100 frames from 20 sequences from a validation subset of Human3.6M. Finally, we quantitatively evaluate our method on real-world internet videos released under creative commons licenses. For additional details, refer to our supplementary material.

Metrics. We report the standard pose metrics such as mean per-joint position error in millimeter (MPJPE-G), mean Procrustes aligned joint error (MPJPE-PA), per-frame translation aligned error (MPJPE), and 2d mean per-joint error in pixels (MPJPE-2d). Note that many papers do not report global position errors since they consider only root-relative poses. We, however, are interested in measuring the pose error, including translational errors, since unnatural translation is a common (non-physical) reconstruction artifact. In addition, we also measure foot skating and the total variation in the joint acceleration per frame (TV). We measure foot skating as percentage of frames where a foot moves more than 2cm while in contact with the ground in two adjacent frames. Unlike [39], we do not assume foot contact annotations but instead heuristically detect foot contacts based on the distance between the foot mesh and the ground-plane. The total variation in acceleration is computed as $\frac{1}{T}\sum_{t\in T}\sum_{k\in K}|\ddot{x}^k_{t+1} - \ddot{x}^k_t|$, for the 3d acceleration $\ddot{x}^k_t$ of joint k at time t estimated using finite differences. Thus, high TV indicates motion jitter, and high foot skate implies motion that slides along the ground.

Implementation Details. We use the Tiny Differentiable Simulator [17] running at $1,000$ Hz with the gradients computed using the auto differentiation framework CppAD [3]. In addition, we use a Python implementation of Basin-Hopping and BFGS [50]. Since the length of the optimized trajectory may be great, we follow [1] and perform optimization in overlapping windows of length $N = 960$. The simulation steps take $\approx$ 5s. For the large datasets in Tab. 5.3, we compute the windows in parallel and stitch them together in order to speed up computation. We initialize the control targets $\hat{q}_{1:T}$ to 3d poses estimated by

Table 5.3: Quantitative evaluation on the Human3.6M and AIST datasets. Our full dynamic model improves over the kinematic estimates used as initialization with respect to standard joint position error metrics as well as reducing motion jitter and unnatural foot skating.

Dataset	Model	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d	TV	Foot skate (%)
Human3.6M	VIBE [24]	207.7	68.6	43.6	16.4	0.32	27.4
	PhysCap [41]	-	97.4	65.1	-	-	-
	SimPoE [60]	-	**56.7**	**41.6**	-	-	-
	Shimada *et al.* [43]	-	76.5	58.2	-	-	-
	Xie *et al.* [56]	-	68.1	-	-	-	-
	Kinematics	145.3	83.0	55.4	13.4	0.34	47.5
	DiffPhy	**139.1**	81.7	55.6	**13.1**	**0.20**	**7.4**
AIST	Kinematics	155.7	107.4	66.9	**10.4**	0.52	50.9
	DiffPhy	**150.2**	**105.5**	**66.0**	12.1	**0.44**	**19.6**

our kinematics. See our supplementary material for details.

4.1 Results

We compare DiffPhy against both state-of-the-art kinematic video models (VIBE [24]) and against physics-based methods. The results are summarized in Tab. 5.3. Since VIBE predicts root-relative poses, we estimate the global translation (required to compute MPJPE-G) by minimizing 2d projection errors using a method similar to the one in [41]. For VIBE, we use the publicly available implementation. For the other methods, we give numbers presented by the authors. On both Human3.6M and AIST, our model improves with respect to the physical metrics (TV and foot skate) compared to the kinematic initialization. On Human3.6M, foot skating is only 7.4% compared to 47.5% for the kinematic initialization and 27.4% for VIBE. On AIST, foot skating is reduced from 50.9% to 19.6%. We believe that increased skating on AIST is due to actual skating motions performed as part of the hip-hop dances. On total variation, our model similarly improves over kinematics with 0.20 and 0.44 on Human3.6M and AIST, respectively. Furthermore, we note that our full model improves the global joint position error (MPJPE-G), a metric that measures pose and translation errors. On Human3.6M, DiffPhy has an error of 139.1 compared to 145.3 and 207.7 mm/joint for kinematics and VIBE, respectively. If we look at the error for foot joints only, we see an even larger improvement by including physics compared to kinematics alone (166.8 vs. 174.1 mm/joint). This result aligns with prior work [39], showing that physics improves foot position estimation. Furthermore, our method aligns well with image evidence when comparing 2d error, i.e., 13.1 px/joint vs. VIBE's 16.4 px/joint on Human3.6M. In terms of joint error including translation error (MPJPE), SimPoE [60], Xie *et al.* [56], Shimada *et al.* [43] outperform DiffPhy (56.7 vs. 68.1 vs. 76.5 vs. 81.7 mm/joint respectively), though in the case of SimPoE and Xie *et al.* this might stem from initializing from the already strong VIBE predictor (68.6 mm/joint). Furthermore, SimPoE is a neural network requiring extensive training using the 3d ground-truth from Human3.6M, whereas DiffPhy is a general method that requires no additional training (*cf.*

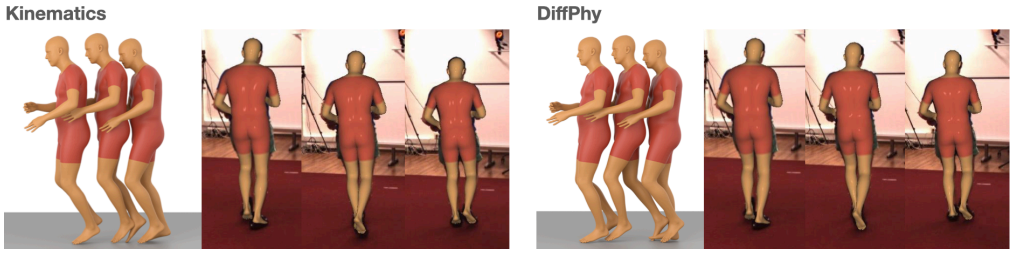


Figure 5.5: Qualitative examples on Human3.6M. DiffPhy infers plausible leg motion while kinematics skates unrealistically forward.

Tab. 5.1). Xie *et al.*, PhysCap [41], Shimada *et al.* [43] on the other hand, focus only on feet-ground contacts while DiffPhy supports complex contacts. Unfortunately, this advantage cannot be demonstrated on subsets that exclude sequences with complex contacts.

Fig. 5.5 presents qualitative results where kinematics fails to estimate the positions of legs due to depth ambiguities. The reconstructed poses align well when projected into the image but are unrealistic since the model skates rather than walks forward. Since DiffPhy reconstructs the motion with physics in the loop, it must propel the model forward through bipedal locomotion, thus inferring feasible leg poses. Similarly, in Fig. 5.4 kinematics estimates a pose that projects well into the image. However, when viewed from a side, it becomes clear that kinematics estimates a pose that leans unnaturally. Since DiffPhy is constrained by gravity, it must find a pose that is both physically plausible *and* aligns with 2d evidence. Fig. 5.4 also includes examples of object interactions and rolling motions requiring complex contacts. We manually modeled the chair as a box since DiffPhy does not estimate scene geometry. For the rolling motion, the kinematics were too noisy for DiffPhy to converge; hence, we manually corrected the worst kinematic frames before running DiffPhy. Finally, Fig. 5.2 shows two reconstruction examples for sequences in-the-wild. These videos exhibit poses and activities missing from standard laboratory-captured datasets.

Ablation studies. In Tab. 5.6 we validate our choice of loss components in (5.5). We note that the 2d projection loss, as expected, plays an important role in aligning the reconstruction with the image evidence (17.1 vs. 12.6 px/joint). Furthermore, since 2d keypoints do not suffer from depth ambiguities, they are generally more reliable than 3d keypoints and thus serve as a strong signal. Therefore removing 2d evidence significantly increases MPJPE-G from 144.9 to 158.5 mm/joint. Removing the root position loss (5.6) has the largest impact on global position error (165.7 mm/joint) since without it, we do not provide DiffPhy with any supervision with respect to world positioning. This allows for suboptimal reconstructions that align well with the projected image (12.8 px/joint) but do not transition correctly in world space. Without the joint angle loss (5.7), DiffPhy is deprived of the per-frame 3d pose estimates, which, when predicted by neural networks such as HUND or VIBE that are trained on large pose datasets, provide useful guidance

as long as their predictions do not contradict any physical constraints. Removing the joint angle limit regularizer (*cf.* (5.9)) demonstrates the usefulness of constraining the reconstructed motion to the space of anatomically valid poses even for everyday motions like those in Human3.6M. Finally, we validate the usefulness of optimizing the initial starting pose and velocity (see §3.6). Without it, the kinematic estimates for the initial frames must be accurate. If not, the simulation may start from an initial state from which DiffPhy may fail to recover, as seen by the largest MPJPE-PA in the ablation of 65.1 mm/joint.

Next, results in Tab. 5.2 show that gradient-based methods are vastly more efficient for our physics loss compared to the commonly used gradient-free approach CMA-ES [16]. BFGS obtains a lower MPJPE-G error (160.1 vs 206.7 mm/joint), and requires a fraction of the computations (122 vs. 80k loss evaluations per windows). Next, We note that BFGS converges to suboptimal minima, but by combining BFGS with Basin-Hopping, we can reduce the errors further to 144.9 mm/joint. As Basin-Hopping can explore infinitely many basins, we set the limit to 5 basin steps, each with 50 BFGS iterations as a trade-off between accuracy and speed.

In Tab. 5.5 we study the effect of the optimization window size. We find that a window of 960 simulation steps (containing 0.96s of video) is optimal for our setup. A larger window size increases the errors, most likely due to a larger search space combined with a larger gradient variance, as noted in [33]. On the other hand, smaller windows provide scarcer visual evidence and are sensitive to a few occluded frames, or to noisy estimates. Interestingly, a smaller window size performed better for experiments on ground-truth data (see supplementary material). This indicates that smaller apertures are better for noise-free inputs.

Several methods (*cf.* Tab. 5.1) introduce "residual forces" acting on the root link of the physical body. This non-physical force allows the method to translate and rotate the body to align with visual evidence at the expense of physical realism. Tab. 5.4 confirms that this indeed can be used to lower DiffPhy's joint errors (MPJPE-G from 144.9 to 140.2 mm/joint and 2d error from 12.6 to 11.6 px/joint when applying 50N for each of the six degrees of freedom). Interestingly, applying a too great residual force (100N) increased error, perhaps since it allows the model to circumvent some of the constraints of physical simulation. In this work, we avoid using residual forces, in order to keep all forces realistic, and avoid non-physical artifacts.

5 Discussion

In order to improve the realism of 3d human sensing, we have introduced *DiffPhy* – the first differentiable physics-based model for full-body articulated human motion estimation, that

Table 5.4: Results on experiments on the effects of residual force. We note using a residual force decreases the error metrics, but we refrain from using it to avoid unexplained non-physical forces.

RF	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d
0	144.9	84.6	61.1	12.6
5	141.4	82.2	60.7	11.8
10	140.1	79.9	60.2	11.7
25	146.3	81.9	60.0	12.7
50	140.2	79.4	60.3	11.6
100	154.0	87.7	61.5	14.4

Table 5.5: Results on the effects of optimization window size. A balance needs to be found between a larger window size which allow for more visual evidence to be taken into account while a smaller reduces the dimensionality of the search space.

Window	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d
240	390.1	224.1	96.6	40.3
480	165.6	97.2	63.8	13.2
720	148.9	87.2	61.8	12.6
960	144.9	84.6	61.1	12.6
1440	155.6	92.5	65.7	15.9

Table 5.6: Ablation of the model components introduced in §3. *No root* means without root position loss (5.6), *No 2d* without 2d keypoint loss (5.8), *No pose* without joint angle loss (5.7), *No 3d loss* without both root link position loss and joint angles losses, *No limit* without anatomical joint limits (5.9), and *No init. opt.* is without optimizing the initial state, *cf.* §3.6.

Variant	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d
Full model	144.9	84.6	61.1	12.6
No root	165.7	84.8	60.7	12.8
No 2d	158.5	98.3	65.7	17.1
No pose	156.8	91.8	64.4	13.0
No 3d loss	216.6	122.4	76.3	12.6
No limits	146.8	86.5	62.2	13.0
No opt. init.	151.5	92.1	65.1	14.1

supports complex contacts, does not assume a known ground plane, and avoids reliance on non-physical forces. This has the benefit of a human model with realistic physics interactions, that are constrained end-to-end by visual losses. Furthermore, such a model can provide a valuable non-learning-based component, which is always valid, complementing the statistical kinematic prediction and optimization techniques prevalent in the current state of the art. Visual 3d human motion reconstruction experiments on multiple datasets demonstrate that our methodology is competitive with other state of the art physics-based approaches.

Limitations and Future Work. An inherent limitation to physics-based approaches is the need to model objects in the scene. We hope to address this challenge in future work

by integrating with 3d scene reconstruction techniques [4]. Ideally, we would be able to jointly optimize the control of the body and the world to match visual evidence. Another limitation is our current assumption of constant camera extrinsics. This limits our technique to videos captured using a static camera but can be easily relaxed. Finally, our reconstructions are limited to a single subject. Reconstructing multiple people interacting is interesting since these scenes are complex, and learning statistical models of interaction between humans is challenging [12]. A physics-based approach could help infer constraints and affordances.

Ethical Considerations. Our construction of physics-based models is motivated by the breadth of transformative 3d applications that would become possible, including fitness, personal well-being or special effects, or human-computer interaction, among others. In contrast, applications like visual surveillance and person identification would not be effectively supported, given that the model's output does not provide sufficient detail for these purposes. The same is true for the creation of potentially adversely-impacting deepfakes, as an appearance model or a joint audio-visual model are not included for photorealistic visual and voice synthesis. While our method is fundamentally applicable to a variety of human body types, we have not evaluated this aspect extensively and consider such a study an important objective for future work.

References

[1] Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann. Trajectory optimization for full-body movements with complex contacts. In *IEEE transactions on visualization and computer graphics*, volume 19, pages 1405–14, 08 2013. doi: 10.1109/TVCG.2012.325. 188, 191

[2] Mazen Al Borno, Ludovic Righetti, Michael J. Black, Scott L. Delp, Eugene Fiume, and Javier Romero. Robust Physics-based Motion Retargeting with Realistic Body Shapes. In *Computer Graphics Forum*, 2018. 183, 184, 188

[3] B. Bell. Cppad: a package for c++ algorithmic differentiation, 2021. URL `https://projects.coin-or.org/CppAD`. 191, 207

[4] Bharat Lal Bhatnagar, Xianghui Xie, Ilya Petrov, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Behave: Dataset and method for tracking human object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2022. 196

[5] Zhe Cao, Hang Gao, Karttikeya Mangalam, Qi-Zhi Cai, Minh Vo, and Jitendra Malik. Long-term human motion prediction with scene context. In *European Conference on Computer Vision*, pages 387–404. Springer, 2020. 184

[6] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019. 184

[7] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019. 184

[8] Rishabh Dabral, Anurag Mundhada, Uday Kusupati, Safeer Afaque, Abhishek Sharma, and Arjun Jain. Learning 3d human pose from structure and motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 668–683, 2018. 183

[9] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf`. 182

[10] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 0387743146. 182

[11] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 0387743146. 187

[12] Mihai Fieraru, Mihai Zanfir, Teodor Szente, Eduard Bazavan, Vlad Olaru, and Cristian Sminchisescu. Remips: Physically consistent 3d reconstruction of multiple interacting people under weak supervision. *Advances in Neural Information Processing Systems*, 34, 2021. 196

[13] Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, NY, USA, 1987. 189

[14] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL `http://github.com/google/brax`. 184

[15] Erik Gärtner, Mykhaylo Andriluka, Hongyi Xu, and Cristian Sminchisescu. Trajectory optimization for physics-based reconstruction of 3d human pose from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 183, 184, 185, 186

[16] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*, pages 75–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32494-2. doi: 10.1007/3-540-32494-1_4. URL `https://doi.org/10.1007/3-540-32494-1_4`. 184, 194

[17] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL `https://github.com/google-research/tiny-differentiable-simulator`. 182, 183, 184, 186, 191, 207

[18] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 68–84, 2018. 183

[19] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019. 183, 184

[20] Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018. doi: 10.1109/LRA.2018.2792536. 187

[21] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (7):1325–1339, jul 2014. 183, 190, 204, 205, 206

[22] H. Joo, T. Simon, and Y. Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8320–8329, 2018. doi: 10.1109/CVPR.2018.00868. 183

[23] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 182

[24] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. 182, 183, 192

[25] Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3(22):500, 2018. doi: 10.21105/joss.00500. URL https://doi.org/10.21105/joss.00500. 184

[26] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Learn to dance with aist++: Music conditioned 3d dance generation, 2021. 205

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015. 183

[28] Zhengyi Luo, Ryo Hachiuma, Ye Yuan, and Kris Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. *Advances in Neural Information Processing Systems*, 34, 2021. 183

[29] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, October 2019. 183

[30] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 183

[31] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. In *2018 International Conference on 3D Vision (3DV)*, pages 120–130. IEEE, 2018. 183

[32] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. 189

[33] Luke Metz, C. Daniel Freeman, Samuel S. Schoenholz, and Tal Kachman. Gradients are not all you need, 2021. 184, 194, 207

[34] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311. 184

[35] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. In *SIGGRAPH*, 2018. 184

[36] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 2017. 188

[37] Stanley Plagenhoef, F Gaynor Evans, and Thomas Abdelnour. Anatomical data for analyzing human motion. *Research quarterly for exercise and sport*, 54(2):169–178, 1983. 188

[38] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pages 8661–8671. PMLR, 2021. 184, 207

[39] Davis Rempe, Leonidas J. Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. Contact and human dynamics from monocular video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 181, 182, 183, 184, 185, 191, 192

[40] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021. 184

[41] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics*, 39(6), dec 2020. 181, 182, 183, 184, 185, 191, 192, 193

[42] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics*, 40(4), aug 2021. 183

[43] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick P'erez, and Christian Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics (TOG)*, 40:1 – 15, 2021. 183, 184, 185, 192, 193

[44] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3d human tracking. In *CVPR*, 2003. 185

[45] Jakub Stepien. *Physics-Based Animation of Articulated Rigid Body Systems for Virtual Environments*. PhD thesis, 10 2013. 182

[46] David Stewart and J.C. (Jeff) Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. volume 1, pages 162–169, 01 2000. doi: 10.1109/ROBOT.2000.844054. 187

[47] Jakub Stępień. *Physics-Based Animation of Articulated Rigid Body Systems for Virtual Environments*. PhD thesis, Silesian University of Technology, 2013. 187

[48] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109. 184

[49] Shuhei Tsuchida, Satoru Fukayama, Masahiro Hamasaki, and Masataka Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 501–510, Delft, Netherlands, November 2019. 190, 204, 206

[50] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2. 191

[51] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision (ECCV)*, sep 2018. 183

[52] David J. Wales and Jonathan P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The*

Journal of Physical Chemistry A, 101(28):5111–5116, 1997. doi: 10.1021/jp970984n. URL https://doi.org/10.1021/jp970984n. 189

[53] Keenon Werling, Dalton Omens, Jeongseok Lee, Ionnis Exarchos, and C. Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *ArXiv*, abs/2103.16021, 2021. 183, 184

[54] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39(4), 2020. URL https://doi.org/10.1145/3386569.3392381. 184

[55] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.01122. URL http://dx.doi.org/10.1109/CVPR.2019.01122. 182

[56] Kevin Xie, Tingwu Wang, Umar Iqbal, Yunrong Guo, Sanja Fidler, and Florian Shkurti. Physics-based human motion estimation and synthesis from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11532–11541, October 2021. 181, 183, 184, 185, 192, 204

[57] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. GHUM & GHUML: Generative 3d human shape and articulated pose models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6184–6193, 2020. 183, 185, 188, 190

[58] Xinyu Yi, Yuxiao Zhou, and Feng Xu. Transpose: Real-time 3d human translation and pose estimation with six inertial sensors. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459786. URL https://doi.org/10.1145/3450626.3459786. 184

[59] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In *Advances in Neural Information Processing Systems*, 2020. 183

[60] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 183, 184, 185, 190, 192

[61] Andrei Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes-the importance of multiple scene constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2148–2157, 2018. 183

[62] Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Neural descent for visual 3d human pose and shape. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 183, 185, 186

[63] Mihai Zanfir, Andrei Zanfir, Eduard Gabriel Bazavan, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Thundr: Transformer-based 3d human reconstruction with markers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. 182, 183

[64] Jason Y. Zhang, Sam Pepose, Hanbyul Joo, Deva Ramanan, Jitendra Malik, and Angjoo Kanazawa. Perceiving 3d human-object spatial arrangements from a single image in the wild. In *European Conference on Computer Vision (ECCV)*, 2020. 184

Supplementary Material

This supplement presents addition results (§A), a description of the datasets used (§B) together with a description of the usage of data with human subjects (§B.2), and additional details of the simulation setup (§C). Please refer to our video for qualitative results at tiny.cc/diffphy.

A Additional Results

Tab. A.7 presents an ablation on window size performed using mocap data as initialization and reference trajectory rather than using the kinematic initialization. In this case, we note that a smaller window size of 480 outperforms the larger window size of 960 used in the main paper. We hypothesize that when the reference signal lacks noise, a smaller window is easier to optimize since the dimension of the problem is reduced. However, with noisy observations, a larger window is required for the method to be robust to missing or poor kinematic reconstructions.

Table A.7: Ablation study of the optimization window size. Experiments were carried out on motion capture rather than the kinematic initialization as input. The experiment was performed on the same Human3.6M sequences as in the ablation in the main paper. Note that when using mocap rather than noisy observations, a smaller window size is better (480 vs. 960 in main paper).

Window	MPJPE-G	MPJPE	MPJPE-PA
240	112.8	75.9	40.1
480	39.4	33.4	21.9
720	46.1	42.1	29.4
960	77.8	68.4	44.9

B Datasets

We evaluate our method on the two established datasets Human3.6M [21] and AIST [49]. In addition, we evaluate our method on "real-world" internet videos.

Human3.6M. When comparing to the state-of-the-art methods, we evaluate on the Human3.6M Protocol P2 sequences while excluding the same sequences as by Xie et al. [56]. That leaves the sequences: *Directions, Discussions, Greeting, Posing, Purchases, Taking Photos, Waiting, Walking, Walking Dog and Walking Together*. We evaluate the motions using only camera *60457274*. Similar to [56], we down sample the Human3.6M data from 50 FPS to 25 FPS.

Table B.8: Human3.6M [21] sequences used for ablation studies. Note that we downsampled the sequences from 50 FPS to 25 FPS.

Sequence	Subject	Camera Id	Frames
Phoning	S11	55011271	400-599
Posing_1	S11	58860488	400-599
Purchases	S11	60457274	400-599
SittingDown_1	S11	54138969	400-599
Smoking_1	S11	54138969	400-599
TakingPhoto_1	S11	54138969	400-599
Waiting_1	S11	58860488	400-599
WalkDog	S11	58860488	400-599
WalkTogether	S11	55011271	400-599
Walking_1	S11	55011271	400-599
Greeting_1	S9	54138969	400-599
Phoning_1	S9	54138969	400-599
Purchases	S9	60457274	400-599
SittingDown	S9	55011271	400-599
Smoking	S9	60457274	400-599
TakingPhoto	S9	60457274	400-599
Waiting	S9	60457274	400-599
WalkDog_1	S9	54138969	400-599
WalkTogether_1	S9	55011271	400-599
Walking	S9	58860488	400-599

The ablation studies were performed on a smaller subset of four-second clips (frames 400-599) from a random camera, see Tab. B.8.

AIST. AIST provides dynamic dance motions not present in Human3.6M. We evaluate our method using the pseudo-ground-truth provided by [26]. We use the first four seconds (120 frames) using a randomly selected camera from the sequences in Tab. B.9.

Internet Videos. Finally, we perform qualitative evaluation of our method on internet videos made public under creative common licences.

B.1 Metrics

Total variation. We compute the total variation of the 3d joint acceleration as a measurement of the jitter in motion. This is given as

$$\frac{1}{T}\sum_{t\in T}\sum_{k\in K}|\ddot{x}^k_{t+1} - \ddot{x}^k_t|, \tag{10}$$

where $\ddot{x}^k_t$ is the 3d joint acceleration of joint k at time t. We estimate the acceleration through finite differences.

Foot skating. We track unnatural foot skating artifacts by measuring the percentage of frames where either foot is "skating" along the ground. Our formulation doesn't rely on foot contact annotations but instead heuristically detect when foot contacts occur by measuring the distance between the foot mesh and the ground-plane. A contact is defined as $N = 10$ foot mesh vertices being within d mm of the ground-plane. For kinematics we use $d = 5$ mm and for dynamics $d = 1$ mm to account for the capsule approximation being smaller than the foot mesh. We define skating as a foot moving ≥ 2 cm between two frames while being in contact with the ground.

B.2 Usage of data with human subjects

In this work, we employ two established pose benchmarks that are commonly used in the field of human pose estimation. Human3.6M [21] was recorded in a laboratory setting with the permission of the actors, and AIST [49] contains *"a shared database containing original street dance videos with copyright-cleared dance music. This is the first large-scale shared database focusing on street dances to promote academic research regarding Dance Information Processing"*[1]. As for the "in-the-wild" videos, these were released under creative common licenses granting express permission to *"copy and redistribute the material in any medium or format"* and *"remix, transform, and build upon the material for any purpose, even commercially"*. Finally, we do *not* intend to release these videos as part of a dataset. Instead we only use them to demonstrate our method on videos with poses and motion uncommon in laboratory captured datasets.

Table B.9: AIST [49] sequences used for evaluation.

Sequence	Frames
gBR_sBM_c06_d06_mBR4_ch06	1-120
gBR_sBM_c07_d06_mBR4_ch02	1-120
gBR_sBM_c08_d05_mBR1_ch01	1-120
gBR_sFM_c03_d04_mBR0_ch01	1-120
gJB_sBM_c02_d09_mJB3_ch10	1-120
gKR_sBM_c09_d30_mKR5_ch05	1-120
gLH_sBM_c04_d18_mLH5_ch07	1-120
gLH_sBM_c07_d18_mLH4_ch03	1-120
gLH_sBM_c09_d17_mLH1_ch02	1-120
gLH_sFM_c03_d18_mLH0_ch15	1-120
gLO_sBM_c05_d14_mLO4_ch07	1-120
gLO_sBM_c07_d15_mLO4_ch09	1-120
gLO_sFM_c02_d15_mLO4_ch21	1-120
gMH_sBM_c01_d24_mMH3_ch02	1-120
gMH_sBM_c05_d24_mMH4_ch07	1-120

[1] https://aistdancedb.ongaaccel.jp/

C Differentiable Physics for Human Motion

Tiny Differentiable Simulator (TDS) [17] is a C++ simulator where the data type is templetized. In our experiments, we use the scalar from the automatic differentiation (AD) framework CppAD [3] to compute the simulation gradients. That is, we compute the gradients of the loss with respect to the input control variables at each time step:

$$\frac{\partial L}{\partial \hat{\mathbf{q}}_{1:T}} = \frac{\partial L}{\partial \mathbf{q}_{1:T}} \frac{\partial \mathbf{q}_{1:T}}{\partial \boldsymbol{\tau}_{1:T}} \frac{\partial \boldsymbol{\tau}_{1:T}}{\partial \hat{\mathbf{q}}_{1:T}}, \tag{11}$$

where L is objective function of the trajectory optimization, $\mathbf{q}_{1:T}$ are the simulated body's joint positions, and $\hat{\mathbf{q}}_{1:T}$ are the per-timestep control signal to the PD controllers in the body joints.

To speed up the optimization we implement our simulation as a fixed computational graph of the simulation rollout for a fixed number of steps and then repeatedly use it to compute the values of the gradients in (11). This greatly speeds up the optimization since the automatic differentiation framework doesn't need to setup the computational graph for each backward pass. To that end, we make the following adaptations to TDS to make it support a fixed graph.

Differentiation and contact points. Since at the time of graph construction it is not known in advance which contact points will be active for particular inputs we always include all contact points into the LCP formulation. This increases the graph size based on the number of contacts considered. The issue of large graph can be address by e.g. "checkpointing" the computation as described in [38].

Dealing with exploding gradients. As noted in [33], gradients from differentiable simulators may explode or vanishing when the window size is large. In this work, we experimentally found it possible to mitigate the issue by setting the LCP solver iterations to $K = 1$ without noticeable degradation of reconstruction quality.

Implementation Details In our experiments we run TDS with a step size of 1ms. This is partly due to the simpler PD controller, which requires smaller simulation steps to allow for stable control. We set the ground-plane friction to 0.8 and the controller gains to $k_p = 200$ and $k_d = 5$. Evaluating our loss function and computing the gradients for a window of 960 simulation steps takes approximately ≈ 5 seconds on a standard desktop computer with only feet contacts enabled. Enabling more contacts or simulating multiple objects increases memory and computation time.

Printed by Media-Tryck, Lund 2023 NORDIC SWAN ECOLABEL 3041 0903

Department of Computer Science, Lund University
Box 118, SE-221 00 Lund, Sweden

ISBN 978-91-8039-471-0
ISSN 1404-1219
Dissertation 70, 2023
LU-CS-DISS: 2023-01