Utvärdering av objektspårningsalgoritmer i realtid med en rörlig robot

Evaluation of Real-Time Single-Object Tracking Algorithms in a Non-Stationary Robotic Agent

Pierre Klintefors

handledare / supervisor
Birger Johansson

KOGM21

2021-06-03

Masteruppstats i kognitionsvetenskap
Avdelningen i kognitionsvetenskap
Filosofiska instutitionen
Lunds universitet

Master's thesis (2 years) in Cognitive Science
Lund University Cognitive Science
Department of Philosophy
Lund University

Evaluation of Real-Time Single-Object Tracking Algorithms in a Non-Stationary Robotic Agent

Pierre Klintefors
pierre.klintefors@gmail.com

Visual object tracking is a fairly easy task for humans but a challenging problem in computer vision and thereby in humanoid implementation. Most of the existing object tracking evaluations are performed with prerecorded video footage, often with a stationary camera. This is not representative of a humanoid platform. The aim of the present thesis was to evaluate different object tracking algorithms' suitability for being implemented in a humanoid by testing the algorithms' performance in real-time using a non-stationary robotic agent. The results reflect a general trade-off between accuracy and computational cost. Kernelised correlation filters are depicted as a suitable choice for single-object tracking systems with limited computational power. Deep learning tracking algorithms is argued to be the better choice for systems with sufficient computational power.

1 Introduction

Tracking an object is a task that seems seamlessly easy for humans but is a difficult challenge in computer vision. Humans perform multiple objects tracking daily in dynamic environments such as: keeping track of the trajectories of pedestrians and other cars while driving, following multiple moving actors in sports events or when parents are monitoring their children in the playground (Zelinsky & Neider, 2008). The skill of tracking multiple objects seems to be fundamental in humans where even two-year-old infants display this ability (Cheng, Kaldy & Blaser, 2019) and it drastically improves from childhood to young adulthood (Ryokai, Farzin, Kaltman & Niemeyer, 2013). Object tracking is thereby a fairly autonomous process in adult humans that endeavours the process of targeting its cognitive underpinnings. In cognitive robotics, the lack of prerequisite knowledge in an artificial system can be utilised to investigate these types of autonomous cognitive processes. In the present thesis, the task of tracking objects will be approached by implementing tracking algorithms in a non-stationary camera to evaluate different methods to track an object.

Object tracking in humans

Humans have the ability to quickly shift between visual-spatial content through fast ocular saccades as well as track trajectories of moving objects through smooth ocular pursuits (Purves, 2012). In the context of object tracking, these two categories of ocular movements are governed by information about position and velocity (Rashbass, 1961). Furthermore, in order for the eye to follow an object smoothly and not lag, there is a need for anticipation where the future trajectories are predicted (Kowler, 1989).

The anticipatory element of tracking becomes particularly critical when considering the sensory delay that occurs in neural transmission while processing information. The delay is a consequence of axonal conduction in visual sensory processing and motoric processing of the internal commands to the musculoskeletal system (Purves, 2012). The prevailed sensorimotor delay entails that a system needs to make compensatory actions in order for real-time tracking to be possible (Carlton, 1981). Nijhawan (2008) argues that such compensation is unlikely to be performed solely by the motor system. Visual mechanisms and the organisation of the central nervous system probably contributes. When an object is in motion, time-varying visual information, such as variation of illumination, can be collected and utilised. The compensatory action to neural transmission delay does not have to take place only in late neural processes of sensory-motor interaction. Instead, feed-forward streams of the visual-spatial system could account for the necessary compensatory action resulting in a less lagged sensory signal to the later stages of motor processing. For example, in the act of catching a ball, the visual perception might alter the current perceived position to future predictions of the trajectory to synchronise with motor signals to position the hand instead of the motor system doing all compensatory work.

Approaching motor control as an inference problem has provided a possible explanation for how humans can track in real-time despite the sensory delay that occurs in neural transmission. Probable causation of sensory input can be predicted by intrinsic generative models and underlying mechanism that approximates Bayesian filtering (Perrinet, Adams & Friston, 2014). The suggestion of the brain performing probabilistic inference has resulted in Bayesian models of perception, learning and cognitive development (Knill & Pouget, 2004; Pitkow & Angelaki, 2017; Tenenbaum, Griffiths & Kemp, 2006; Tenenbaum, Kemp, Griffiths & Goodman, 2011). However, the concept of the brain performing probabilistic inference is not new, it was suggested by Helmholtz (1867) over 150 years ago.

Within the Bayesian brain hypothesis, it is postulated that the brain is trying to infer the causation of noisy sensory input. It is argued that the brain must handle the uncertainty obtained with noisy sensory data through mechanisms that approximate, if not perfectly resembles, Bayes optimal inference in order to get perceptual rep-

resentations of such noisy data (Knill & Pouget, 2004). In Bayesian inference, Bayes' theorem (1) is utilised to update conditional probabilities by newly collected information.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

In (1), A is the hypothesis given the observation B. The posterior density $P(A|B)$ can be obtained through dividing the product of the likelihood $P(B|A)$ and the prior $P(A)$ by the marginal likelihood $P(B)$, also termed as model evidence. However, perfect Bayesian reasoning inferred from the Bayes theorem is often not feasible for many real-life scenarios. The model evidence $P(B)$ distribution of such situations becomes complex where the likelihood needs to be marginalised for every possible state of the observation. This makes it intractable to calculate perfectly and it can only be approximated through a body of techniques termed variational Bayesian methods (Galdo, Bahg & Turner, 2020). The Bayesian approach to perception, learning and cognitive function has been useful in modelling how statistical regularities in natural images can be taken advantage of to reduce their complexity and ambiguity. By way of computer vision, these real-life statistics of natural images can be measured to create a basis of how to model the most probable properties of an object based on features of the captured image (Kersten & Yuille, 2003). Given the Bayesian approach of visual perception, computational strategies of artificial systems can be valuable in trying to model and understand human cognition (Allen & Friston, 2016). Furthermore, even if the brain would not perform explicit calculations of shifts in probability densities, the Bayesian approach remains valid. In updating beliefs, it is the Bayesian sampling process that is essential for biological systems, where context guides the choice of hypothesis and governs the sampling, rather than the process of ideal Bayesian reasoning (Sanborn & Chater, 2016).

Object tracking in computer vision

Object tracking has been portrayed as one of the key challenges in computer vision, essential for multiple areas such as autonomous vehicles, surveillance and robotics (Kothiya & Mistree, 2015; Verma, 2017). The challenge arises from changeable factors, such as illumination change, motion blur, deformation of shapes, scaling variations and occlusion, that can occur in a video when an object is in motion (Soleimanitaleb, Keyvanrad & Jafari, 2019). This complicates the association between different frames of the video footage because the appearance of the same object can be substantially different from one frame to another. Furthermore, tracking objects in real-time complicates the task even more where accuracy needs to be balanced with the computational load which decides how many frames per seconds the tracker algorithm can handle. This can be compared to the sensorimotor delays in humans, which limits the amount of information that can be processed within a small time frame. In other words, just as in biological systems, the computational load also highlights the importance of a predicting component in artificial tracking systems.

Object detection is a prerequisite for object tracking where the object needs to be differentiated from the background. A non-stationary camera, as often the case in robotics, further complicates the differentiation task between background and object because the background is less static compared to a stationary camera. In order to track an object's trajectory, its detected features must be associated between different frames of the video sequence that includes the same object (Kothiya & Mistree, 2015). To detect the object in each frame of the video is thereby not equivalent to tracking the object. If the object would be occluded in one frame and then reappear in another frame, a detection algorithm might classify this as an appearance of a new object.

In object tracking, the frames of the video sequence are compared with each other to estimate the likelihood of one object being the same object presented in a previous frame but in a different location (Verma, 2017). There are several approaches to this association problem in object tracking. Commonly used tracking methods can be categorised into the following five categories: feature-based tracking, segmentation-based tracking, estimation-based tracking, appearance-based tracking and learning-based tracking (Soleimanitaleb et al., 2019; Verma, 2017).

In feature-based tracking, unique features from the detection process are used to find an object with the most similar features in the subsequent frames. In segmentation-based tracking, the foreground, usually where the moving objects are located, is separated from the background through background subtraction. The background model can be constituted of neural networks that classify the characteristics of the background and continuously update these characteristics to catch changes in illumination and so forth (Soleimanitaleb et al., 2019). Estimation-based tracking formulates the tracking task as a state estimation problem where the objects are transformed to state vectors that represent the position, velocity and acceleration of the moving object. The changes in the state vectors can be estimated with Bayesian filters, where Kalman filters are often used if the state posterior density is assumed to be Gaussian and the particle filters are used if changes in state variables are assumed to be non-linear (see (Chen, 2003) for more information about Bayesian filtering). Appearance-based tracking targets to incorporate changes in the appearance of the object, which largely can be divided into kernel-based and patch-based (see Verma (2017) for a more elaborated description of kernel-based and patch-based appearance models). Finally, learning-based tracking methods are built upon trained classifiers and supervised machine learning methods to extract features and appearances from objects as well as background characteristics to evaluate if an object is a target or not. Supervised machine learning methods usually require offline training before being deployed, where the classifiers are often exposed to large sample sizes of the objects of interest. Supervised machine learning with neural networks can thereby complex and robust appearance models of an object but does not alter the model online (Emmert-Streib, Yang, Feng, Tripathi & Dehmer, 2020).

The presented categorisation of the tracking methods is quite coarse and done for the purpose of highlighting the commonly used methodology in object tracking. These approaches are not necessarily used independently, rater,

a single tracking algorithm usually contains a combination of the presented methods (Kothiya & Mistree, 2015; Soleimanitaleb et al., 2019; Verma, 2017). The categorisation was done only for the purpose

Visual servoing

The term visual servoing was first coined by Hill (1979) and refers to when a closed system is controlled by visual feedback. Visual servoing can be used to control a robot's motions in real-time based on the visual information captured from one or several cameras. There are two main approaches for visual servoing, termed position-based visual servoing and image-based visual servoing (Corke & Hutchinson, 2000). In position-based visual servoing, features from the captures image are extracted to reconstruct a representation in Euclidean space where geometric calculations can be used to estimate the target position. In image-based visual servoing, the view directly decides the motion, if the view of an object matches the desired view of that object then the object is in the correct position (Hutchinson, Hager & Corke, 1996). Visual servoing has been a successful method for controlling the motion of an end-effector in humanoid for targeting and grasping behaviour (Ardón, Dragone & Erden, 2018; Vahrenkamp et al., 2008). Systems with visual servo control are thereby dependent on the quality of the visual feedback. If tracking algorithms are to be used to control movement they need to be accurate in their estimation of an object's position.

Aim of thesis

The aim of the present thesis is to implement a set of different tracking algorithms in a system with a non-stationary camera in order to evaluate their suitability for humanoid implementation. The evaluation will be based on specific performance metrics that target appropriate properties of interest for a humanoid in an object tracking task. These include the tracking algorithms' suitability to function as visual servo control as well as their robustness to occlusion and deflection. The majority of evaluations of object tracking algorithms are conducted on stationary cameras or with prerecorded footage (Kristan et al., 2016; Yin, Makris & Velastin, 2007; You, Zhu, Li & Li, 2019). The present thesis project evaluated tracking algorithms in a setting that aspired to resemble a humanoid platform. The algorithms were implemented in a mobile artificial agent that receives visual input to track an object and control the agent's movement in real-time. This setting is more appropriated, in terms of ecological validity, for evaluating the algorithms' suitability for humanoid implementation compared to the conventionally conducted object tracking evaluations.

2 Method

To achieve the aim of the present thesis project, the tracking algorithms needed to be evaluated in an appropriate context where parameters of interest could be isolated. The subsequent sections describe the used algorithms and the experimental setting where the algorithms were tested. The full code that was used for the experiments along with installation instructions can be found in the following GitHub-repository: `https://github.com/pierreklintefors/MasterThesisObjectTracking`

Used tracking algorithms

The tracking algorithms that were tested are specified and briefly described in this section. The reader is advised to follow the referred original papers in the aspiration to learn more about the technical details of the algorithms. This paper will only briefly touch upon the used methods of each algorithm. The trackers: MOSSE, CSRT, GOTURN, KCF, MEDIAN FLOW, TLD and MIL were part of the main and extra modules of the OpenCV library (version 3.18) (Bradski, 2000). The YOLOv4-Deep SORT tracker is a combination of the fourth iteration of "You Only Look Once" detection algorithm (Bochkovskiy, Wang & Liao, 2020) and the tracking algorithm "Simple Online Real-time Tracking" (SORT) (Bewley, Ge, Ott, Ramos & Upcroft, 2016) with a deep association metric (Wojke, Bewley & Paulus, 2017). To be suitable for humanoid deployment, a tracking algorithm needs to track an object at close to real-time speed, approximately 20 frames per second. This criterion was aspired to be fulfilled without utilising a high-end CPU or GPU which excluded many of the available, state of the art, deep learning trackers.

MOSSE stands for 'Minimum Output Sum of Squared Error' and was the first object tracking algorithm to use correlation filters. Correlation filters are classifiers that take features to be associated as input and returns peaks of correlation. In terms of object tracking, correlation filters are used to associate image features of one frame with a subsequent frame in a video sequence (for more information about correlation filters see (Liu, Liu, Srivastava, Połap & Woźniak, 2020)). MOSSE is argued to be robust against changes in illumination, scale as well as pose and is claimed to be able to update by a rate of 649 frames per second when run on a 2.4 GHz Intel Core 2 Duo Processor (Bolme, Beveridge, Draper & Lui, 2010).

CSRT is based on discriminative correlation filters and was created by Lukezic, Vojír, Zajc, Matas and Kristan (2017). The tracker works by having correlation filters trained on compressed features in form of histograms of oriented gradients and colour names. Histograms of gradients represent occurrences of change in the most intensive pixel values in the picture. Furthermore, the orientations of the gradients can be obtained through those changes and probable estimations of coherent objects can thereby be extracted. The correlation filters are aided by channel and spatial reliability maps that help the filter to track suitable parts of the object of interest. Lukezic et al. (2017) argues that the CSRT tracker can run close to real-time speed on a CPU.

KCF is a tracker based on kernelised correlation filters (Henriques, Caseiro, Martins & Batista, 2015). The KCF increases the number of samples for training the filters by applying cyclic shifts to every sample to strive for better learning. It utilises circulant matrices for speeding up linear ridge regression of the filters and also allows for non-linear regression by applying kernel trick. The KCF also uses histograms of oriented gradients instead of raw pixels to improve the accuracy of tracking.

MEDIAN FLOW is a tracker that uses forward and

backward error estimation in a point tracking algorithm (Kalal, Mikolajczyk & Matas, 2010). The algorithm predicts and tracks the trajectory of a set of points in the image that is selected from a bounding box. A forward and backward error estimation then filters out half of the predictions with the highest amount of error. Estimation of the new bounding box is done by taking the median of each spatial dimension of the remaining points.

TLD is an implementation of the Tracking-Learning-Detection framework that breaks down long-term tracking into tracking, learning and detection (Kalal, Mikolajczyk & Matas, 2012). This implementation of TLD uses the MEDIAN FLOW tracker described above. The idea is to use detection to establish an appearance model of the object from the previous frames which can potentially correct the tracker. The learning part of the algorithm estimates the errors made by the detection and updates the detection model to avoid these errors in future frames.

MIL stands for "Multiple Instance Learning" and is a tracking-by-detection-based algorithm (Babenko, Yang & Belongie, 2009). The tracker utilises multiple instances learning for an appearance model where a bag of several image patches around the contour of the object of interest updates the appearance model. This is combined with a simple motion model that estimates the trajectory of the object. Babenko et al. (2009) argues for MIL tracker being able to run at real-time speeds.

GOTURN stands for "Generic Object Tracking Using Regression Networks" and constitutes of convolutional neural networks that have been trained offline on a combination of videos and still images (Held, Thrun & Savarese, 2016). If a powerful GPU is utilised the GOTURN algorithm can be used to track objects at a speed of 100 frames per second and can still reach a speed of above 30 with a less powerful GPU.

YOLOv4-Deep SORT is a tracking-by-detection algorithm that utilises the convolutional neural network YOLOv4 (Bochkovskiy et al., 2020) to detect and classify an object. The detection network has been trained the COCO-dataset from Microsoft (Lin et al., 2015), containing 80 different classes of objects. The full weights of the pre-trained network make the detection algorithm quite computational heavy and require a powerful GPU to achieve real-time detection. As a result, a compressed version of the YOLOv4 network was used, termed YOLOv4-tiny, which runs significantly faster but with a cost regarding reduced accuracy. The YOLOv4-tiny was combined with SORT (Simple Online and Realtime Tracker) with a deep learning association metric (Wojke et al., 2017). This version of SORT uses deep convolutional networks to integrate appearance information and improves the association of the object's features between different frames. YOLOv4 was built and trained with the API Darknet (Redmon, 2013–2016) and had to be converted to a TensorFlow model to be compatible with the deep SORT. This was done by implementing a protocol from TheAIGuy (2020).

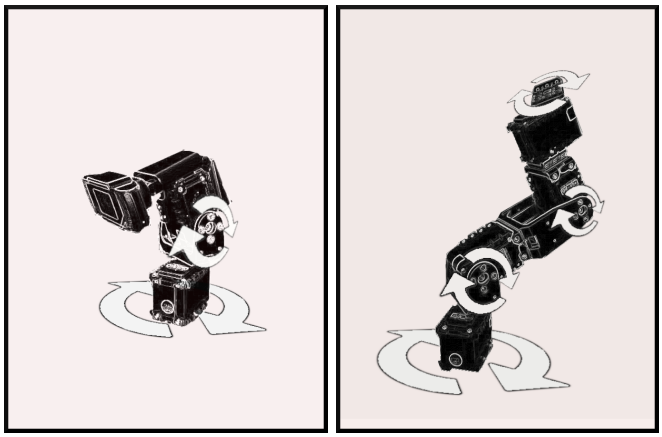
Experimental setting

The tracking algorithms were tested in an experimental setting constituted of two moving robotic agents, one tracker and one object mover. The tracker received visual input through a webcam, capable of capturing 21 frames per second with dimensions of 640 by 360 pixels. The camera was mounted on two Dynamixel AX-12 servomotors from Robotis. This gave the tracker two degrees of freedom, allowing it to pan and tilt to keep the object of interest centred in the image (see Figure 1a). The object mover constituted a chain of four Dynamixel AX-12 servomotors, corresponding to four degrees of freedom, where two motors were used for panning motion and two for tilting motion (see Figure 1b). This allowed the object to be rotated, which altered its appearance, as well as moved in two orthogonal directions with correcting rotation to keep the front of the object faced toward the tracker. The servomotors of both agents were connected in one chain to a computer via a USB2DYNAMIXEL device. The USB signals were converted to TTL (Transistor-Transistor Logic) signals before being sent to the servos which were connected to a 12 V power source. The experiments were conducted with a computer running Ubuntu 20.04, equipped with an AMD Ryzen 5 3600 6-core CPU and an NVIDIA GeForce GT 1030 GPU with 2-gigabyte memory.

The experimental setup aimed to control the lighting to avoid the difference in illumination between trials. This was done by darkening the room and using artificial lights positioned above the robotic agents. To make sure that the illumination was constant, the hue, saturation and lightness of the object were measured in the initiation of every trial. The background was a white wall decorated with a patterned and coloured wall sheet hanging approximately 50 centimetres above the object moving robotic arm. This made the background static for horizontal movements, with no changes in colour, but not for vertical movements where the camera angle included the wall sheet.

Experimental tasks

The algorithms were tested in two experimental tasks which aimed to capture situations that would be representative of a scenario where a humanoid is tasked to follow an object. The first experiment task involved visual servoing, where the tracking agent in Figure 1a was tasked



(a) Tracker **(b)** Object mover

Figure 1. The degrees of freedom of the robotic agents, (a) the tracker had two degrees of freedom, (b) the object mover had four degrees of freedom.

to keep the object's position centred in the image. The object was attached to the robotic arm in Figure 1b that moved in a predetermined choreography. The tracking of each trial was initialised by the manually annotated ground truth position of the object and visual servoing was then based on the algorithms' continued estimation of the object's position. The second experiment aimed to test if the tracking of the algorithm could handle deflection from the object. This was done by turning the camera away from the stationary object interim of tracking and then having it returned to the start position to see if the tracking was recovered.

Moving sequence of the object

The object that was used for experimental tasks was a small orange plastic cup. This single object was used continuously for all algorithms in both of the experimental tasks. This particular object was chosen because it needed to be an existing class of the COCO-dataset, in order to test the YOLOv4-Deep SORT algorithm, as well as being lightweight and easy to attach to the robotic arm.

The choreography for the object moving agent was designed in a way that would adequately cover a range of different movement and is displayed in Figure 2. The object started from a base position where it was aligned with the tracking agent, showing the front of the object at a distance of approximately 40 centimetres from the camera. The object moving robotic arm then performed a sequence of movements where the object returned to the base position between every transition. The object was first moved in linear trajectories, both in horizontal and vertical directions, which only changed the position of the object but not the appearance. As displayed by image 2 and 4 of Figure 2, these movements kept the front side exposed to the tracking agent's point of view. However, the horizontal linear movements of image 2 and 6 altered the scale of the object, moving it slightly closer to the camera due to the way these movements was executed. This is implied by the degrees of freedom displayed in Figure 1b, the object was moved horizontally by rotation of the bottom motor while keeping the other sections of the arm at angles of 90 degrees. In the second half of the moving sequence, the object was moved in ways that altered its appearance. First by partial occlusion by a screen, shown in image 6 of Figure 2, and then by rotating the object around its axes, shown in image 8 and 10.

Visual servoing

The motion of the tracking agent was controlled by the visual feedback from the camera, captured via the OpenCV library (Bradski, 2000), where pixel coordinates of the centre of the image and the centre of the tracked object were extracted. The motion of the tracking agent was determined by position-based visual servoing. The servomotors of the tracking agent received signals to rotate in ways that would pan and tilt the camera to minimise the distance between the image centre and the estimated object centre. The motors rotated different degrees dependent on the measured distance and stopped moving when the image centre and object centre was within a range of 30 pixels. By way of position-based visual servoing, the pixels of the image were represented in a Cartesian coordinate system where the measured distance on the x-axis and the y-axis controlled rotation of the pan and tilt motors respectively.

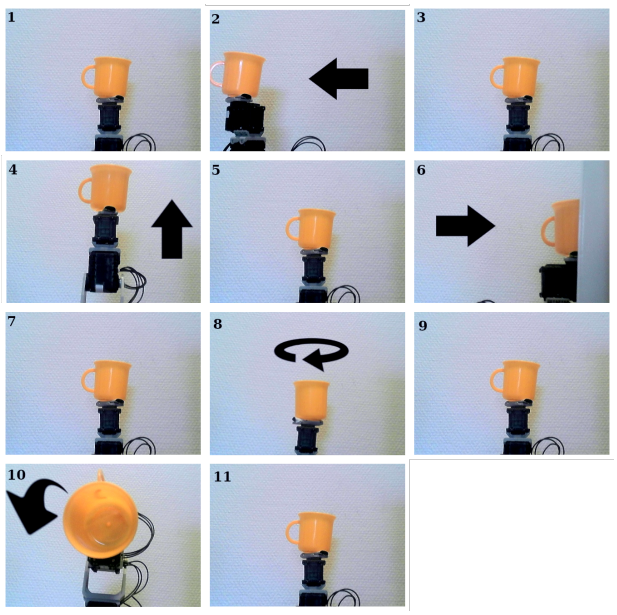


Figure 2. Moving sequence for object during one trial. The object returns to base position (1, 3, 5, 7) between each movement. The object was moved in ways that did alter the appearance of the object (6, 8, 10) and in ways that did not alter the appearance (2, 4)

Performance metrics

The algorithms were evaluated by the following performance metrics: proportion of frames the object was centred, computational efficiency (number frames processed per second), tracking accuracy, tracking loss frequency and the ratio between tracking loss and tracking recovery. The proportion of frames within the trial that the object was kept in the centre of the image served as a good overall performance metric of the visual servoing because it entailed spatial and temporal accuracy. The other metrics gave a more detailed picture of the shortcomings of the trackers. The tracking accuracy was obtained by comparing the overlap between the tracker's bounding box and the ground truth of the object's position. The ground truth constituted of manual post hoc annotation of the object's position in each frame from saved video footage of the trial.

3 Results

All the tracking algorithms were tested on the same moving sequence, which amounted to 41 seconds. Their performance was evaluated according to the metrics that were described in the previous section. The results are summarised in Table 1. Video footage of each trial can be found at: `https://github.com/pierreklintefors/MasterThesisObjectTracking/TrackingVideos`

Computational speed

By comparing the frames per second (FPS) rates in Table 1, the MOSSE tracker was the fastest, with an

Table 1. The table contains performance metrics of the tracker algorithms in the visual servoing task, the two best values of every metric are displayed in bold. Frames per second (FPS) is a measure of computational speed. Average precision is the average overlap between the algorithms' estimating bounding box and ground truth annotation. The region of interest (ROI) is defined as when the absolute distance in pixels was ≤ 30 along both the X-axis and Y-axis of the image. The proportion of frames with detectable object refers to the frames where the object is at least partially visible and not outside of the captured image.

Tracker	FPS	Average precision	Prop. of frames with ROI centred (≤ 30 px)
CSRT	12.23	**.75**	.46
GOTURN	10.68	.28	.52
KCF	14.39	**.66**	.47
MEDIAN FLOW	**15.97**	.32	**.85**
MIL	10.36	.26	.65
MOSSE	**16.26**	.60	**.70**
TLD	11.19	.46	.43
YOLOv4-Deep SORT	12.36	.18	.09
	Reported tracking losses	Recovered tracking	Prop. of frames with detectable object
CSRT	-	-	**1**
GOTURN	-	-	.49
KCF	1	-	**1**
MEDIAN FLOW	-	-	.47
MIL	-	-	.61
MOSSE	2	1	**1**
TLD	-	-	.98
YOLOv4-Deep SORT	19	19	.94

average of 16.26 FPS. Close behind, the second-fastest algorithm was MEDIAN FLOW with an average of 15.97 FPS. MIL and GOTURN were the slowest among the tracking algorithms with FPS rates of 10.36 and 10.68, respectively.

Spatial tracking accuracy

As shown in Table 1, CSRT and KCF were the two algorithms with the highest average precision per frame. The precision was measured in terms of how much the estimated bounding box of the tracking algorithm overlapped with the ground truth bounding box where 1 would be a perfect score. Furthermore, Figure 3 shows the success rate, in terms of the proportion of frames where the overlap exceeded a given threshold. CSRT, KCF and MOSSE were the only trackers that had 70% overlap for more than half of the frames during the visual servoing task. CSRT showed a small degree of overlap for a high proportion of frames but for more precise estimations, i.e higher overlap thresholds, KCF had a higher success rate in the estimation of the object's true position.

The divergences from the ground truth annotations are visualised in Figure 5. The axes of the coordinate systems correspond to the distance in pixels between the centre of the image and the x-coordinates as well as the y-coordinates of the object's centre. In every plot there are two paths, the blue corresponds to the divergence between the centre of the image and the estimated object centre, the green paths show the divergence from the ground truth annotations of the object's centre. A higher degree of overlap between the two paths corresponds to a more accurate estimation of the object's position. The paths are only visualised for frames when the object was visible and tracked. For example, MEDIAN Flow's low proportion of frames with a detectable object (47%) can be seen by the shorter length of paths in Figure 5.

Visual servoing

The visual servoing performance can be reflected by the proportion of frames where the region of interest (ROI) was centred in the image. The region of interest was defined as the centre point of tracked bounding box $\pm$ 30 pixels along both axes. In this regard, the MOSSE tracker kept the ROI centred in the image 70% of frames and KCF did so for 47% of the frames, as displayed in Table 1. MEDIAN FLOW reached the highest rate at 85%, however, MEDIAN FLOW had false positives where it lost track of the object but attached the bounding box to the static background, this is displayed in Figure 4. As a result, MEDIAN FLOW had a high divergence from the ground truth along the Y-axis, seen in Figure 5 as well as

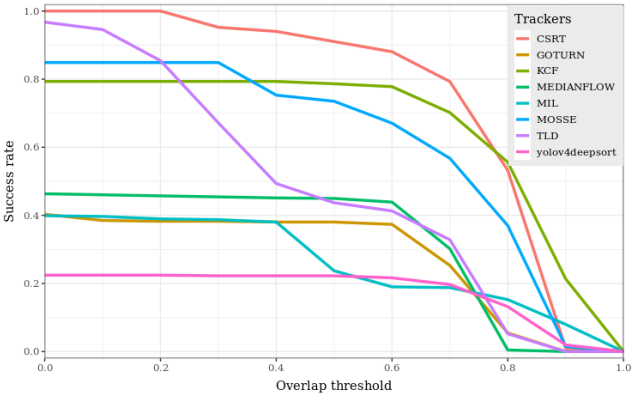


Figure 3. The proportion of frames where the algorithms' bounding box overlapped with the ground truth annotation, termed success rate, for different overlap threshold values.

the low proportion of frames where the object was detectable in Table 1. This was also the case with GOTURN, and MIL which both wrongfully attached the bounding box to the background, resulting in a low proportion of frames where the object was detectable.

Visual servoing performance is also reflected by how centred the blue path is around the origin and axes of Figure 5. Paths that are centred around the origin indicate a close correspondence to the tracking and the visual servoing. The blue paths are therefore more centred than the green paths because they relate to the tracked ROI which underpinned the visual servoing of the tracking agent. By looking at the MEDIAN FLOW, KCF and MOSSE plots of Figure 5 the close overlay between the blue paths and the axes of the coordinate systems are reflected by their high values in "Proportion of frames with ROI centred" in Table 1. However, just as the parametric of centring ROI, the central tendency of paths in Figure 5 does not entail if the object was lost. Therefore, the visual servoing parametrics ought to be combined with the average precision and proportion of detectable frames in Table 1 to get the correct representation of the trial.

Tracking loss

Tracking loss was measured in both the visual servoing task and the second task of tracking recovery. The bottom half of Table 1 shows how many times the tracking algorithms reported tracking loss during the visual servoing task and if they recovered tracking after the loss. Tracking loss, in this regard, corresponds to the algorithms' reports of losing the tracking and not when the bounding box is attached to the background as in the case with MEDIAN FLOW, shown in Figure 4. There were more examples of this phenomena during the visual servoing task which is reflected by 'Proportion of frames with detectable object' in Table 1. This parameter shows instances where the object no longer was visible within the frame due to tracking futility. This was either a result of a terminated tracking or erroneous tracking of another feature of the background. Together with MEDIAN FLOW, the GOTURN, MIL and TLD algorithms also failed to report when they lost track of the object, attaching the bounding box to something else.

In contrast, KCF, MOSSE and YOLOv4-Deep SORT successfully reported when the tracking was lost during the visual servoing task. The KCF lost track of the object in the last appearance altering move where the object was rotated along both its axes (see image 10 in Figure 2. The MOSSE algorithm lost track of the object two times and recovered the tracking from one of the times. The YOLOv4-Deep SORT algorithm lost track of the object most frequently but always recovered the

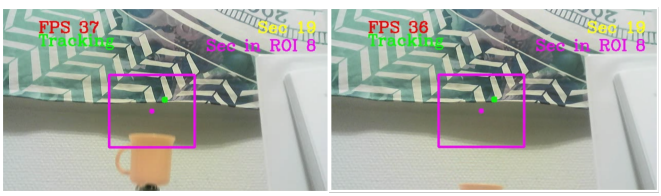


Figure 4. MEDIAN FLOW losing track of the object but keeps bounding box attached to the background.

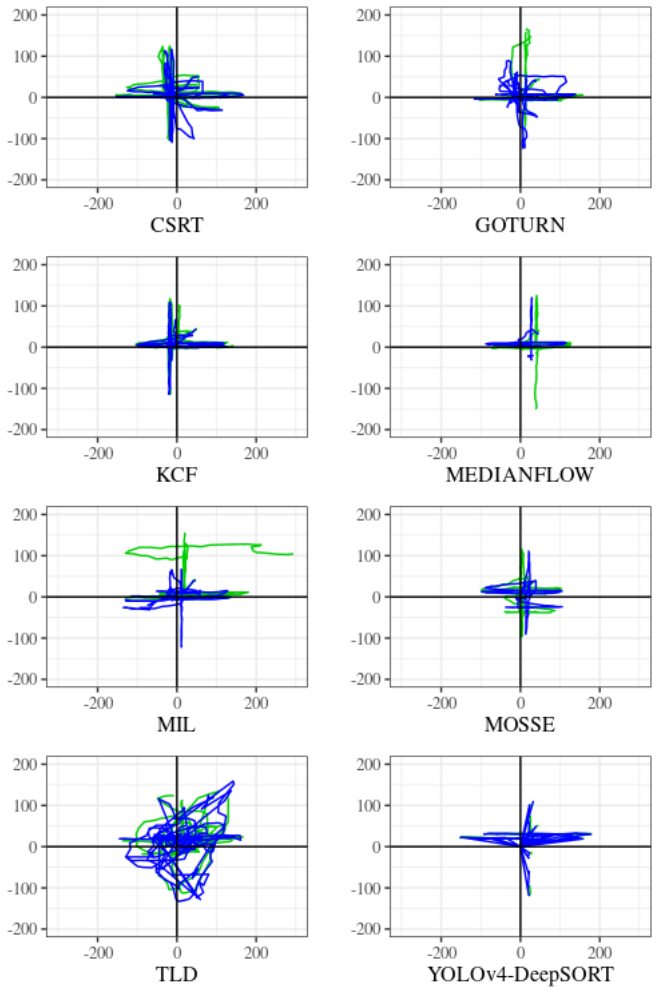


Figure 5. Coordinate systems that displays accuracy in the visual servoing task. The x-axis and y-axis corresponds to the x-axis and y-axis of the captured image where the origin is the centre. The blue path shows the difference between the image centre and the centre of the tracking bounding box. The green path shows the difference between the image centre and the centre of the ground truth annotated bounding box. Both of the trajectories displayed in the plots are measured and displayed in pixels. If the tracking trajectory and visual servoing would be perfect, both paths would be centred around the origin.

tracking. However, the association between the detection of the YOLOv4 network and the SORT metric failed on several occasion, resulting in that the algorithm classified the same object as multiple objects. This is illustrated in

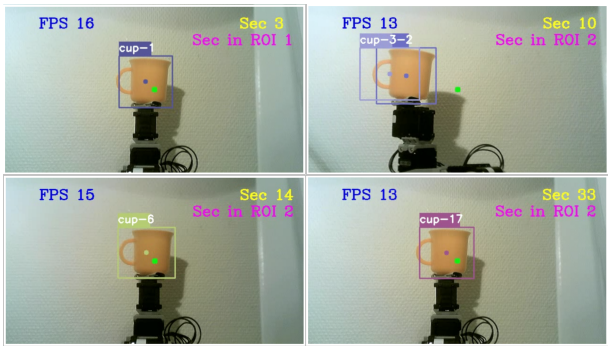


Figure 6. The YOLOv4-Deep SORT tracker erroneously classifies the same object as different objects.

Table 2. Reports of tracking loss and recovery of tracking in the turning away task. The time metric is the time it took, in milliseconds, to recover tracking from when the object was visible again.

Tracker	Reported	Recovered	Time
CSRT	Yes	No	-
GOTURN	No	No	-
KCF	Yes	Yes	590
MEDIAN FLOW	No	No	-
MIL	No	No	-
MOSSE	Yes	No	-
TLD	Yes	Yes	43
YOLOv4-Deep SORT	Yes	Yes	90

Figure 6. Furthermore, the CSRT was the only tracker that did not lose track of the object and kept it in sight throughout the whole trial.

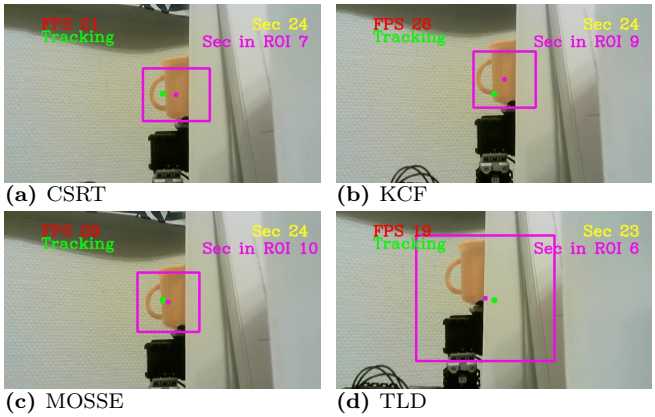
The result of the second task, which was aimed to measure the ability to recover lost tracking is summarised in Table 2. The performance of this task was measured with two dichotomous variables as well as a continuous variable of time. The first dichotomous variable conveyed whether the algorithms were successful or futile in reporting tracking loss when the object was no longer visible. The second dichotomous variable informs whether the tracking was recovered when the object became visible again. The time variable measured how long time it took for the algorithms to recover tracking after the tracking agent was returned to face the object again. The algorithms that were successful in both reporting tracking loss and recovering tracking were KCF, TLD and YOLOv4-Deep SORT, whereof TLD had the fastest recovery.

Occlusion

The moving sequence of the object mover in the visual servoing task included a movement where the object was partially occluded by a screen, as displayed in image 6 of Figure 2. The algorithms: CSRT, KCF, MOSSE and TLD were successful in tracking the object during its partial occlusion, which is shown in Figure 7. The bounding boxes of CSRT, KCF and MOSSE maintained a good representation of the object during the partial occlusion while TLD enlarged its bounding box. The algorithms GOTURN, MEDIAN FLOW, MIL and YOLOv4-Deep SORT lost track of the object before it was occluded.

4 Discussion

The tracking algorithms were evaluated based on their computational efficiency, tracking accuracy, visual servoing and recoverability. The fastest algorithm, in terms of the number of processed frames per second, was MOSSE. The algorithm with the highest tracking accuracy, in terms of estimating the real position of the object, was CSRT followed by KCF and MOSSE. Despite CSRT's high average tracking precision, it was not successful in keeping the object centred in the image, neither did it prove any strong recoverability. In contrast, the KCF, TLD and YOLOv4-Deep SORT algorithms were all able to recover from tracking loss in the turning away task, but YOLOv-Deep SORT also showed strong recoverability when accumulating the recovered tracking from the visual servoing task.



(a) CSRT **(b)** KCF **(c)** MOSSE **(d)** TLD

Figure 7. The four algorithms that handled occlusion, (a-b) CRST and KCF, respectively, tracking the partially occluded object with a tight bounding box, (c) the MOSSE tracker centre the tracking bounding box around the part of the object that is not occluded, (d) the TLD tracker expanded the bounding box when the object was partially occluded.

Based on these results, the discussion will mainly highlight and compare the KCF, MOSSE and YOLOv4-Deep SORT algorithms in order to emphasise their strength and weaknesses regarding humanoid implementation. It should be noted that the results from the present experiments were dependent on the used equipment and experimental setting. If the experiments were to be repeated with a system with more processing power, high-speed cameras or different image resolution the results might have been different. However, due to limited financial resources and arrangement space, it is not always feasible to use powerful CPUs, GPUs and high-speed cameras, especially in smaller or mobile robots. The results from this thesis project could thereby serve as an indicator of what type of algorithms perform well with equipment that is highly accessible and relative cheap in the present market.

Computational efficiency

The computational efficiency is an important criterion for tracking algorithms that would be implemented in a humanoid. If visual servoing is to be used for controlling movements, the chances of calculating the correct position of an object and the desired trajectory of a humanoid's end-effector are improved with increased processing capacity. It is important to note that the computational efficiency (FPS) obtained in the visual servoing task in the present thesis project is not a true measurement of the algorithms' computational speeds. This measurement was dependent on the used code for visual servoing and the computational speed can be increased by further code optimisation. However, the same code was used consistently for all algorithms and their results are therefore comparable with each other but not necessarily representative of their true capacity.

The MOSSE algorithm had the highest FPS rate and was also one of the top-performing algorithms in keeping the object centred in the image. This replicates previous results that have shown that MOSSE is faster than the other used trackers (Chen, Hong & Tao, 2015; Mi & Yang, 2019; Yue & Li, 2018; Zhang, Wang, Sun,

Yao & He, 2015). The difference in speed between the MOSSE algorithm and the deep-learning algorithms such as GOTURN and YOLOv4-Deep SORT can be accounted for by the use of simple correlation filters instead of advanced appearance models in the association of frames (Liu et al., 2020). Fast Fourier transformation is used with correlation filters to correlate features in the Fourier domain, which speeds up the calculations, and then use inverse Fourier transformation to acquire the spatial position. The MOSSE algorithm predicts the position of the next frame by taking the maximum correlation values from a search window and then performs an online update in that location (Bolme et al., 2010). The high FPS rate of MOSSE was probably a contributing reason to its high proportion of frames where the ROI was centred in the image. This can be contrasted to CSRT and KCF, both had higher average precision than MOSSE but not as successful in keeping the object centred. This could be a result of their slower processing capability.

Deep neural networks rely on large numbers of distributed and different weighted computational units, termed neurons. The combinatorial activations of those neurons, in terms of summation exceeding a threshold, can represent multiple complex features in an image (Emmert-Streib et al., 2020). The weights of the artificial neurons are tuneable parameters and calculated through simple matrix multiplication but in tremendous quantities. The huge amount of tunable parameters makes neural networks appropriate for building complex appearance models but also requires many computations to be done and a large amount of data to be kept in memory (Sze, Chen, Yang & Emer, 2017). For training (i.e tuning the parameters) deep neural networks often require parallel processing on GPUs to be feasible. Using a trained network for inference (i.e making predictions) can be done on both CPU and GPU but still requires more memory allocation compared to correlation filters with calculations in the Fourier domain. Even with the compressed deep network YOLOv4-tiny, which is one of the fastest detection networks available (Bochkovskiy et al., 2020), correlation filter-based trackers such as MOSSE and KCF were faster in the present experiment.

Tracking accuracy

Tracking accuracy entails both spatial and temporal accuracy and is crucial in order for a humanoid to successfully manoeuvre in relation to an object's position, both adjusting the angles of the cameras and orient its end-effectors in grabbing tasks. After CSRT, the algorithms with the highest tracking accuracy were KCF and MOSSE. Previous accuracy comparisons that included KCF and MOSSE algorithms have shown mixed results, but KCF has been reported most frequently to have the better average accuracy of the two (Chen et al., 2015; Mi & Yang, 2019; Yue & Li, 2018; Zhang et al., 2015).

The basis of KCF's success in tracking accuracy could be its robust learning of the object's appearance without a drastic increase in computational cost. KCF uses the initial frame of the captured image to create a positive sample of the object and then performs cyclic shifts to produce negative samples of the object (Henriques et al., 2015). Ridge regressions are then performed on the cyclic shifted samples in form of circulant matrices to model the appearance of the object. Circulant matrices have the property that enables diagonalisation which is used in a discrete Fourier transformation (Yadav & Payandeh, 2018). This enables more dense sampling without gaining too much computational load where the KCF could have a more complex object model and still have a relatively high FPS rate of 14.39, as seen in Table 1. This is noticeably faster than the other used tracking algorithms with complex appearance models such as the deep learning algorithms (GOTURN, YOLOv4-Deep SORT), as well as CSRT and TLD, with FPS rates of 10.68, 12.36, 12.23 and 11.19, respectively.

As its name entails, KCF uses kernelised correlation filters, which allows for a computationally efficient way of using non-linear regression functions. By way of non-linear regression, more complex relationships between features can be predicted, which makes it more powerful than linear regression. The way non-linear regression can be performed without the heavy computational cost is by the 'kernel trick' (Henriques et al., 2015). Kernel trick can be used to train a classifier to classify support vectors that are only linearly separable if they were to be transformed into higher dimensions. Transforming data that contains a lot of features into higher dimension requires multiple polynomial combinations of the features which quickly becomes impractical due to the computational cost. The kernel trick is a solution to that problem where a kernel function can calculate the dot product of higher transformed data using input from the original feature space. As a result, kernel functions can be used to obtain properties from the higher dimensional feature space without explicitly performing the computational costly transformations (Schölkopf & Smola, 2018).

The robust learning models and computationally efficient proprieties makes KCF algorithm a suitable choice for object tracking in a system with insufficient computational power for deep learning algorithms. However, with enough computational power deep learning algorithms tend to outperform KCF in terms of accuracy (Li, Wang, Wang & Lu, 2018; X. Wang, Li, Li, Shen & Porikli, 2017). Furthermore, trackers with more advanced appearance models than KCF might be more robust to appearance alternating situations with long occlusion and scale changes (X. Wang et al., 2017).

Recoverability

Recoverability is also an important aspect of tracking in a humanoid. If something causes the humanoid to deflect from the object meanwhile tracking, or if the object is fully occluded, it is important that it can recommence the tracking when the object is visible again. For humans, this is not a difficult task as long as the object has some unique features that make it distinguishable from the background or other objects (Horowitz, Birnkrant, Fencsik, Tran & Wolfe, 2006). Even if multiple objects are highly similar, humans still perform well when tracking up to 4 objects(Zelinsky & Neider, 2008). Moreover, humans also have the ability to classify novel object that they never encountered before (Morgenstern, Schmidt & Fleming, 2019).

In artificial systems, this task is not as trivial. An appearance model of an object often needs to be explicitly trained with a large number of examples in order for the model to be robust and generalisable. However, artificial systems' ability to generalise is continuously improving with clever engineering and new machine learning techniques (Barbiero, Squillero & Tonda, 2020). A robust appearance model is key for the system to successfully recognise an object that reappears after being undetectable for long period. This complicates real-time tracking where the appearance of the object needs to be modelled online. With this said, it is not surprising that a deep learning tracking by detection algorithm, such as YOLOv4-Deep SORT, successfully recovered tracking lost in the conducted experiments. The YOLOv4-network had been trained on thousands of images of each class of the COCO-dataset, meaning that it has been exposed to thousands of images of cups of various colours, shapes and sizes. This makes it likely to detect an object a second time, even if the object has not been visible for a longer period.

In contrast, the correlation filter tracker MOSSE uses the peak-to-sidelobe ratio (PSR), which measures the strength correlation peaks, to determine if the object is no longer visible. If the PSR value is below some set threshold, it will stop the online update of the appearance model until the PSR-value is up and thereby resume the tracking when the object shows a similar appearance again (Bolme et al., 2010). This reliance on online training makes the MOSSE algorithm less likely than a deep learning algorithm to recover tracking if the object is not visible for a long period. KCF also uses correlation filters but has a more robust appearance model of the object which could give it a better chance to recover the tracking (Henriques et al., 2015; Yadav & Payandeh, 2018). This was reflected by the second experiment with the turning away movement where both YOLOv4-Deep SORT and KCF were successful in recovering tracking but where MOSSE, with the simpler appearance model, was futile in this regard.

In the present experiments, the tracked object was part of the classes in the dataset that has been used to train the detection network YOLOv4. This means that as long as the object was clearly visible, the YOLOv4-Deep SORT had a good chance of detecting the object and establish tracking. However, the YOLOv4-Deep SORT misclassified the object as being different objects repeatedly after short periods of detection fails. Even if the misclassification was not a problem for this specific experiment because it only involved single object tracking, the frequent tracking lost resulted in jittery movements of the tracking agent instead of the desired smooth motion as when humans track an object (for a more detailed view of this phenomena, see video: `https://github.com/pierreklintefors/MasterThesisObjectTracking/blob/main/TrackingVideos/YOLOv4DeepSort_output.avi`). The jittery motion can be improved by altering the update rate of movement in the visual servoing code but that would also affect the agent's ability to follow fast and long movements. Therefore, a more robust detection model is preferred. This could potentially be achieved with the fully weighted YOLOv4-network but that conveys an increased computational cost. To reduce this computational cost, the YOLOv4-network could be scaled with a cross-stage partial (CSP) network approach. The CSP approach has been shown to significantly improve the speed of the YOLOv4-network, by utilising parallel processing, while still maintaining its accuracy when tested on the COCO-dataset (C.-Y. Wang, Bochkovskiy & Liao, 2021).

The deep association metric attribution to the SORT network is supposed to improve the tracker algorithm's performance to track an object with long periods of occlusion (Wojke et al., 2017). The deep association metric facilitates the association between frames in order to keep track of the detected object. As mentioned earlier, the experiment was conducted with single object tracking so the misclassifications of the detection network of the YOLOv4-Deep SORT did not affect its performance in this task. However, in multiple tracking, this deep association metric becomes more important to keep track of different objects' trajectories.

In order to recover from tracking loss, the algorithm must first accurately report the loss. This is an important and desired quality in humanoid to not be misguided by background noise. For this reason, algorithms such as GOTURN, MIL and MEDIAN FLOW, which failed to report tracking loss and erroneously started to track the background, are not suitable for humanoid implementation.

Limitations

The design of the conducted experiments intended to target important aspects of object tracking that are desired in a humanoid. However, there are some limitations with the present design that reduces the generalisability of the results.

The conducted experiments constituted of single object tracking to make a clean comparison between the algorithms and reduce the data for analysis. To increase the external validity, multiple trials of multiple objects could be used in a multiple object tracking task. The used algorithms are all compatible with multiple object tracking. A further expansion of the experiments could have been to vary the image dimensions and resolution of the capturing camera to see how this affected the speed and accuracy of the algorithms.

The different set of motions that were tested in the visual servoing task was part of one movement sequence. This meant that movements of the later stages of the sequence, such as the partial occlusion, were only tested for algorithms that had not lost track before that point. To get a more thorough evaluation of how the algorithms handle specific types of movement, the sequence could have been dived into multiple segments where each algorithm where tested with multiple trials. However, this will increase the amount of data for the analysis and thereby also increase the time dedicated for ground truth annotations.

Another quite obvious limitation with the present experimental setup was that it did not actually involve a full humanoid robot but instead used a simplified robotic agent. This was a result of the restrictions due to the Covid-19 pandemic, which limited access to the lab of

the LUCS cognitive robotics group. A humanoid robot has a higher number of servo motors with longer signal chains and bigger parts in motion causing vibrations that might influence the algorithms' performance in tracking an object. Furthermore, with a humanoid robot, the algorithms' suitability as a visual servo control tool could also be extended to control the end-effectors for grabbing purposes.

Future directions

With continuing developments of powerful GPUs and CPUs as well as computational optimisation of deep learning networks, more robust tracking algorithms are becoming more available and feasible for implementation in mobile artificial agents. However, the statistical calculations of artificial neural networks do not necessarily replicate the underpinnings of human cognition (Peterson, Abbott & Griffiths, 2018; Sejnowski, 2020). Nevertheless, the future holds great opportunities for mutually beneficial exchange between neuroscience and artificial intelligence (Savage, 2019). In terms of cognitive robotics, the development of artificial systems should be done in the aspiration to better understand the performed calculations of biological systems. In this regard, there is potential to further use Bayesian estimation methods in the development of tracking algorithms that approach human tracking behaviour as well as being feasibly computed by the cortical architecture of the biological brain (Friston, 2005a).

Implementing the Bayesian brain hypothesis to the functional anatomy of the brain has resulted in hierarchical prediction-driven Bayesian models known as predictive processing or predictive coding (Clark, 2013; Friston, 2005b; Lee & Mumford, 2003; Rao & Ballard, 1999). Within predictive processing, perception and action are driven by predictions, formulated as top-down signals. The predictions are produced based on intrinsic generative models that map the causational relationship between the world and sensory information. In other words, the top-down stream of information is used to infer the most probable causes of sensory input given the current knowledge. In this context, sensory input serves as error measurements of those predictions. This is along the line with Barlow's efficient coding hypothesis (2012) where it can be argued to be more efficient for a system to emphasise uncertainty and the unexpected instead of continuously coding a stream of bottom-up information.

Further, Karl Friston has applied his free energy principle (Friston, Kilner & Harrison, 2006) to the predictive processing and predictive coding framework. Free energy constitutes as an upper bound of surprisal (i.e the negative log probability distribution of Bayesian model evidence) and can be used in variational Bayes methods. Friston also emphasises the role of enactivism in the process of inferring causation to sensory inputs and making predictions, termed active inference (Friston, Daunizeau, Kilner & Kiebel, 2010). According to the active inference framework, the free energy bound can be minimised, and thereby the discrepancy between prediction and perception, by either changing the prediction or acting upon the environment. The active inference framework is thereby suitable for embodied agents and has been successfully used for optimal control of humanoid movements based on information from the perceptual input channels (Oliver, Lanillos & Cheng, 2021). The free energy, or variational free energy in Bayesian belief update terms, can be defined as a complexity term subtracted with an accuracy term as formulated in (2) (Friston, 2010).

$$F = D_{KL}[q(s|\pi)||p(s|\pi)] - E_{q(s|\pi)}[ln(p(o|s))] \quad (2)$$

Complexity serves as Bayesian surprise, which is the difference between the prior and posterior Bayesian beliefs. This is constituted by the Kullback-Lieber divergence between the recognition density given a policy for action $q(s|\pi)$, which holds the posterior beliefs, and the prior density $p(s|\pi)$ in 2. The accuracy term is the arisen surprise given the recognition density. This formulation of variational free energy advocates that the accuracy should be maximised while decreasing complexity. In other words, the modelling of causation should be as precise as possible without changing the prior beliefs too drastically. By this nature, minimising variational free energy incorporates the explore-exploit relationship where epistemic action can be made in order to increase the likelihood for more accurate future predictions (Friston et al., 2016). This makes the active inference suitable for modelling the task of object tracking where the model can be learned and updated online.

Conducted simulations have shown that active inference can be used to successfully navigate in both deterministic and stochastic environments (Sajid, Ball, Parr & Friston, 2021). With the appropriate generative model, Sajid et al. (2021) argues that active inference can be scaled to robotic agents; something that has been realised by Oliver et al. (2021) in using active inference in body perception of a humanoid. There has also been theoretical work where active inference has been used in simulations to create dynamical models of ocular motion and sensory delays in humans (Adams, Aponte, Marshall & Friston, 2015; Perrinet et al., 2014).

In conclusion, the active inference framework is suggested to be incorporated in object tracking in artificial systems in future work. Gradient descent of variational free energy can be used to optimise predictions, compatible with other Bayesian filters and neural networks (Ueltzhöffer, 2018). Active inference has also been used as inspiration of the human visual system to implement generative adversarial networks (GAN) for quality assessment without a reference image. Furthermore, combining GAN in object detection and tracking can potentially improve the inference of distorted object caused by occlusion or non-rigid movement (Prakash & Karam, 2019). This is in line with the active inference framework where generative models are used to infer probable causation based on feature information from the sensory input.

5 Conclusion

As the general case in computer vision, there was a trade-off between accuracy and computational cost in the present evaluation of the object tracking algorithms. Based on the results, the correlation filter-based trackers MOSSE and KCF are argued to be suitable options for implementation in humanoids with low computational

powered systems. KCF has the more complex learning model of the two and is more robust to occlusion whereas MOSSE is the faster but less accurate alternative. KCF is therefore ultimately recommended for a low computational powered system but MOSSE will serve as the appropriate choice if speed is of the most important essence. Moreover, in high computational powered systems, deep learning algorithms will probably be more robust than KCF and should be used instead. With continuing development and optimisation of the deep neural networks they could potentially become the more suitable choice for low-end systems as well. Finally, active inference, either in conjunction with neural networks or as an independent framework, could be suitable for object tracking due to its biological feasibility as well as its high relevance in prediction task and further pursuit of the aforementioned is encouraged.

Acknowledgements

I wish to express my gratitude to Birger Johansson and Christian Balkenius for their expertise and guidance in supervising this thesis project. I would also like to express my appreciation for the LUCS cognitive robotics lab lending me the servomotors that were used to build the robotic agents used in the conducted experiments.

References

Adams, R. A., Aponte, E., Marshall, L. & Friston, K. J. (2015). Active inference and oculomotor pursuit: The dynamic causal modelling of eye movements. *Journal of Neuroscience Methods*, *242*, 1-14. doi: https://doi.org/10.1016/j.jneumeth.2015.01.003

Allen, M. & Friston, K. J. (2016). From cognitivism to autopoiesis: towards a computational framework for the embodied mind. *Synthese*, *195*(6), 2459–2482. doi: 10.1007/s11229-016-1288-5

Ardón, P., Dragone, M. & Erden, M. S. (2018). Reaching and grasping of objects by humanoid robots through visual servoing. In D. Prattichizzo, H. Shinoda, H. Z. Tan, E. Ruffaldi & A. Frisoli (Eds.), *Haptics: Science, technology, and applications* (pp. 353–365). Cham: Springer International Publishing. doi: 10.1007/978-3-319-93399-3_31

Babenko, B., Yang, M.-H. & Belongie, S. J. (2009). Visual tracking with online multiple instance learning. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 983-990.

Barbiero, P., Squillero, G. & Tonda, A. (2020). Modeling generalization in machine learning: A methodological and computational study. *ArXiv*, *abs/2006.15680*.

Barlow, H. B. (2012). Possible principles underlying the transformations of sensory messages. In *Sensory communication* (pp. 216–234). The MIT Press. doi: 10.7551/mitpress/9780262518420.003.0013

Bewley, A., Ge, Z., Ott, L., Ramos, F. & Upcroft, B. (2016). Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*. doi: 10.1109/icip.2016.7533003

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv e-prints*, arXiv:2004.10934.

Bolme, D., Beveridge, J., Draper, B. & Lui, Y. (2010). Visual object tracking using adaptive correlation filters. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2544-2550. doi: 10.1109/CVPR.2010.5539960

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Carlton, L. G. (1981). Processing visual feedback information for movement control. *Journal of Experimental Psychology: Human Perception and Performance*, *7*(5), 1019–1030. doi: 10.1037/0096-1523.7.5.1019

Chen, Z. (2003). Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, *182*. doi: 10.1080/02331880309257

Chen, Z., Hong, Z. & Tao, D. (2015). An experimental survey on correlation filter-based tracking.

Cheng, C., Kaldy, Z. & Blaser, E. (2019). Two-year-olds succeed at mit: Multiple identity tracking in 20- and 25-month-old infants. *Journal of Experimental Child Psychology*, *187*, 104649. doi: https://doi.org/10.1016/j.jecp.2019.06.002

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *The Behavioral and brain sciences*, *36*, 1-24. doi: 10.1017/S0140525X12000477

Corke, P. I. & Hutchinson, S. A. (2000). Real-time vision, tracking and control. , *1*, 622-629 vol.1. doi: 10.1109/ROBOT.2000.844122

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S. & Dehmer, M. (2020). An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, *3*, 4. doi: 10.3389/frai.2020.00004

Friston, K. (2005a). A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *360*(1456), 815-836. doi: 10.1098/rstb.2005.1622

Friston, K. (2005b). A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *360*(1456), 815-836. doi: 10.1098/rstb.2005.1622

Friston, K. (2010). The free-energy principle: a unified brain theory? nat. rev. neurosci. 11, 127-138. *Nature reviews. Neuroscience*, *11*, 127-38. doi: 10.1038/nrn2787

Friston, K., Daunizeau, J., Kilner, J. & Kiebel, S. J. (2010). Action and behavior: a free-energy formulation. *Biological Cybernetics*, *102*(3), 227-260. doi: 10.1007/s00422-010-0364-z

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., ODoherty, J. & Pezzulo, G. (2016). Active inference and learning. *Neuroscience Biobehavioral Reviews*, *68*, 862 - 879. doi: https://doi.org/10.1016/j.neubiorev.2016.06.022

Friston, K., Kilner, J. & Harrison, L. (2006). A free energy principle for the brain. *Journal of Physiology-Paris*, *100*(1), 70 - 87. (Theoretical and Computational Neuroscience: Understanding Brain Functions) doi: https://doi.org/10.1016/j.jphysparis.2006.10.001

Galdo, M., Bahg, G. & Turner, B. M. (2020). Variational bayesian methods for cognitive science. *Psychological*

Methods, *25*(5), 535 - 559. doi: 10.1037/met0000242

Held, D., Thrun, S. & Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. *Springer International Publishing*. doi: 10.1007/978-3-319-46448-0_45

Helmholtz, H. v. (1867). *Handbuch der physiologischen optik*. Leipzig: Voss.

Henriques, J. F., Caseiro, R., Martins, P. & Batista, J. (2015). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), 583–596. doi: 10.1109/tpami.2014.2345390

Hill, J. (1979). Real time control of a robot with a mobile camera. *9th Int. Symp. on Industrial Robots, 1979*, 233-246.

Horowitz, T. S., Birnkrant, R. S., Fencsik, D. E., Tran, L. & Wolfe, J. M. (2006). How do we track invisible objects? *Psychonomic Bulletin & Review*, *13*(3), 516–523. doi: 10.3758/bf03193879

Hutchinson, S., Hager, G. & Corke, P. (1996). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, *12*(5), 651–670. doi: 10.1109/70.538972

Kalal, Z., Mikolajczyk, K. & Matas, J. (2010). Forward-backward error: Automatic detection of tracking failures. , 2756–2759. doi: 10.1109/ICPR.2010.675

Kalal, Z., Mikolajczyk, K. & Matas, J. (2012). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(7), 1409–1422. doi: 10.1109/TPAMI.2011.239

Kersten, D. & Yuille, A. (2003). Bayesian models of object perception. *Current Opinion in Neurobiology*, *13*(2), 150–158. doi: 10.1016/s0959-4388(03)00042-4

Knill, D. C. & Pouget, A. (2004). The bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, *27*(12), 712–719. doi: 10.1016/j.tins.2004.10.007

Kothiya, S. & Mistree, K. (2015). A review on real time object tracking in video sequences. , 1-4. doi: 10.1109/EESCO.2015.7253705

Kowler, E. (1989). Cognitive expectations, not habits, control anticipatory smooth oculomotor pursuit. *Vision Research*, *29*(9), 1049–1057. doi: 10.1016/0042-6989(89)90052-7

Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., ... Čehovin, L. (2016). A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(11), 2137-2155. doi: 10.1109/TPAMI.2016.2516982

Lee, T. S. & Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. *Journal of the Optical Society of America. A, Optics, image science, and vision*, *20*(7), 1434–1448. doi: 10.1364/JOSAA.20.001434

Li, P., Wang, D., Wang, L. & Lu, H. (2018). Deep visual tracking: Review and experimental comparison. *Pattern Recognition*, *76*, 323-338. doi: https://doi.org/10.1016/j.patcog.2017.11.007

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... Dollár, P. (2015). *Microsoft coco: Common objects in context.* doi: 10.1007/978-3-319-10602-1_48

Liu, S., Liu, D., Srivastava, G., Połap, D. & Woźniak, M. (2020). Overview and methods of correlation filter algorithms in object tracking. *Complex & Intelligent Systems*. doi: 10.1007/s40747-020-00161-4

Lukezic, A., Vojír, T., Zajc, L. C., Matas, J. & Kristan, M. (2017). Discriminative correlation filter with channel and spatial reliability. *ArXiv*, *abs/1611.08461*.

Mi, T.-W. & Yang, M.-T. (2019). Comparison of tracking techniques on 360-degree videos. *Applied Sciences*, *9*(16). doi: 10.3390/app9163336

Morgenstern, Y., Schmidt, F. & Fleming, R. W. (2019). One-shot categorization of novel object classes in humans. *Vision Research*, *165*, 98-108. doi: https://doi.org/10.1016/j.visres.2019.09.005

Nijhawan, R. (2008). Visual prediction: Psychophysics and neurophysiology of compensation for time delays. *Behavioral and Brain Sciences*, *31*(2), 179–198. doi: 10.1017/s0140525x08003804

Oliver, G., Lanillos, P. & Cheng, G. (2021). An empirical study of active inference on a humanoid robot. *IEEE Transactions on Cognitive and Developmental Systems*, *PP*, 1-1. doi: 10.1109/TCDS.2021.3049907

Perrinet, L. U., Adams, R. A. & Friston, K. (2014). Active inference, eye movements and oculomotor delays. *Biological Cybernetics*, *108*(6), 777–801. doi: 10.1007/s00422-014-0620-8

Peterson, J. C., Abbott, J. T. & Griffiths, T. L. (2018). Evaluating (and improving) the correspondence between deep neural networks and human representations. *Cognitive Science*, *42*(8), 2648-2669. doi: https://doi.org/10.1111/cogs.12670

Pitkow, X. & Angelaki, D. E. (2017). Inference in the brain: Statistics flowing in redundant population codes. *Neuron*, *94*(5), 943–953. doi: 10.1016/j.neuron.2017.05.028

Prakash, C. D. & Karam, L. J. (2019). It gan do better: Gan-based detection of objects on images with varying quality. *ArXiv*, *abs/1912.01707*.

Purves, D. (2012). *Neuroscience*. Sunderland: Sinauer Associates Inc.

Rao, R. P. N. & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, *2*(1), 79-87. doi: 10.1038/4580

Rashbass, C. (1961). The relationship between saccadic and smooth tracking eye movements. *The Journal of Physiology*, *159*(2), 326–338. doi: 10.1113/jphysiol.1961.sp006811

Redmon, J. (2013–2016). Darknet: Open source neural networks in c. http://pjreddie.com/darknet/.

Ryokai, K., Farzin, F., Kaltman, E. & Niemeyer, G. (2013). Assessing multiple object tracking in young children using a game. *Educational Technology Research and Development*, *61*(2), 153–170. doi: 10.1007/s11423-012-9278-x

Sajid, N., Ball, P. J., Parr, T. & Friston, K. J. (2021). Active Inference: Demystified and Compared. *Neural Computation*, *33*(3), 674-712. doi: 10.1162/neco_a_01357

Sanborn, A. N. & Chater, N. (2016). Bayesian brains without probabilities. *Trends in Cognitive Sciences*, *20*(12), 883–893. doi: 10.1016/j.tics.2016.10.003

Savage, N. (2019). How AI and neuroscience drive each

other forwards. *Nature*, *571*(7766), S15–S17. doi: 10.1038/d41586-019-02212-4

Schölkopf, B. & Smola, A. J. (2018). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press. doi: 10.7551/mitpress/4175.001.0001

Sejnowski, T. J. (2020). The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, *117*(48), 30033–30038. doi: 10.1073/pnas.1907373117

Soleimanitaleb, Z., Keyvanrad, M. A. & Jafari, A. (2019). Object tracking methods:a review. *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, 282 - 288. doi: 10.1109/ICCKE48569.2019.8964761

Sze, V., Chen, Y.-H., Yang, T.-J. & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *105*(12), 2295-2329. doi: 10.1109/JPROC.2017.2761740

Tenenbaum, J. B., Griffiths, T. L. & Kemp, C. (2006). Theory-based bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, *10*(7), 309–318. doi: 10.1016/j.tics.2006.05.009

Tenenbaum, J. B., Kemp, C., Griffiths, T. L. & Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, *331*(6022), 1279–1285. doi: 10.1126/science.1192788

TheAIGuy. (2020). Yolov4-deepsort. *GitHub repository*. https://github.com/theAIGuysCode/yolov4-deepsort.

Ueltzhöffer, K. (2018). Deep active inference. *Biological Cybernetics*, *112*(6), 547–573. doi: 10.1007/s00422-018-0785-7

Vahrenkamp, N., Wieland, S., Azad, P., Gonzalez, D., Asfour, T. & Dillmann, R. (2008). Visual servoing for humanoid grasping and manipulation tasks. , 406-412. doi: 10.1109/ICHR.2008.4755985

Verma, R. (2017). A review of object detection and tracking methods. *International Journal of Advance Engineering and Research Development*, *4*, 569-578.

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network. *ArXiv*, *abs/2011.08036*.

Wang, X., Li, H., Li, Y., Shen, F. & Porikli, F. (2017). Robust and real-time deep tracking via multi-scale domain adaptation. In *2017 ieee international conference on multimedia and expo (icme)* (p. 1338-1343). doi: 10.1109/ICME.2017.8019450

Wojke, N., Bewley, A. & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 ieee international conference on image processing (icip)* (p. 3645-3649). doi: 10.1109/ICIP.2017.8296962

Yadav, S. & Payandeh, S. (2018). Understanding tracking methodology of kernelized correlation filter. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 1330 - 1336.

Yin, F., Makris, D. & Velastin, S. (2007). Performance evaluation of object tracking algorithms. *10th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2007)*.

You, S., Zhu, H., Li, M. & Li, Y. (2019). A review of visual trackers and analysis of its application to mobile robot. *CoRR*, *abs/1910.09761*.

Yue, F. & Li, X. (2018). Improved kernelized correlation filter algorithm and application in the optoelectronic tracking system. *International Journal of Advanced Robotic Systems*, *15*(3), 1729881418776582. doi: 10.1177/1729881418776582

Zelinsky, G. J. & Neider, M. B. (2008). An eye movement analysis of multiple object tracking in a realistic environment. *Visual Cognition*, *16*(5), 553-566. doi: 10.1080/13506280802000752

Zhang, L., Wang, Y., Sun, H., Yao, Z. & He, S. (2015). Robust visual correlation tracking. *Mathematical Problems in Engineering*, *2015*, 1-13. doi: 10.1155/2015/238971