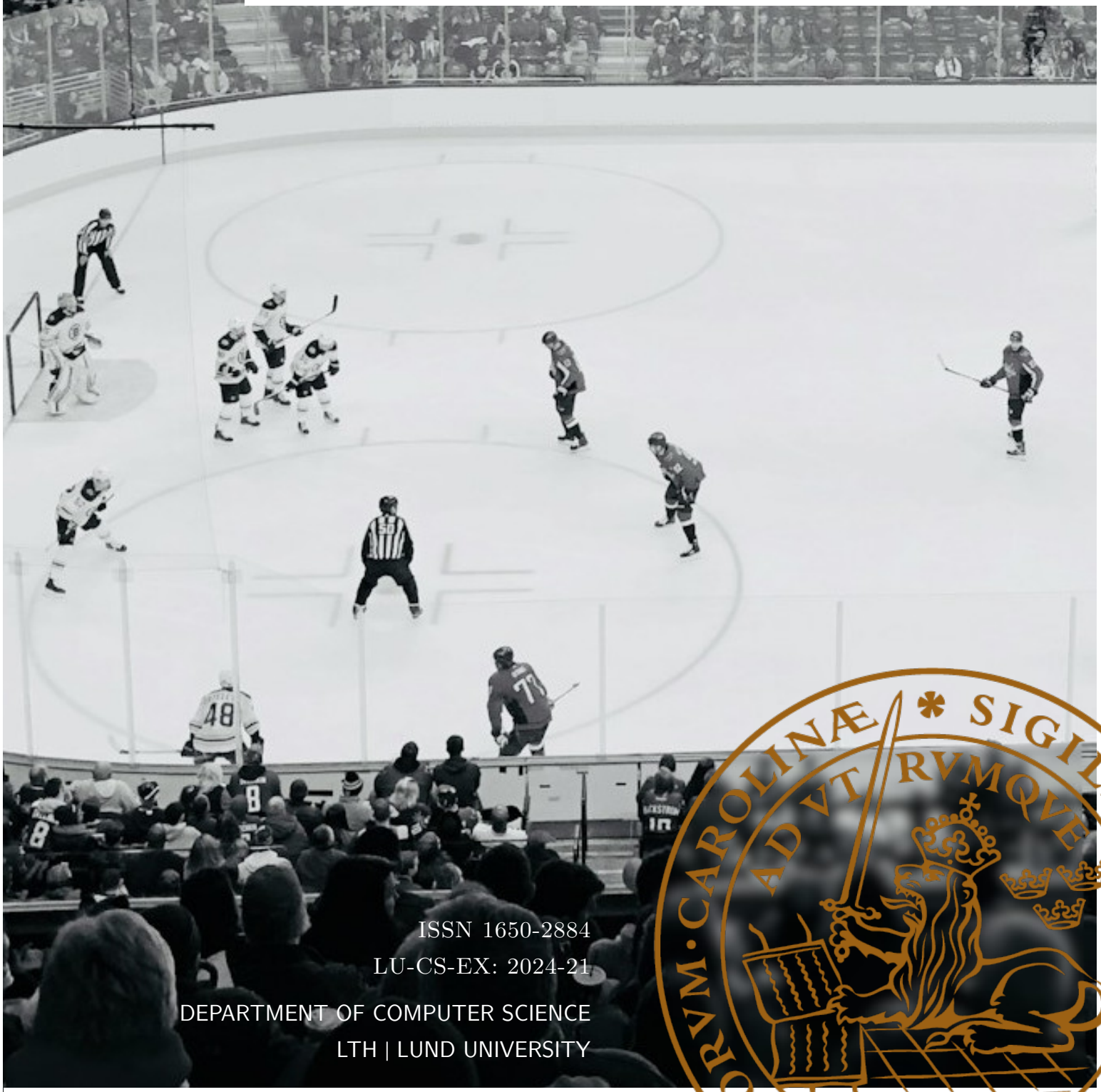
MASTER'S THESIS 2024

# Analysis of seam algorithms in video stitching with static scenes and dynamic objects

Amin Alian, Ludvig Gierup



ISSN 1650-2884
LU-CS-EX: 2024-21

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-21

# Analysis of seam algorithms in video stitching with static scenes and dynamic objects

Analys av sömalgoritmer i videostitchar med statiska scener och dynamiska objekt

**Amin Alian, Ludvig Gierup**

# Analysis of seam algorithms in video stitching with static scenes and dynamic objects

Amin Alian
alianamin00@gmail.com

Ludvig Gierup
lgierup25@gmail.com

May 30, 2024

Master's thesis work carried out at Spiideo AB.

Supervisors: Håkan Ardö, hakan.ardo@spiideo.com
Michael Doggett, michael.doggett@cs.lth.se

Examiner: Per Andersson, per.andersson@cs.lth.se

**Abstract**

The focus of this paper is video stitching with a dual camera setup, static sports scenes and dynamic objects, as Spiideo wants to explore solutions to frequently occurring optical issues. Video stitching is closely related to image stitching but poses a different range of challenges. To create a video without visual artifacts, one solution is to find a seam through each video frame image pair so that the pixels to the left of the seam are from the left image, and the pixels to the right of the seam originate from the right image. In this paper we adapt and improve an already existing seam algorithm with optimisations in both performance and visual quality. We have developed three different versions of the baseline seam algorithm that we call temporal, index, and segmented. The conclusion is that the segmented variant was preferred over other variants as it was the best performing while also minimising visual artifacts.

**Keywords**: computer vision, video stitching, seam algorithm, parallax effect

# Acknowledgements

We would like to give a big thank you to the group we have been studying with in Studiecentrum for our entire time spent studying at LTH. This would not have been possible without your company.

We also want to thank the other master thesis students at Spiideo. It's been a pleasure getting to know you, and we really appreciate all the conversations and banter in the table tennis room.

We want to give thanks to our supervisor Michael Doggett who has trusted us and given us support for this thesis.

Finally, we also want to thank Spiideo and Håkan who gave us this opportunity to conduct our thesis at their company as well as giving us continuous advice and support.

# Contents

# Chapter 1
# Introduction

Sports is becoming more and more data driven. To collect data you need to manually keep track of the statistics or acquire help from cameras and computers. Spiideo automates sports streaming, recording and analysis with video cameras placed with differing angles at the same spot in an arena. These cameras have to process and correctly stitch their frames where the cameras overlap to give a good experience to the end user. When the cameras are installed, they are manually calibrated for use. This poses an issue when a camera is put out of its calibration in later use. This results in a slight disarray when the video images are stitched together, resulting in blurry details in the processed video.

Another issue is that it's difficult to adequately stitch on different distances at the same time e.g. the cameras are calibrated on ground level but players' bodies and heads are closer to the camera. This is because of differing distance to the focal points, leading to blurry details or ghosting, also known as parallax. Currently, stitching is done with preset parameters for a soft average blend between images. By finding an algorithm to smoothly cut and blend images the video will look closer to what a human would expect. This algorithm can either be efficient enough to use in a real-time stitching or to later analyse the video to find optimal stitches.

In an optimal scenario, the cameras would use an algorithm to stitch the video while being cost efficient enough to ensure the live streaming product is delivered with sharp details and a balanced colouring of the images.

## 1.1 Problem statement

Spiideo has offered us the opportunity to research this field as they have experienced issues with videos with parallax, especially for indoor sports. Their current approach of overlaying and blending the images is a solution but to improve their product they want to begin researching this field. This study will explore seam stitching algorithms for video, which will hopefully open a path for Spiideo to continue on in the future.

## 1.2 Camera setup

The cameras in the setup which will be studied are two Axis P1468 cameras pointing outwards from each other as shown in Figure 1.1. The two cameras cover the entire field and have an overlapping area in the middle. This area is the main focus of this Master's thesis. The overlapping area is where a stitch between the two frames generated by the cameras is needed. In order to create a stitch a camera calibration for each camera is needed which will be expanded upon in Section 2.2.

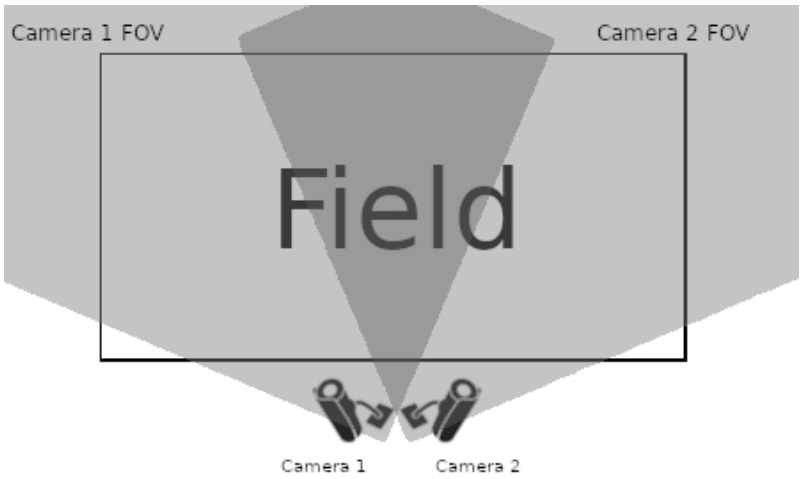



**Figure 1.1:** Camera setup from bird's eye view

## 1.3 Research topics

The research topics can be divided into three different parts.

### Research the algorithms that already exist in the frame-stitching field

Do we need to alter the algorithms to fit our use case?
If none of the algorithms we find can be applied to this problem we would have to write our own instead.

### Measure performance of each algorithm

How do the different algorithms perform?
Is the algorithm good enough to run in real-time or only for already processed videos?

### Analyse the output of each algorithm

Which algorithm produces the best looking stitched video?
How well does the output of the best performance algorithm look?

## 1.4 Scientific contribution

In this Master's thesis we contribute with a solution for creating a high quality video with a dual-camera setup. We have taken inspiration from previous work regarding seam algorithms in the area of video stitching and expanded the field with our own additions such as only updating parts of the seam and optimisations such as recalculating the seam strictly when necessary.

## 1.5 Contribution statement

The authors have worked closely together during the whole project. Amin

# Chapter 2
# Background

This chapter explores theoretical and practical knowledge on the topic related to this thesis. First and foremost, the camera setup for the video capture will be introduced, followed by camera calibration which explains how the parallax effect arises. To understand how a frame can be warped, relevant mathematics follows. Subsequently, image and video stitching will be introduced along with the different issues they pose. Finally, the foundation and key concepts of the seam algorithm, which is the focus of this thesis, will be defined.
There are multiple ways of solving the parallax problem. The focus of this thesis is on the seam algorithm using dynamic programming. The seam algorithm is applied to individual frames in a video sequence that have been stitched.

## 2.1 Computer Vision Mathematics

This section will briefly introduce the relevant mathematics related to the areas of computer vision in this thesis. These equations are a core part of the alignment algorithm which will be expanded upon in Section 3.2.

### 2.1.1 Geometric translation

A geometric translation is a transformation that moves every point of a figure, shape or space by the same distance in a given direction [1]. A translation for an image in x-axis is a shift where every point of the image moves by the same distance. A trivial equation for a 2D point, $(x, y)$, the translation $(\Delta x, \Delta y)$ where $(x', y')$ is the new location has the form:

$$(x', y') = (x + \Delta x, y + \Delta y) \tag{2.1}$$

### 2.1.2 Rotation matrix

A rotation matrix in linear algebra is a transformation matrix that is used to perform a rotation in Euclidean space [2]. A standard 2D rotation matrix has the form:

$$(x', y') = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.2}$$

This rotation matrix rotates a 2D point, $(x, y)$, $\theta$ degrees counterclockwise.

### 2.1.3 Affine transformation

An affine transformation is a geometric transformation that preserves lines and parallelism [3]. A combination of a rotation matrix and a translation can produce an affine transformation where the equation has the form:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.3}$$

$$\boldsymbol{x'} = A\boldsymbol{x} + \boldsymbol{b} \tag{2.4}$$

The invertible matrix $\boldsymbol{A}$ is represented with the aforementioned rotation matrix $\boldsymbol{R}$ (2.2) and the vector $\boldsymbol{b}$ can be represented with the translation $\boldsymbol{T}$ (2.1) resulting in this equation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & \Delta x \\ \sin\theta & \cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

## 2.2 Camera Calibration

This section briefly explains how camera calibration functions as well as its relation to the parallax effect. In this thesis the intrinsic matrix, which will be introduced below, is known and the extrinsic matrix has been estimated. This estimation is known as camera calibration.

### 2.2.1 Intrinsic Matrix

The intrinsic matrix is a representation of the inner parameters of a camera. It is represented by an invertible $3x3$ matrix of the form:

$$K = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

This equation is derived from the pinhole camera model where the skew has been omitted [4]. In this equation $f$ is the focal length, which is a measure for how strongly a system converges light, and $(x_0, y_0)$ represents the principal point, which is the intersection of the optical axis and the image plane. These parameters are constant, meaning they are not affected by the cameras position or orientation when installed.

### 2.2.2 Extrinsic Matrix

The extrinsic matrix is a representation of the external parameters of a camera. The matrix describes the cameras position and orientation in the world coordinate system and is represented by a $3x3$ rotation matrix $R$ and a $3x1$ translation vector $t$ of the form:

$$\begin{bmatrix} R & t \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix} \tag{2.7}$$

The extrinsic matrix is subjected to change when the position of the camera changes [4]. The extrinsic matrix is not dependant on the camera, but rather explains the cameras relation to the world coordinates.

### 2.2.3 Camera Matrix

To map a 3D point, such as an object in the world, to an image plane a $3x4$ camera matrix is used. The camera matrix is the product of the intrinsic and extrinsic parameters mentioned above. The intrinsic parameters changes as the camera changes and the extrinsic parameters changes as the position and rotation of the camera changes. The camera matrix is used to map 2D pixel coordinates to 3D world coordinates. Since an image is a 2D representation, the 3D coordinate is somewhere on a line going out from the camera. In our case, the points are chosen to lie in the ground plane.

### 2.2.4 Parallax effect

When an object is located far above the ground plane, the left and right image will start to disagree on its position. In Spiideo's case, this can happen with players' heads and upper bodies when they are close to the camera. This creates a parallax effect on the object, as seen in Figure 2.1. This effect is not visually appealing to viewers, thus this thesis will attempt to deal with the parallax effect by processing the stitched frames with a seam algorithm.

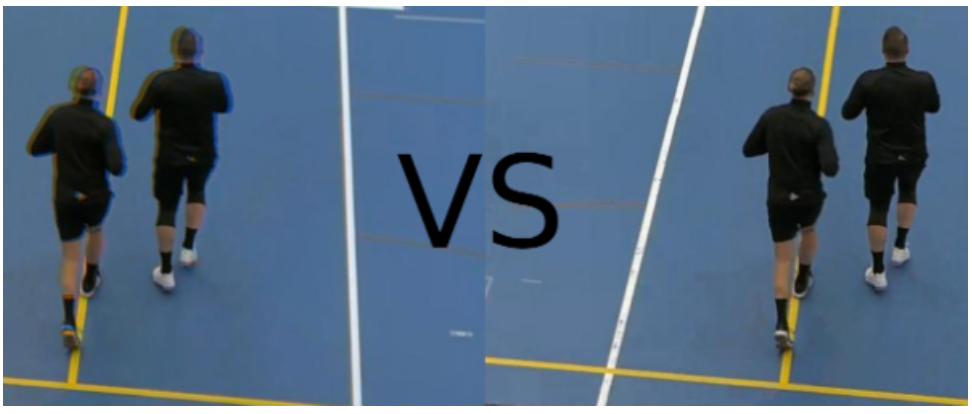



**Figure 2.1:** Parallax effect shown on the left image. Right image is the comparison.

## 2.3 Image stitching

Stitching images is a well researched field with many applications in several industries [5]. On the iPhone 5, released in 2012, Apple released a panorama feature [6]. At the time it was a revolutionary addition to the smartphone, but today it is regarded as a trivial one. The feature uses image stitching to create a panorama. The concept of image stitching can be divided into three sections: feature extraction, computing transformations and warping the images.

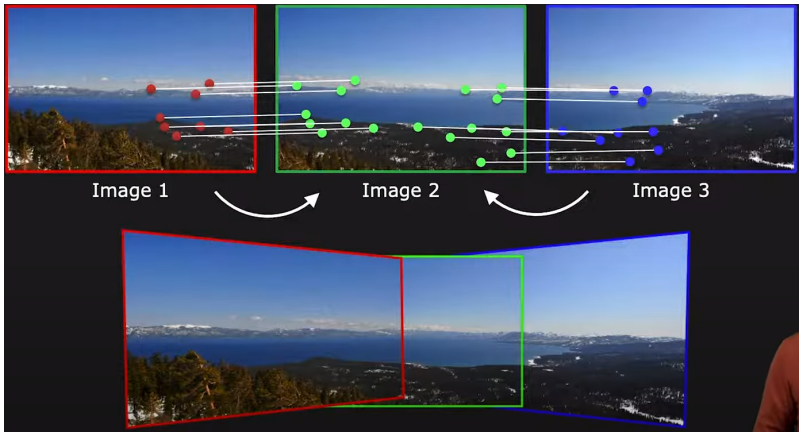



**Figure 2.2:** Image stitch example by Shree K. Nayar [7].

Feature extraction will be disregarded as this thesis has access to the full camera calibration. Therefore the areas of interest is computing the transformation and warping images. With the help of OpenCV, a popular open source computer vision library [8], the rotation matrix and translation will be generated in addition to the warping of the image. This topic will be expanded upon in Section 3.2.

## 2.4 Video stitching

Video stitching is a significantly less explored field than image stitching. There are numerous approaches for implementing video stitching as there are additional aspects compared to image stitching. One important aspect is the sensitivity of the human eye to unnatural objects or lines. When only processing one single frame for an image, it is easier to create a good looking stitch. When processing multiple frames, there must be a degree of consistency in colour, objects and a satisfactory temporal quality.

### 2.4.1 Analysing video quality

One of our research topics was to analyse the output, referring to stitched videos. How can we measure the quality of a video with respect to jittering and other visual artifacts? We will combine quantitative metrics with subjective video quality, see Section 3.5. This is because objective quality assessment algorithms do not consistently correlate to subjective ratings.

### 2.4.2 Performance of frame processing

Another topic is whether or not we can process the video in real-time. To meet this criterion, running everything in real-time on two 50 frames per second camera, the entire processing of each frame has to be done in less than 0.02 seconds in the worst case. This is quite a demand as there are multiple computationally heavy steps in processing frames. The thesis will be consistently run on a personal computers for simplicity. Therefore, the performance results are only interesting in their relativity as our machine is not up to par with the computers that run the Spiideo software. The specifications of the machine are: Intel® Core™ i7-8550U CPU @ 1.80GHz × 8 CPU.

## 2.5 Seam algorithm

This section will describe how a seam algorithm is constructed and implemented by introducing cost functions and detail the concept and baseline variant of the seam algorithm used in this thesis. A seam algorithm chooses where to cut off and combine two images rather than to simply overlay the two frames. This is done after the frames have been warped according to their respective camera calibrations. The result of the warp is that the overlapping area for the cameras now have correct coordinates in relation to each other.

### 2.5.1 Cost Function

To be able to find which pixels to run the seam through, something called a cost function is introduced. In the overlapping region, compare the RGB (Red, Green, Blue) values of each pixel in the left image, $I_1$, to the right image, $I_2$, using e.g. mean squared error.

$$C(x, y) = \frac{1}{3} \sum_{R,G,B} (I_1(x, y) - I_2(x, y))^2 \tag{2.8}$$

The resulting matrix is visualised in figure 2.3. The light areas are where the pixels differ a lot between the images, and the dark areas are where the pixels are similar. We want to run the seam through the low cost dark areas so that the seam is not noticed.

There are multiple ways of calculating the difference between two pixels, which will be further discussed in 3.4.

### 2.5.2 Seam algorithm simply explained

To determine a path for the seam each pixel in the area of interest, the overlapping area, is given a value according to a cost function. An example of a cost function focuses on how similar the colours of each overlapping pixel is in the two individual warped frames.

If the difference between two pixels is high, the corresponding cost of the coordinate will also be high. The seam algorithm then proceeds by finding a continuous path from the top of the image to the bottom of the image with a minimum accumulated cost. The only limitation being that it has to go down straight or diagonally, never horizontally or upwards.

**Figure 2.3:** Mean Squared Error Cost Function.

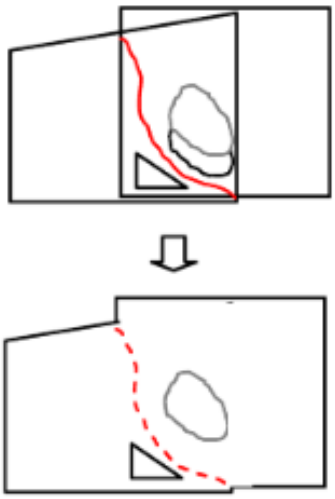


**Figure 2.4:** Figure created by He and Yu [9] showing how a seam is visualised.

The best, lowest cost, path is then used to combine the two frames with the left frame going from the starting x-coordinate to the respective x-coordinate for the seam and then from there the right frame continues to fill the rest of the coordinates. If the cost function and seam combine well the resulting image will not contain any trace of objects being incoherent as seen in Figure 2.4.

### 2.5.3 The baseline seam algorithm

The main focus of this thesis is the seam algorithm. The baseline seam algorithm explained below was adapted and improved to our use case.

The baseline variant of the seam algorithms following Avidan and Shamir's [10] description of their findings. Their dynamic programming algorithm is a foundation to start with as it solves the most basic problems faced with parallax. Despite this, there were multiple improvements to be made to reach a real-time customisable algorithm which had practical use. For efficiency and coherency every pixel has to be at a maximum one pixel diagonally away from the previous row as seen in Figure 2.5.

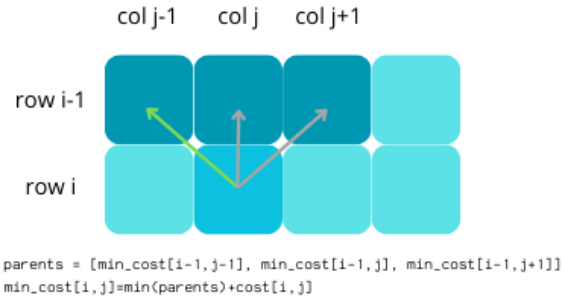



**Figure 2.5:** Visualisation of pixel choice of cost from previous row.

The baseline variant iterates through the rows of an image from top to bottom and gives them a cost, based on a cost function. Since the pixel on one row can only be one pixel away diagonally to the left and right from the pixels on the previous row, the only relevant costs are the three pixels closest from the previous row. While iterating through the image the cost of a pixel is the value of itself accumulated with the minimum cost of the three previous pixels. It is important to keep track of which pixel on the previous row contains the minimum cost to create a minimum path. By iterating through the entire image with this logic while keeping track of each direction the pixels are choosing, the last row will contain the minimum cost path for each pixel. Choosing the pixel on the last row with the smallest cost will then give us the path through the image with the least cost.

## 2.6 Related work

Image stitching is a well researched area, and has been continuously expanded since the early 2000s [5][11]. Articles that reference the same seam algorithm that we have implemented have been around for image stitching for years [10]. The same is not true for video stitching, which is a more recent advancement in computer vision. Video stitching poses new problems

that do not exist in image stitching, like seam coherence between frames, seam updating, and real-time performance [12]. The paper by Botao He and Shaohua Yu [9] proposes a way to update the seam when an object is passing over it. We have implemented their solution and also expanded it, see Section 3.3.

Seam coherence between frames has also been researched, and is often called temporal coherence because the frames appear after each other in time. We have taken inspiration from [13], but we haven't implemented their exact solution.

Different cost functions for computing the seam were used in our solution. One of them was implemented from a paper by Wu et al. [14]. This cost function takes both the colour and geometric difference into account. In the same paper they also discuss an optimisation of the seam algorithm. Instead of considering the three pixels above in the seam, the optimization takes the entire row in to consideration when deciding which path to take, which is something we have not implemented but is relevant for future work.

There are other ways of solving the aforementioned problems with stitching, like graph-cut solutions [15] and warping solutions using different transformations [16]. The reason for not trying out these solutions is that there just wasn't enough time to do so. In this thesis we focus on seam-based solutions.

# Chapter 3
# Approach

This chapter will detail our approach to exploring and solving the problem faced at Spiideo, described in the introduction. The pipeline explains the different steps of processing a frame to give an overview before diving in to the details. After understanding the overall process we will focus on the different parts of the pipeline and explain how they are created starting with the alignment algorithm and why it might be needed.

## 3.1 Pipeline configuration

The pipeline, the procedure we have created to process each frame of a video, will be explained as it lays the foundation for our approach. As seen in Figure 3.1 the initial stage of the configuration loads a frame. This frame may originate from a local or external video source. In the pipeline OpenCV was used to create a video capture where we could load each frame of the video. The first frame serves as the basis for configuring the camera calibration

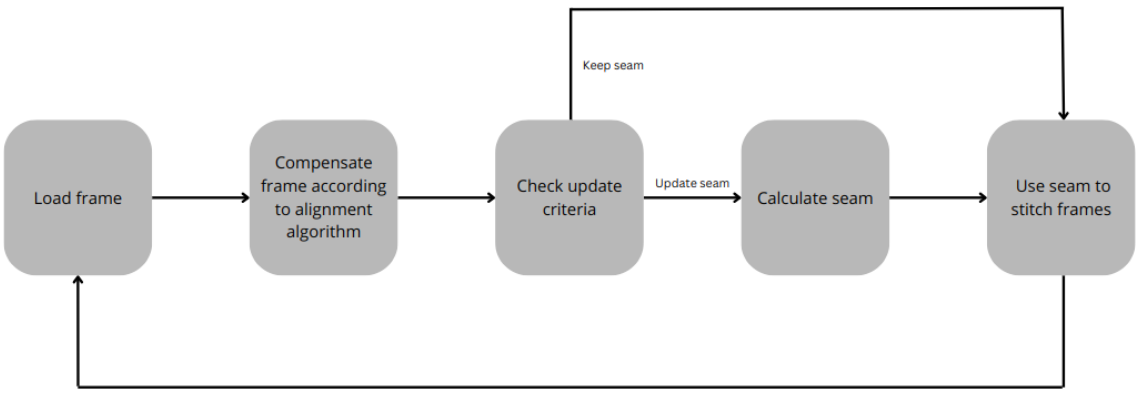



**Figure 3.1:** Pipeline flow chart

for all of the coming frames in the capture. During this setup, the alignment algorithm can also be initialised. Subsequent frames will then automatically use the alignment algorithm to mitigate the discrepancies before progressing down the pipeline. These frames will progress to an update criteria checker where the current seam is compared to the previous one to detect any colour changes in the pixels lying on the seam. If the threshold is not met, the pipeline is finished and the next frame is loaded. If it is met, a recalculation of the seam is triggered. The new seam will then be used to stitch the frames optimally.

## 3.2 Programmatical alignment

For the seam algorithm to successfully create an adequate stitch, it needs a frame each from two cameras which are well calibrated. This is not always the case, which is where the alignment algorithm serves its purpose. It is an algorithm to find an affine transformation for one of the frame to align with the other. This transformation will then be used for subsequent frames to be adjusted with the help of OpenCV. By aligning the frames we will generate a better seam resulting in a better video.

### 3.2.1 Alignment algorithm

To align frames from two different cameras the general idea of the alignment algorithm is to transform the right frame to match the left frame. In other words, the algorithm is finding a rotation matrix for, and warping, the right frame while leaving the left frame unchanged. The main algorithm is seen in Algorithm 1. However there is also a preprocessing algorithm which drastically reduces the number of possible rotations and translations the main algorithm should process. This reduction compares the cost of sample transformations and discards the irrelevant rotations and translations. It is a greedy algorithm meaning it does not necessarily guarantee the global optimum transformation is not discarded, however after testing it almost always finds the global optimum.

The result of this algorithm is a rotation matrix which is then used to warp future frames of the cameras. This rotation matrix is the transformation with the lowest cost according to the cost function. The images before and after the algorithm can be seen in Figure 3.2-3.9.

## 3.3 Detecting pixel changes

The baseline variant of the seam runs on every frame. By running it on every frame, it might use unnecessary computing power. As seen in Figure 3.10 there are no meaningful object changes near the seam. In such a scenario ideally, there should not be any calculations conducted for the seam.

To make the entire procedure more efficient there needs to be a detection threshold which is triggered whenever there are object changes in the scene. For this thesis we have used and adapted He and Yu's change detection [9]. Their work compares every pixel on the seam in the previous frame to the next frame to detect whether the colours have changed significantly. The parameters in their method are $\delta$, a threshold for each pixel and $C$ is the percentage of

**Algorithm 1** Alignment algorithm

```
1:  procedure Find Minimum Rotation Matrix(img1, img2)
2:      Rs ← array of possible rotations
3:      bs ← array of possible translations
4:      min_R ← 3x3 Identity matrix
5:      min_cost ← cost_function(img1, img2)
6:      for i ← 0 to size of Rs do
7:          R ← Rs[i]
8:          for j ← 0 to size of bs do
9:              R[0][2] += b[j][0]
10:             R[1][2] += b[j][1]
11:             tmp_img2 ← warpAffine(img2, R)
12:             tmp_cost ← cost_function(img1, tmp_img2)
13:             if tmp_cost < min_cost then
14:                 min_cost ← tmp_cost
15:                 min_R ← R
16:     return min_R
```

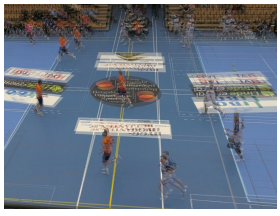


**Figure 3.2:** Stitch before alignment algorithm

**Figure 3.3:** Stitch after alignment algorithm

**Figure 3.4:** Stitch before alignment algorithm

**Figure 3.5:** Stitch after alignment algorithm

**Figure 3.6:** Focus area of stitch before alignment algorithm

**Figure 3.7:** Focus area of stitch after alignment algorithm

**Figure 3.8:** Focus area of stitch before alignment algorithm

**Figure 3.9:** Focus area of stitch after alignment algorithm


**Figure 3.10:** Seam with no changes near it. Purple line is the seam visualised.

pixels in the entire seam which meet the $\delta$ threshold for change. When $C$ is large enough the entire seam is updated. This method improves the performance drastically as the seam is no longer run on every frame, but we have further improved it for our use.

Our improvement finds the first pixel, $p_0$, which meets $\delta$ and then checks whether a $C'$ percentage after $p_0$ also meet $\delta$. This results in a procedure where we only have to update the seam after $p_0$ rather than all of the pixels in the seam. While this slightly improved performance of the overall runtime, it could be further improved.
Our next improvement finds the first pixel, $p_0$, which meets $\delta$ and then checks whether a $C'$ percentage of $N$ pixels after $p_0$ also meet $\delta$. If it does it creates a segment of the seam which we want to update, rather than recalculating the entire seam. To ensure that there are no unnecessary updates further down the seam, this segmented procedure continues $N$ pixels after $p_0$. The next pixel which meets $\delta$ creates a new segment for $p_1$ and so forth until we have reached the end of the frame, giving us multiple segments which will be recalculated significantly improving performance.

## 3.4 Seam

The first variant we worked with was the baseline variant. It is optimised for image stitching and always finds the best possible seam for an image. However, the mathematically best path is not necessarily the most visually appealing for the human eye. The baseline is not especially well performing in video as it always has to recalculate the entire seam for the entire height of the image, which might not always be necessary. In this section we will explain the different adaptations we made to the cost functions and seam algorithm to improve both performance and visual appeal in a video.

### 3.4.1 Cost Functions

We implemented several cost functions, but ended up testing the algorithms with just the base version. All cost functions gave similar results and the base function was the fastest by a small margin. see 4.1.

#### Base

Takes the difference between the left, $I_1$, and right, $I_2$, image's pixels, $(x, y)$, of each RGB value and squares it. Then takes the average of those values to make $C_{\text{base}}$.

$$C_{\text{base}}(x, y) = \frac{1}{3} \sum_{R,G,B} (I_1(x, y) - I_2(x, y))^2 \tag{3.1}$$

This cost function is very fast to calculate and yields a well functioning cost matrix where both colour difference and edges are visible. A downside to this function is that it doesn't take neighboring pixels or pixel gradients into account. Since we didn't see a big difference in performance between the cost functions, this is the one we went with for all our tests because of its fast execution time.

**Figure 3.11:** Base Cost Function

#### Intensity

The intensity cost function is similar to the base function. The difference is that instead of squaring each RGB value and then taking the average of the difference, the intensity function sums up each RGB value to one single value and then takes the squared difference. The result is a more sensitive cost matrix, in our tests it was strictly worse with respects to image quality than the base version.

$$C_{\text{int}}(x, y) = \Big( \sum_{R,G,B} I_1(x, y) - \sum_{R,G,B} I_2(x, y) \Big)^2 \tag{3.2}$$

#### Laplacian

The Laplacian [17], or the Laplace operator, is an operator that uses the second order derivative and is often used in computer vision as an edge detector. Because we're using it on a 2D

**Figure 3.12:** Intensity Cost Function

matrix and not on a continuous space, we used the discrete version of the operator [18] and we implemented it using OpenCV [19]. This cost function is great at finding edges, but gives low cost to small colour difference.

**Figure 3.13:** Laplacian Cost Function

## Roberts

The Roberts operator [20], or Roberts cross, is a simple operator that is used for edge detection. It approximates the gradient of an image by convolving the image with the two filters:

$$S_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } S_y \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$C_{\text{roberts}} = \left((I_1 \times S_x) \times S_y - (I_2 \times S_x) \times S_x\right)^2 \tag{3.3}$$

The resulting cost matrix was very similar to when using the Laplacian, which makes sense as they are both used for edge detection.

## Sobel

The Sobel operator is also an edge detector that uses filters to estimate the gradient of an image [21]. The kernels that are used for this operator are:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

**Figure 3.14:** Roberts Cost Function

Here, we used OpenCV [22] to implement the cost function. Once again, this cost matrix is very similar to the Roberts and the Laplacian cost functions.

**Figure 3.15:** Sobel Cost Function

## Lidong

We call this cost function Lidong because we found it in a paper by Lidong et al[14]. They divide the cost function into two parts: colour and geometry.

$$C_{\text{lidong}}(x,y) = C^2_{\text{colour}}(x,y) + C^2_{\text{geometry}}(x,y) \tag{3.4}$$

where $C_{\text{colour}}$ is something similar to our base cost function and $C_{\text{geometry}}$ is defined as:

$$C^2_{\text{geometry}}(x,y) = S_x \times (I_1(x,y) - I_2(x,y))^2 + S_y \times (I_1(x,y) - I_2(x,y))^2 \tag{3.5}$$

where $S_x$ and $S_x$ is the Robert operator that was previously described. After the calculation of $C_{\text{lidong}}$ they also take the 8 neighboring pixels of each pixel into account.

## Temporal

To solve the problem of temporal coherence, meaning that frames that follow each other in time should look somewhat alike, we implemented a new cost function that we call a temporal cost function. It takes the seam coordinates of the previous frame as input and weighs it so that pixels far from the previous seam are given a higher cost.

$$C_{\text{temp}}(x,y,x_{\text{prev}}) = C_{\text{base}}(x,y) + |(x_{\text{prev}} - x)| \tag{3.6}$$

**Figure 3.16:** Lidong Cost Function

**Figure 3.17:** Temporal cost function

### 3.4.2 Temporal seam

By using a temporal cost function, as seen in Figure 3.17, the seam will always prefer finding a path close to the seam in the preceding frame. This is done by saving the coordinates of the previous seam and weighing the next calculation of the cost function. This has multiple upsides in terms of visual appeal in a video sequence as a longer jump with the seam will alter a larger area of the image. A longer jump is more noticeable to the human eye as multiple coordinates slightly shift compared to a shorter jump. This temporal variant is not an

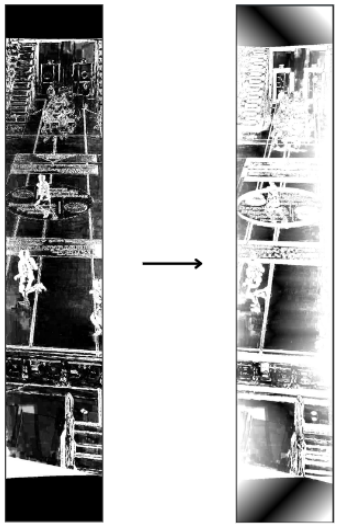


**Figure 3.18:** Temporal Cost Function

improvement runtime performance wise as it still has to evaluate and recalculate the entire seam. The next variant will focus on improving this area.

### 3.4.3 Index seam

To improve performance an adaptation was made to the pixel change detection algorithm to return the coordinate of the first pixel, which will be referred to as start index, which has changed. By using this start index the Index variant can start recalculating the seam in the following coordinates rather than the entire height of the image as seen in Figure 3.19.

This reduces the time it takes to run the algorithm with 38% on average, seen in 4.2.2 as there are not always objects moving across the seam triggering the pixel change threshold in the upper part of the image.

### 3.4.4 Segmented seam

Further improving from the previous variant is to not only start when we find a pixel area which needs recalculation but to also limit the end of area when we reach a new unchanged part in the seam. Finding a start and an end in the seam which needs updating creates a segment between the coordinates, hence the name of the variant. Since the seam has to be continuous the segmented variant is inherently temporally induced as the segment which will be recalculated still keep the same start and end points. As seen in Figure 3.20 this creates a

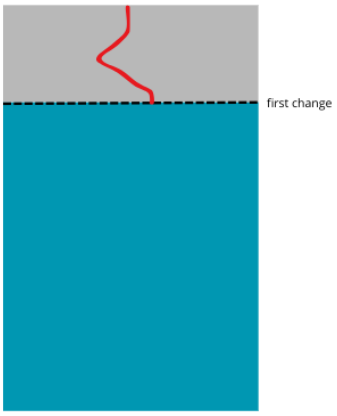



**Figure 3.19:** Index variant showing the area it has to recalculate.

"box" of possible outcomes for the path of the seam which drastically reduces the runtime of the algorithm.

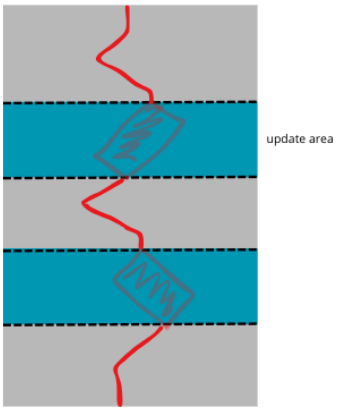



**Figure 3.20:** Segmented variant showing the area it has to recalculate.

Since the pixel change detection finds the first pixel which needs changing and the seam has to be continuous, the segmented variant did not always perform as well as expected. The answer as to why was that the new seam did not have enough space to move around to recalculate a better seam. If it found the start index pixel which needed changing it could only choose from the alternative one step to the left or to the right of it, which was not always better. To overcome this issue a buffer above and below the pixels was created to create alternative routes for the seam to choose from. For example, if the start index of the pixel area which needs recalculation is $(x_0, y_0)$, a buffer of $B$ pixels, 50 in our case, in the y-axis is created to avoid poor paths at a higher rate. This buffer is created at the cost of slight performance as there are more paths for the seam to relate to.

## 3.5 Video quality assessment

To evaluate the different variants of the seam we will use both a quantitative and qualitative approach. This is both to see if there are correlations between the two and also to include a user study as the most important factor for the output is how it is perceived by humans. We will use different metrics to analyse the videos for the quantitative assessment and to gather data for subjective video quality ratings, a panel will be created where 10 people will compare four different variants of the seam algorithm.

### 3.5.1 Metrics

#### Seam evaluation score

Introduced by Gao et al [23] in 2013, this is a way to evaluate a seamed image. For each pixel, $p$, in the seam, $S$, define a patch, $P$, around it (in this case a 17x17 patch was chosen). Compare this patch in the seamed image to the same patch in the left, $P_1$, and right, $P_2$, image. Choose the smallest of these two values. The error along a seam is the average of the calculated error values and the total error over a sequence of frames, $F$, is the average error of the frames.

From Gao et al: "The idea is that a patch along the seam is perceptually plausible if it resembles a patch found in either $I_1$ or $I_2$. If a patch along the seam cannot be found in either source images, it is likely to be an artifact and therefore assigned a larger error".

We hope that this metric can tell us how visible the seam is, both in terms of colour difference and in number of artifacts.

$$E(p) = \min_{P_i \in I_1, I_2} ||P - P_i||^2 \tag{3.7}$$

$$E_{\text{frame}}(\phi) = \frac{1}{|S|} \sum_{p \in S_\phi} E(p) \tag{3.8}$$

$$E_{\text{seq}} = \frac{1}{|F|} \sum_{\phi \in F} E_{\text{frame}}(\phi) \tag{3.9}$$

#### Seam move distance

Average of how far the seam has moved between two frames. This metric gives us a way to understand how much the seam moves horizontally between frames. If an algorithm scores high in this metric we can expect it to make big jumps between frames, which can look like jittering in the ground plane.

$$D(p) = |p.x_t - p.x_{t-1}| \tag{3.10}$$

$$D_{\text{frame}}(\phi) = \frac{1}{|S|} \sum_{p \in S_\phi} D(p) \tag{3.11}$$

$$D_{\text{seq}} = \frac{1}{|F|} \sum_{\phi \in F} D_{\text{frame}}(\phi) \tag{3.12}$$

### Percentage of seam moved

Average of how much of the seam is moved between two frames. This metric is also tied to noticeability. If we only update a small portion of the frame, that should be less noticeable compared to updating the entire frame.

$$M(p) = \begin{cases} 1, & \text{if } |p.x_t - p.x_{t-1}| \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

$$M_{\text{frame}} = \frac{1}{|S|} \sum_{p \in S_\phi} M(p) \tag{3.14}$$

$$M_{\text{seq}} = \frac{1}{|F|} \sum_{\phi \in F} M_{\text{frame}}(\phi) \tag{3.15}$$

### Mean seam runtime

The mean seam runtime, Avg(r), is the average time to run the seam algorithm over the frames. The time is only measured when the seam actually run, not on the frames where the algorithm might skip recalculating the seam.

### Update frequency

Since it will always be faster to not run the seam, the update frequency $f$ tells a good story of how well the seam performs in general. With a low update frequency the general time to process each frame is drastically reduced and the frequency is determined by the pixel change algorithm. The frequency is calculated by dividing the number of times the threshold for pixel change is met by the total number of frames.

### Total seam uptime

To get a good overview of the previous two performance metrics the total seam uptime $T_{\text{uptime}}$ gives an adjusted value to punish seams who produce a higher update frequency. This metric is calculated as such:

$$T_{\text{uptime}}(p) = \text{Avg}(r) \times f \tag{3.16}$$

Where Avg($r$) is the mean seam runtime and $f$ the update frequency for the seam.

## 3.5.2 Panel evaluation

As the end goal of this thesis is to create a video which is well received by humans rather than mathematical metrics, we created a panel of sports interested humans to review the videos processed by our algorithms to get an opinion which is subjective. As with any experiment including subjective opinions this comes with risks and margin for error.

As the goal of the form is to evaluate the seam in the video we focused the attention of the participants to where the seam is in action, around the center of the field. To get a

better evaluation of the general video we could have left the focus comment out, but since we wanted an assessment on the seam we chose to keep the focus.

The form created for the panel consisted of 5 major categories. The focus was on their evaluation of the seam regarding overall impression, unnatural colour changes, jittering or jumpy frames, quality of players and lastly quality and coherence of other objects in the video. To receive a well-rounded response to analyse we offered quantitative and qualitative choices where they were relevant.

# Chapter 4
# Results

In this chapter the results of cost functions, quantitative metric analysis as well as the user study will be presented.

## 4.1 Cost functions

Although we compared multiple cost functions the end result was that there was no decisive difference between the different cost functions, with the exception being temporal which is used for the temporal seam. Therefore the base cost function was used for all result gathering, except for the temporal cost function.

## 4.2 Metric scores

The quantitative metrics will be presented in tables in this section. They give us a numeric approach at evaluating the seams performance and visual appeal.

### 4.2.1 Visual metrics

These metrics are based on 300 frames of footage from ice hockey, handball, and basketball. The algorithm settings were the same for all three sports. The values in the "highest" columns are the averages of the 5 largest values in the 300 frame sequence. We included this to get some sense of how large the outliers are compared to the average.

In the tables below, "SES" is the seam evaluation score, "SMD" is the seam move distance, and "Change %" is the percentage of seam moved. For all the metrics, lower numbers are better.

### Ice Hockey

| Algorithm | SES | Highest SES | SMD | Highest SMD | Change % | Highest change % |
|---|---|---|---|---|---|---|
| baseline | **25** | **57** | 15.1 | 45.4 | 34 | 72 |
| temporal | 72 | 120 | 1.4 | 7.2 | 37 | 61 |
| index | 26 | 63 | 10.2 | 30.4 | 28 | 66 |
| segmented | 87 | 145 | **1.1** | **5.7** | **9** | **20** |

**Table 4.1:** Metrics for all four seam algorithms run on an ice hockey scene.

### Handball

| Algorithm | SES | Highest SES | SMD | Highest SMD | Change % | Highest change % |
|---|---|---|---|---|---|---|
| baseline | 40 | **55** | 9.5 | 65.5 | 36 | 86 |
| temporal | 110 | 145 | 7.4 | 32.6 | 33 | 70 |
| index | **39** | **55** | 10.6 | 39.2 | 30 | 76 |
| segmented | 130 | 158 | **2.5** | **11.8** | **11** | **23** |

**Table 4.2:** Metrics for all four seam algorithms run on a handball scene.

### Basketball

| Algorithm | SES | Highest SES | SMD | Highest SMD | Change % | Highest change % |
|---|---|---|---|---|---|---|
| baseline | **66** | **105** | 5.6 | 45.7 | 39 | 54 |
| temporal | 101 | 125 | 1.1 | **7.8** | 37 | 55 |
| index | 67 | **105** | 4.7 | 46.6 | 36 | 54 |
| segmented | 96 | 136 | **0.7** | 12.5 | **9** | **31** |

**Table 4.3:** Metrics for all four seam algorithms run on a basketball scene.

In the tables above we see that the baseline algorithm scores low on the seam evaluation metric. This is because it always picks the optimal seam for every frame with no respect to which frames or seams came before it. This also leads to a SMD.

The reason the seam evaluation score is so high for segmented and temporal is because the seams will slowly drift when processing a sequence of frames. Since these algorithms always take the last seam into account, they will stray further and further from the optimal seam for the individual images.

## 4.2.2 Performance metrics

The performance metrics are of a simpler kind. In Table 4.4 the significant impact the update frequency has on the total uptime of the seam is seen. The segmented variant of the seam is

by far the best performer in these metrics with an impressive 14.6 times reduction in total seam uptime compared to the baseline variant.

The measurements were run on an Intel® Core™ i7-8550U CPU @ 1.80GHz × 8 CPU.

| Algorithm | Mean seam runtime (ms) | Update frequency (%) | Total seam uptime (ms) |
|---|---|---|---|
| baseline | 84 | 100 | 84.0 |
| temporal | 86 | 42 | 36.1 |
| index | 70 | 74 | 51.8 |
| segmented | **23** | **25** | **5.8** |

**Table 4.4:** Performance metrics for seam variants.

## 4.3 User study

The results of the user study is present in Table 4.5 and Figure 4.1 with the individual responses found in Appendix A. All of the videos shown to the users in the study was of handball sequences where all of the seam variant were implemented separately. The videos were only named with aliases for the respondents. In the table the results are presented in mean values of the 1-5 rating. In questions 1 and 2 the rating 1 indicated "Poorly" and 5 indicated "Well". In question 3 the rating 1 indicated that it was "Noticeable" and rating 5 indicated "Natural look".

| | **Questions** | **Baseline** | **Temporal** | **Index** | **Segmented** |
|---|---|---|---|---|---|
| 1 | What did you think of the overall quality of the players and the field? | 2.9 | 3.3 | 3.2 | 3.8 |
| 2 | How well did the colours blend around the seam? | 4.3 | 4.4 | 4.4 | 4.5 |
| 3 | How noticeable was the jittering? | 1.7 | 2.8 | 2.7 | 3.3 |

**Table 4.5:** A table with mean values from the user study.

As seen in Table 4.5, the general trend of the results indicates that Baseline was the worst when perceived by humans and Segmented was the best looking video.

The low value of Baseline on Question 3 can be explained with the fact that it tries to find the best possible seam on every frame, which might result in it finding very good seam in consecutive frames but they might be far away in the x-axis which often results in a jittering video. Since Segmented uses the pixel change detection and only updates the segments of the seam which are in need of recalculation, it is generally perceived as less jittery.

When the respondents were presented with instructions to focus only on the players in the video however this general trend which was seen in the table is no longer clear. The respondents found the same artifacts and issues in all videos. One important note is that we did not ask how many times in the video they noticed what they did, only if they did or not. If we had included that the results might have been different.

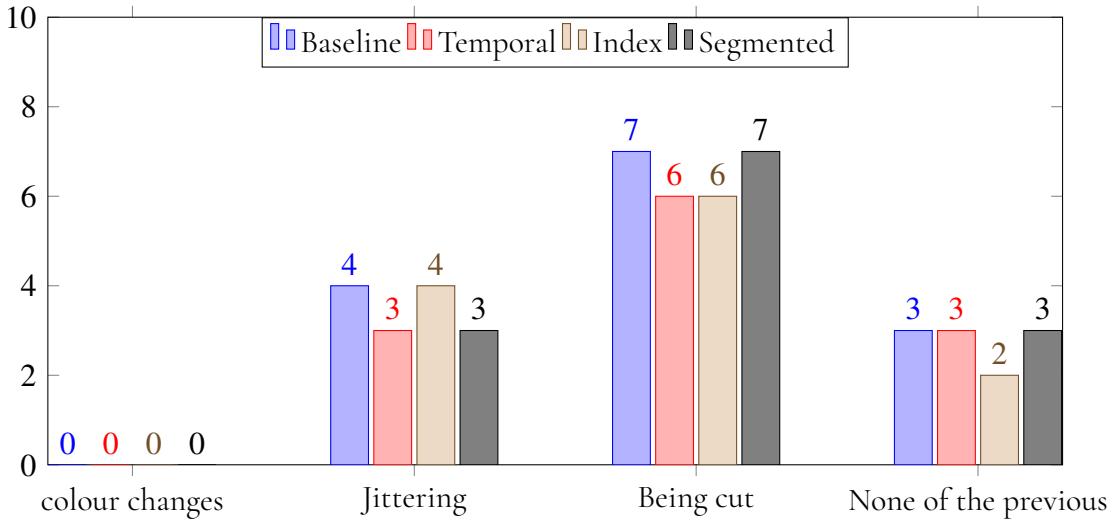



**Figure 4.1:** Bar chart of responses from user study where respondents could choose what they noticed in a video where we asked them to focus on the players.

The results from the user study show that the respondents slightly favor the segmented variant. It is not possible to draw any concrete conclusions from the data however because of the small sample size of the study.

# Chapter 5
# Discussion

This chapter will consist of a summary of the results for each algorithm followed by a discussion based around the results.

## 5.1 Summary

The segmented variant produced the best performance metrics and results from the user study. The temporal and index variants had opposite visual metrics, but were perceived with similar scores in the user study. The baseline variant consistently produced good SES, but was by far the worst in all other metrics.

## 5.2 Detailed discussion

As seen in the results, the baseline seam yielded the best or the second best scores for the Seam Evaluation Score in every measurement. This is to be expected as it recalculates the seam for every frame in the sequence and is configured to always find an optimal path for each individual frame. If the goal was to find the best way to stitch an image, this would be the best alternative. Nonetheless, this thesis focuses on video which is why the baseline is not as impressive when it comes to every other metric. If SES was excluded it would be by far the worst performing algorithm by every combined measure. This case truly highlights the differences between image and video stitching.

This is displayed by the user study as it was the worst performing in Question 1. When asked about the general quality the respondents did not find it appealing. It also seems as if Questions 1 and 3 are somewhat related as jittering, jumping of the seam, reduces general appeal of the video. The metric SMD, Seam Move Distance, should also be directly related to jittering which seems to be the case as a high SMD gave a worse score on Question 3 in the user study.

There are various risks with a user study. Firstly, it is difficult to draw any stark conclusions based on the responses. The respondents might be affected by some kind of altering physical or psychological matter e.g. tiredness, loss of concentration, slight colour blindness. This can and most likely will affect the results as it takes full concentration to notice slight differences in the provided videos. Secondly, they might not fully follow our instructions for each section of the evaluation which might be necessary to be able to adequately give their opinion. The result that we get from the panel is only as good as the questions we ask and the instructions we provide.

The comparison of the temporal variant and the index variant is an interesting study. The temporal does in fact have a high SES but also a lower SMD than the baseline which is the opposite of index. Even though they have opposite metric values they produce a similar result in the user study in all of the questions.

One could think that the temporal variant should perform better than the index variant in the user study as it is generally better to punish high jittering. This might however not always be the case as the temporal might follow a player who is running from one side of the focus area to the other side as it cannot make drastic jumps to improve the overall seam. The side effect of this can be that it creates a worse environment for future frames as the seam usually looks the most natural around the center line and not in the outskirts of the focus area. The index variant however always calculates the optimal seam from its start index rather than trying to reduce the move distance of each pixel in the seam. Finding an optimal seam seems to be as good of an alternative as it is on a reduced area compared to the baseline. This causes a smaller jittering effect which is good enough to compete with the temporal variant. Index can maneuver the entire image better than temporal which makes sure that the stitch is optimal. If the temporal variant had a different punish equation it might produce a better result and be more similar to the baseline variant while producing a better video. One such equation could allow x-values at the outskirt to not face as high of a cut off value as it does in our temporal cost function. One explanation to the results of the user study for the temporal variant is that the handball videos were the ones that produced the highest SMD values out of all of the different sports. If SMD was lower it might have resulted in a better video quality.

The focus of the segmented variant is to not do unnecessary work. It should only update when and where it is needed. It does well in this regard as it consistently produces the lowest SMD, Change % and Update frequency which results in the clearly lowest Total seam uptime. This is great from a performance point of view as it runs very fast. As previously mentioned, the high SES is because it never updates a new global optimal seam,but rather drifts towards the outskirts with each recalculation. This is because it might be following an object which is moving in one direction. This variant is sensitive to the parameters used. According to the user study this is still a better alternative as it had the best score for each question with a remarkable increase in the overall quality compared to the other variants. By keeping SMD low it reduces jittering which created a more appealing video.

A way of mitigating the drifting problem and the high SES score for the segmented algorithm would be to occasionally run a baseline algorithm to reset the seam. This is not something we have implemented but is definitely worth looking into for future work. One way to do this would be to run an update every once in a while when the play is detected to be close to the edges, meaning it is outside of the overlapping area. This detection is something that Spiideo has access to, since they have an AI feature called AutoFollow which automatically follows the interesting area of play.

We have not investigated what happens to the drift over longer sequences. We noticed that for the segmented seam, the SES slowly went up as we ran our tests on 300 frames. It would definitely be interesting to see what happens to the drift over an hour or two, and the seam reset could be a good solution to this problem.

The similar results in Figure 4.1 can be attributed to the parameters of the cost functions in combination with the differences of the algorithms.

### Different scenes

Another thing that's worth mentioning is that these algorithms will not perform the same for all sports and arenas. For example, in the ice hockey scene we used there was a large screen at the top of the image. This lead to very frequent updates of the seam, since the screen was following the play and was always changing. The index algorithm would perform very closely to the baseline in this case, as there would always be pixels changing in the seam near the top of the image.

From what we understand, seam algorithms generally perform better for indoor scenes than they do for outdoor scenes. This is because the lighting conditions indoor stay roughly the same, while it can change quite a bit for outdoor scenes. In football scenes it is not uncommon for a cloud to cast a shadow on only one of the two cameras, which would lead to a difference in colour balance and a very noticeable seam. The cameras are also usually located closer to the ground for indoor sports, which makes the parallax error more prevalent. Furthermore, some indoor sports like handball and basketball have a tendency towards play happening mostly close to the edges of the field. This is of course beneficial for a seam algorithm, since there will be less objects passing over the middle of the image where the seam is located.

It's worth noting that the optimal parameters for update functions will not take the same values for all scenes and sports. There is probably some future optimisation work to be done here, for example a program that tests a range of parameters and from the metrics can calculate good values for the parameters. Currently we do not have a better way of finding these parameters than to try out a few and see which values produce a good update frequency, i.e. the update function is not so sensitive that it reacts to individual pixel changes but at the same time not so rigid that it performs an update too late or not at all.

# Chapter 6
# Conclusion

Lastly, we will draw conclusions about our approach and the results by tying them to our research questions.

The first research topic was regarding finding existing algorithms. We did in fact find working algorithms, the baseline seam, but there were necessary alterations needed to produce a high quality video. By adapting different areas of the baseline we could create an algorithm which better fit our use case.

Measuring the performance was the second topic. Here, there was a clear difference between the different algorithms. The segmented algorithm was 14.6, 8.9 and 6.2 times faster than baseline, index and temporal respectively. No other algorithm could compare to the low total uptime of the segmented variant. This is an area for further work as more parameter tuning could improve the performance of all three algorithms in regards to update frequency. The segmented variant did nevertheless perform faster in runtime compared to the rest. We can not draw any conclusions whether or not the algorithms could be run in real-time as we did not have access to production hardware, however the performance of the segmented variant on our hardware suggests that it has potential for real-time processing.

Finally, the video output which was analysed with visual metrics and a user study also favoured the segmented seam. Our visual metrics gave somewhat of an indication of the results from the user study as a low Seam Evaluation Score and high Seam Move Distance correlated to lower subjective video quality. It is however difficult to draw any decisive conclusions on the video quality as the metrics of the user study could have been more concise and a larger population is needed. There was also no clear correlation between the qualitative and quantitative metrics.

It was interesting to see that the algorithm with the shortest execution time managed to produce the highest scoring results. It *seems* that sometimes, less is more.

# References


[1] Wikipedia. Translation (geometry) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Translation%20(geometry)&oldid=1178641124, 2024. [Online; accessed 11-March-2024].

[2] Wikipedia. Rotation matrix — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Rotation%20matrix&oldid=1212197050, 2024. [Online; accessed 11-March-2024].

[3] Wikipedia. Affine transformation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Affine%20transformation&oldid=1168715119, 2024. [Online; accessed 11-March-2024].

[4] Kenji Hata and Silvio Savarese. Cs231a course notes 1: Camera models. https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf. [Online; accessed 26-March-2024].

[5] Nidhal El abbadi, Safaa Alwan Al Hasaani, and Ali Hussein Abdul Khaleq. A review over panoramic image stitching techniques. *Journal of Physics: Conference Series*, 1999:012115, 09 2021.

[6] Wikipedia. IPhone 5 — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=IPhone%205&oldid=1211468263, 2024. [Online; accessed 21-March-2024].

[7] S. K. Nayar. Overview | image stitching. https://www.youtube.com/watch?v=J1DwQzab6Jg, March 3 2021.

[8] Home — opencv.org. https://opencv.org. [Accessed 26-04-2024].

[9] Botao He and Shaohua Yu. Parallax-robust surveillance video stitching. *Sensors*, 16(1), 2016.

[10] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 10–es, New York, NY, USA, 2007. Association for Computing Machinery.

[11] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 377–389, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[12] Wei LYU, Zhong ZHOU, Lang CHEN, and Yi ZHOU. A survey on image and video stitching. *Virtual Reality  Intelligent Hardware*, 1(1):55–83, 2019.

[13] Jie Hu, Dong-Qing Zhang, Heather Yu, and Chang Wen Chen. Discontinuous seam cutting for enhanced video stitching. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2015.

[14] Yihan Wu, Lidong Liu, Tong Niu, Yuying Zou, and Zhenling Yang. An adaptive seam line panoramic video stitching algorithm. In *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 133–138, 2022.

[15] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. *Graphcut Textures: Image and Video Synthesis Using Graph Cuts*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023.

[16] Kaimo Lin, Nianjuan Jiang, Loong-Fah Cheong, Minh Do, and Jiangbo Lu. Seagull: Seam-guided local alignment for parallax-tolerant image stitching. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 370–385, Cham, 2016. Springer International Publishing.

[17] Wikipedia. Laplace operator — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Laplace_operator`, 2024. [Online; accessed 25-April-2024].

[18] Wikipedia. Discrete Laplace operator — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Discrete_Laplace_operator#Image_Processing`, 2024. [Online; accessed 25-April-2024].

[19] OpenCV. Laplace operator. `https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html`, 2024. [Online; accessed 25-April-2024].

[20] Wikipedia. Roberts cross — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Roberts_cross`, 2024. [Online; accessed 25-April-2024].

[21] Irwin Sobel. An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*, 02 2014.

[22] OpenCV. Sobel operator. `https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html#gacea54f142e81b6758cb6f375ce782c8d`, 2024. [Online; accessed 25-April-2024].

[23] Junhong Gao, Yu Li, Tat-Jun Chin, and Michael S. Brown. Seam-Driven Image Stitching. In M.-A. Otaduy and O. Sorkine, editors, *Eurographics 2013 - Short Papers*. The Eurographics Association, 2013.

# Appendices

# Appendix A
# User study results

The detailed results of the user study with Respondents annotated as R1, R2 etc.
Question 1: What did you think of the overall quality of the players and the field?
Question 2: How well did the colours blend around the seam?
Question 3: How noticeable was the jittering?

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Q1 Vid 1 | 4 | 3 | 4 | 2 | 4 | 2 | 2 | 2 | 3 | 3 |
| Q1 Vid 2 | 4 | 4 | 4 | 2 | 4 | 3 | 2 | 2 | 4 | 4 |
| Q1 Vid 3 | 3 | 3 | 4 | 3 | 4 | 4 | 2 | 2 | 3 | 4 |
| Q1 Vid 4 | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 2 | 5 | 5 |
| Q2 Vid 1 | 4 | 5 | 2 | 5 | 5 | 5 | 4 | 5 | 5 | 3 |
| Q2 Vid 2 | 4 | 5 | 3 | 5 | 5 | 5 | 4 | 5 | 5 | 3 |
| Q2 Vid 3 | 3 | 5 | 3 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| Q2 Vid 4 | 4 | 5 | 3 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| Q3 Vid 1 | 3 | 1 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 2 |
| Q3 Vid 2 | 4 | 3 | 4 | 2 | 2 | 3 | 2 | 1 | 4 | 3 |
| Q3 Vid 3 | 3 | 2 | 4 | 3 | 1 | 3 | 2 | 1 | 5 | 3 |
| Q3 Vid 4 | 4 | 4 | 4 | 4 | 3 | 3 | 2 | 1 | 5 | 3 |

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2024-05-24

**EXAMENSARBETE** Analysis of seam algorithms in video stitching with static scenes and dynamic objects
**STUDENTER** Amin Alian, Ludvig Gierup
**HANDLEDARE** Michael Doggett (LTH), Håkan Ardö (Spiideo AB)
**EXAMINATOR** Per Andersson (LTH)

# Undersökning av algoritmer för videostitchning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Amin Alian, Ludvig Gierup**

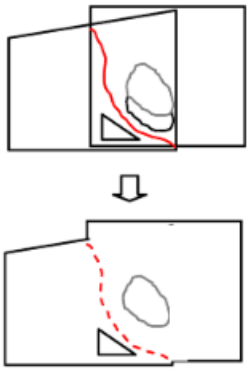
De senaste åren har sport har blivit mer datadriven och beroende av kameror. För att kunna automatisera datainsamlingen av sportinnehåll behövs högkvalitativ video. Detta arbete har analyserat olika algoritmer för att skapa bra videos där två kameror riktas över samma plan.

För att kunna livesträmma och spela in högkvalitativ video av sport behövs det ofta mer än en kamera för att täcka vidden av en hel plan. När man använder fler än en kamera blir det direkt en svårare uppgift att få ihop en video. Tekniken man använder kallas *stitch* som innebär att man kombinerar flera videoklipp till ett enda klipp.

För att förbättra stitchtekniken har detta arbetet undersökt vilka algoritmer som redan finns inom detta fält och testat dem. De algoritmer som var i framkant inom generell videostitchning fungerade inte så bra när det kom till sport. Problemet som algoritmerna stötte på var att olika objekt på planen såg olika ut från olika vinklar.

Ett sätt att förbättra en stitch är att försöka hitta en söm genom de olika kameravinklarna. Detta undviker att sömmen skär genom viktiga detaljer i bilden. Om en söm skär genom objekt i bilden kan det resultera i att objekten förvrängs. Efter att man har hittat en bra söm sammanfogas de olika vinklarna. Med hjälp av sömmen skapas ett sammanhängande videoklipp. Ett exempel på hur det skulle kunna se ut syns i figur 1.

Efter vår undersökning där fokus var på prestanda och att video ska se naturlig ut har vi sett att en uppdelad algoritm som endast uppdaterar vissa delar av sömmen är bättre än att i varje bild av videon försöka hitta det optimala sättet att stitcha bildflödena från kamerorna.



Figur 1: Exempel på en söm genom två olika vinklar.

Detta skapar en naturligare upplevelse för tittaren, då mindre uppdateringar går obemärkta oftare än större uppdateringar.