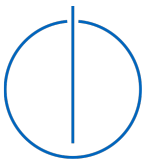
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Three-Dimensional Tracking of Fast Moving Objects for use in Augmented Reality Systems

Felix Neumeyer



DEPARTMENT OF INFORMATICS

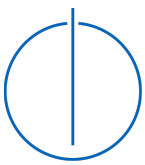
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Three-Dimensional Tracking of Fast Moving Objects for use in Augmented Reality Systems

Drei-Dimensionales Tracking von schnell bewegenden Objekten für Augmented Reality

Author:	Felix Neumeyer
Supervisor:	Prof. Gudrun Klinker, Ph.D.
Advisor:	Frieder Pankratz
Submission Date:	April 15, 2019



I confirm that this Bachelor's Thesis in Informatics: Games Engineering is my own work and I have documented all sources and materials used.

Munich, April 15, 2019 Felix Neumeyer

Acknowledgments

I would like to express my sincere gratitude to my advisor, Frieder Pankratz, for his continuous guidance, support and availability thoughout this thesis.

Abstract

This thesis proposes a detection and tracking algorithm for fast-moving spherical objects. These objects are often barely visible on camera frames due to their speed and appear as semi-transparent streaks. Balls are tracked in two synchronized camera images simultaneously to estimate their 3D position from 2D-correspondences. We demonstrate the the ability to find a ball's impact point on a surface, calculate its speed and predict its motion.

Diese Arbeit stellt einen Detection- und Tracking-Algorithmus für sich schnell bewegende kugelförmige Objekte vor. Solche Objekte sind auf Kamerabildern häufig kaum sichtbar und erscheinen als halb-transparente Striche. Bälle werden parallel in den Bildern zweier synchronisierter Kameras getrackt, um anhand dieser 2D-Positionen ihre 3D-Position zu bestimmen. Wir zeigen wie der Aufprallpunkt eines Balls auf einer Oberfläche bestimmt werden kann und schätzen seine Geschwindigkeit und zukünftige Flugbahn.

complete

Contents

List of Notations

Notation	Description
ε	The camera's exposure time.
FPS	The camera's frame rate.
C_I	The camera's intrinsic matrix.
C_E	The camera's extrinsic matrix.
C_d	The camera's distortion parameters.
A	A contour's area.
A_{max}	Predefined maximum contour area.
A_{min}	Predefined minimum contour area.
O	Object (FMO).
r	Radius of a FMO.
$\varnothing$	Diameter of a FMO.
ℓ	Length of a FMO.
$\mathbf{EA}(O)$	Expected Area of object O.
X_{cm}	Position (or center of mass) of a FMO (or contour).
n	Number of points.
X_i	The i'th point of a contour.
$\overrightarrow{d}$	Direction-vector of elongation of a contour (i.e. of a FMO).
$\overrightarrow{d}^{\perp}$	Vector perpendicular to $\overrightarrow{d}$.
$\overrightarrow{d_{ij}}$	Direction vector of a pair of contours, i.e. normalized difference of their positions.
$a \circ b$	Dot product of vectors a and b.
$\ell_{\overrightarrow{d}}(P)$	Projection length of points P on vector $\overrightarrow{d}$.
Δt_{ij}	Time difference (in seconds) between frames i and j.
v	Speed of a FMO in pixels per second.
v_{ij}	Estimated speed of a FMO in pixels per second from two frames i and j.
$v_{\min}$	The minimum speed a FMO needs to have to be detected.
θ	Angle between two vectors.

Abbreviations

Term	Definition
FPS	Frames Per Second
FOV	Field Of View
DOF	Depth Of Field
ART	Advanced Realtime Tracking
IR	Infrared
SRG	Spatial Relationship Graph
CL	Camera Left
CR	Camera Right
CB	Chessboard
CBC_i	Chessboard Corner i
BAT	Board ART-Target
CAT	Camera ART-Target
C	Camera
IP	Image Plane
MTAT	Measuring-Tool ART-Target
MTT	Measuring-Tool Tip

1 Introduction

1.1 Motivation

Tracking of objects with high velocity is common in sports such as tennis, football or soccer to help with umpiring and to provide additional information to viewers, such as the speed of a serve. A player and their trainer could also use tracking data to review certain matches and training sessions afterwards, find weaknesses and therefore train more efficiently. Commercial solutions exist for television and umpiring purposes, such as the ”Hawk-eye” system for tennis and many other sports: it is used to provide additional information and statistics to viewers or assist umpires in case it’s unclear whether a ball was in or not [1]. Since the table tennis world cup in 2017, a tracking solution for ping pong balls has been used.

Beside the obvious use cases for television and umpiring, such tracking can be used to directly augment sports. The Londoner entertainment brand Bounce created an interactive ping pong table that reacts to balls hitting its surface by showing visual effects and adding gameplay elements to the regular ping pong game [2]. Virtual targets that need to be hit by the ball can be displayed on the table.

These tracking-solutions usually require specialized cameras with high frames per second (FPS) since objects with high velocity, such as balls in sports, otherwise appear as semi-transparent streaks on camera footage and cannot be tracked using standard visual object tracking [3]. To track those objects specialized methods are required. While there has been extensive research on visual object tracking in general, there are only three publications available on tracking fast-moving objects that appear as semi-transparent streaks [3, 4, 5]. Research available focuses on tracking in 2D only. In our approach ping pong balls are tracked with two synchronized cameras at frame rates from 30 to 60 FPS, effectively processing 60 to 120 FPS for real-time usage on a single laptop.

1.2 Contributions

This thesis proposes an algorithm to detect and track fast-moving objects over a large range of speeds while standard tracking-methods fail to track fast-moving objects

completely. The initial detection requires the object to be moving fast, but objects can slow down to a halt after detection without losing track. The main contributions of this thesis are:

- A new algorithm for tracking fast-moving objects that estimates their trajectory and velocity in both 2D and 3D is introduced. It runs on standard hardware (i.e. a single modern laptop) and is written in C++.
- The first detailed public research on tracking such objects in 3D.
- Two cameras are used simultaneously to derive the objects position in 3D.
- Small dataset with reference 3D positions of ball-positions.
- The estimation of the bounce location of a fast-moving object enables the creation of AR games that make use of it as a way of interaction.

1.3 Thesis Structure

The next chapter (2) discusses the related work. The appearance of fast-moving objects on camera images is explained while also discussing the various camera parameters that influence it in chapter 3. The hardware used and the calibration process necessary for both tracking and validation is discussed in chapter 4. Chapter 5 explains the detection process and how some properties of fast-moving objects are computed, while chapter 6 talks about tracking them and combining the output of two 2D-trackers to calculate 3D position estimates. The tracking's accuracy is then evaluated in chapter 7. The last chapter summarizes the thesis and proposes ideas for improvement and future work.

2 Related Work

The Hawk-Eye system for sports broadcasting was first used to track tennis balls and provide information about the balls speed and trajectory. It is one of the best examples of tracking fast-moving objects with high precision both in 2D and 3D by combining multiple cameras from arbitrary viewing angles. Its major design considerations have been described by Owens, Harris, and Stennett [1]. The paper describes the first commercially used version of the Hawk-Eye system at the UK's Davis Cup 2003 [1]. No further research on later iterations of the Hawk-Eye system is publicly available.

Researchers from the Czech Technical University in Prague introduced the notion of a fast-moving object (FMO) – an object that moves more than its own size in a single camera frame's exposure time – and have since published three papers on the topic of tracking them in 2D [3, 4, 5]. A proof-of-concept algorithm was presented and its performance evaluated on a new annotated dataset by Rozumnyi, Kotera, Sroubek, et al. [3]. Their dataset consists of short image sequences containing FMOs from a number of different sports. Other benchmarks for visual object tracking do not contain any FMOs [3]. They succeeded in tracking FMOs in videos with low frame rate and unknown exposure time and extract properties such as trajectory, rotation angle and velocity. The proof-of-concept algorithm was refined for precision and extended to also track FMOs that slow down after detection by Rozumnyi [4]. It was further improved to run smoothly on mobile devices in C++ by Hrabalík [5].

This paper goes more into the direction of the Hawk-Eye system, as we want to track FMOs in 3D in a controlled environment with configurable cameras, such that we can optimize the tracking results. While the proposed method works on some of the samples from the FMO-dataset it was optimized for tracking ping pong balls at 60Hz.

Does this belong here?

3 Appearance of Fast-Moving Objects

FMOs appear as semi-transparent streaks on camera frames [3]. There are two main parameters responsible for how long the streak is: the exposure time, also known as shutter speed, and the speed of the FMO itself. A longer exposure introduces motion blur to camera images for any moving object. Other camera parameters can influence the overall look of an image, such as its brightness or depth of field, but not the appearance of moving objects exclusively. These parameters and their direct or indirect effects for FMO-tracking are discussed below. In order to get a properly exposed image, i.e. not too bright or too dark, the exposure time can be dependent on these parameters though.

Frame Rate The frequency with which images are captured by a video camera is called *frames per second (FPS)*. By increasing the FPS an object can be tracked more precisely, since more data-points are available and the distance between the objects' positions in consecutive frames is smaller. It is generally beneficial to have a camera run at a high frame rate for tracking purposes. For example, the infrared tracking-cameras we use for validation run at a maximum of 300 FPS. In fact, even FMOs can be tracked with standard-tracking techniques if a high-speed camera is used, as the FMO will only move slightly in consecutive frames. Most standard cameras, however, will only run between 24 and 60 FPS.

Exposure The camera's exposure ε is the time the camera sensor receives light during image formation of a frame. It is limited by the frame rate and can only be smaller than or equal to $\frac{1}{FPS}$. The longer the exposure, the longer is the streak of a FMO on an image. By reducing it to a very small value (e.g. $\varepsilon \leq \frac{1}{1000}$ seconds) even FMOs can be seen with close to no motion-blur. However, under normal circumstances with standard cameras, such small ε values are only possible in direct sunlight, as the image's brightness depends on ε. For detection, long exposures are advantageous as FMOs are easier to distinguish from other (slow) moving objects or noise when they exhibit clear motion-blur.

Speed of FMO The faster a FMO, the longer is its streak on an image, since it is moving over a larger distance during the exposure time. If exposure is known, a FMO's

speed can be inferred from the length of its streak (see Section 5.2.3).

Aperture The camera's aperture is both a property of the lens and a physical component inside the lens. It's the opening inside a lens that lets light through to the sensor. The size of the opening can be changed and therefore, it also affects the image brightness, as more light can hit the sensor when the aperture is opened. However, it does not only affect image brightness, but also the depth-range around the focus-plane in which it is sharp, called the *depth of field (DOF)*. When the aperture is opened (i.e. it lets more light in) the DOF is shallow. For tracking ping pong the DOF should at least cover the whole table, thus the aperture cannot be opened completely. Under dark lighting conditions you want the aperture to be as widely opened as possible without blurring the tracking-space.

Gain The gain setting of a camera is the sensibility of the sensor to light. It is often called ISO. The higher the gain, the brighter, but also the noisier the image. As noise can be disadvantageous for tracking, gain should usually not be set too high if it is possible to properly expose the image by changing exposure time and aperture. However it makes it possible to record videos in fairly dark environments when exposure time and aperture are already at their limits.

Lighting While lighting cannot always be controlled, it can affect what exposure, gain or aperture settings are required. If the environment is dark, the camera may need to be set to a long (the maximum being $\frac{1}{FPS}$) exposure, resulting in longer streaks and brighter images. If lighting is highly uneven, the contrast between bright and dark regions in the image might be too high, resulting in bright regions being completely white, or dark regions being completely black. This depends on the camera model used, as the dynamic range (i.e. the ratio between the brightest and darkest pixel that are still distinguishable from the rest of the image) is a property of the image sensor.

Add images to show observations, possibly in one large figure

4 System Setup

This chapter introduces the hardware for tracking and validation. The calibration process of the stereo camera setup is explained.

4.1 Reference Tracking System (ART)

To validate our tracking results and for registration (see Section 4.3) an optical infrared 6dof (degrees of freedom) tracking camera system by ART [6] was used. Their proprietary software Dtrack2 combined with ARTTRACK5 cameras is able to track targets at up to 300Hz and allows for submillimeter-accurate positioning. The cameras have multiple IR emitting LEDs surrounding the lens to illuminate the scene and allow for tracking of the markers and targets described below.

Markers A (passive) marker, in the context of the IR tracking system, is a single retro-reflective (i.e. incoming light is mostly reflected in the direction where it came from) ball. It can be seen as a bright spot by the cameras, as it reflects the IR light emitted by the cameras back at them, while the rest of the scene is significantly less bright, since most normal objects have diffuse reflections. While active markers exist, that emit IR light themselves, we only used passive markers. In our recordings we throw a single large marker instead of a ping pong ball as it can be tracked by both the ART system and our tracking and has about the same size as a ping pong ball.

Targets A target consists of a minimum of four markers with fixed positions in relation to each other. The target can be recognized by the system by using the distances between markers as a key [6]. The pose (position and rotation) of a target, like the one mounted above the cameras in figure 4.1, is tracked once detected by the ART system. By attaching a target on an object, the pose of the latter can be determined relative to the target, after registration (see Section 4.3).

4.2 Cameras

RGB-cameras running at a maximum of 120 FPS (max. 60 FPS for real-time use) with a resolution of 1280 × 1024 and wide-angle lenses were used for tracking. The exposure time, which is required to be known in the detection algorithm (see Chapter 5), can be set manually. To determine the 3D-position of objects two calibrated and synchronized cameras are required. We mount them horizontally next to each other in a stereo-camera setup, but any relative orientation is supported after camera registration.

Figure 4.1: Two cameras mounted on a frame with an infrared (IR) tracking target above. A cable for hardware-synchronization connects both cameras. Tape to prevent IR-reflections. Background: one of four IR tracking cameras for validation and calibration.

4.2.1 Single Camera Calibration

Images of today's cameras are often significantly distorted. We use wide-angle lenses that make it possible to capture a large *field of view (FOV)* and are known for their strong

distortion. Camera calibration is a procedure to determine the camera's properties, i.e. their intrinsic matrix C_I, extrinsic matrix C_E, as well as distortion parameters C_d. After camera calibration, images taken by the camera can be undistorted. Lines that are straight in reality should not be bent on images afterwards (see Figure 4.2). Calibration also enables us to determine the relation between the image units (pixels) and the real world units (e.g. meters). Multiple images are taken of a calibration pattern (circular grid). It's important that the whole image area has been covered by the calibration pattern. Images should be taken from different angles, positions and distances in order to get accurate results. We use the OpenCV calibration algorithm based on Zhang [7] and Y. Bouguet [8]. Both cameras need to be calibrated separately.

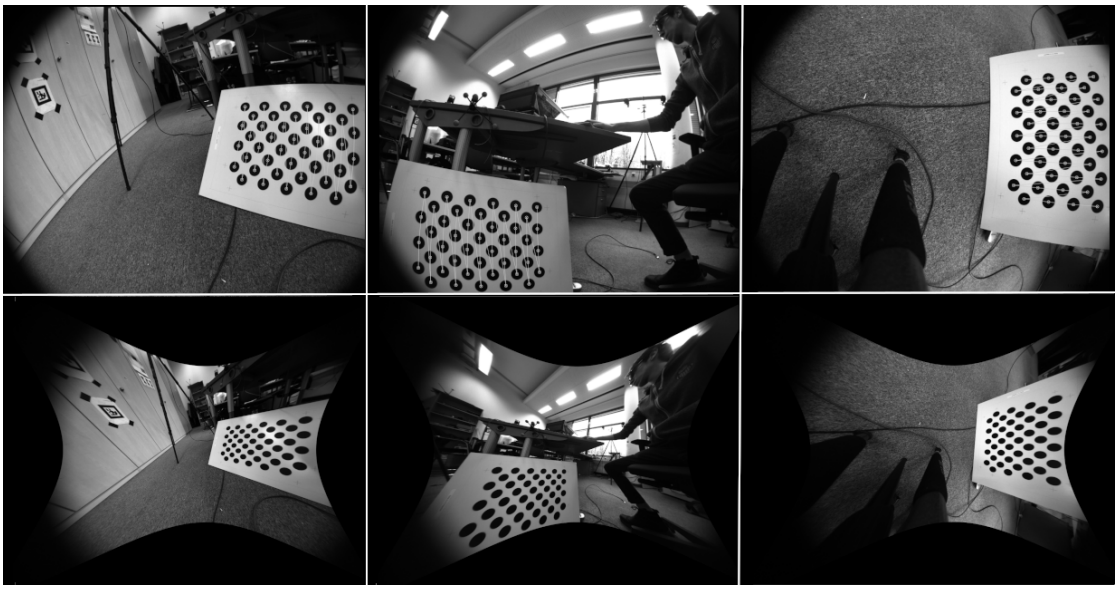


Figure 4.2: Sample images from the calibration process of a single camera. Top row: Distorted calibration pattern with circle detection, lines are bent. Bottom row: Corresponding undistorted images with straight lines, before cropping them to only show relevant regions.

4.2.2 Stereo Camera Setup

A standard stereo-calibration, as available in OpenCV, requires cameras to be mounted horizontally next to each other. A different approach has to be used to support any relative transformation between cameras. In order to improve the depth-accuracy of a standard stereo-setup, it was planned to have cameras located on the side and above of the tracking space. Especially in the context of a well defined tracking space as

a ping pong table, as described in this thesis, this kind of setup could be beneficial. Due to time and equipment restrictions (i.e. the cameras being shared among multiple research teams) only a manual calibration with horizontally aligned cameras was tested. A standard stereo-calibration would have probably been the better choice in this setup.

4.2.3 Hardware-Synchronization

Synchronization between multiple cameras is very beneficial for tracking objects in 3D. If there is no synchronization, images taken with two cameras are usually not from the same point in time, making it necessary to interpolate values obtained from one of the cameras. This would not only result in less accurate, but also delayed calculations. For hardware-synchronization the cameras are connected via cables. The master camera sends a trigger-signal to the slave camera so that it takes an image simultaneously. Therefore, 2D positions calculated in each cameras' synchronized frames can be used directly to calculate 3D positions, without interpolation.

4.2.4 Choice of Frame Rate and Exposure

The elongation of a FMO on images is helpful for detecting them and for estimating their speed and direction and is assumed by the detection algorithm. Short exposure times result in less elongated FMOs and makes it harder to differentiate between them and other image motions or noise. With small exposure settings the detector produces more false positives. Since the exposure cannot exceed $\frac{1}{FPS}$ we don't want to set the frame rate too high for FMOs to keep a certain amount of motion-blur. On the other hand a higher frame rate is generally beneficial for tracking as more data is available and objects can be tracked with higher granularity. We have come to the conclusion that a combination of a frame rate of 60 FPS and the corresponding maximum exposure of $\frac{1}{60}$s is beneficial for both tracking and detection. At these settings FMOs still have their characteristic shape and give information about their direction and speed in a single frame. A frame rate of 60 FPS is also high enough to track objects at fairly high speeds while not exceeding the processing power on a single laptop. Note that 120 FPS need to be processed as two cameras are used simultaneously.

4.3 Calibration (ART $\leftrightarrow$ Camera)

The process of finding a rigid transformation between two objects' poses is called registration. This section explains all used tools, steps and registrations required to enable the transformation of poses from the ART coordinate system to the left camera's coordinate system (CL) and vice versa. This is required to calibrate the stereo-camera

setup and later to evaluate our results. We use *Spatial Relationship Graphs (SRG)* and patterns as described by Pustka, Huber, Bauer, and Klinker in [9] to explain the relationships and operations between objects in the tracking system.

4.3.1 Measurement Tool

The Measurement Tool (MeaTool) is an ART-target that resembles a large pencil with a pointy tip (see Figure 4.3). After tool-calibration, the tip's 3D-position can be measured.

Figure 4.3: The MeaTool ART-target to measure positions in 3D at the tip.

4.3.2 Marker Board

We use a marker board (see Figure 4.4) as the reference point between the two coordinate systems (the ART system and the stereo-camera system). The board has been printed with a large chessboard pattern and four smaller square-markers in order to be tracked by a camera. To track the board from ART it has been made into a custom target consisting of four ART-markers that are screwed into the board.

The square-markers are similar to QR-Codes, which are widely known today. They are just a binary-encoded matrix, that is interpreted as an identifier, enclosed by a black border. In augmented reality, markers are used not only to be identified. Their pose can be determined in the camera coordinate system, if their real world size is known. If identification is irrelevant, a simple chessboard-pattern of known size can be used. Both, the square-markers and chessboard patterns are tracked using OpenCV [10]. After initial accuracy issues with square-markers we ultimately used the chessboard pattern. Due to its larger size and feature count, its pose can be measured more accurately.

Registration: Board ART-Target to Chessboard-Pattern

The transformation between the origin of the local chessboard pattern's coordinate system (CB) and the tracked board's ART target (BAT) is the missing link between the ART and camera coordinate systems and is marked red in Figure 4.5. The MeaTool

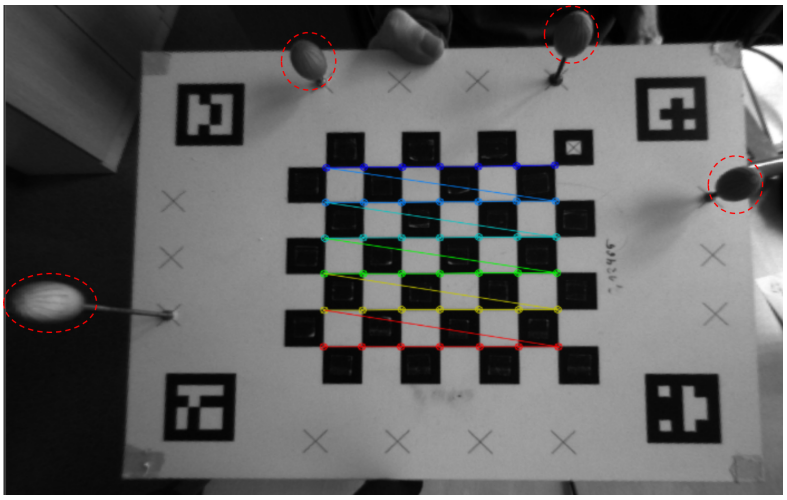


Figure 4.4: Chessboard detection using OpenCV. Detected corners are highlighted and a colored line indicates the order in which corners are handled, starting at the upper right corner (blue). The first corner is the origin of the boards local coordinate system. Four ART markers (circled red) form a custom ART target (see 4.1).

(MTAT) (see Section 4.3.1) enables us to measure the positions of the chessboard's corners (CBC_i) by touching every corner with the tools tip (MTT). This is done in a specific order, starting from the origin of the chessboard's coordinate system, into the x-direction, and finishing one row after the other in y-direction (see Figure 4.4). The local corner positions of the pattern are given by its real world size and number of rows and columns. We use 3D-3D pose estimation, which is also known as the absolute orientation problem, to compute rotation and translation BAT $\rightarrow$ CB between the ART and the chessboard's local coordinate system over time [9]. Only a minimum of three measurements from corresponding pairs of BAT $\rightarrow$ MTT and CB $\rightarrow$ MTT are required for an estimation, but all corner points should be measured. Before the pose estimation BAT $\rightarrow$ MTT is calculated by inverting ART $\rightarrow$ BAT and concatenating the result with ART $\rightarrow$ MTAT and concatenating this (BAT $\rightarrow$ MTAT) with MTAT $\rightarrow$ MTT again.

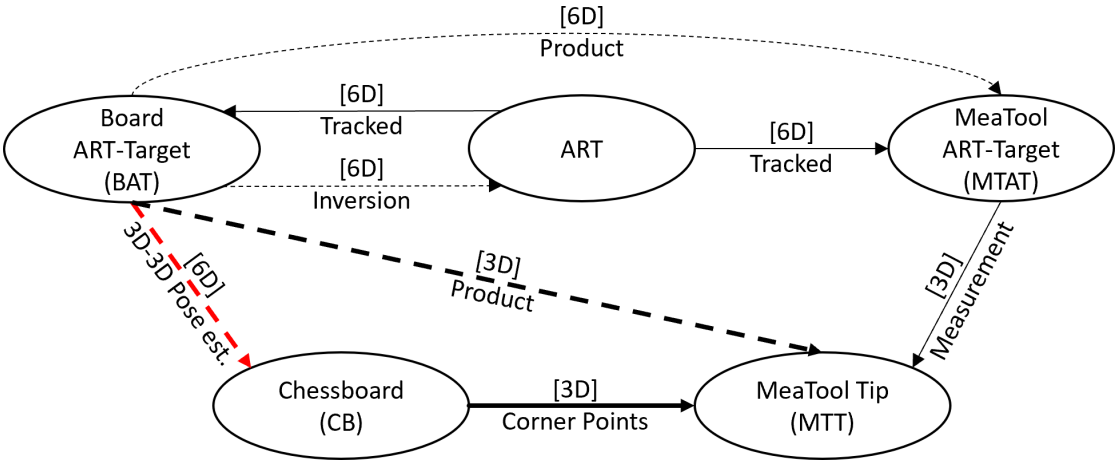


Figure 4.5: SRG of marker board ART-target to chessboard pattern registration.

4.3.3 Camera ART-Target to Camera

We want to find the transformation from the camera's ART-target (CAT) to the camera coordinate space (C). To do so, the marker board is tracked by the camera and ART simultaneously. The desired pose CAT $\rightarrow$ C can again be calculated with a 3D-3D pose estimation (see Figure 4.6). Before this is possible, we need to first find the pose CAT $\rightarrow$ CB by inversion of ART $\rightarrow$ CAT and two concatenations (CAT $\rightarrow$ ART $\rightarrow$ BAT and CAT $\rightarrow$ BAT $\rightarrow$ CB). Secondly, the chessboards pose C $\rightarrow$ CB is estimated using 2D-3D pose estimation. This procedure is done for both left and right cameras separately.

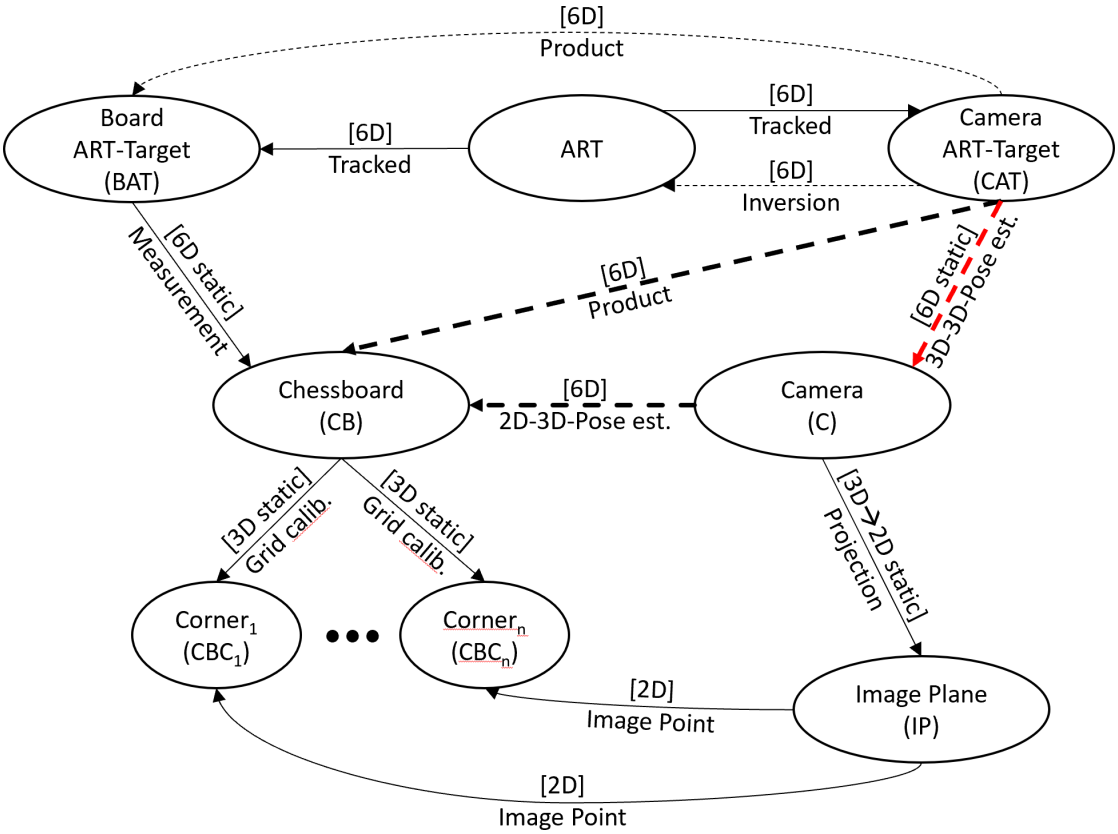


Figure 4.6: SRG of camera ART-target to camera registration. The resulting transformation (red) is obtained by 3D-3D pose estimation.The pose C → CB in camera coordinate space is obtained by a 2D-3D pose estimation

4.3.4 Left to Right Camera

The left and right cameras need to be registered, as it is required for the 3D-estimation from 2D point correspondences of a tracked FMO. After registration of both left camera (CL) and right camera (CR) to the camera ART-target (CAT), the transformation CL → CR can be calculated easily as visualized in Figure 4.7. Inversion of CAT → CL results in CL → CAT. By concatenation of CL → CAT and CAT → CR we find the desired transformation CL → CR.

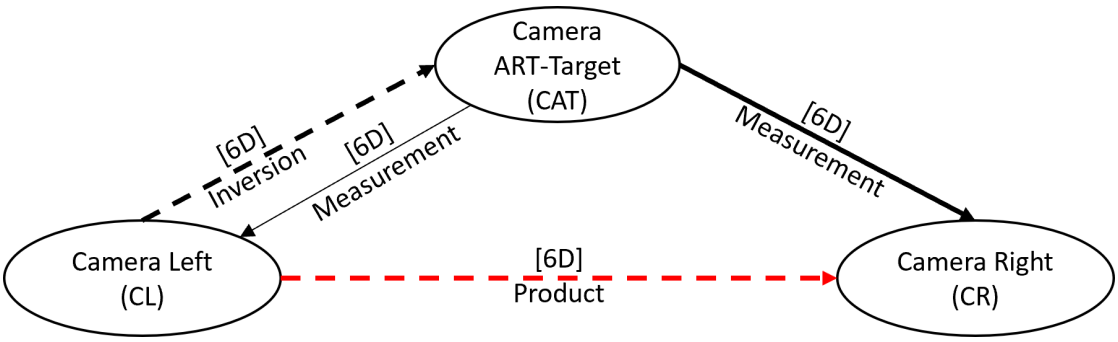


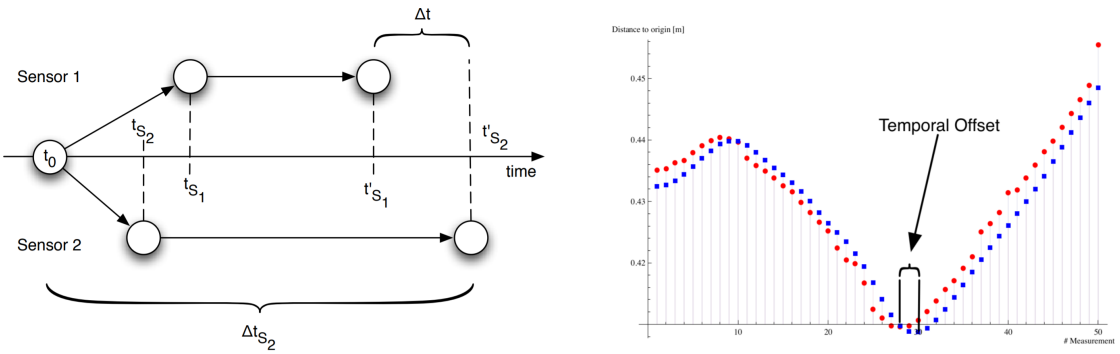
Figure 4.7: SRG of left to right camera registration.

4.4 Time Delay Estimation

In order to be able to compare tracking data from the ART and the stereo-camera system, we need to know the relative time delay between their outputs, so that the timestamps of one system can be adjusted and both systems refer to a common time base [11]. Figure 4.8 shows the relevant points in time:

> Considering an event happening in the real world at time t_0, two sensors S_1 and S_2 sense this event as an analog physical input and convert it into digital representations at times t_{S_1} and t_{S_2}. Further delay will be caused by the various communication stacks of the operating system or by network transport of the measurement data. We assume the measurements to arrive at the tracking framework at times t'_{S_1}, t'_{S_2} and to be tagged with according timestamps at that point in time. As soon as a reliable timestamp is attached, all further processing delays can be managed by the tracking framework. For sensor fusion it is only necessary that all sensors are temporally aligned relative to each other; the offset to the unknown true point in time t0 is not relevant. To align the sensor data, it is sufficient to determine the temporal offset $\Delta t = t'_{S_1} - t'_{S_2}$.
>
> Huber, Schlegel, and Klinker [11]

We track the marker-board from both ART and the camera. The board is then quickly moved back and forth in front of the camera to perform a time delay estimation based on the board's motion as visualized in Figure 4.8 (b).



(a) (b)

Figure 4.8: (a) Schematic visualization of the relevant points in time; (b) Temporal offset in sensor data from two sensors. Taken from *Temporal Calibration in Multisensor Tracking Setups* by Huber, Schlegel, and Klinker [11].

5 Detection

Finding a previously unrecognized object in an image is the task of a detection algorithm. Recognition is the process of identifying (classifying) an object. Usually, a detection algorithm is looking for a number of previously learned objects, like animals, human faces or the tip of a needle. In case of FMOs we want to detect streaks that move in the direction of their elongation, at a speed that can be explained with their length.

5.1 Preprocessing

In the preprocessing stage moving regions are found in an image-sequence and further filtered to extract objects that are possibly a FMO while also removing noise.

5.1.1 Foreground/Background Segmentation

Moving objects can be observed in temporal difference images [3]. This is because of the color-difference in the images at the position of moving objects. Static objects on the other hand don't change in color between camera frames, except for when the scene's illumination changes, or due to shadows. It is possible to determine FMOs using simple difference images of consecutive frames, but this only works for objects that move more than their own size from one frame to the next. If objects overlap in two consecutive frames and their appearance is uniform, the overlapping region may be removed from the difference images. By modeling the background throughout a longer sequence of frames and applying a background subtraction objects that move less than their own size in two consecutive frames will be visible in the resulting foreground mask. We use the Gaussian Mixture-based background subtraction algorithm in OpenCV, which does not only help find moving objects, but also handles shadows and changes in illumination and is based on two papers by Zivkovic [12, 13, 10]. Depending on lighting conditions and camera properties like ISO, aperture and sensor size there may be noise visible in the foreground mask which can be removed using an opening operation [10]. Contours are then extracted from the foreground mask. Objects that remain at the same position for too long are lost and can only be tracked again, if they move fast again, due to the nature of the background-subtraction used. The algorithm allows us

to chose a history length, i.e. how fast the background model changes, or how many previous frames should affect it. We limit the history to 120 instead of the default value of 500 images to make the background model update faster.

5.1.2 Foreground Filtering

The foreground mask obtained through background subtraction contains any moving object, not just FMOs. E.g. in a video of ping pong the two players and their rackets. While these can be moving fast, we are only interested in objects that can be explained by the properties of a FMO. A contour is considered a candidate for detection if all of the following properties can be confirmed.

Minimum and Maximum Area Very small contours can be the result of image noise and need to be ignored. While the opening operation on the foreground mask already removed most of the tiny noise, there may be some left. We use the pre-defined limits A_{max} and A_{min} to check whether a contour's area A is too large or too small. Therefore the inequality

$$A_{min} \leq A \leq A_{max} \tag{5.1}$$

must hold. If the minimum/maximum size of the object to track is known in advance this can be used to further reduce the number of candidates.

Area Model A FMO's contour can be assumed to be a rectangle enclosed by two half-circles. We can therefore calculate the expected area the streak of an object O should have considering its radius r, diameter $\varnothing$ and length ℓ [4]:

$$\mathbf{EA}(O) = \varnothing * \ell - (d^2 - \pi * r^2) \tag{5.2}$$

5.1.3 Pseudocode

This is the preprocessing process as explained in the sections above. It returns the candidate contours that might be a FMO.

Algorithm 1 Preprocessing

```
1: function PREPROCESS(I)                                  ▷ Image
2:     FG ← backgroundSubtraction(I)                       ▷ see 5.1.1
3:     FG ← removeNoise(FG)                                ▷ By morphological opening
4:     C_t ← findContours(FG)
5:     C_t ← filterContours(C_t)                           ▷ see 5.1.2
6:     return C_t
7: end function
```

5.2 Algorithm

The detector algorithm's inputs are the current and previous frames as well as the camera's exposure time. It requires a FMO to be visible in at least two consecutive frames to be detected. It then tests pairs of contours from the current and previous frames, i.e. elements of the Cartesian product of the two sets of candidates from each frame. Several checks are performed on each pair of candidates. If one of the checks fails, the pair is rejected. Only pairs that pass all checks are considered a FMO and added to the tracked objects.

5.2.1 Position of Contour

Since a FMO is depicted as a contour in the image, its position could be anywhere within that contour. However we want to have a consistent way of determining a single position for the whole contour in a frame. Therefore, we compute the position X_{cm} of a contour as the center of mass, which is the average of all contour points' positions:

$$X_{cm} = \frac{\sum_i X_i}{n} \tag{5.3}$$

where n is the number of points and X_i is the position of a single point of the contour.

5.2.2 Radius and Length Estimation

Diameter $\varnothing$, radius $r = \varnothing/2$ and length ℓ of a FMO's streak are main properties to consider when detecting them. Since FMOs appear as streaks on camera frames we can assume that the smaller side is the diameter of the object, while its length indicates its speed in relation to its diameter. There are various ways to obtain these properties that differ in their computational expensiveness and accuracy which we discuss in the upcoming sections.

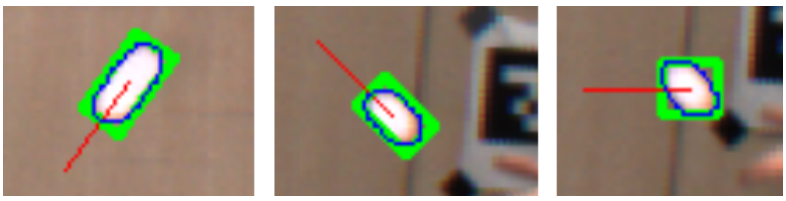


Figure 5.1: Length and radius estimation with minimum area rectangle. Rotated rectangle (green), direction from rectangle rotation (red) and FMO contour (blue). Left and middle images show a successful estimation. Estimation in the right image is incorrect due little elongation of FMO, even though slight elongation is visible.

Distance Transform

An accurate estimate of r (and of an objects trajectory) can be found by calculating the distance transform, which is a relatively computationally expensive operation [3]. Due to the real-time constraint with two cameras at up to 60 FPS per camera we need to use a different, less expensive method.

Minimum Area Rectangle

We find the minimum area rectangle containing the streak to obtain ℓ, $\varnothing$, r and the FMOs direction $\overrightarrow{d}$ using OpenCV [10]. While this method is very fast, it may fail to produce acceptable results if the object doesn't show a significant amount of blur (see Figure 5.1). If the ratio $\frac{\ell}{\varnothing}$ between length and diameter is close to 1, the streak does not show a clear direction and the minimum area rectangle might not be rotated so that its major axis corresponds to the object's length. However, for the initial detection of a FMO, which is supposed to have blur, it is sufficient.

Motion-Dependent Projection

This method requires contours from two consecutive frames i and j. It projects all contours points onto the (normalized) direction-vector

$$\overrightarrow{d_{ij}} = X^j_{cm} - X^i_{cm} \tag{5.4}$$

between two contours' positions X^i_{cm} and X^j_{cm}.

Projection of a vector a onto another vector b is defined as the dot product $a \circ b = s$, where s is a scalar value. s can be considered the position of the point a on b, or a's length in the direction of b. By finding the minimum and maximum values of all

contour point projections on $\overrightarrow{d_{ij}}$ we can calculate the length of an object's contour c along $\overrightarrow{d_{ij}}$:

$$\ell_{\overrightarrow{d_{ij}}}(c) = max(\{p \circ \frac{\overrightarrow{d_{ij}}}{\left\|\overrightarrow{d_{ij}}\right\|} | p \in c\}) - min(\{p \circ \frac{\overrightarrow{d_{ij}}}{\left\|\overrightarrow{d_{ij}}\right\|} | p \in c\}) \tag{5.5}$$

Assuming the object has nearly the same velocity and direction in both frames, the object should have about the same length along $\overrightarrow{d_{ij}}$ in i and j. Their diameters (or radii) can also be estimated analogously by projecting onto a direction-vector $\overrightarrow{d_{ij}}^{\perp}$ perpendicular to $\overrightarrow{d_{ij}}$. $\overrightarrow{d_{ij}}^{\perp}$ can be obtained by rotating $\overrightarrow{d_{ij}}$ counterclockwise by 90°:

$$\overrightarrow{d_{ij}}^{\perp} = \begin{pmatrix} -\overrightarrow{d_{ijy}} \\ \overrightarrow{d_{ijx}} \end{pmatrix} \tag{5.6}$$

$$\varnothing_{\overrightarrow{d_{ij}}}(c) = max(\{p \circ \frac{\overrightarrow{d_{ij}}^{\perp}}{\left\|\overrightarrow{d_{ij}}^{\perp}\right\|} | p \in c\}) - min(\{p \circ \frac{\overrightarrow{d_{ij}}^{\perp}}{\left\|\overrightarrow{d_{ij}}^{\perp}\right\|} | p \in c\}) \tag{5.7}$$

Note that the lengths and radii are obtained under the assumption that the FMO is moving in the same direction in both frames. A ball's trajectory is actually a parabola, so the radius will be overestimated and the length underestimated by using this method.

5.2.3 Speed Estimation

The speed of an object can be estimated either from a single or from two consecutive frames as described below.

Streak Length

The length of a FMO indicates its speed along the camera plane. We can therefore estimate the speed in pixels per second as

$$v = \frac{\ell - \varnothing}{\varepsilon} \tag{5.8}$$

and in diameters per second as

$$v_{\mathrm{d/s}} = \frac{v}{\varnothing}. \tag{5.9}$$

Therefore, if the real world diameter of the object is known, the objects real speed on the camera plane could be estimated from just one camera frame [3].

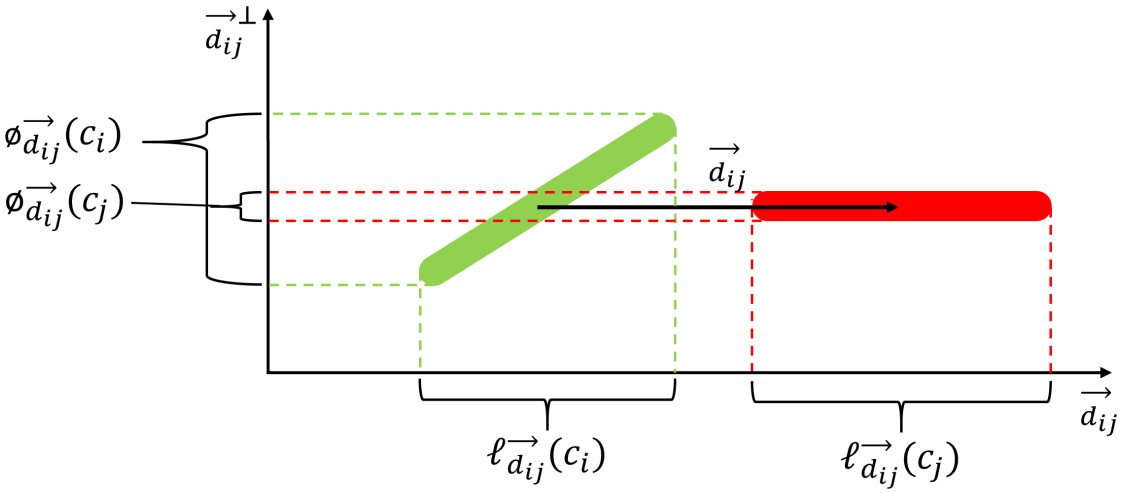


Figure 5.2: Motion-dependent projection along $\overrightarrow{d_{ij}}$. Two contours (green [i] and red [j]) that have the same length but different orientations are projected onto $\overrightarrow{d_{ij}}$. Due to the large deviation in projection lengths and diameters the pair is rejected. If the green contour had the same orientation as the red one, the check would be successful.

Relative Motion

The speed per second can also be derived from the Euclidean norm of $\overrightarrow{d_{ij}}$ (5.4) and the time-difference Δt_{ij} between frames i and j:

$$v_{ij} = \frac{\left\|\overrightarrow{d_{ij}}\right\|}{\Delta t_{ij}} \tag{5.10}$$

Calculating the speed from relative motion it is more accurate than estimating it from just a single streak length and should be preferred.

5.2.4 Contour-Pair Checks

Every check is performed on a pair of contours (c_i, c_j), where c_j is the more recent contour obtained from frame j. All checks need to be satisfied for a FMO to be detected.

Diameter- and Length-Projections

The motion-dependent projections (5.2.2) are compared. If the diameters or lengths obtained differ too much, the pair is rejected. Given the parameters for maximum length

and diameter deviations $\Delta\ell_{max} = 0.3$ and $\Delta\varnothing_{max} = 0.25$, the following inequalities must be satisfied:

$$\left| \frac{\ell_{\overrightarrow{d_{ij}}}(c_j)}{\ell_{\overrightarrow{d_{ij}}}(c_i)} - 1 \right| \leq \Delta\ell_{max} \tag{5.11}$$

$$\left| \frac{\varnothing_{\overrightarrow{d_{ij}}}(c_j)}{\varnothing_{\overrightarrow{d_{ij}}}(c_i)} - 1 \right| \leq \Delta\varnothing_{max} \tag{5.12}$$

Speed Validity

test other calculation

Since the speed of a FMO can be estimated from a single frame (5.8) and from two consecutive frames (5.10), we check if both measurements yield similar results. If they don't, the pair's contours most likely do not come from the same physical object, have large variation in speed, or it's a non-spherical FMO. The speed calculated from a streaks length can be inconsistent depending on the contrast between the FMO and background. It's mostly underestimated. This is especially true for long exposures and very high speeds. If the contrast is high, the streak is going to be longer than with low contrast. This has to do with how images are formed by the camera sensor. At the start of the exposure the objects backwards-facing outline will immediately move away from its initial position and the camera can only capture the light coming from it for a very small fraction of the exposure time. Therefore the outlines are more transparent than the rest of the object or not visible at all. Thus, they can be cut off from the foreground mask. Consider the two most extreme cases where the object has:

1. high contrast to the background, its outlines are perfectly visible and nothing is cut off in the foreground mask
2. low contrast to the background, its outlines get very transparent and gradually fade into the background, however still visible in the foreground mask

In case of 1., the actual motion of the FMO can be estimated to a fairly high accuracy given ℓ, $\varnothing$ and ε in a single frame (see 5.8). In case of 2. however, the real object motion during the exposure time is larger than what is visible on the foreground mask. We estimate the maximum it could have moved in addition to what is visible to be $\varnothing$. Therefore, under the assumption that v_{ij} is the actual speed of the object, we can allow a range of speeds to be valid, as described in Algorithm 2.

Low Speed

The low speed check is supposed to reject pairs of contours that move slower than a predefined threshold v_{min} relative to each other. v_{min} should be chosen to match a

Algorithm 2 Speed Validity Check

function CHECKSPEEDVALIDITY(c_i, c_j) ▷ Pair of contours
  $\varnothing_{\text{average}} \leftarrow \frac{\varnothing_i + \varnothing_j}{2}$
  $\ell_{\text{average}} \leftarrow \frac{\ell_i + \ell_j}{2}$
  $v_{\text{min}} \leftarrow \frac{\ell_{\text{average}} - \varnothing_{\text{average}}}{\varepsilon} \cdot \Delta t_{ij}$
  $v_{\text{max}} \leftarrow \frac{\ell_{\text{average}}}{\varepsilon} \cdot \Delta t_{ij}$
  if $v_{ij} \geq v_{\text{min}}$ **and** $v_{ij} \leq v_{\text{max}}$ **then**
    check passed
  else
    check failed
  end if
end function

specific tracking application and can be determined empirically. In case of ping pong the minimum speed of a ball on camera frames is dependent on the camera's distance to the table and its focal length or FOV. The check is considered passed if the following requirement is met.

$$v_{ij} \geq v_{\text{min}} \tag{5.13}$$

In an alternative implementation the low speed check may be dependent on the diameters of the contours in question and pose the following requirement

$$v_{ij} \geq f \frac{\varnothing_i + \varnothing_j}{2\Delta t_{ij}} \tag{5.14}$$

where $\varnothing_i$ and $\varnothing_j$ are the diameters of the corresponding contours in frames i and j and f is a parameter that defines how fast the contours need to move. As an example, for $f = \frac{1}{4}$ objects would have to move at least one fourth of their diameter in consecutive frames.

Lateral Motion

Lateral motion of objects can lead to contours in consecutive frames with similar appearance as FMOs [3]. FMOs are moving along the direction of their elongation. If $\overrightarrow{d_{ij}}$ (5.4) does not point in the same direction as $\overrightarrow{d_i}$ and $\overrightarrow{d_j}$, which are obtained by fitting the minimum area rectangle (5.2.2), the pair is rejected. The angle θ between $\overrightarrow{d_{ij}}$ and $\overrightarrow{d_i}$

can be calculated using the dot product, as long as $\theta \neq 0$, i.e. $\cos\theta \neq 1$:

$$\theta = \arccos \frac{\overrightarrow{d_{ij}} \circ \overrightarrow{d_i}}{\left\|\overrightarrow{d_{ij}}\right\| \left\|\overrightarrow{d_i}\right\|} \tag{5.15}$$

Given the preconfigured maximum angle θ_{max} the following inequation needs to be true for a contour pair to pass this check:

$$\theta \leq \theta_{max} \tag{5.16}$$

Due to the minimum area rectangle only being oriented correctly if the contour shows significant elongation, this check is only performed if the ratio $\frac{\ell}{\varnothing}$ is larger than 1.5.

visualization of lateral motion?

5.2.5 Pseudocode

While the following code does not completely represent the actual implementation it gives an overview of the detection-process.

Algorithm 3 Detector

```
1:  function DETECTBALLS(C_t, C_{t-1})          ▷ Contours obtained in preprocessing (1)
2:      for all c_t in C_t do
3:          for all c_{t-1} in C_{t-1} do
4:              if c_{t-1} is already tracked then
5:                  continue with next contour
6:              end if
7:              pair ← {c_t, c_{t-1}}
8:              if projectionLengthCheck(pair)                    ▷ see 5.2.4
9:                  and projectionRadiusCheck(pair)               ▷ see 5.2.4
10:                 and lowSpeedCheck(pair)                       ▷ see 5.2.4
11:                 and speedValidityCheck(pair)                  ▷ see 5.2.4
12:                 and lateralMovementCheck(pair)                ▷ see 5.2.4
13:                 and similarityCheck(pair) then                ▷ see 5.2.4
14:                 add pair as tracked object
15:                 break loop
16:             end if
17:         end for
18:     end for
19: end function
```

6 Tracking

6.1 Two-Dimensional

The motion of a previously detected FMO is predicted by the tracker. It takes the last 3 positions of the FMO to predict its location based on its previous position, velocity and acceleration.

6.1.1 Prediction of Motion

Future positions of a FMO can be predicted using simple numerical integration schemes, as long as its direction doesn't change abruptly. Given a timestep Δt, the time differences $\Delta t(t_1, t_2)$ between timestamps t_1 and t_2 and the latest three positions $X_{cm}(t)$, $X_{cm}(t-1)$, $X_{cm}(t-2)$ of the FMO its current velocity $v(t)$ and acceleration $a(t)$ as well as future positions can be estimated using Leap-Frog integration:

$$\begin{aligned} v(t-1) &= \frac{X_{cm}(t-1) - X_{cm}(t-2)}{\Delta t(t-1, t-2)} \\ v(t) &= \frac{X_{cm}(t) - X_{cm}(t-1)}{\Delta t(t, t-1)} \\ a(t) &= \frac{v(t) - v(t-1)}{\Delta t(t, t-1)} \\ v(t+1) &= v(t) + \Delta t * a(t) \\ X_{cm}(t+1) &= X_{cm}(t) + \Delta t * v(t+1) \end{aligned} \tag{6.1}$$

While more sophisticated integration methods could be used, this is sufficient for predicting an objects location in the next few frames and can handle temporary loss or occlusion as observable in Figure 6.1.

6.1.2 Estimation of Impact-Point

The location where a ball bounces off of a surface is not necessarily visible in a single frame, depending on the camera's frame rate, exposure time and speed of the FMO. Especially at low frame rates FMOs will often only be visible before and after impact.

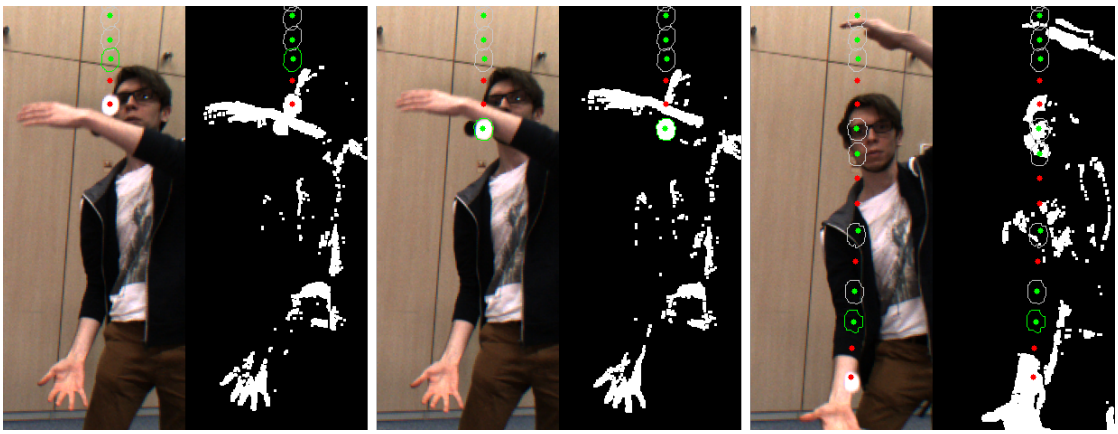


Figure 6.1: Pairs of camera images with corresponding foreground-masks. Tracked trajectory (grey contours) with last contour available in green. Dots indicate predicted locations where prediction was successful (green) or failed (red). Tracking is lost when the ball overlaps with motion of the body, but recovers thanks to multi-frame prediction.

Even if an image's exposure time-frame happens to coincide with the bounce, the shape of the FMO will possibly no longer be a straight streak and fail the requirements of the tracking and detection algorithms.

Intersection of Motion-Vectors The intersection point between the FMOs direction-vectors before and after the impact mark the position where its direction changed, i.e. their location of bounce. This introduces a delay of $\frac{2}{FPS}$s - $\frac{3}{FPS}$s, as the FMO needs to be re-detected after the bounce. Given the support vectors X^i_{cm}, X^{i+t}_{cm} and the directional vectors $\overrightarrow{d_i}$, $\overrightarrow{d_{i+t}}$ before and after impact respectively, their intersection can be calculated.

If the angle between $\overrightarrow{d_i}$ and $\overrightarrow{d_{i+t}}$ happens to be small, this method is inaccurate and shouldn't be used.

Add Code/Equations for how to calculate intersection?

6.1.3 Algorithm

After successful detection the next position $\hat{p}$ of a tracked FMO is predicted using numerical interpolation as described in Section 6.1.1. After the FMO-candidates of the next frame C_{t+1} are determined (see Algorithm 1), and before the detector is executed, the tracker tests if $\hat{p}$ lies inside of a contour in C_{t+1}. If a contour c is found, the diameter- and length-projection contour-pair check (see Section 5.2.4) is performed on it together with the previous frames contour of the tracked FMO. If the check is satisfied c is

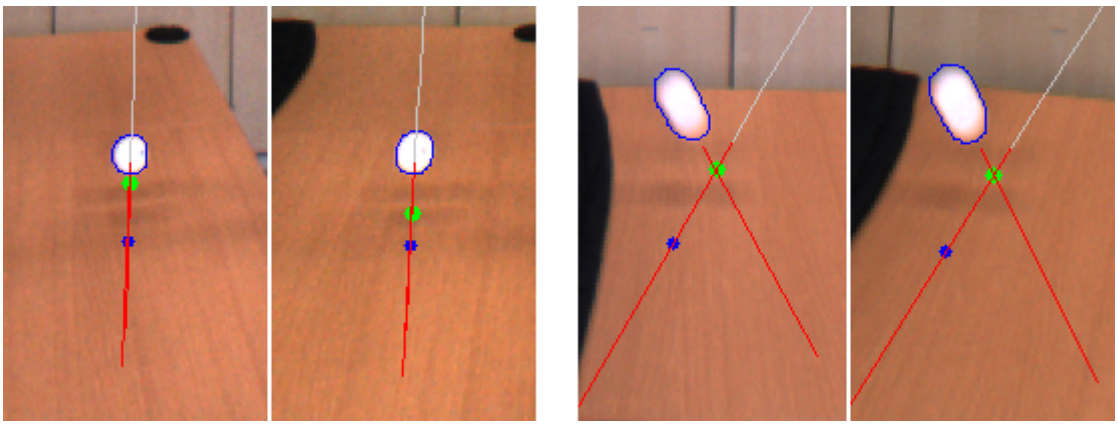


Figure 6.2: Pairs of left and right camera images. Tracked trajectories in grey. Direction before and after impact in red. Green: estimation of impact point by intersecting direction vectors. Blue: prediction of position without knowledge of impact. Left pair of images: the small angle between the direction vectors leads to inconsistent intersections. On the left camera image the impact point was estimated to be much higher than in the right camera image.

accepted as a contour of the tracked object. See Algorithm 4 for pseudocode.

6.2 Three-Dimensional

In this section the process of obtaining a 3D-position is explained.

6.2.1 Calculation of 3D-Position

well... add the math... it's just a little bit of geometry!

6.2.2 Reprojection Error

The reprojection error (RE) is a measurement for how accurate an estimation of a 3D point is. By calculating a 3D-position from multiple camera images (see 6.2.1) and projecting it back into the image, we can calculate the error as the Euclidean distance between the projected point and the original point. If the error is larger than the expected accuracy, the measurement can be marked as an outlier. Given an estimated 3D position $\hat{X}$ and the camera's projection matrix P we calculate the image projection of $\hat{X}$:

$$\hat{x} = P\hat{X} \tag{6.2}$$

Algorithm 4 Tracker

```
1: function trackBalls(trackedBalls, C_t)          ▷ C_t obtained in preprocessing (1)
2:     for all ball in trackedBalls do
3:         p̂ ← predictLocation(ball, t)                                  ▷ see 6.1.1
4:         for all c_t in C_t do
5:             if c_t is already tracked
6:                 or p̂ is not inside of c_t then
7:                 continue with next contour
8:             end if
9:             pair ← {c_t, ball.getLastContour()}
10:            if projectionLengthCheck(pair)                             ▷ see 5.2.4
11:                and projectionRadiusCheck(pair) then                   ▷ see 5.2.4
12:                add c_t to ball
13:                break loop
14:            end if
15:        end for
16:    end for
17: end function
```

Let x be the original position of $\hat{X}$ on the left camera's image and $\hat{x}$ be the image projection of $\hat{X}$ on the corresponding image. The reprojection error is then defined as the Euclidean distance between the image positions:

$$RE(\hat{X}) = \|\hat{x} - x\| \tag{6.3}$$

6.2.3 Matching of Tracked 2D-Objects

The (3D-)position of a FMO is calculated from two 2D-positions obtained by 2D-trackers. Both 2D-trackers are running simultaneously and report their tracked objects to the 3D-tracker. The 3D-tracker has to find pairs of tracked objects that are representations of the same physical object. To realize this we take pairs of recently detected and tracked 2D-objects, assume they are corresponding representations of the same object, and triangulate their 3D-position. If the re-projection error of the estimation is small enough, the pair can be considered a match [1]. Even if there happen to be some incorrect detections from the 2D-trackers, the epipolar contraint is a powerful noise filter [1].

6.2.4 Estimation of Gravity-Vector

When an object is flying through air the only force acting upon it is gravity, if we assume aerodynamics to be negligible. Therefore, the direction of gravity in camera-coordinate-space can be estimated by calculating an objects acceleration from its tracked trajectory. This can be used to predict an object's path during free fall.

7 Evaluation

The 3D-estimation accuracy is calculated by comparing it to reference data recorded with the ART system. Limitations, problems and future works are discussed.

7.1 Dataset

Using the ART system and our stereo-cameras simultaneously we recorded several small clips where an ART-marker is thrown. The marker has about the same size as a ping pong ball, but its grey color makes it hard to track from RGB cameras. Therefore we added lights to our cameras so that the marker's visibility is increased. Since the ART system is probably using standard tracking-techniques (i.e. not FMO-tracking) we couldn't throw the ball with high initial speed. It also loses track of the ball as it bounces off of a table and doesn't re-detect it until it slows down. We are especially interested in the accuracy while the ball is in free fall after being thrown.

7.1.1 Sequences

The tracking is evaluated on nine short image sequences. Exposure time varies and has been set to either 5, 10 or 15ms. Most of them are similar, but three are worth mentioning seperately.

60FPS 5SHUTTER LIGHT BOUNCE FAIL The marker is released from a hand and falls onto a table in a straight line, without any curvature. Therefore the impact-point estimation is not working.

60FPS 15SHUTTER LIGHT OCCLUSION The marker is thrown upwards and intersects the body of the thrower, making it necessary to re-detect the ball after intersection, if possible at all. For our method re-detection was working in only one camera of the two, due to the slightly varying perspective

60FPS 15SHUTTER THROW3 This sequence is an outlier, as the mean absolute error was the highest, but the mean reprojection error was the lowest among all sequences.

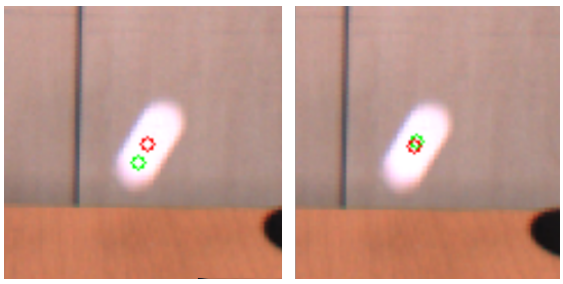


Figure 7.1: Green circle: ART reference position projected into the camera frame. Red circle: Estimation re-projected. Left: No offset, ART data at the end of exposure time of the camera. Right: Corrected timestamps of ART data by half the exposure time of the camera.

The reason for this has not been found and may be due to some mistake during recording.

7.2 Results

To test the accuracy of an 3D estimation $\hat{X}$, it is compared to its corresponding reference position X captured by the ART system (see Section 4.1). Specifically, we transform the reference-positions obtained through ART into the coordinate system of the left camera based on the calibration process described in Section 4.3. Even though we have measured the time-delay between both systems (see Section 4.4) and corrected for it there's still a problem: the systems have largely different exposure times. Whereas the ART system has a fast shutter speed, our stereo cameras expose for up to 15ms. The positions calculated by ART are therefore from the end of the exposure of the cameras (see Figure 7.1). Due to the ART system being proprietary we do not know its actual shutter speed and way of calculating the marker's position. As it runs at up to 300 FPS, its exposure time should be less than $\frac{1}{300}$ seconds. We calculate the position of the ball as the center of the contour, which is at the middle of the exposure time. We therefore correct the timestamp t_{ART} from every ART measurement adding half of the camera's exposure time:

$$t_{\text{ART}} = t_{\text{ART}} + \frac{\varepsilon}{2} \tag{7.1}$$

As the timestamps between reference data and our estimations are not synchronized, we use linear interpolation on the reference data. Interpolation is only performed if reference-data within $\frac{1}{FPS}$ seconds before and after our estimation is available.

Mean Absolute Error (AE) We compute the Euclidean distance between the estimation $\hat{X}$ and reference point X for all n points where both estimation and reference data is available:

$$AE = \frac{\sum_i \sqrt{(\hat{X}_i - X_i)^2}}{n} \tag{7.2}$$

Mean Absolute Reprojection Error (RE) Given the reprojected 2D position $\hat{x}$ of the 3D position estimation $\hat{X}$ of the FMO and its image position x, the mean absolute reprojection error (see Section 6.2.2) is calculated:

$$RE = \frac{\sum_i \sqrt{(\hat{x}_i - x_i)^2}}{n} \tag{7.3}$$

Errors During Flight (AEF and REF) The errors we are interested in the most are during the flight of the marker. 279 of 654 measurements are of the ball being in free fall after being thrown and before hitting a surface. We abbreviate the (mean) absolute error and (mean) reprojection error during flight as *AEF* and *REF* respectively. As we can see in Table 7.1 the error is significantly lower during flight. The marker's contour is not accurate while rolling on the table due to shadows and reflections on the table. This is most likely because the marker is rolling on the table after bouncing off of it. The box-plots in Figure 7.2 show the error in all camera-axes and reveal that a large portion of the absolute error is related to inaccurate depth-estimation (camera Z-axis). This could most probably be improved by using a standard stereo-calibration, or different camera-placement, as originally planned. The outliers are mainly from the (*60FPS 15SHUTTER THROW3*) recording, which is included nonetheless, as there's no indication for a bad capture, except for the sub-par results.

7.3 Limitations and Problems

Background-Subtraction While the background-subtraction works for FMOs that are brighter than their background, dark objects, like a grey or darker blue ball, might be considered a shadow by the algorithm used and not show up in the foreground mark, making it impossible to detect or track them. If the object is rolling on a surface, its own shadow makes the contour larger and reduces accuracy up to a point where values are unusable. We can observe the higher errors in Table 7.1. Alternatives should be considered.

maybe add image

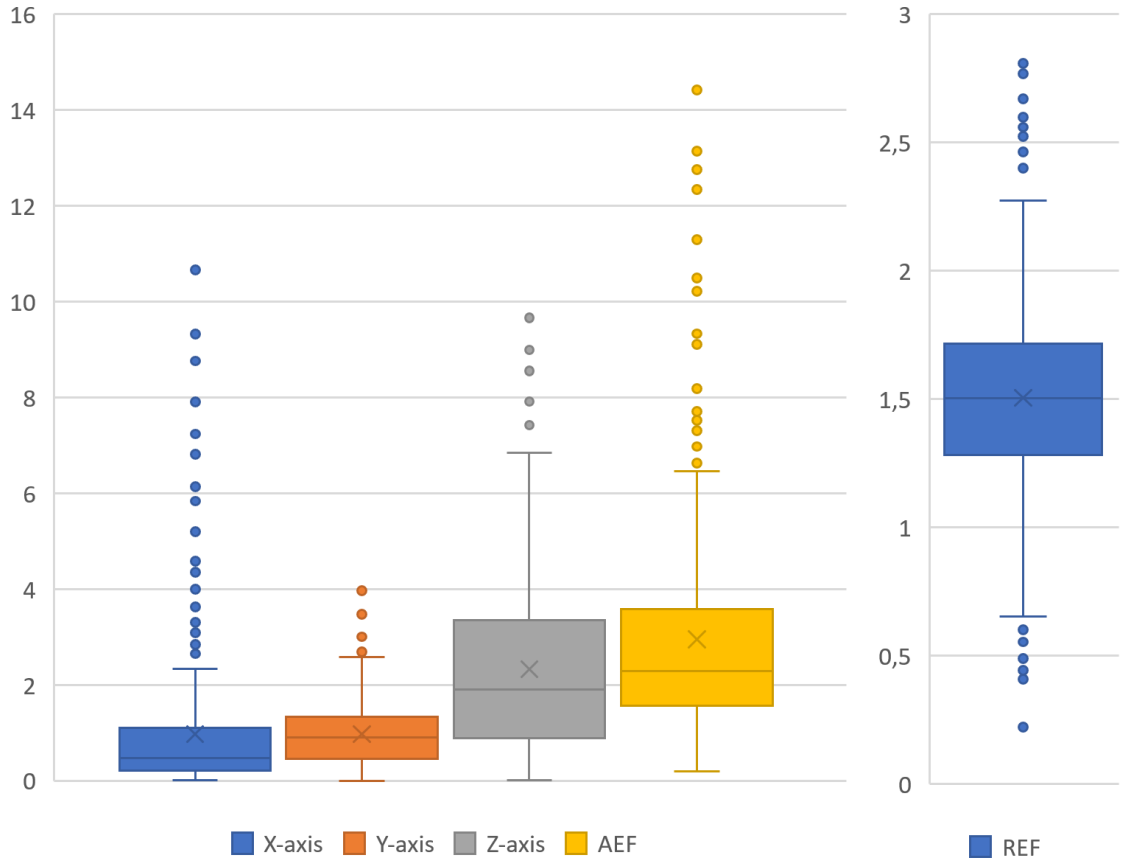


Figure 7.2: Box-plot of absolute error during flight (AEF) over all sequences; split into the corresponding camera-axes. Reprojection error during flight (REF) over all sequences.

Recording	AE	AEF	RE	REF
60FPS THROW	2.3	2.3	1.228	1.228
60FPS 5SHUTTER LIGHT BOUNCE	4.6	2.8	1.301	1.486
60FPS 5SHUTTER LIGHT BOUNCE FAIL	2.3	2.6	0.923	0.945
60FPS 10SHUTTER LIGHT BOUNCE	4.5	2.3	1.08	1.515
60FPS 15SHUTTER LIGHT OCCLUSION	1.9	1.9	1.962	1.962
60FPS 15SHUTTER LIGHT BOUNCE	5	2.5	1.219	1.529
60FPS 15SHUTTER THROW	4	2.8	1.495	1.516
60FPS 15SHUTTER THROW2	4.3	2.7	1.263	1.61
60FPS 15SHUTTER THROW3	7.7	7.7	0.687	0.687

Table 7.1: Mean absolute errors (AE) and mean absolute errors during flight (AEF) in centimeters. Mean reprojection errors (RE) and mean reprojection errors during flight (REF) in pixels.

False-Positives Mainly in 2D we can observe a number of false positives. It increases with lower exposure time, because of the missing elongation of the FMO. Even circular shapes can be FMOs, if the exposure time is small enough. Chosing a higher ε-value (e.g. 15ms) decreases false positives reliably and should be preferred. Some object motions can be considered a FMO, like moving shoes. The issue almost vanishes in 3D, due to the epipolar constraint [1].

Impact Point Estimation Since the FMO has to be redetected after its impact, there is a delay of 2-3 frames to the estimation. If it changes direction again, before re-detection, the impact point cannot be estimated. This is especially unfavorable in very fast sports as ping pong. Also, if the angle of intersection is small between incoming and outgoing directions, the estimation gets inaccurate.

8 Conclusion

8.1 Summary

Stuff text

8.2 Future Work

Foreground/Background Segmentation To track a larger color-range of objects, especially darker objects, the background subtraction needs to be replaced with a different implementation.

Camera Setup Other stereo-camera arrangements (e.g. one camera from the top and side) should be considered. A standard stereo-calibration as offered by OpenCV would probably give better results in the typical side-by-side stereo-vision setup.

Motion Prediction The prediction of ball-motion could be done more accurately in 3D than directly on camera images, as gravity could be considered. The predicted 3D location could be projected onto camera frames for tracking in 2D. Also, given the pose of the ping pong table's surface, the impact of the ball on the table could be predicted and factored into the predicted trajectory. Instead of simple numerical integration a parabola could be fitted to the tracked points.

Interactive Projection-Mapping The ball's impact point or trajectory could be used as an input to an augmented reality game that is projected onto a (table tennis) table.

List of Figures

List of Tables

Bibliography

[1] N. Owens, C. Harris, and C. Stennett. "Hawk-eye tennis system". In: *2003 International Conference on Visual Information Engineering VIE 2003*. July 2003, pp. 182–185. DOI: 10.1049/cp:20030517.

[2] Bounce. *Wonderball*. URL: https://www.bouncepingpong.com/wonderball/.

[3] D. Rozumnyi, J. Kotera, F. Sroubek, L. Novotny, and J. Matas. "The World of Fast Moving Objects". In: (Nov. 23, 2016). arXiv: http://arxiv.org/abs/1611.07889v1 [cs.CV].

[4] D. Rozumnyi. "Tracking, Learning and Detection over a Large Range of Speeds". Bc. thesis. FEE, CTU, Prague, 2017.

[5] A. Hrabalík. "Implementing and Applying Fast Moving Object Detection on Mobile Devices". MA thesis. Czech Technical University, 2017.

[6] A. R. T. GmbH. *Advanced Realtime Tracking (ART)*. URL: https://ar-tracking.com/.

[7] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[8] J. Y. Bouguet. "Matlab camera calibration toolbox". In: *IEEE Transactions on Reliability - TR* (Jan. 2005).

[9] D. Pustka, M. Huber, M. Bauer, and G. Klinker. "Spatial Relationship Patterns: Elements of Reusable Tracking and Calibration Systems". In: (2006).

[10] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[11] M. Huber, M. Schlegel, and G. Klinker. *Temporal Calibration in Multisensor Tracking Setups*. 2012.

[12] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* IEEE, 2004. DOI: 10.1109/icpr.2004.1333992.

[13] Z. Zivkovic and F. van der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". In: *Pattern Recognition Letters* 27.7 (May 2006), pp. 773–780. DOI: 10.1016/j.patrec.2005.11.005.