Universiteit Leiden

Master Computer Science

Tracking long track ice skaters in the bend

Name:	Rik Zandbelt
Student ID:	s1847503
Date:	July 6, 2021
Specialisation:	Data Science
1st supervisor:	Arno Knobbe
2nd supervisor:	Rens Meerhoff

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In long track speed skating, there are many different paths to take and techniques to use when skating through the bend of a long track ice skating rink. In order to find out which path and technique yields the fastest time, it is very useful to be able to examine the trajectories for both the left and right skate of a skater that is skating through the bend.

In order to do this, a camera is installed in the ceiling of the Thialf ice skating stadium, providing top-view video footage of skaters that skate through the bend. This research contributes to the end goal of determining the best path and technique by attempting to automatically track the skates of skaters in the video footage. The video footage is annotated manually in order to create the ground truth, being the real positions of the skates throughout each video.

This research gives an overview of the existing state-of-the-art methods for tracking objects and for tracking ice skaters in general, after which it presents two novel CNN-based methods for tracking both separate skates for each skater. The first method analyzes an image of a skater and then predicts the coordinates for the two skates, while the other analyzes a combination of a skater image and a potential skate and predicts whether the potential skate is actually one of the correct skates of the skater. Finally, this research presents a way of getting from the predicted skates to skate trajectories that can be analyzed.

Although the experiments that are conducted during this research show that the models are not as accurate as they need to be in order to use them to analyze the skate trajectories, this research does provide a solid start for this skate tracking task as well as an analysis of which approach is likely to produce a model that is accurate enough.

Contents

1 Introduction

1.1 Long Track Speed Skating

Long track speed skating is an Olympic sport where skaters try to cover a set distance as fast as possible. Regardless of the distance that needs to be covered, long track skating competitions usually take place on an oval-shaped rink with a total length of 400 meters.

Besides being in physical top condition, skaters also require a good skating technique to skate as fast as possible. In particular when skating through the corners of the rink, they need to find the optimal strategy because simply taking the shortest path will not necessarily yield the fastest time; by going for the shortest path, a skater might lose speed and thus be slower than by taking a seemingly longer route. Moreover, different techniques can be used for approaching the bend and for moving different parts of your body (most notably the legs) while skating through the bend.

This research assists in finding out how to skate through the bend as fast as possible by attempting to track skaters that are going through the bend using top-view video footage. Different tracking methods are employed in an attempt to track both skates of the ice skaters. When the skaters can be tracked successfully, the data of the skaters' trajectories can be used to determine the fastest way to go through the bend.

1.2 Problem Statement

The camera footage that is used to track the skaters, comes from a camera that is installed in the ceiling of the Thialf speed skating stadium in Heerenveen, the Netherlands. The camera is pointed at the part of a bend of the 400 meter rink. The data consists of videos from this camera, in which skaters skate into the view of the camera on one side of the bend and skate through a part (1/6) of the bend before exiting at the other side of the camera view. More information about the camera and the data will be given in section 3.1. In order to track the movement and the technique of the skaters, this research focuses on tracking both skates of each skater, specifically the blades of the skates. Tracking the skates instead of the whole skater allows for finer insights in the technique of the skater. The blades are thin, straight objects so they can each be represented by a straight line in the video frames.
Therefore, the research question for this thesis is the following: How can we track the trajectories of the skates of long track skaters in a video frame by finding the line that corresponds to a skate blade for each skate in each video frame?

1.3 Tracking Challenges

While building a model that can automatically track speed skaters in to-view camera footage, multiple challenges must be taken into account.
First of all, a skater wears two skates while skating. Not only do both of their skates need to be tracked, but we need to keep track of the identities of both skates in order to know which skate is on the left foot and which one is on the right foot at any time. This is made difficult by the movements of the skaters, especially because the skaters are in the bend of the rink when filmed. While skating through the bend, they often put one leg over the other as shown in Figure 1. This causes the skates to be occluded occasionally, making them harder to track. Additionally, the skates are close to each other during this process of crossing legs, which could make it hard to distinguish the left and right skate.

Figure 1: *A speed skater putting one leg over the other while skating through the bend. This makes it difficult to distinguish which skate is the left one and which is the right one.*

Another challenge is that there can be multiple skaters in a frame. All skaters need to be tracked and both skates need to be tracked for each skater. Furthermore, a skater can sometimes occlude the body or skates of another skater, which makes the tracking even harder.

Although the blades itself are rigid objects, the skaters that wear them are not. Because the positions of the skates depend on the position of a skater, which is a fast-moving and non-rigid object, following the trajectory of the skates can be challenging. Furthermore, the body of the skater can be in various unconventional positions during skating, which is hard to track and could cause occlusions.

In short, the main tracking challenges that were mentioned in this section can be summarized by the following points:

1. A skater wears two skates that both need to be tracked.
2. Multiple skaters can be present in a frame, which might cause occlusions. Besides, this also causes the individual skaters to be harder to detect and distinguish before tracking the skates.
3. The positions of the skates depend on the position of the skater, which is a non-rigid and fast-moving object.
4. The appearance of a skater (especially their silhouette) changes and can take on unconventional shapes during skating.

1.4 Goals

The first goal of this research is to give an overview of the state-of-the-art methods for tracking objects, as well as to note what work has already been done on tracking ice skaters specifically.

The second goal of this research is then to use these state-of-the-art methods to create a novel method that attempts to track the skates of long track speed skaters, who are skating through the bend of an ice skating rink, as accurately as possible. This novel method has to solve the problem that is described in section 1.2, meaning that it can handle the challenges that are mentioned in section 1.3.

The aforementioned research goals are reached through the following steps: Section 2 gives an overview of the state-of-the-art work on Video Object Tracking in order to select the methods that might be useful for creating a novel method. Section 3 describes the characteristics of the data, as well as the process of data annotation that we use to create the ground truth for tracking the skates. We use the information about the state-of-the-art to present two novel methods for tracking the skates. The first one is introduced, evaluated and discussed in section 4 and the same is done for the second method in section 5. In section 6, we describe the process of getting a trajectory for a skater after predicting its skates. In section 7 we conclude the research by discussing how the experimental results relate to the research goals, as well as providing some suggestions for future work.

2 Existing Work

2.1 General Object Tracking

A lot of work has previously been done on Video Object Tracking, so there are a lot of existing algorithms that could be used for the task of tracking ice skates on camera footage. Object Tracking algorithms in general use an 'appearance model' to represent the object that has to be tracked [1]. Such an appearance model consists of two components:

1. Visual Representation, which uses some features to represent the object and its location. Many different kinds of features exist that can be used for this [1].
2. Statistical Measuring, which uses the visual representation to calculate the relation (according to some measure) between two frames. The result shows how the object has moved between the frames.

Two types of object tracking methods can be distinguished: Detection-Free Tracking (DFT, also known as Visual Object Tracking) and Detection-Based Tracking (DBT). Those two categories differ in the way the objects are initialized in the first frame of a video, as will be discussed in the following subsections.

2.1.1 Detection-Free Tracking

The steps in the process of a DFT method are shown in Figure 2. DFT often focuses on one object. The object that needs to be tracked is initialized manually, in contrast to the automatic detection that DBT methods use (see section 2.1.2). This means that DFT methods need a human to determine when the object first appears in the video and where it is located in that first frame. After this initialization the object is tracked in the subsequent frames, giving the trajectory of the tracked object as output.

Figure 2: *The process of Detection-Free Tracking.*

DFT relies on human view for object initialization, which can determine the location of objects very precisely. Because the human annotator can indicate what the object looks like, DFT is suitable for tracking any type of object. However, the need for human intervention is a drawback of DFT.

A common way to benchmark DFT algorithms is through the Visual Object Tracking (VOT) challenge [2], which is updated each year with a new ranking of the best DFT algorithms that participate in that year's challenge.

Because the VOT challenge is a well-known benchmark in the Object Tracking field, looking at the best performing trackers in the most recent challenges gives a good overview of the state-of-the-art DFT methods. At the time of writing, the most recent edition of the VOT challenge is the VOT2020 challenge.

In the most recent VOT challenges, the primary measure that is used to evaluate the trackers is called the 'Expected Average Overlap' (EAO), which measures the estimated average overlap between the predicted object and the real object over all the frames in a video. [3] The VOT challenge uses the 'state-of-the-art bound' (SotA bound) to determine whether an algorithm is state-of-the-art. This bound is equal to the average EAO score of several participating algorithms that have recently been published by prominent computer vision journals or conferences. Algorithms that exceed the bound in a VOT challenge, are considered state-of-the-art when that VOT took place.

Although the participating algorithms in the VOT challenges of 2018 [4], 2019 [5] and 2020 [6] were based on several different tracking methods, the algorithms that exceeded the SotA bound either used Siamese Networks, Correlation Filters, Convolutional Neural Networks or a combination of those methods.

Siamese Networks became a popular method for object tracking when Bertinetto et al. published SiamFC in 2016 [7]. Siamese networks learn a similarity function and compute the similarity between two frames. Then they compute the similarity between the target object and each region of the frame in order to localize the target. Siamese Networks are trained offline and that makes them relatively fast.

Using Correlation Filters for tracking objects became popular when Bolme et al. used it for their MOSSE tracker in 2010 [8]. Correlation Filter-based trackers use a filter, which represents the target object, to perform convolution. The output of the convolution indicates which part of the frame is most similar to the filter and thus where the target object is most likely located. The filter is computed online and is updated over time using the frames that were observed before. Because of this online computing, Correlation Filters are often slower than Siamese Networks.

In the latest VOT challenges, many participating algorithms use a combination of Correlation Filters and Siamese Networks. Examples of this are ATP (Accurate Tracking by Progressively refining) in the VOT2019 challenge [5] and RPT [9] and AlphaRef [10] in the VOT2020 challenge [6], which all were amongst the top-performing algorithms in their respective challenges. Furthermore, all of those combination-based methods use a two-phase approach: first they estimate the location of the target and then it searches in that estimated location for the precise location and state of the object. This approach was first introduced by ATOM (Accurate Tracking by Overlap Maximization) [11], which makes it an important inspiration for those two-phase methods.

2.1.2 Detection-Based Tracking

The steps in the process of a DBT method are shown in Figure 3. As the figure shows, the objects that need to be tracked are detected by a trained object detector as soon as they are visible to the camera, after which they are tracked and this yields a trajectory for each object. DBT methods often track multiple objects in the same frames. This makes the tracking more difficult, as a DBT method needs to keep track of the identities of all objects in case of confusion between the objects (for example because of occlusions). This makes DBT algorithms more complex than DFT algorithms (as described in section 2.1.1), which use human view.

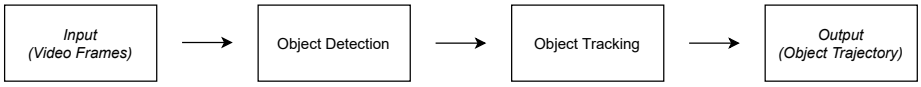

Figure 3: *The process of Detection-Based Tracking.*

Automatically initializing the objects means that no human involvement is needed during DBT, which is convenient. However, this also increases the complexity of the problem as the tracker has to also detect the object(s) that it has to track.

Like the VOT challenge is a benchmark for DFT algorithms, the MOT (Multiple Object Tracking) challenge [12] provides a benchmark for DBT algorithms. The results of the most recent MOT challenges show that there are no types of DBT algorithms that perform better than all others on most tasks, probably due to the complexity of DBT problems. For some tasks complex methods do outperform simple ones, but for others it is the other way around. Therefore it seems that the best DBT algorithms are trained to solve a specific task and there is no one method that can solve most DBT problems.

As expected due to the complexity of DBT, the field of DBT has not progressed as far as DFT. However, the DBT field has been inspired by the progress of DFT and DBT has seen improvements over the recent years. Because DFT and DBT do both use an appearance model, some researchers have successfully used DFT methods as a basis for creating DBT algorithms.

The most prominent example of this is the rise of Convolutional Neural Networks (CNNs), which are neural networks that are designed to handle Computer Vision-tasks and that take inspiration from the Correlation Filters that were mentioned in section 2.1.1. While methods based on Correlation Filter use one filter that represents the whole object, CNNs use multiple filters which each represent a part (feature) of the object.

Each filter is used for convolution, creating a neural network with one or multiple 'convolutional layers' for extracting features that can be alternated with 'max pooling' layers, which enhance the features. Because CNNs divide the object into different features, CNN-based methods are able to deal better with occlusions than Correlation Filters. CNNs are more complex and take longer to run than the methods mentioned before, but they are able to handle the complex DBT challenges better than simple methods. Because the computing power has increased over the past years, CNNs have become increasingly more popular and are a big part of the state-of-the-art for complex object tracking tasks.

For commonly tracked object such as humans or vehicles, trained CNNs have already been created that can detect those objects very accurately. Those pre-trained detectors can be used to initialize objects as soon as they appear in the video. Using such an external pre-trained model to solve another related problem is called `transfer learning`. Because pre-trained CNNs exist for detecting and tracking humans, transfer learning could be applied in the task of tracking ice skaters as an intermediate step towards tracking their skates. This is explained further in section 2.2.

However, there might be no pre-trained model when tracking objects with uncommon shapes. In that case transfer learning might be impossible and a new object detector needs to be trained in order to perform DBT. This makes it more likely that the detector is not able to initialize the location of the objects precisely, especially when there is not much data available for training the detector. Not being able to precisely initialize the objects is a huge problem for object tracking because tracking algorithms rely heavily on the correct initialization of the objects. Unfortunately, there is no pre-trained detector for ice skates, so this issue applies to this research.

2.2 Object Tracking in the Field of Ice Skating

2.2.1 Models for Tracking Short-Track Ice Skaters

While much work has been published about object tracking in general, little work has been published on the subject of tracking long track ice skaters specifically. However, some DBT algorithms have been developed for tracking skaters in short track ice skating [13–16]. Although those methods are designed for tracking the whole human body and we aim to track the separate skates of a skater, the methods could be useful as an intermediary step towards tracking the skates. Knowing where the body of the skater is, should make it easier to detect the position of the skates. However, this intermediary step does make the method more complex.

Although the methods are designed to track short track ice skaters, they are likely still capable of tracking long track skaters because tracking short track skaters is even more complex. Short track skaters are more difficult to track than long track skaters because more skaters participate in a short track race and they are all skating on the same stretch of ice where they often pass by each other or crowd together (especially in the bend of the rink), as shown in Figure 2.2.1. This causes skaters to be occluded by each other more often and it causes the occlusions to be much more severe than in long track skating.

Figure 4: *A video frame of short track. Because the skaters are trying to pass by each other, they are closer together. This causes more occlusions than with long track skating. Adapted from the paper by Yuxuan Wang. [14]*

Although skaters have their own lane in regular long track skating matches, there are often multiple skaters in the same lane in the long track skating camera footage that is used for this research. However, in those cases the skaters are skating behind each other the whole time and do not intend to pass by each other. Therefore, there are fewer occlusions between skaters in the long track skating footage used for this research than in short track skating. Additionally, the occlusions that do take place in the long track data are less severe.

Furthermore, a lot of the existing approaches for tracking short track skaters use a panning camera to track the skaters over the whole rink, while the cameras that are used for collecting our data are not moving because they only focus on the bend of the rink. Because there are relatively few occlusions and there is no moving camera used for this research, some of the challenges of tracking short track ice skaters do not have to be solved or are present to a lesser extent in this research. For the reasons mentioned above, it is likely that a model that is designed to track short track skaters, is also able to track long track skaters.

2.2.2 Using Existing Methods for a Novel Two-Step Approach

Now that we have established which existing methods could be useful for this research, we have to decide whether we are going to use any of those methods that are designed for tracking human bodies, for transfer learning. We argue that the extra information that the location of the body of the skater provides for the task of tracking the skates, is worth the complexity of an intermediary step. As discussed in section 2.1.1, it is common to use a two-phase approach for tracking objects, like aforementioned Atom algorithm [11]. In the case of this research, the location of the skater is detected in the first stage and then that estimated location is searched for the precise location of the skates.
In this approach, the second stage could utilize the information that the two skates of a skater are always located under the feet of the skater (one skate under each foot).

An algorithm that is trained on tracking human bodies could first be used to track the body of the skater. In order for an algorithm to be suited as this intermediary step, it should be able to deal with the challenges as mentioned in section 1.3. Because we want to be able to track the skaters without human intervention, the chosen method should be a DBT method rather than a DFT method. However, because DBT methods can be created using DFT methods as inspiration, the state-of-the-art methods for DFT that are discussed in section 2.1.1 could still be used for this research. Another reason why the problem of tracking skates is a DBT problem, is that multiple skaters can be present in a frame and thus we need to keep track of their identities to handle occlusions.

In order to select an algorithm to track the body of the skater, it is crucial to make a distinction between algorithms that only track the whole human body and algorithms that track the whole body and also the position the body has. Algorithms in the latter category might be more suitable for tracking the skates as they make it possible to locate the feet specifically, while ones in the former category do not provide enough information about the exact position of the body and will thus not be able to produce suitable estimations of the skate locations.

When looking at the methods that have been designed to track short track skaters and that are mentioned in section 2.2.1, they all fall in the category of algorithms that only track the body without including any information about the position that the body is in. Although it is better to use methods that do include that information, methods that do not include it could also help. The methods for short track that we consider using are the ones that track the whole silhouette of the skaters and are DBT methods [14,16]. The other methods only draw a rectangle around the skaters, which is much less informative to use in the second stage.

One of the latest DBT methods that has been designed for tracking skaters and does track the whole silhouette of the body of skaters, is a method from 2012 by Yuxuan Wang [14]. This method uses Gaussian Mixture Models [17] and novel 'Fuzzy Memberships' in order to track the bodies of skaters and deal well with occlusions. Although this method does deal well with occlusions, it does not deal well with the changing shape of the skaters during skating because the appearance model for the skaters is not updated after it is initialized after the first frame. The approach by Chenguang Liu [16] seems to be more suitable for this research, because it does deal well with the changing silhouette of the skaters. Their method is also designed to track short track skaters and it uses a Random Forest based method to fuse multiple features in order to update the appearance model and deal with the changing appearance of the skaters. Furthermore, the method uses 'Blob Growing' in order to classify each blob (silhouette) to the correct skater and thus keep track of the skaters' identities during occlusions.

Although the method by Liu [16] seems to be a suitable first step of finding the skates, it might be difficult to locate the foot from the resulting silhouette of the algorithm. As mentioned before, there are also algorithms that do include information about the position of the body. Those algorithms are called `Pose Estimation` algorithms, and they track a body in the form of a skeleton that indicates where each body part is located. OpenPose [18] is the most prominent example of a pose estimation algorithm, and algorithms like that could be useful to get the pose of the skater and thus get the locations of the left and right foot. OpenPose uses CNNs to produce an estimation of the pose of a human in a picture, and is even robust to occlusions. This makes it suitable for detecting the feet of multiple skaters in a frame.

2.2.3 From Body to Skates

After using an existing method to locate the body of the skater, the second step is to use this information to locate both skates of the skater. The blades of the skates are located directly under the feet of the skaters and the skates have an angle that is roughly the same as the angle of the foot they are connected to, so the location of the feet of a skater substantially narrows down the possible locations of the skates a lot and the area under the feet can be used as estimated locations for the skates. Knowing the angle of the skate could also help a lot when the skates are (partially) occluded, as the precise locations of the skates can be estimated by finding the most likely candidate for a line with the correct angle in the small area under each foot.

Now that the state-of-the-art in Object Tracking and the related methods on tracking ice skaters are clear, those existing methods have to be extended to be able to track the ice skates specifically. This process is described in section 4. Before describing that, we describe the data that we are going to use for this in section 3.

3 Data

3.1 Data Characteristics

As mentioned in section 1, the data that is used in this project consists of videos from a camera that is attached to the ceiling of the Thialf long track ice skating rink in Heerenveen, the Netherlands. The videos are filmed using a 'Basler acA2500-60uc USB 3.0' camera [1] and a 'Kowa 8mm LM8HC 1", Sensor F1.4 C-mount' lens. [2]

Each video is recorded using the following camera settings:

- The video duration is 2200 miliseconds (2.2 seconds)
- The framerate is 148 frames per second
- Each video contains 326 frames

The camera captures about 1/6 of the bend of the rink and the videos from the camera show one or more skaters that skate through the frame, seen from above like in Figure 5. They skate behind each other without passing by one another for the whole duration of the video. The videos are saved as `.avi` files. The amount of skaters per video ranges from 1 to 9 with a median of 3 skaters per video and a mean of approximately 4 skaters per video.

Figure 5: *An example of a frame from one of the videos. In this video, 7 skaters are skating through the bend*

At the time of writing this thesis, it is expected that more of those cameras will soon be installed throughout the bend. Together, the cameras will be able to capture the whole bend so that the whole trajectory of the skater through the whole bend can be determined. This will create opportunities for future work, for example to analyze those trajectories and determine the optimal trajectory to follow when skating through the whole bend.

[1] `https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2500-60uc/`

[2] `https://lenses.kowa-usa.com/hc-series/472-lm8hc.html`

3.2 Data Annotation

In order to create a model that can accurately determine the positions of the skates of a skater, we need to produce a ground truth for the videos in order to train the model. We do that by manually annotating videos. For each video that we want to annotate, we first select each skater in the video by finding a frame where all skaters are visible and then selecting a rectangle around each skater which serves as the Region Of Interest (ROI) for that skater. After the skaters are selected, we use a simple Blob Detector from OpenCV[3] to follow them through the frames. For each skater in a frame, we manually draw two straight lines (one on each skate). An image with the ROI and the positions of the lines for the left and right skates in that image are then saved and the lines serve as the ground truth. We only generate such annotated images once every five frames to prevent overfitting on certain images, because subsequent frames are too similar. If a skates for a skater in a frame is not visible enough to annotate them, we skip the annotation for that skater in the frame. Therefore, there are some gaps in the annotations, but this is unavoidable as the skates are sometimes occluded.

4 Experiment 1: Coordinate-Predicting CNN

4.1 Method

As mentioned in section 2.2, we chose to use a two-step approach, where we track the whole body of a skater in the first step and then we use the information of where the body is to get the location of the skates in the second step.

4.1.1 Step 1: Enhancing Data

For the first step, we tried different ways of tracking the bodies of the skater. We first try to acquire the code for methods that are trained to track the bodies fast-moving short-track skaters, such as the method by Liu [16]. This method estimates the outline of the skaters that it tracks. However, we did not succeed in getting the code for such a method as it is not open-source and contacting the authors yielded no results.

Because we could not get access to a method that is specifically made for the task of tracking ice skaters, we used general methods for extracting the outline of objects. First of all, we used a basic computer vision technique called `background subtraction` to find the silhouette of the skater and remove any noise in the background. This technique scans the video frames and creates a 'background mask' consisting of all non-moving parts of the video, and then it makes the background black and the moving objects white.

[3] `https://docs.opencv.org/master/d0/d7a/classcv_1_1SimpleBlobDetector.html`

We use the OpenCV library, which includes many different techniques for background subtraction[4]. We tried the three most prominent ones, which are all based on Gaussian Micture Models: KNN [19], MOG [20] and MOG2 [21] (an improvement of MOG). We dropped the MOG method because it also detects the shadows of the skaters, which we do not want because the shadows of the skaters are often visible on the ice. The KNN and MOG2 methods have an option to remove the shadows. We tried both methods and while they both did better than MOG, the shadow removal worked visibly better on the KNN method than on the MOG2 method, as there were regularly still parts of the shadow visible after using the latter. For those reasons, we decided to use the KNN method for background subtraction. An example of the effect of this background subtraction method is shown in figure 6. It is visible in the figure that background subtraction has some issues when used on skaters wearing white skates because they have roughly the same color as the background, but it does still indicate the blade of the skate and white skates will also be an issue when using the plain images

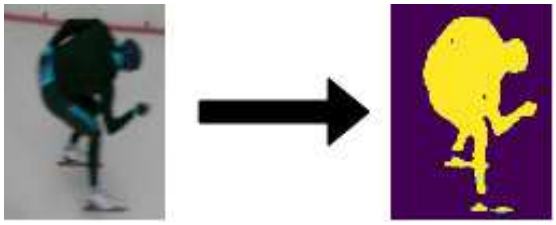

Figure 6: *An example of an image before (left) and after (right) background subtraction. The noise (the red line above the skater) is removed because it is part of the static background.*

We applied background subtraction on all videos and this way we obtained a version of each annotated skater image where the background is subtracted. We did this because we believe that it becomes easier to track objects when there is less noise in the background. When creating a model, we try to train the same models on this 'enhanced' dataset to see what impact background subtraction has on the model performance.

Although background subtraction does indicate where the skater is located, it does not retain the details on the skater's body that are in the original image. Because of this, background subtraction might also make the tracking challenge more difficult and it could be a better idea to enhance the original images with only the outline of the skater. Fortunately, we can use the silhouettes from the background subtracted images to find the contours (outline) of the skater in the original image. For this we used a general method for extracting the outline of objects (`contour detection`).

[4] `https://docs.opencv.org/3.4/d7/df6/classcv_1_1BackgroundSubtractor.html`

For the contour detection we used a method by Suzuki [22], which is a simple method from as early as 1985. Despite the method being old and simple, it does a very good job of finding the contours of the skater when we use it on the images where the background is already subtracted because there is no background noise and the only object to draw a contour around in those images is the skater. When we then draw the found contours back onto the original image, we get an enhanced image (as shown in figure 7) that contains the extra information of the outline of the skater's body. Just like with background subtraction, we save another enhanced version of each image after contour detection to train each model on. We do this in order to see what effect contour detection has on the model results.

Figure 7: *An example of an image before (left) and after (right) contour detection. The contour (outline) of the skater is added to the image.*

In section 2.2, it was also mentioned that it would be even better to add extra information about the location of the different body parts to the image than to only add the outline of the skater. In order to accomplish this, we tried to use the prominent pose detection algorithm OpenPose [18] to detect the pose of the skaters, after which we could then use the estimation of the pose to find the skates. However, OpenPose is not trained on the poses of fast-moving bodies such as skaters. Therefore the pose estimations were almost always incorrect and this led to bad skate estimations.

We also tried to use AlphaPose [23], which is suitable for estimating poses of bodies that are fast-moving, but even AlphaPose did produce an incorrect pose in almost all cases. We suspect that in order for those algorithms to work properly, they must first be partly re-trained on some of our skater data to properly get acquainted with the angle of the camera. However, we do not have any way of annotating the poses of the skaters in each frame within the time constraint of this research.

So after trying different ways of incorporating extra information about the location of the body of the skater into the images, we have three different datasets:

- The original images of the skaters.
- The black-and-white images after background subtraction.
- The images where the contour of the skater is added as described in this section.

When the data is enhanced with extra information about the location of the body of the skater, we have to find the location of both skates using a CNN model. The basic workflow that is followed from the original videos until feeding the data to the CNN is shown in figure 8.

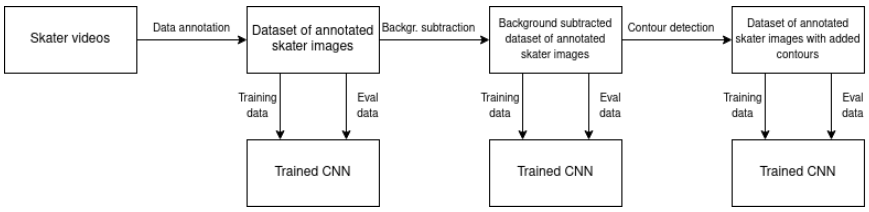

Figure 8: *The workflow from the original skater videos to feeding the data to the CNN*

The process of using the CNN to find the skates, which is step two of this two-step approach, is explained in section 4.1.2.

4.1.2 Step 2: Predicting Skate Coordinates

As mentioned in section 2, Convolutional Neural Networks (CNNs) are a big part of the state-of-the-art in computer vision at the time of writing, especially for complex tasks. As this task has a lot of challenges such as the great speed of the skaters or the need to keep track of the identities of the left and right skate, we argue that this task is complex and is thus best solved by a CNN. Because no prior solution to this task can be found on the internet, we have to design a CNN that is hand-tailored for this task from scratch.

CNNs are able to detect objects well, provided that they are fed enough data. The big disadvantage of the hand-tailored CNN approach is that we need to create the data ourselves by annotating skater videos (as described in section 3.2) and this is a very time-consuming process because CNNs typically need a lot of data in order to perform well.

When I started with this project, a basic CNN had already been created for this by a fellow project member. This first CNN got the image of a skater as the input layer and it produced 8 numbers as output. Each skate is seen as a straight line between two points. The outputs of the CNN are the predicted x-coordinate and y-coordinate for both of these skate points for both skates of the skater.

That first CNN consists of two convolutional layers with kernel size 3, each followed by a max pooling layer with kernel size 2. Those layers are followed by three fully connected linear layers and then by the output. For the loss function, it uses a custom 'skater loss' function. This loss function is defined as the sum over all distances between the predicted skate points and the annotated 'ground truth' skate points in an example image. The loss is then the mean over all examples of this summed distance.

As we show in section 4.2.4, we found that the CNN with this structure was overfitting on the training data. The network was too complex and has too many layers, because after experimenting with the structure of the network we found that removing one convolutional layer and one linear layer caused the CNN to perform better. Therefore, the final structure of this CNN consists of one convolutional layer, followed by a max pooling layer and two fully connected layers. We also tried including a `dropout rate` in the fully connected layers, which introduces a probability for each node to be turned off. This causes the network to be less dependent on certain nodes, which combats overfitting. However, we decided to not include it in the model as it only increased the loss function.

4.2 Experiments

4.2.1 Evaluation Metrics

For evaluation of the models that we use in this research, we use three metrics: precision, recall and F1 score. The metrics are evaluated per skater, and as evaluation videos we use 5 (annotated) videos with 11 skaters in total. Those videos contain a variety of data characteristics: they vary in number of skaters per video, color of skater suits and color of skates. This makes sure that a model that is overfitted on one type of video does not yield good evaluation scores. The metrics for each skater are averaged over all frames.

First, a margin of error (MOE) is defined. Then we loop over each frame in the video and for skater x in one frame the metrics are defined as follows:

- $\text{Precision}_x = \frac{|\{\text{skate points}_x \text{ are predicted}\} \cap \{dist(\text{predictions}_x, \text{ real points}_x) < MOE\}|}{|\{\text{skate points}_x \text{ are predicted}\}|}$
- $\text{Recall}_x = \frac{|\{\text{real skate points}_x \text{ exist}\} \cap \{\text{skate points}_x \text{ are predicted}\}|}{|\{\text{real skate points}_x \text{ exist}\}|}$
- $F1_x = 2 * \frac{\text{Precision}_x * \text{Recall}_x}{\text{Precision}_x + \text{Recall}_x}$

4.2.2 Data Separation and Experimental Settings

The CNNs in this research are trained on 1161 annotated images from 35 skater videos. The videos are split in a training set and a test set, where for each video either all images are placed in the training set or they are all placed in the test set. This is done because otherwise some images in the training set might be too similar to images in the test set and this may cause overfitting. The videos are divided over the training and test set in such a way that the split is approximately 80(train)/20(test).

The models are evaluated on the 11 skaters in the 5 evaluation videos. The margin of error for the evaluation metrics is set to 20 because a lower evaluation metric causes very low precision and f1 scores, as barely any skates are found that are close enough to the real skates.

The precision, recall and F1 scores for each skater are shown per model in box plots to make it clear which models are the best ones.

4.2.3 Baseline

For the baseline that we are comparing the CNN to, we use a 'simple' (especially when compared to the rather complex CNN) method. This baseline method is a function that finds all possible skate lines for each image of a skater, using a Canny edge detector first and then a Hough transform. Then it predicts which of those are the correct lines for the left and right skates based on the characteristics of the lines.

With this method, first a dataset is created that is used to predict which lines are the correct skate lines using the annotated videos. For each annotated example of a skater in a frame, the characteristics (x and y coordinates and degree) of the lines for the left and right skates are saved as features in a Pandas Dataframe together with a label, which is 1 if it is the left skate and 0 if it is the right skate. When this is done for all examples, this dataset is turned into a vector space, called the The baseline that we use to compare our models against is called the 'Probability Map'. Every point in the Probability Map belongs to either class 0 (the 'right skates') or class 1 (the 'left skates').

When an unseen skater image is processed, the method retrieves all possible skate lines and then uses the a kNN classifier to determine which line is the most likely to be a left skate (when the predicted label is closest to 1) and which is most likely to be a right skate (when the predicted label is closest to 0), according to the Probability Map.

4.2.4 Experimental Results

The following box plots show the results for the baseline and the different coordinate-predicting CNNs (the initial CNN with two convolutional layers and the final CNN with one convolutional layer, both having versions for the plain images, the images after background subtraction and the images after contour detection) on the 11 skaters in the 5 evaluation videos. All CNNs are trained on 35 videos. The dots show the data behind the box plots: each dot is the average score for a skater in one of the evaluation videos over the whole video. The box plots then show the distribution of those dot values. Figure 9 shows that the probability map baseline has the best precision scores. The initial CNN model is by far the worst, while the final CNN is better than the initial CNN, but overall worse than the probability map.

While the CNN that is trained on images with added contours is slightly better than the CNN that is trained on the plain images, the CNN that is trained on the background subtracted images has a far lower precision than both of them.

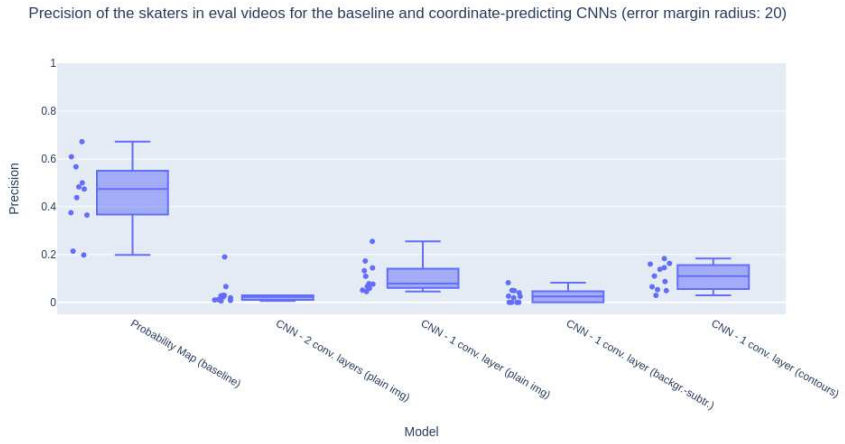

Figure 9: *The precision scores per model on the evaluation videos*

Figure 10 shows that the recall yields a different model ranking. This is because the CNNs predict skates in almost every frame, while the baseline is picky and often predicts no skates. This might be a reason for the relatively low precision scores from the CNNs. The CNN that is trained on the background subtracted images has also a worse recall than the other CNNs.

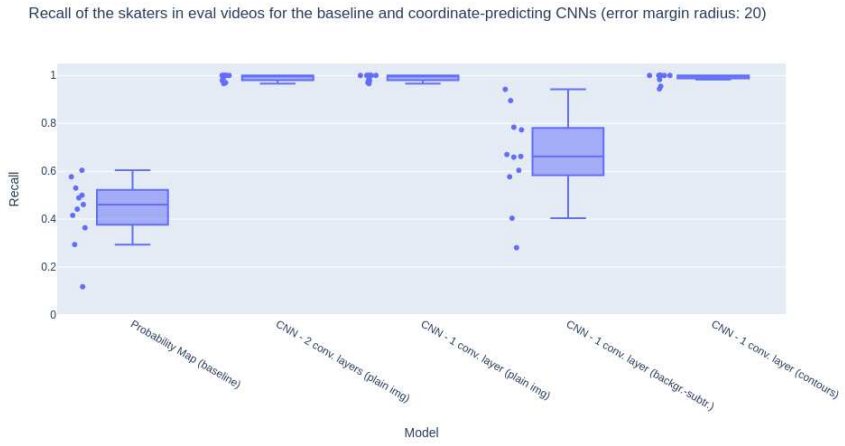

Figure 10: *The recall scores per model on the evaluation videos*

Those precision and recall scores result in the F1 scores that are shown in Figure 11. The ranking is the same as with the precision scores, although the difference between the baseline and the CNNs is mitigated by the recall scores.

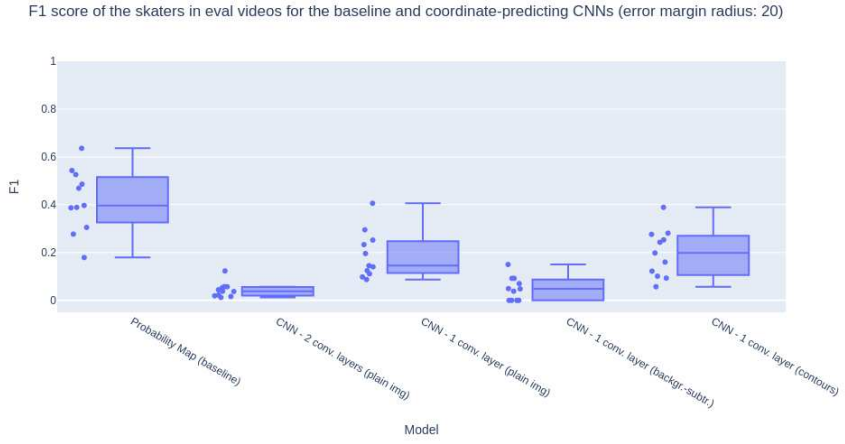

Figure 11: *The F1 scores per model on the evaluation videos*

4.3 Discussion

The results in section 4.2.4 show that the Probability Map baseline is the best performing model, it performs far better than all CNNs that we trained. Because CNNs typically need a lot of data in order to perform well, we hypothesized that the lack of training data is the main reason why the CNNs performed so badly.

We tested this by training each version of the CNN multiple times, each time using more data, until we reached the total amount of 35 annotated videos that we have. Figure 12 shows the results of this, showing the progress of the average F1 score over all 11 skaters in the same 5 evaluation videos that were used for the results showed in section 4.2.4. The figure shows that there is some progress for the CNNs as more videos are used, but this progress stagnates already as the full set of 35 videos is used for training the models.

Figure 12: *The progress in F1 score of all models as more videos are annotated and the dataset grows larger. The progress of the New CNN is better than the old one, but even the new CNN stagnates before it is trained on 35 videos.*

The figure also shows that there is no notable progress at all for the model that is trained on background subtracted images. Because of this and the fact that it performs far worse than the other CNNs, we decided to not train any more models on those background subtracted images as they are evidently not informative enough for the CNNs.

Because of the stagnation of the progress and the disappointing performance of the model, we reject the hypothesis that this CNN approach performs so badly because there is not enough data. In order to test this further and to make sure that this stagnation is not due to the random data separation into train and test set, we divided the videos into 5 folds with roughly the same amount of examples and trained each CNN with the new structure 5 times, each time with one fold as test data and the rest as training data. They are evaluated on the same videos as in figure 12. This yields figure 13, which shows a similar trend as figure 12 and even shows a drop in F1-score after 25 videos. This figure therefore confirms the thought that a lack of data is not the issue here.

Figure 13: *The progress in F1 score of the new CNNs over the amount of videos with a 5-fold cross validation. The figure shows a similar stagnation as without cross validation.*

This makes it even more remarkable that such a simple method as the Probability Map performs much better than a complex CNN. This suggests that although the Probability Map uses simple methods, its core approach of selecting lines and then using the line characteristics to classify whether it is no skate, a left skate or a right skate is a better one than processing the image through a CNN to get the raw coordinates. This idea is substantiated even further by the fact that the CNNs are forced to always predict skates, regardless of whether the real skates are easily detectable. This causes the recall for the CNNs to be high, but it might be a reason why their precision is so low. It would be better if the CNNs could, like the Probability Map, give some kind of score of how certain they are that a skate is detected. This way they can skip frames where they are not certain of their detected skates.

For those reasons we suggest another method, which uses the same approach as the Probability Map but also processes the images using a CNN. This combination of Probability Map and CNN is described, evaluated and discussed in section 5.

5 Experiment 2: CNN and Probability Map Combination

5.1 Method

As discussed in section 4, the approach of a coordinate-predicting CNN does not work for this task. That CNN takes an image of a skater as input and gives 8 numbers as output, which are the coordinates of the left and right skate points. Those 8 outputs that can each be any number between 0 and the image length/width and have to be exactly right, which makes the task very hard.

From the Probability Map baseline, we realized that the task can be done in an easier way. As the Probability Map finds all possible skate lines and then uses characteristics of those lines to determine whether a line is a skate line, we can use a similar approach for this Probability Map and CNN combination. Just like with the Probability Map, we first generate all possible skate lines for a skater image using Canny edge detection and Hough transform. But where the Probability Map classifier simply uses only characteristics of each line itself to estimate the probability of it being a skate line, a CNN can be complex enough to take both the image and the coordinates of a line as input and output two numbers: the predicted distance of the line to the left skate, and the predicted (Euclidean) distance to the right skate.

We did try to turn this distance outputs into probabilities of being each skate by inversing the distance to get output numbers between 0 and 1, but this did not work as the network predicted a probability of 0 for all inputs. Because of this, we decided to just use the predicted distances as outputs for the network. The lower the predicted distance from a line to the real skate is, the higher the probability should be that the line is in fact the skate.

This approach makes the task less complicated because the combination CNN only has two numbers as output, and because those numbers have to be less precise. In the coordinate-predicting CNN, the output numbers are coordinates for the skates, so they all had to be very precise for the prediction to be accurate. For this combination CNN, the only thing that matters is that good skate lines output a low distance and bad skate lines output a high distance.

Furthermore, the combination CNN has a built-in confidence mechanism. The coordinate-predicting CNN structure was obliged to always predict some coordinates for the skates, even if the skates were not clearly visible (for example if they were occluded). An added benefit for this approach is that a threshold for the distance can be set, so that the model does not predict any skates if it is not confident that it has found the right ones. This may increase the precision of the model, and the frames where no skate is predicted can later be predicted by interpolation.

The combination CNN has the following architecture: First the image and the line are processed separately. The images goes through one convolutional layer and a max pooling layer, while the skate passes through a fully connected layer. Then both are concatenated using the 'cat' function from Pytorch[5]. When concatenated, they go through two fully connected layers to get the two outputs. The first of those fully connected layers after concatenation has a dropout rate of 0.25 to prevent overfitting. The outputs of the model are the predicted distance from the input line to the left skate of the input image, and the predicted distance from the line to the right skate. This final architecture is established after systematically training models with many different architectures and watching how low the loss function went during training. This CNN uses a simple standard loss function: the MSE loss between the predicted output distances and the real distances.

In order for this CNN to work properly, a new dataset with combinations of images and lines has to be prepared. For each image, all possible skate lines are determined. In order to keep our dataset manageable for the computer that we are training the model on, we do not include every combination of image and possible line in the dataset. We select the possible lines that have a distance to a real skate of less than 5 for the dataset, and then we randomly pick the same number of lines from the other possible lines (that are farther than distance 5 from a real skate line). This limits the dataset and it also keeps the dataset balanced, with as many bad lines as there are good ones.
Then for each combination of an image and possible skate line, the two labels are calculated as the distance between the possible skate line and the annotated target left and right skate respectively.

After this new dataset is created, the model is trained and evaluated using the same settings, baseline and evaluation metrics as the coordinate-predicting CNN (which are described in section 4.2.2). For the combination CNN, we set the distance threshold to the same value as the error margin of the evaluation metrics (20). This means that the CNN will only predict skates in a skater image if it has found lines that are at a distance of at most 20 from the real lines, according to its predictions.

[5] `https://pytorch.org/docs/stable/generated/torch.cat.html`

The coordinate-predicting CNN is added as extra baseline. The results of the model are described in section 5.2.

5.2 Experimental results

The following box plots show the results for the Probability Map, the coordinate-predicting CNN and the combination CNN. Figure 14 shows that the combination CNN outperforms the coordinate-predicting CNN, and is close to the Probability Map baseline, although the baseline does still have higher scores overall.

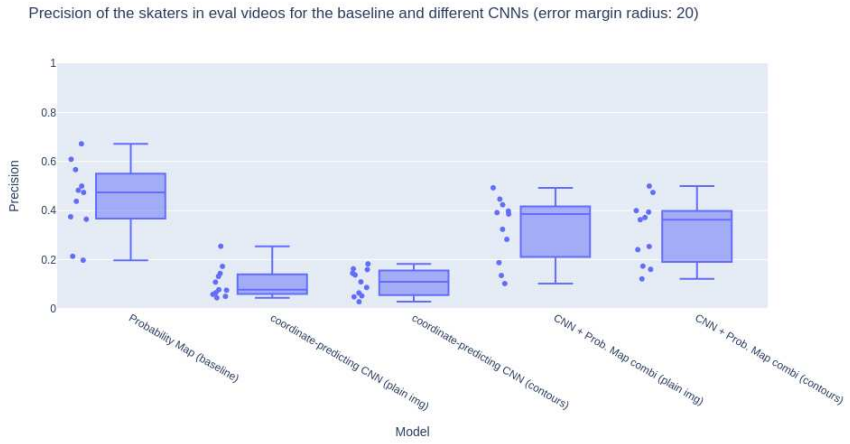

Figure 14: *The precision scores per model on the evaluation videos*

Figure 15 shows that although the combination CNN does have the possibility of not predicting a skate, its recall values are still very high. We did try evaluating it with a higher distance threshold, which did lower the recall but it also lowered the precision instead of raising it.

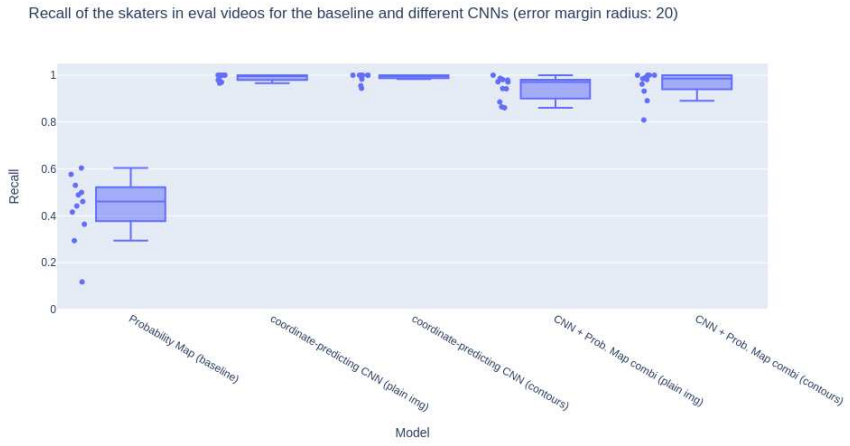

Figure 15: *The recall scores per model on the evaluation videos*

Those precision and recall scores result in the F1 scores that are shown in Figure 16. While the precision scores of the combination CNN were on par with the baseline, this figure shows that the high recall of the combination CNN causes it to overall surpass the baseline.

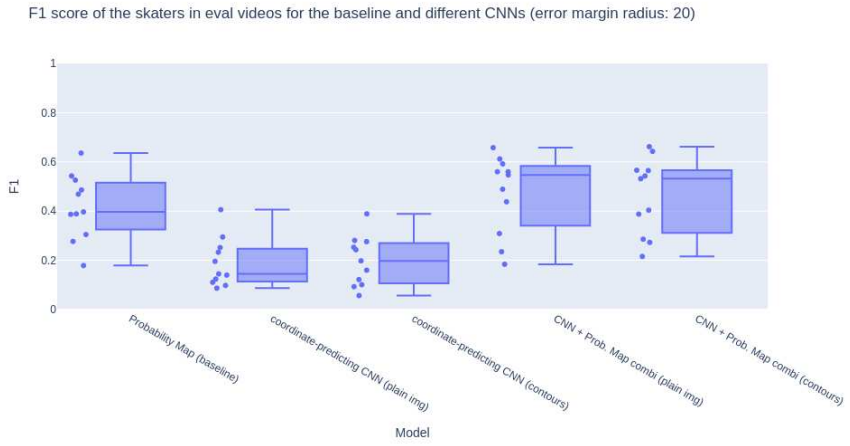

Figure 16: *The F1 scores per model on the evaluation videos*

For the combination CNN, adding the contours does not seem to have an impact as the scores for the CNNs with and without contours are roughly the same.

5.3 Discussion

The results in section 5.2 show that the combination CNN does surpass the Probability Map when it comes to F1 scores. However, this is partly because the combination CNN has very high recall scores.

Those high recall scores are remarkable, because the CNN is able to predict no skate at frames where it is unsure about the skates thanks to the distance threshold. Although the recall drops when the threshold is set to a lower value, the CNN still does not skip the difficult frames because the precision score also drops.

It is also remarkable that the precision scores of the combination CNN are still lower than those of the Probability Map, although the Probability Map is a simple method that only looks at a potential line characteristics to make a prediction and the CNN is more complex and analyzes the image as well as the potential line. So although the overall F1 score from the combination CNN is better than the overall F1 score from the Probability Map, it does not seem like the CNN does a better job at predicting the skates.

While it is weird that such a simple method such as the Probability Map can beat a complex method such as a CNN, there are some things that can explain this. As mentioned before, a CNN typically needs a lot of data. Although we have already found that the coordinate-predicting CNN does not need more data, this can still be the case for the combination CNN.

Besides that, the method of retrieving potential skate lines could also be an issue. For this model we used a simple Canny edge detector and a Hough transform, but other line detectors might yield better skate lines. After all, if no good skate lines are detected then the results are bad regardless of the competence of the CNN.

Furthermore, we need to keep in mind that we have only evaluated the models with a margin of error of 20. When using a skate detection model, we want each prediction to be within a far smaller margin of error. When we reduce the margin of error for the current models, the scores drop to (almost) 0. This indicates that we are unfortunately still far from reaching the goal of creating a skate tracker that can be used to analyze the trajectories and technique of skaters that are skating through the bend.

Although we were not able to experiment with predicting skates any further due to time constraints, we did work on the process after the skates are predicted: getting from the predicted skates to a trajectory for each skate and then to a plot of the speed of each skate across the video frames. This process is described in section 6.

6 From Predicted Skates to Trajectory

After predicting the skates, a goal of the project that this research builds on is to use the predicted skates to retrieve the trajectory for each skate. Although the models that we created during this research are not accurate enough to come close to the real trajectories of the skates from a skater, we can look at the process of getting from the predicted results to the trajectory for each skate and then to a plot of the speed for each skate over time.

Firstly, we can plot the real trajectories of the skaters by plotting the coordinates for each annotated skate line on a background frame of the video. For one skater, this is shown in figure 17, where the red dots represent the left skate and the green dots represent the right skate. The dots are the point in the middle of a skate line, because representing them as dots makes the trajectories more clear. The figure shows that there are some dots missing, especially at the beginning and the end of the video, and then the skates cross each other. This indicates that the skates could not be annotated properly in those parts of the video. This is the case because the blob detector is not able to properly detect the skater (at the beginning and end of the video) or the skates occlude each other (when they cross each other) and are thus not visible enough to be annotated.

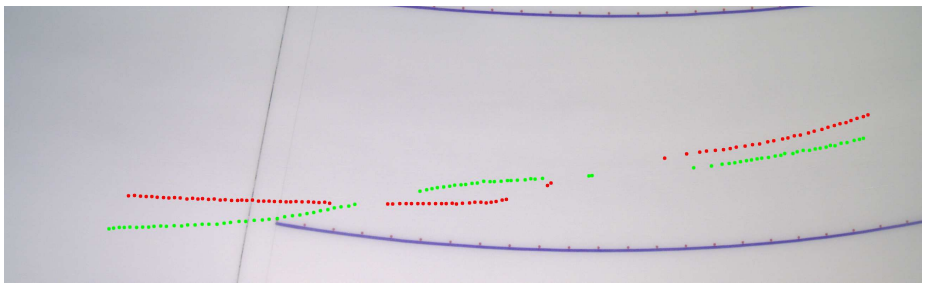

Figure 17: *The real trajectories for the left skate (red) and the right skate (green) of one skater*

Those gaps in the skate coordinates can be fixed by interpolating the line of coordinates. We use a standard interpolate function from Scipy for this[6], which yields the trajectory that is shown in figure 18.

[6] `https://docs.scipy.org/doc/scipy/reference/reference/interpolate.html#module-scipy.interpolate`

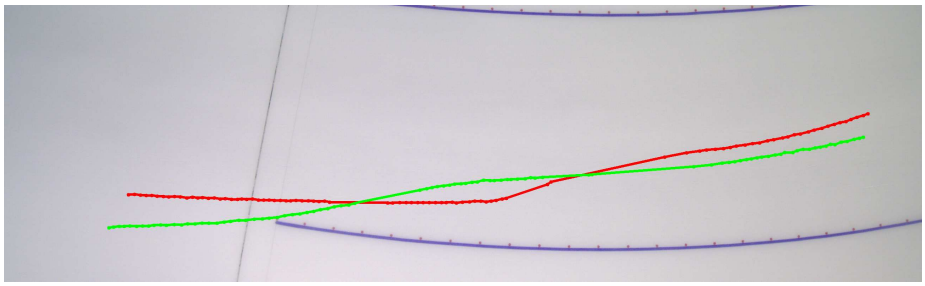

Figure 18: *The real interpolated trajectories for the left skate (red) and the right skate (green) of one skater*

When we have the whole trajectory for both skates, we can also plot the trajectories as predicted by our combination CNN. This is shown in figure 19. Here, the dark red points represent the predicted left skates and the dark green dots represent the predicted right skates.

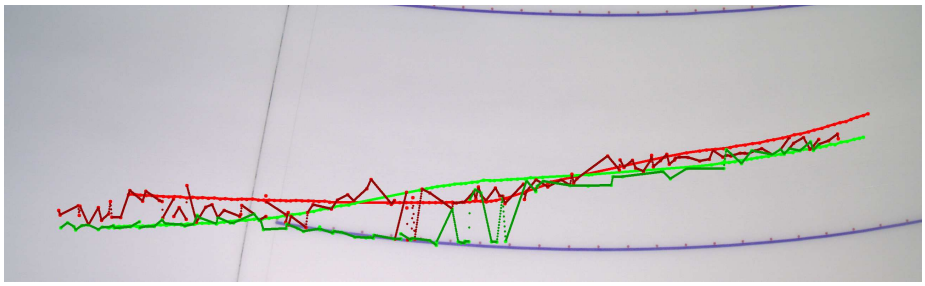

Figure 19: *The real (light) and predicted (dark) interpolated trajectories for the left skate (red) and the right skate (green) of one skater*

It is clearly visible that the model does not yield predictions that are good enough to come close to the real trajectories, because the predicted points are all over the place instead of forming a trajectory. However, it is remarkable that the predictions for the left skate are often exactly on the real line for the right skate and vice versa. This means that it is hard for the model to keep track of the identities of the left and right skate, which could be a huge factor in the bad evaluation scores of the model.

Another factor is that the blue line on the ice that indicates the edge of a skating lane is often predicted as a left or right skate. Finding a way to remove this line from the background for the model could also improve the evaluation scores.

After plotting the trajectories, we can use those to get the speed of the skates by calculating the distance between two subsequent skate points per frame. As the camera uses a calibration of 0.6075 cm per pixel and the videos are all 2200 ms long, we can calculate the speed of a skater between two frames in km/h, for both the real trajectory and the predicted trajectory. As those lines are not smooth at all, we smooth them using a 2nd order low pass Butterworth filter. The result for the skater with the trajectories in figures 17-19, is shown in figure 20. As expected, the predicted speed line are not even close to the real lines. When calculating the RMSE between two left lines (128.0) and between the two right lines (211.7), it is no surprise that both are high. This analysis of the trajectories and skater speeds will be more useful when the skate predictions are better.

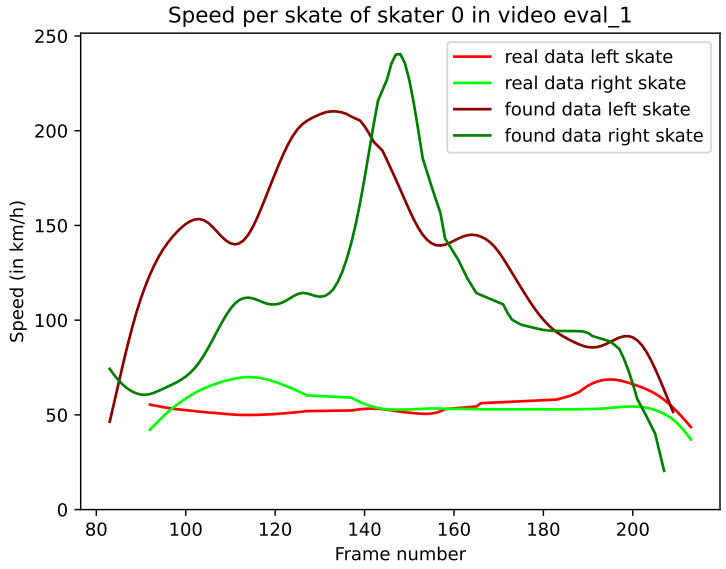

Figure 20: *The real (light) and predicted (dark) skater speed lines (in km/h) for the left skate (red) and the right skate (green) of one skater*

The workflow of getting from a skater video to a plot of the skate speed, is shown in figure 21.

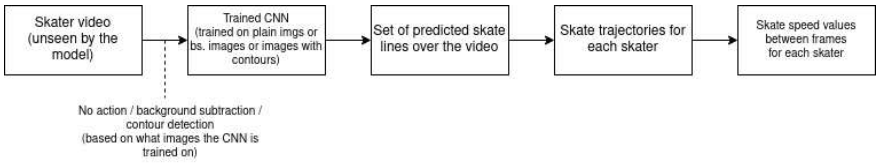

Figure 21: *The workflow from an original skater video to a plot of the skate speed*

7 Conclusions

In this research, we introduced two novel methods for detecting the skates for long track ice skaters. However, our experimental results showed that no model produces predictions that are accurate enough to actually analyze the trajectories of the skaters through the video frames. As our research question was "How can we track the trajectories of the skates of long track skaters in a video frame by finding the line that corresponds to a skate blade for each skate in each video frame?", we have to conclude that the research question remains unanswered.

However, we did contribute a lot to the process of eventually answering that research question. First of all, the first goal of this research (as mentioned in section 1.4) was to give an overview of the state-of-the-art methods for tracking objects and of what work has already been done on tracking ice skaters specifically. We did manage to do this and in doing so we learned a lot about the field of Computer Vision in general, as well as about the previous work on the combination of Computer Vision and sports (especially ice skating).

We also made a lot of progress on our second goal, which entails creating a novel method that attempts to track the skates of long track speed skaters, as accurately as possible. Although we did not manage to track skaters accurately, we did try several approaches of doing so, after which we learned that the best approach (so far) in solving the task is to first generate all possible skate lines and then predicting which one is the best skate line.

Despite this progress, the question remains why the models perform badly and thus why we were not able to answer our research question. First of all, the task is a lot harder than we thought beforehand. Because the field of Computer Vision is very advanced and a lot of tasks that seem very difficult can be solved by now, it makes it looks like this relatively 'simple' task of tracking skaters can be solved easily. However, we found that most of the already existing work is not available or not suitable to use for this task. Because of this, we had to build a model from scratch, which makes the task a lot harder than we thought it would be.

This task also requires a complex workflow in which every step could have some flaws that could make the model performance worse. The annotation of the data could introduce problems as a flaw in the Blob Detector could cause a skater to not even be detected in frames where it is visible and thus should be annotated. Another problem with the annotation is that it takes time, which limits the amount of data that we can use for our models. As we mentioned before, CNNs typically need a lot of data and this might cause the model to perform badly.

Furthermore, the simple potential line detection algorithm (Canny edge detection and Hough transform) that we use to create the combination dataset could also be an issue, because if that produces only bad lines then the model cannot be trained well.

Besides the problems with the workflow, there are also some things about the task itself and the data that make the task hard. As shown in section 6, the model often thinks that the right skate is the left skate and vice versa. Therefore we can conclude that keeping track of the identities of the skates is very hard for the model. This is to be expected, as the skates often cross and occlude each other. Furthermore, it was also shown that the model often thinks that the line that separates the skating lanes is actually a skate, or a skate is on that line. This is also a cause for bad performance.

Future research on this topic could build upon the model that we created, while it tries to remove or mitigate the aforementioned possible causes for the bad performance of the model. For example, they could look at different ways to track the skater bodies and to retrieve all possible skate lines in order to increase the quality of the dataset. Removing background noise such as the line that indicates the border between two lanes would also be a step that could help the model. They could also simply add more annotated data to see what that does to the model's performance.

Lastly, another recommendation for future research is to try to do what we could not do: gain access to existing work that is suitable for this task. We were not able to access the model by Liu [16] and the pose detection algorithms that we tried did not work, but if such a method that does work for this task is found and accessible, then that could also help a lot in solving this task.

References

[1] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, p. 103448, 2021.

[2] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2137–2155, Nov 2016.

[3] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2015 challenge results," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.

[4] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Bhat, A. Lukežič, A. Eldesokey, G. Fernandez Dominguez, A. Garcia-Martin, Iglesias-Arias, A. Alatan, A. Gonzalez-Garcia, A. Petrosino, A. Memarmoghadam, A. Vedaldi, and A. Muhič, "The sixth visual object tracking vot2018 challenge results," in *Computer Vision – ECCV 2018 Workshops*, pp. 3–53, Springer International Publishing, 2019.

[5] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J. Kämäräinen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, A. Eldesokey, J. Käpylä, G. Fernández, A. Gonzalez-Garcia, A. Memarmoghadam, and et al., "The seventh visual object tracking vot2019 challenge results," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2206–2241, 2019.

[6] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Čehovin Zajc, A. Lukežič, O. Drbohlav, L. He, Y. Zhang, S. Yan, J. Yang, G. Fernandez Dominguez, A. Hauptmann, A. Memarmoghadam, A. Garcia-Martin, A. Robinson, and Z. Ma, *The Eighth Visual Object Tracking VOT2020 Challenge Results*, pp. 547–601. 01 2020.

[7] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," *CoRR*, vol. abs/1606.09549, 2016.

[8] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, 2010.

[9] Z. Ma, L. Wang, H. Zhang, W. Lu, and J. Yin, "Rpt: Learning point set representation for siamese visual tracking," in *ECCV Workshops*, 2020.

[10] B. Yan, D. Wang, H. Lu, and X. Yang, "Alpha-refine: Boosting tracking performance by precise bounding box estimation," *CoRR*, vol. abs/2007.02024, 2020.

[11] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," 2020.

[13] G. Liu, X. Tang, H. Cheng, J. Huang, and J. Liu, "A novel approach for tracking high speed skaters in sports using a panning camera," *Pattern Recognition*, vol. 42, no. 11, pp. 2922 – 2935, 2009.

[14] Y. Wang, "A novel and effective short track speed skating tracking system," in *Utah State University: Theses and Dissertations (4620)*, 2012.

[15] H. Cheng, J. Shan, and Y. Wang, "Multiplayer tracking system for short track speed skating," *IET Computer Vision*, vol. 8, pp. 629–641, 12 2014.

[16] C. Liu, "A computer-aided training (cat) system for short track speed skating," in *Utah State University: All Graduate Theses and Dissertations (2188)*, 2014.

[17] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, pp. 28–31 Vol.2, 2004.

[18] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021.

[19] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[20] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," pp. 135–144, 2002.

[21] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, pp. 28–31 Vol.2, 2004.

[22] S. Suzuki, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[23] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "Rmpe: Regional multi-person pose estimation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2353–2362, 2017.