Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

DISSERTATION

Predicting the Trajectory of the Flying Object with the Use of k-Nearest Neighbors

Determination of Thrown Object Impact Position in Manufacturing Transportation Systems

Submitted at the Faculty of Electrical Engineering and Information Technology,
Technische Universitaet Wien in partial fulfillment of the requirements for the degree of
Doktor der technischen Wissenschaften (equals Ph.D.)

under the supervision of

Prof. Dr. Dietmar Dietrich
Institute of Computer Technology
Technische Universitaet Wien

and

Prof. Dr. Liliya Chernyakhovskaya
Institute of Technical Cybernetics
Ufa State Aviation Technical University

by

Konstantin Mironov
Matr.Nr. 1227747
Ehrensteingasse 3/I/4, 1220 Wien

Vienna, April 2016

Kurzfassung

Automatisiertes Werfen und Fangen stellt einen vielversprechenden Weg dar, um den Transport von Objekten und Teilen in einer industriellen Umgebung zu gewährleisten. Um ein geworfenes Objekt mit dem Greifer zu fangen, wird das Wissen über die Flugbahn des Objektes innerhalb des Arbeitsbereiches des Greifers gebraucht, weshalb die Flugbahn vorhergesagt werden muss. Die meisten existierenden Algorithmen für diese Aufgabe basieren auf der Modellierung der ballistischen Eigenschaften des geworfenen Körpers. Der in dieser Forschung vorgeschlagene Algorithmus erfordert keine genaueren physischen Daten zum Flug; er basiert auf dem Lernen von Beispielen des Werfens. Als Basistechnik zur Prognose von Objektkoordinaten wird das Nächste-Nachbar-Prinzip angewendet. Zwei Modifikationen wurden vorgenommen, um die Leistungsfähigkeit und die Genauigkeit der Vorhersage zu erhöhen. Zuerst wird eine Suche nach dem nächsten Nachbarn innerhalb einer relativ kleinen Teilmenge von ähnlichen Beispielen anstatt innerhalb der gesamten Datenbank gestartet, was eine rasche, skalierbare Verarbeitung großer Datenbanken ermöglicht. Zweitens werden die Vorhersagen in einem zweidimensionalen, Trajektorie-bezogenen Koordinatensystem anstelle eines dreidimensionalen Weltkoordinatensystems gemacht. Dadurch wird der Algorithmus Unabhängigkeit von der Startposition, der Wurfrichtung und der Tracking-Umgebung. In der Simulation wurde gezeigt, dass diese Methode die Vorhersagegenauigkeit für die Trajektorie der geworfenen Objekte im Vergleich zu den anderen existierenden Ansätzen erhöhen konnte. Die vorgeschlagenen Änderungen verbessern neben der Genauigkeit der Vorhersage auch die Berechnungszeit für die Prognose. Die Fangversuche mit einem Roboterarm zeigten, dass dieser Vorhersagealgorithmus es erlaubt, Kugeln in der Echtzeitanwendung zu fangen.

Abstract

Automated throwing and catching is a promising way to provide the transport of objects and parts in an industrial environment. For the gripper to catch the thrown object, prior knowledge about the trajectory of the object within the gripper workspace is needed. Therefore, the trajectory must be predicted. Most of the existing algorithms for this task are based on modeling the ballistic properties of the thrown body. The algorithm proposed in this research does not require the physics of flight to be exactly known; it is based on learning from example throws. As a basic technique to forecast object coordinates, the nearest neighbors principle is applied. Two modifications are made in order to increase the performance and accuracy of prediction. First, a search for the nearest neighbors is done within a relatively small subset of similar examples instead of the entire database. This allows the fast, scalable processing of large databases. Second, forecasts are made within a two-dimensional trajectory-related coordinate system instead of a three-dimensional world coordinate system. This provides algorithm invariance to the launching position, direction of throw and tracking environment. Once this algorithm was applied to the trajectories of thrown objects, it demonstrated that it could increase prediction accuracy in comparison with other existing approaches. The proposed modifications not only improve the accuracy but also the speed of prediction. Catching experiments with a robotic arm showed that the predictor allowed balls to be caught in real time.

Acknowledgements

Doing my doctoral research at two universities at the same time was a challenging task for me, and I could not have accomplished it without the help I received from so many people on both ends. First of all, I would like to thank my supervisors: Professor Dietmar Dietrich from the Technische Universitaet Wien and Professor Liliya Chernyakhovskaya from the Ufa State Aviation Technical University. It was a great honour to work under their supervision. They have not only supervised me but have continually motivated me to work for nearly four years. I would also like to thank my colleague Martin Pongratz, who is the leader of the research project "transport-by-throwing". He has helped me solve a number of issues during my research - scientific, administrative, and social. Without his help, finalizing my work would have been impossible. A special thanks goes to the Professors Vladimir Vasilyev, Murat Guzairov, Rustem Munasypov and Doctor Gouzel Shakhmametova. I discussed some specific topics of my thesis with them, and they were very helpful. I also received significant help from these students and junior scientists working on the research project: Manuel Huber, Maximillian Goetzinger, Balint Toth, and Gilbert Markum in Vienna, and Irina Vladimirova in Ufa. My work was provided by the Erasmus Mundus Action 2 MULTIC Scholarship, and I would like to thank all the people that helped me in obtaining it and managing administrative issues. Last but not least, I want to say thank you to my family and my friends. They have believed in me and been supportive since I began this work, and it has been really helpful and encouraging.

Table of Contents

1 Introduction **1**
- 1.1 Material transportation . . . 1
- 1.2 Transportation by throwing and catching . . . 2
  - 1.2.1 Overview . . . 2
  - 1.2.2 Throwing . . . 4
  - 1.2.3 Catching . . . 5
  - 1.2.4 Trajectory observation and prediction . . . 5
- 1.3 Challenge of trajectory prediction . . . 9
- 1.4 Learning-based prediction . . . 12

2 State of the art **15**
- 2.1 Aerodynamics of ballistic motion . . . 15
  - 2.1.1 Common view . . . 16
  - 2.1.2 Arm ballistics and aerodynamics of gliders . . . 19
  - 2.1.3 Research on the aerodynamics of sport balls and similar objects . . . 20
- 2.2 Statistical estimation of ballistic curves . . . 24
- 2.3 Visual tracking of moving objects . . . 28
  - 2.3.1 Motion capture with object-integrated sensors . . . 30
  - 2.3.2 Structured light and time-of-flight range measurement . . . 30
  - 2.3.3 Monocular vision . . . 33
  - 2.3.4 Stereo vision . . . 34
- 2.4 Robotic catching and transportation by throwing . . . 38
  - 2.4.1 Throwing and catching . . . 38
  - 2.4.2 Transport by throwing . . . 44
  - 2.4.3 Discussion . . . 46
- 2.5 Trajectory prediction in robotic catching and transport-by-throwing . . . 50
  - 2.5.1 Catching without prediction . . . 51
  - 2.5.2 Prediction using physical models . . . 52
  - 2.5.3 Neural network prediction . . . 54
  - 2.5.4 Premises for use of nearest neighbors algorithm . . . 56

3 Observing object trajectory **59**
- 3.1 Throwing experiments . . . 59
  - 3.1.1 Throwing . . . 60
  - 3.1.2 Tracking . . . 62

3.1.3 Triangulation 68
3.1.4 Acquired datasets 70
3.2 Accuracy of positioning static object 72
3.3 Accuracy of positioning the flying object 76
3.3.1 Influence of background subtraction 77
3.3.2 Errors in range measurement over long distances 80
3.3.3 Measuring object's velocity 84
3.3.4 Summary 85

4 Algorithm for trajectory prediction 87
4.1 Means of initial validation 87
4.2 Coordinate transformations 89
4.2.1 Overview 89
4.2.2 Gravity-related coordinate system 92
4.2.3 2D representation for 3D coordinates 94
4.2.4 Invariance to release point 101
4.2.5 Reverse transform and robot coordinate system 102
4.3 Predictor 103
4.3.1 Forecasting operation 103
4.3.2 Search for nearest neighbors 106
4.4 Allocating the subset of neighbors 107
4.5 Predictor summary 114

5 Implementation and experiments 117
5.1 Numerical experiments with the dataset 117
5.1.1 Simple and weighted nearest neighbours 118
5.1.2 Size of the cluster 119
5.1.3 Value of k 121
5.2 Implementation 122
5.2.1 Integration into the transportation system 123
5.2.2 Real-time predictor 124
5.3 Catching experiments 126

6 Conclusion and Future Work 129
6.1 Results of research 129
6.2 Future work and outlook 132

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AOI	Area of Interest
A-AOI	Algorithmic AOI
C-AOI	Camera AOI
CA	Calibration errors
CCTV	Closed-Circuit Television
CPU	Central Processing Unit
CSV	Comma-Separated Values
CWC	Catching With Consideration of shock contact
CWOC	Catching WithOut Consideration of shock contact
DSP	Digital Signal Processor
EKF	Extended Kalman Filter
FOV	Field of View
FPGA	Field-Programmable Gate Array
GA	Grasping Area
GAG	Gaining Angle of Gaze
GAT	Grasping Area Trajectory
GPU	Graphic Processing Unit
HPA	High Precision Angle
HPT	High Precision Throw
HSV	Hue-Saturation-Value color space
IPE	Image Processin Errors
KF	Kalman Filter
KFO	k Nearest Neighbors Forecasting Operation
k-NN	k Nearest Neighbors
KSO	k Nearest Neighbors Search Operation
LfE	Learning from Examples
LOO	Leave-One-Out rule
LPT	Low Precision Throw
LS	Least Squares
MA	Measurement Area
MAT	Measurement Area Trajectory
MLE	Maximum Likelihood Estimation
NNTP	Neural Network Trajectory Predictor
PC	Personal Computer
QE	Quantization errors
RANSAC	RANdom SAmple Consensus algorithm
RBF	Radial Basis Function
RGB	Red-Green-Blue color space

RLS	Robust Least Squares
SCC	Stereo Camera Coordinates
SLS	Structured Light Scanner
SVR	Support Vector Regression
PoF	Plane of Flight
TbT	Transportation by Throwing
ToF	Time-of-Flight
TCR	Throw-Catch Route
UFL	Used Frames List
UKF	Unscented Kalman Filter
USB	Universal Serial Bus
WKNN	Weighted k Nearest Neighbours

1 Introduction

The contribution described in the current thesis is part of a bigger research project that aims to develop a new approach in the transportation of material objects from one place to another by throwing and catching. The task of object transportation (material handling) often arises in industry, for example when there is the need to relocate an object from one machine tool and process it to another or when the product is moved from machine tool to warehouse. Approaches to this task may differ in various production systems in various situations, and different means can be used for transportation. The development of this field is motivated by the increasing requirements of the performance, quality and flexibility of production systems.

This introductory chapter of the thesis is organized in the following way. Section 1.1 is a brief introduction to modern transportation systems. Then in 1.2, an overview of the research into transportation by throwing and catching (Transport-by-Throwing, TbT) is given. The research described in this thesis concentrates on one aspect of the TbT system: predicting the trajectory of a flying body. This challenging aspect is discussed in 1.3The challenges and open issues that motivated the research are listed in that section, and the proposed solutions to those challenges are laid out in section 1.4.

1.1 Material transportation

Material transportation is a natural part of the common industrial manufacturing process. The means of manufacturing include the machine tools that process the products, the workers that perform the work that cannot be performed by the machine tools and the environment (transportation network) that provides the flow of products or its parts between the workers and the machine tools. History shows that the allotment of the tasks that are performed by human workers decreases with the progress of industrial technologies. Mass production, the standard in industrial processes since the beginning of the 20th century (the introduction of mass production is usually associated with the opening of Highland Park Ford Plant in 1910), requires the automated sequential transportation of products and parts between the machine tools and the workers who process these products and parts. In a fully-automated industrial process, all transportation and the processing of objects is done by machine tools.

The traditional networks for object transportation are based on various conveyor systems (belt conveyors, screw conveyors, small rail-guided vehicles, etc.) that transport multiple small objects over short distances and special vehicles that transport larger objects over longer distances. In some specific conditions, other techniques exist, e.g. pneumatic tubes. The TbT concept could replace conveyor belts in some specific cases and conditions.

Commonly, the conveyor belt is a mean that takes up a significant volume of factory workspace and determines the disposition of the machine tools. The machine tools are arranged sequentially along the production line associated with the conveyor belt. The following properties of conveyor systems should be mentioned [Sul09]:

- *Ability to move a large number of parts*: multiple objects may be transported by the conveyor belt at the same time
- *Speed of transportation*: may be adjusted according to the requirements of machine tools.
- *Possibility to allocate a temporary storage for the parts between the machine tools.*
- *Highly automated transfer of the objects.*
- *Common environment for object transfer between multiple machine tools*: there are multiple machine tools processing the object between source and destination.

The increasing demand for individual products and number of product variants has increased the requirements of the flexibility and speed of material transportation systems [Pon11]. The flexibility of production in this paper (synonyms for the same or similar phenomenon are adaptability, agility, and changeability) means that a fast and simple reconfiguration of the manufacturing environment is possible if there is a change in the manufacturing process. These conditions spur the development of alternative technologies for object transportation.

1.2 Transportation by throwing and catching

Transport-by-throwing (TbT) is a novel approach to object transportation proposed by Frank [Fra06]. Possibilities for this use of this approach in material handling in industry are now being investigated. TbT systems could replace conveyor-based systems in the high-speed transportation of small, rigid objects over short distances.The main principle of TbT is that an object is thrown by a specific throwing device from the source point towards the destination point and is then caught by a special catching device (gripper) at the destination point. The motion between these points is influenced by gravity and the impulse of throw. It follows the ballistic trajectory.

1.2.1 Overview

An example structure of a transportation network based on throwing is shown in figure 1.1 (a modified version of the figure from [Fra06, p.92]). Various workstations that treat the objects during the manufacturing process are connected via throw-catch routes. The throwing devices are located at the starting point of each route. A gripper is located at each destination point. It catches the flying object and gives it to a processing machine tool.

This type of transportation network when compared to traditional networks based on conveyor belts is different in several ways:

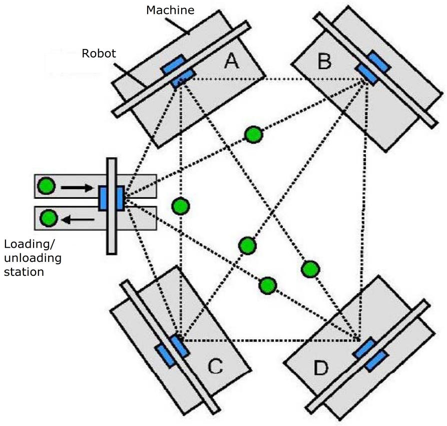


Figure 1.1: A transportation Network based on throwing- a modification of the picture from [Fra06, p.92].

- There is no need to set up a special conveyor infrastructure on the floor between two workstations. Therefore, applying TbT factory workspace to be saved. Another aspect of this network is that there is no need to mechanically connect various machine tools.
- In conveyor-based systems, it is typical for one conveyor to provide sequential object transportation through many workstations. Workstations in this situation are usually located along the conveyor. In the throw-catch approach, each pair of sequential workstations have their own route. If there is a need to transport an object between two distant workstations, it must be done via several throw-catch routes. However, it is also possible for one throwing device to throw objects towards several grippers or one gripper may catch objects thrown from several throwing devices.
- It is unnecessary to position machine tools in a straight line like in those transportation networks based on conveyor belts.
- The ratio between the object velocity and route capacity as well as power consumption could be much better. The expense of energy in the throw-catch approach is due to the throwing movement and the mechanical actions of the gripper. In conveyor-based systems, large energy expenses are connected with the moving conveyor belt.
- The distance is limited by the power of the throwing device.

The development and exploration of the approach created in [Fra06, Fra08a, Fra08b, Bar08, Pon10, Pon11, Fra11, Fra12] showed that this could become a viable method of object transportation under the following conditions:

- Size of the object is around 10 cm or less. In most of the experiments tennis balls were used [Fra08, Bar08, Pon10] while some other compact object were also considered [Fra11, Fra12].
- The throwing distance would have to be that of several meters. An increase in distance leads to an increase in deviation from the interception point. Hence, the area of catching expands with an increase in distance. Long distances also require higher throwing velocities and more equipment-free space for object trajectories.

- The velocity of throw must be up to 10 meters per second. A higher velocity would lead to higher energy consumption. Throwing velocity has a significant influence on the maximum distance of flight. For example, throwing the ball at a velocity of 10 m/s and at a =4 angle to the horizon leads to a distance of 9 m with a maximum height of 2 meters [2]. High velocities over small distances can lead to an increase in the relative velocity of both the object and the gripper at the point of interception, which could damage fragile objects.

One throw-catch route will now be discussed, which would provide object transportation from source point A, where throwing-device is located, to destination point B, where the gripper is located. One throwing device throws identically-shaped objects towards one gripper. The following four main tasks of the system are considered in [Pon11, p. 138]:

1. The throwing device must throw the object from the source point towards the destination point.
2. The catching device must capture the object within its workspace around the destination point.
3. The trajectory of the object within the gripper's workspace must be predetermined in order to define the catching movement of the gripper. In other words, the trajectory of the object must be predicted.
4. Its prediction requires information about the initial stage of flight; therefore, the trajectory of the object must be observed in real time.

A simple throw-catch route includes two electromechanical devices (the thrower and the gripper) and a computer system for tracking the objects and gripper control. This system may be distributed on the controllers of the gripper, the thrower and additional devices (e.g. sensors for object tracking).

1.2.2 Throwing

The thrower's task is to throw objects in such a way that its impact position is within the gripper catching area. In principle, the thrower should be adjusted each time it throws in order to provide the identical initial parameters required for each object (velocity, angles, object orientation).

The first step in the development of a working Transport-by-Throwing system is the creation of a throwing device, which would allow objects to be thrown more or less precisely. This means that such a device would throw objects without strong deviations in velocity and direction of throw. If all thrown objects intercept the gripper workspace, the throwing device meets the requirements. Several works exist that are connected with the development and control of these types of devices, e.g. [Fra08a, Fra12]. At this time the question of how to improve upon these devices is still open, but devices that meet these requirements do already exist. A throwing device cannot be perfect; deviations in throws will always exist (i.e. the velocity and the direction of the thrown object will not be exactly the same every time). The object?s trajectory will always be different due to these deviations and how the object interacts with the air flow. An exploration of the deviations of a throwing device used in the experimental evaluation of this research is given in subsection 3.1.1.

The quality of throwing is sufficient if the device can make the object land within a certain area. Accurate throwing lead to relatively small size of this landing area. The throwing of objects was considered to be a self-contained robotic task in research [Miy10, Kob11, Nem11, Zha12, Kan12]. For example, in [Nem11] balls thrown from a distance of 2.5 meters were able to reach the precisely the 5 cm diameter landing area.

1.2.3 Catching

The gripper's task is to catch the flying object and give it to the machine tool. Two main types of mechanical catching are considered by [Fra06]: hard and soft catching. "Hard" catching is where the gripper is positioned at the point of expected intersection and waits there for the flying object. At the moment of intersection, it grasps the object. The velocity of the gripper at this moment is equal to zero; therefore, the relative velocity between the object and the gripper is equal to the velocity of the object's flight. Hard catching was implemented in [Fra08b] with a gantry Cartesian robot that has two degrees of freedom; it can move both vertically and horizontally, which is perpendicular to the distance from the throwing device. Hence, the gripper can move within a vertical gripping plane defined by these directions. The predictor gives the data to the gripper about the point where the object intersected this plane, and then the gripper moves to this point and waits there for the object.

In the "soft" catching, the gripper has even more freedom. The end-effector of the catching device has its own trajectory that matches the trajectory of the object at the catching point (and at the same moment in time) [Pon13]. The trajectory of the end-effector is determined in such a way that the relative velocity of the object is minimal at the moment of catching. Therefore, the interception overloads are minimized, and soft catching may be used with more fragile grippers and for more fragile objects.

A similar dichotomy of catching movements for human object-grasping has already been discussed in [Kaj99]. There CWOC (catching without consideration of shock contact) and CWC (catching with Consideration of shock contact) are specified and correspond to hard and soft robotic-catching respectively. It was shown that in order to catch a moving object a human must either move its arm in a way that enables it to reach the object as quickly as possible (CWOC) or move in a way that minimizes the overload (CWC). In this way the concept of soft catching can be considered to be bio-inspired.

The gripper's actions depend on the received information about the object's flight. Specific sensors and software are used to extract this information. The requirements for extraction algorithms and the trajectory prediction task are discussed in subsection 1.2.4. After the interception point is determined, the instructions for the catching device must be generated. The catching device must perform the capturing movement right before the moment of expected interception, and this movement must not damage the object, the gripper or the environment. After a successful catch, the gripper must transfer the object to the next tool, processing it with the required orientation. This task also has various challenges, depending on the construction of the capturing device and the type of object.

1.2.4 Trajectory observation and prediction

A successful catch can only be achieved by defining the point in gripper workspace where the grasp should take place with a particular throw. To do this accurate and immediate data about the thrown object's trajectory is required. Therefore, a sensor system for measuring object coordinates in 3D space is needed. The use of a digital camera setup for this type of system is researched in most of the work connected with TbT (e.g. [Bar08, Bar09, Pon09, Pon12]). A single camera is able to track the ball with the use of additional information provided by other sensors or by specific theoretical assumptions. For example, in [Bar08] visual tracking of the flying ball with a single camera is supported by measuring the launch coordinates with light barriers and a specific model of the object's motion. Another factor in determining object spatial coordinates from a single camera is that information on the flying object's size must be accurate..

A stereo camera setup (two synchronized cameras observing the same scene) allows data about object coordinates to be directly extracted from a pair of simultaneous images. The accuracy of extraction depends on the parameters of cameras and on the quality of the extraction algorithms.

The two main stages of trajectory processing can be summarized as the collection of data about the current trajectory (tracking) and the forecast of parameters values into the catching area (prediction). These stages were described as separate tasks in subsection 1.2.1. In this subsection, the first stage will be discussed. Trajectory prediction is one of the main topics of this dissertation and will thus be discussed in more detail in a separate section (1.3). Flight information processing may be divided into the following steps:

1. *Scene observation before the flying object?s appearance*: This stage begins when the throwing device signals that it is about to throw.
2. *Object detection and positioning when the object flies into the observer's field-of-view*: The tracking system must first detect that the object is flying through the scene and then determine its coordinates in 3D space..
3. *Object tracking*: This stage includes the collection of data about object position changes during a period of time. To increase the accuracy and performance of the algorithm, data on previous object positions may be used.
4. *Future trajectory prediction.*
5. *Determination of the interception point where the capture must take place.*
6. *Determination of instructions for the gripper.*
7. *Gripper actuation.*

Consider the throw-catch route where the coordinate system is positioned with regard to the throwing device. The gripper and the gravity vector are located as shown in figure 1.2. The y-dimension here is perpendicular to the plane of the figure. The gripper is located near the destination point. It has some workspace where it can move more or less freely according to its degrees of freedom. For the 2-DoF gripper used in [Fra08b], the workspace will be within a planar rectangle and is restricted to the size of the gantry. The workspace for the robotic arm is restricted by the lengths and maximum angles of the joints. The capturing point must lie inside the gripper workspace. The area of space where the capturing point may lie is called the Gripping Area (GA). The part of the trajectory that lies within the GA is called the Gripping Area Trajectory (GAT).

Information about the initial velocity of the object can be provided by the specific sensors on the throwing device (e.g. light barriers in [Bar08]) or estimated based on camera measurements.

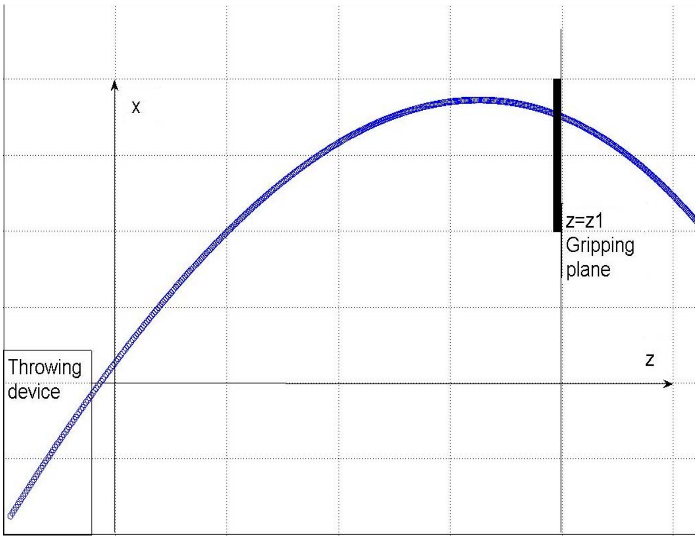


Figure 1.2: Position of throwing device and gripping plane in a transportation system based on hard catching.

Essentially, the throwing device is adjusted to throw the object towards the destination point at a constant velocity and at an angle to the horizon. The differences between various trajectories are caused by deviations in initial velocity, the direction of throwing and unpredictable interactions with the air. For example, in [Coo00] it was shown that the trajectory of a tennis ball is unstable in the initial stage of the flight. After reaching a distance equal to that of approximately ten times that of its diameter from the launching point, the ball came to a dynamically stable state, and its trajectory from that point on could be determined more or less accurately. Thus, information about the object's trajectory after the throw is needed for prediction.

The set of sensors (e.g. synchronized stereo pair) is used to measure the object's position. Cameras should be positioned in such a way that maximal accuracy of measurements can be provided (if such a position is allowed by the construction of the guild and enables communication with the controller).

Similar to GA and GAT, the measurement area (MA) and measurement area trajectory (MAT) are defined. In this area, starting from the launching point (figure 1.3), the coordinates of the flying object are measured by sensor system. After the object flies out of this area, a prediction of its trajectory in the catching area is made. The space between the measurement and the catching area can be called the processing area (PA). While the object is flying through the processing area, the following system must complete the following:

1. Process the data, received from sensors;
2. Predict GAT of the object;
3. Determine instructions for the catching device;
4. Move the catching device to the starting point for grasping.

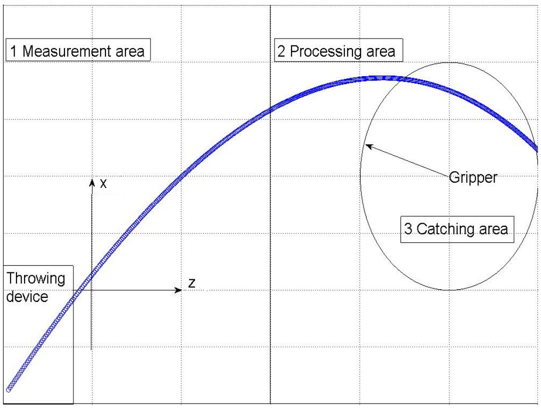


Figure 1.3: Division of motion space into three areas

If the initial velocity of the object is around 5-10 m/s, and the distance is 1.5-3 meters, the duration of flight would be around 0.7-1.5 seconds [3]. This period must be divided into a measurement and processing interval- except the time of catching.

A large MA has some advantages. A large number of measurements makes the data more reliable. The smaller the size of the "blind" processing area means the deviation of the trajectory will be less unpredictable. Even for various different trajectories, the values of the coordinates are not much different at the beginning of the flight. Hence, in a small measurement area large deviations at the end of flight should be able to be precisely predicted because of low deviations at the beginning of the flight. In that situation, it is not likely that small measurement errors will strongly influence prediction accuracy. The time performance of the system greatly limits the size of the measurement area. The time needed for the object to fly through the processing area must not be less than the time the gripper needs to prepare to catch the object.

The division of the trajectory into three parts is conditional. In reality the measurements could even be made within the gripper's workspace. If the predictor is fast enough, it can update the prediction after each new frame is received. The time shift between the moment the prediction is made and the expected moment of catching is first determined by the time the gripper's servo needs to perform the catching movement. Therefore, the processing area trajectory (PAT) is considered to be the part of the trajectory that corresponds to the specific time period that is sufficient for performing the catching movement before the object reaches the gripper's workspace. Once the first prediction operation has been completed, the dynamics of the object in the MA is an input for predictor, and dynamics of the object in the GA is the required output. Later on, the new prediction may be obtained, and the new results may be used to correct the path of the gripper. However, the gripper may not deviate strongly from its trajectory if it has already started to move. Hence, the decision on the gripper's trajectory to make the catch must be made based on the data received from the measurement area.

The goal of tracking is to obtain information about the motion of the object through the measurement area. To do this the system must be able to perform the following tasks:

1. *Detection*: to establish that the object is present in the image, i.e. it is in the measurement area.

2. *Recognition*: to define the position of the object in the image. This and the previous step are implemented based on the object recognition algorithm. This algorithm must be able to distinguish the object's contour from the industrial scene (which can contain moving equipment) or to detect the specific characteristics of the object. The goal at this stage is to define the pixel position of the object's center of mass in the images. This must be done very quickly- advisably within the timeout between two frames.
3. *Positioning*: to define the object's position in space. The object's center coordinates in the world coordinate system must be defined and the object's orientation for non-point-symmetrical bodies must be determined.

In tracking mode, image processing could determine the position of the object in the current image more quickly if data collected from the previous images is used. A real TbT system may include a number of additional tasks (e.g. throwing objects towards several destinations, catching objects from several sources, transporting various objects through one route). At this time there is no common methodology for constructing such complicated TbT systems. Only once a functioning model for a simple TbT route has been created would it be useful to develop the methodology for more complicated TbT systems.

1.3 Challenge of trajectory prediction

The task of trajectory prediction for hard catching in the situation considered in [Bar08] would have the following state: to determine the values in y- and z-direction when $x = x_{grasp}$ (notations fromfigure 1.2 are used). At the moment of gripping, the catching device must be fixed at the point $x_{grasp}, y_{grasp}, z_{grasp}$ and wait for the object. It does not adjust its motion to the object velocity and orientation, so prediction of these parameters is not necessary for successful hard catching. In related research from the TbT project, the challenge of object orientation prediction is poorly considered because most of thrown objects are point-symmetrical. Predicting the time of an object's interception with x_{grasp}-plane should be as accurate as needed to correctly position the catching device before this interception.

Soft catching requires more complex trajectory prediction as the velocity and the direction of the gripper's movement must be close to the velocity and direction of the object's movement at the moment of catching. Due to the construction of the catching device, the catch must be within the grasping area (GA) where the gripper can move. To determine the instructions for the gripper, the system must know the values of the following parameters of object motion through the trajectory in the gripping area (GAT):

- *Time ranks of object motion through the gripping area*: the moment it arrives in the gripping area t_{in} and the moment when it leaves gripping area if there is no catch t_{out};
- *The dynamics of object center coordinates* $x_1(t), x_2(t), x_3(t)$ for $t_{in} < t < t_{out}$;
- *The dynamics of object velocity* $\mathbf{v}(t)$ for $t_{in} < t < t_{out}$;
- *The dynamics of object orientation for complex-shaped bodies.*

The task is to estimate the most probable values of these parameters. Note that more or less accurate information about the velocity can be derived from the coordinate/time dependence. The calculation is based on the previous trajectory of the object. Three approaches on the use of trajectory prediction for robotic catching exist in related works:

1. *Accurate adjustment of throwing device* [Fra12]: No prediction is used at all.
2. *Catching in real-time* [Ish96, Nam99, Ima04, Fur06]:The gripper moves closer and closer to the actual position of the flying object. No prediction is used at all or very simple models are used.
3. *Long-term prediction* [Hov91, Fre01, Pon09, Bir10, Bae11].

Essentially, the throwing device is adjusted to throw the object towards the destination point with a constant velocity and at an angle to the horizon. The differences between various trajectories are caused by deviations in initial velocity and the direction of throwing and by unpredictable interactions with the air. For example, a tennis ball thrown by a sportsman's rocket is dynamically unstable at the beginning of flight, and its trajectory can vary from the nominal shape. However, after flying through approximately ten times its own diameter, it becomes dynamically stable, and its flight may be more or less accurately estimated by the aerodynamic model [Coo00].

Real-time catching is used when the robot is able to operate with a velocity comparable to the velocity of the object. For these catching strategies, the use of a high speed sensing system is required. For example, in [Ish96] a specific vision system that can react with a feedback time of 1 ms is proposed and is later applied for robotic catching in [Nam03a]. This strategy demonstrated its validity in the experiments, but if the throw's distance is long and the robot cannot react in time, then it is useless.

Algorithms for predicting the trajectory of a flying object were developed in a number of works, e.g. [Hov91, Fre01, Bar08, Bir11, Kim12]. Successful catches were up to 80 percent in [Bir11]. Mostly analytical models were used. A more detailed discussion of these approaches is in section 2.5.

The basic idea of forecasting the trajectory of a flying body is to use the physics of ballistic flight. To determine the physical model of the projectile's motion, the forces influence on the object should be determined. There are a number of such forces [Kar54, pp. 68-108]. The most influential of which are shortly described below:

1. *Gravity*: It is directed downwards and can be calculated using the well-known formula:

   $$F_{grav} = m * g, \tag{1.1}$$

   where m is the mass of the object (obviously, it is a constant for the objects of a same type), g is free-fall acceleration (it is constant for geographical region and has no large deviations on the Earth). All of the following forces are caused by the interaction between the object and the air.

2. *Pressure drag or skin friction*: If the flight is subsonic and takes place in the Earth's troposphere, the value of drag can be considered as proportional to the second power of the object velocity v

   $$F_{drag} = k * v^2 \tag{1.2}$$

The coefficient k The coefficient k here depends on a number of parameters: object size, shape and orientation, the Reynolds number, temperature and humidity of air, etc. The structure of this coefficient is discussed in more detail in the subsection 2.1.1. The coefficient k k can only be calculated analytically for very simple objects (smooth sphere, smooth parallelepiped, etc.) while it is usually measured in aerodynamic tubes for bodies with a more complicated shape.

3. *Lift, side force and inductive drag*: These forces are connected with the nonsymmetrical streamlining of a nonsymmetrical body. Due to this effect, pressure on the different points of the flying body varies. This difference causes the specific force that produces deviation of the trajectory. An airplane's flight is based on these deviations. In aeronautics the different components are differentiated as a lift component (directed upwards), a side component (side force, which is directed to the side) and a drag component or inductive drag (directed backwards), but the parent of these forces is the same. An "upwards-backwards" representation is sufficient for an airplane wing, but it can be in different directions for arbitrary shapes.

4. *Trail drag*: This force is connected with the influence of the vortex street after the flying blunt body. The air after the object curves into a number of vortices which increase the resistive influence of air on a flying object.

5. *The Magnus effect*: This effect takes place if the flying object is rotating. It may have upwards, downwards, backwards and sideways components.

6. *Other streamlining effects*: For compact objects the influence of the force connected with air flow near the moving object surface (specific cases of these force are lift, inductive drag, trail drag and the Magnus effect) can be ignored. For simple bodies in special conditions, it can be calculated using special equations or measurement results. Otherwise it is not possible to accurately determine the influence of these effects.

7. *Influence of wind and external air flow*: If there is a constant air stream, its influence could be considered. The influence of local air fluctuations is unpredictable and could be considered as stochastic deviations.

The creation of a precise deterministic model of object flight seems to be difficult or computationally impossible for objects of a variety of shapes. In most related works, only the influence of gravity [Hov91, Her09] and air drag [Fre01, Bar09] is considered. Other factors are considered to be part of the process noise. However, it can be said that physical rules of projectile motion exist, and under the same entry conditions, two trajectories would not vary strongly. The phrase ?same entry conditions" means that the type of object, its starting point, its initial velocity and the direction of throw would be exactly the same and there would be no significant difference in air flow. This influence of air on projectile motion is difficult but necessary to calculate and motivates the search for a new method to predict the trajectory. This method should take the complexity of these aerodynamic forces into account but not require the exact calculation of their values.

Most of the existing predictive robotic catchers use motion models that take gravity and drag into account (e.g. [Fre01, Bar09]). The rate of successful catches in these systems varies from 66% [Fre01] to 80 % [Hov91, Bae09] of success in catching.

1.4 Learning-based prediction

Most humans are good at intuitively predicting ballistic trajectory. The ability of human brain to solve this task has been examined in a number of related sports, such as tennis, badminton, baseball, ping pong, volleyball, squash, etc. The players in these sports must quickly estimate the destination point of the flying ball. The only input for the player is the flight of the ball that they watch with their eyes. This is not stereo vision: the distance from the ball is much longer than the eyes? baseline; hence, there is no visible difference in object position from the left and right eye. In fact, the decision is made based on the elevation angle of the flying object [Gil99]. This task is similar to the task of vision-based robotic ball-catching.

Children are usually able to play sports at a relatively early age. Beginning in primary school, most children are able to play badminton and volleyball. This is much earlier than when they begin to study physics. Even when they do begin lessons in physics, the aerodynamic properties of the projectiles are not usually a part of the physics curriculum. This knowledge is not necessary to catch the ball successfully. Humans learn to predict ball trajectories by memorizing previous examples. This intuitive process that helps humans to catch a ball is known as the "ball-catching theorem". This prediction principle was used for bio-inspired prediction in other fields of research, predicting electrical power consumptions for example [Mao10].

In the previous sections, the following directions for Transport-by-Throwing (TbT) system development were specified: the throwing techniques, the measurement of trajectory, the determination of an interception point, the defining of the gripper actions, and the further construction of transportation networks. The determination of an interception point (based on the prediction of object GAT) is necessary for a functioning TbT route. Deviations of the throwing device and in the environment make it impossible to for every object to fly with an identical trajectory; therefore, the gripper must act to catch the object successfully. The gripper?s actions are based on prediction results. In turn, prediction is based on visual tracking. The visual tracking of movement (including flying) objects has been researched in many scientific papers. The application of a tracker and the processing of coordinate extraction for TbT was considered in Akhter's dissertation [Akh11]. Hardware and software for tracking are chosen in such a way that they could give accurate coordinate information to the input of a predictor.

This work when compared to current research primarily concentrates on trajectory prediction and can be divided into the following three parts:

1. *Analyzing the structure and accuracy of the tracking information*: The input of the prediction system includes the measured coordinates of the object; hence, an accurate observation system is needed for accurate prediction. The determination of the 3D object?s coordinates is made based on images from the cameras. This determination has known errors due to the inaccuracy of the recognition algorithm. It is for this reason that the mechanism of trajectory estimation is to be implemented. The challenge here is that the exact dependence between time and coordinates stays unknown and it cannot be approximated with exact and simple function.

2. *Development of the trajectory prediction algorithm*: The motivation for this task is necessity of making an accurate prediction for the TbT route to function correctly. Up until now, predictors based on a physical model were used. However, as demonstrated above, physical models at their current stage prediction accuracy are restricted. Hence, an alternative forecasting technique is needed.

3. *The construction of a system that would allow these techniques to be implemented*: i.e. integrating the algorithm into a working system for robotic catching. To correctly evaluate the quality of the prediction, the whole route needs to be implemented because the predictor must interact with other parts of the system.

The main idea of the proposed method for trajectory prediction is the use of sample-based forecasting models. In these models, the measured parameters of previous trajectories (the dynamics of the object's coordinates in the measuring and catching areas) are stored in the database. As the process of flight obeys physical rules, the information collected from previous flights can be used to construct the predictor. The construction process is known as "learning by example". This is similar to how the humans learn to perform various action (e.g. how to catch a ball). Learning by example is the most widespread method of machine learning.

The learning task can be more clearly defined in the following way. Let $\mathbf{X}$ be a set of possible input data for the task, $\mathbf{Y}$ be a set of possible outputs for the solver. $y^* : \mathbf{X} \to \mathbf{Y}$ is unknown target correspondence of $\mathbf{Y}$ from $\mathbf{X}$. The values of this correspondence are known only for finite learning sampling consisting of m samples $\mathbf{L} = \mathbf{X}^M = \{(\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Y}_2), \ldots, (\mathbf{X}_m, \mathbf{Y}_m)\}$. It is needed to create the algorithm $a : \mathbf{X} \to \mathbf{Y}$ based on LS, which allows $y$to be approximated not only for $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m\}$ but for any possible samples from the set $\mathbf{X}$. This means that the learning algorithm must have the ability to construct common knowledge (dependence) from individual knowledge (examples). In the trajectory prediction task, the part of the trajectory in the measurement area is $\mathbf{X}$ and the part of trajectory in the catching area is $\mathbf{Y}$.

Due to the nature of $\mathbf{Y}$ two widespread types of learning tasks are specified:

- Classification: the set of possible $\mathbf{Y}$ is finite;
- Regression: $\mathbf{Y}$ is numerical function (set of possible $\mathbf{Y}$ is infinite).

This special type of regression task is time series forecasting for when we need to predict the future values of a certain function based on the previously observed values of its function. It is obvious that projectile trajectory prediction refers to the time series forecasting tasks. For the case of trajectory prediction task $\mathbf{X}$ is the initial part of trajectory, measured by the observer, $\mathbf{Y}$ is the part of trajectory in catching area, which must be predicted. Commonly this means $\mathbf{X}$ is a sequence of measurements of object coordinates with timing marks (and also velocity and orientation if this is possible and useful), and $\mathbf{Y}$ is an expected dependence of object coordinates on time in catching area.

The prediction environment can be divided into two parts: predictor and learning algorithm. The predictor?s task is to determine $\mathbf{Y}$ from $\mathbf{X}$ using known prediction model a.the task is the development of a learning algorithm, i.e. the algorithm for the construction of a prediction model. This means the creation of an algorithm a based on a set $\mathbf{X}^M$.

The methodological steps of the research are as specified in the following:

1. *An analysis of observer performance and accuracy*: This is based on the throwing and measurement experiments and aims to determine what data are used as the predictor's input and what the likelihood of this data is.

2. *The development of an algorithm for trajectory prediction based on machine learning and the learning algorithm*: These algorithms are developed in parallel as they are strongly related on each other.

3. *The evaluation of the accuracy of the constructed model*: This evaluation consists of two stages. In the first stage, the simulation of projectile flight is used. The simulation setup is defined based on the physics of the flying body, the accuracy of observation system used, and the deviations of the throwing devices used. In the second stage, a large database of the real trajectories of the object is constructed. Examples from this database are used for training and validating the predictor.

4. *The integration of the proposed prediction module into the existing transport-by-throwing system*: This includes the evaluation of the speed of the constructed model and the creation of its real-time version.

5. *The final evaluation of the whole transport-by-throwing route with the integrated learning-based prediction algorithm*: This step includes the catching experiments and the establishment of a successful catching percentage. For these experiments, the workable model of predictor is created. Within these experiments, the developed algorithm be compared to other possible solutions. The proposed approach for trajectory clustering and processing will also be evaluated.

The development and validation of the proposed ideas are implemented with a tennis ball as the object to be thrown. This is because a tennis ball is a well-known object. As shown in subsection 2.1.3 the aerodynamic properties of the tennis ball are explored in a number of scientific works [Ach72, Ste88, Cha00, Coo00, Meh08]. Most existing robotic catchers are also intended for catching small-sized sport balls, e.g. [Slo91, Fre01, Nam03a, Bar08, Bir11]. Tennis balls are well studied aerodynamic objects with not very complicated shape. Physical models of such bodies flight allow to predict trajectory accurately (used in e.g. [Pon09]) and can be used for proving the accuracy of other algorithms. Therefore use of the tennis ball is likely in terms of comparison of the proposed ideas with the existing solutions.

2 State of the art

As previously mentioned, Transport-by-Throwing (TbT) is a concept for the material transportation networks in industry. Since the introduction of this approach in 2006 [Fra06], a number of works published about the project have brought it to its current level and many of the main challenges have been are solved. However, the task of catching the flying object with a robot manipulator was researched in a number of works prior to the introduction of TbT. In these related works, the task of catching was mostly just a subfield of theoretical robotics focused on understanding and improving the possibilities of robotic systems. The task of trajectory prediction belongs to time series forecasting tasks. Time series forecasting is a wide field of research in applied mathematics. Various models have been created in this field. Some are based on known dependences, some on the statistical properties of data, while others are based on analogies and learning.

The aim of this chapter is to analyze the existing methods. Section 2.1 analyzes and summarizes the aerodynamic properties of thrown objects. The next two sections discuss certain aspects of estimating the trajectory from measurements. Section 2.2 considers the situation when we already know the measured coordinates of the flying body depending on time. The ways of how to fit this data into the process model (i.e. how to filter out erroneous measurements and improve measurement accuracy) are mainly discussed here. In section 2.3 , the technical and algorithmic issues of observing ballistic trajectories and extracting 3D coordinates from sensor data are considered. Various technologies that allow the observation and tracking of a moving target are discussed. The aim of section 2.4 is to give an overview of recent developments in the systems for robotic throwing and catching and related tasks (i.e. tracking airborne objects and forecasting the trajectories). This overview includes research from, but not limited to, TbT. Some Prediction aspects are briefly discussed in this section as issues of catching movements and grasping principles. A more detailed review of solving the challenges involved in prediction in these works is given in section 2.5 as prediction is the main topic of this thesis.

2.1 Aerodynamics of ballistic motion

Determining the influence of aerodynamic forces has been mentioned in works that research the motion of specific objects when it is motivated by applications. The most well-investigated types of ballistic flight are bullets and projectiles in motion and the flight of gliders. Other objects are less investigated. Among the objects with properties similar to TbT, the behavior of small sports balls (used for tennis, table tennis, golf, etc.) is more or less well-defined. Here, TbT objects mean rigid, compact bodies of up to ten centimeters in size that do not have high wintage (e.g. a sheet of paper is not an acceptable TbT object).

2.1.1 Common view

The main forces that influence a flying body are listed in section 1.3Newton began the exploration of the behavior of flying bodies; he discovered the law of gravity, proposed the first models for air drag (these models were then rejected [Kar54, p. 61]) and observed that the trajectory of a rotating object is curved. Nowadays, these three effects are still considered to be the most influential on a flying decimeter-sized body: gravity, air drag and the Magnus force. Most of the papers on the aerodynamics of sport balls (see subsection 2.1.3) described below take these three forces into account. In some papers (e.g. [Ala98]), three aerodynamic forces are considered: drag, lift and side force. In these papers "drag" is air resistance directed downwards, while "lift" is the component directed upwards, and "side force" is the component perpendicular to the horizontal projection of the velocity. "Lift" here is connected with the Magnus effect and is of another nature than that of the lift on plane wings. The main source of side force is also primarily the Magnus effect.

To acquire a fundamental understanding of drag requires that details of fluids mechanics and how flying objects interact with the air around them be discussed. The fluid area near the flying body where air flow is influenced by the body is called the boundary layer. The fluid motion around an object may be laminar or turbulent (figure 2.1). In the laminar mode, the space around the object may be divided into a set of virtual "tubes" which correspond to a certain sublayer. "Sublayer" means the specific velocity of fluid motion and its movement only inside itself. These sublayers curve around the object without intersecting one another. When the flow is turbulent the object destroys the structure of the sublayers, which then intersect, and the fluid motion becomes chaotic.

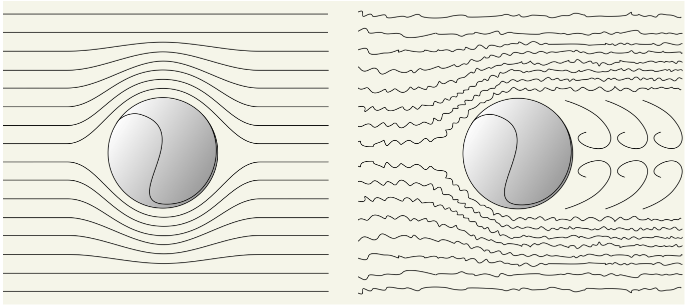


Figure 2.1: Laminar (left) and turbulent (right) air flow around the spherical object moving in the air.

In turbulent mode, the air behind the object curves into a set of vortices, the so-called Karman vortex street [Kar54, p. 101]. As previously mentioned, the vortex trail is an additional source of the drag. However, the viscosity of the air plays a limited role after the transition; hence, the value of drag decreases with the transition from laminar to turbulent flow [Gof13, Kar54].

The motion of fluid in the boundary layer may be characterized by the dimensionless Reynolds number, Re, which is defined as follows:

$$Re = \frac{v * D}{\nu} = \frac{v * D * \rho}{\mu}, \tag{2.1}$$

where v is the speed of the object, D is a characteristic length (diameter for a spherical objects), and ν is the kinematic viscosity which is calculated from viscosity μ and air densityρ. For spherical objects of a known shape and size in the air with a known temperature and humidity, Re depends only on the velocity of the object. When the Reynolds number is low, the flow is laminar. At higher numbers, it becomes turbulent. The transition from laminar to turbulent flow (and the corresponding drop in the value of air resistance) at a certain value range of Re is called ”drag crisis” [Gof13, p. 141]. The transition is not an instantaneous process. The range of turbulence is connected with the growth of a turbulent area after the flying body).

If the flow near the object is laminar, its influence is defined by air viscosity. For flying objects bigger than several micrometers, the flow would be turbulent, and viscosity can be ignored [Kar54, p.100]. The turbulence of air flow near standard flying balls was proven in experiments, e.g. by [Ala10]. To calculate drag in this case, the following formula, called the drag equation, should be used:

$$F_{drag} = \frac{1}{2} S C_d \rho v^n. \tag{2.2}$$

Here v is object velocity, ρ is air density, C_d is drag coefficient depending on the shape (e.g. for sphere $C_d = 0,47$), S is a parameter, characterizing frontal square of the body; for oblong bodies it may be expressed as:

$$S = V^{\frac{2}{3}}, \tag{2.3}$$

where V is object volume. For the sphere characteristic square is equal to the square of its cross section:

$$S = \pi * R^2, \tag{2.4}$$

where R is the radius of the sphere. The value of n depends of flight conditions (e.g. for flight in the air at a supersonic speed $n = 1$, for ships in water $n = 3/4$). For an object flying through the air at subsonic and lower velocities, the pressure drag is proportional to the velocity square, i.e. $n = 2$ [1].

As it is more easily understandable, the equation 2.2 is replaced by this:

$$F_{drag} = k * v^2, \tag{2.5}$$

where the coefficient k is defined as a product of coefficients from 2.2:

$$k = \frac{1}{2} * S * C_d * \rho. \tag{2.6}$$

In addition to k and C_d the so called ballistic coefficient C_b may be used to characterize the drag properties of rigid bodies. For oblong projectiles it may be calculated using the following equation [Wei80, p.722]:

$$C_b = \frac{m}{C_d * S} = \frac{\rho * l}{C_d}, \tag{2.7}$$

where m is the mass of the body, C_d is the drag coefficient, S is the effective square of the body, ρ is density of the body and l is its length. The ballistic coefficient corresponds to the ability of the object to keep its velocity during flight. It is used to characterize the aerodynamic properties of the object e.g. in [Wei80, Far02].

The motion of the body only under the influence of gravity may be expressed by the following differential equation:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}'' = \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix}, \tag{2.8}$$

where x_1, x_2, x_3 are object coordinates in a coordinate system where x_1 is collinear to gravity direction. If the launching velocity $v(0) = \begin{bmatrix} v_1(0) \\ v_2(0) \\ v_3(0) \end{bmatrix}$ equation 2.8 will have the following solution:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} v_1(0) \\ v_2(0) \\ v_3(0) \end{bmatrix} * t + \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix} * \frac{t^2}{2}, \tag{2.9}$$

where t is the time period after the launch. This model of motion represents the object?s motion in a vacuum. For flight in the air it is not exact; however, it was successfully applied for predicting projectile trajectories e.g. [Hov91, Her09]. Inserting the influence of drag into 2.8 will lead to the following differential equation:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}'' = \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix} - \frac{1}{2} * S * C_d * \rho * \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}' * \sqrt{(x_1'(t))^2 + (x_2'(t))^2 + +(x_3'(t))^2}. \tag{2.10}$$

The motion model based on this equation is one of the most popular existing solutions in robotic catching, e.g [Fre01, Bar08, Bir11]. Unlike 2.8 it has no analytical solution, which means it is usually solved by numerical methods in practice, e.g. the Runge-Kutta method in [Bar08].

The Magnus effect appears when the object is rotating. For spherical objects it may be expressed as a similar form of drag [Gof13, p. 138]::

$$F_m = \frac{1}{2} * SC_l^* * \rho * v^2. \tag{2.11}$$

Here C_l^* is a dynamic coefficient dependent on the spin of the object, and all other notations are equal to the drag equation. The dependence of C_l^* on spin is not very useful. Thus, it is sometimes better to use another version of the equation [Gof13, p. 139]:

$$F_m = \frac{1}{2} * SC_l * \rho * r * \omega * v. \tag{2.12}$$

where ω is the spin value, r is the radius of the sphere, C_l is the static Magnus force coefficient. C_l and C_l^* are connected by the following equation [Gof13, p. 138]:

$$C_l^* = \frac{C_l * v}{r * \omega}. \tag{2.13}$$

The direction of the Magnus force is collinear with the vector $\omega \times v$ [Gof13, p. 139]. It is sometimes considered to be a component of Magnus drag (directed backwards), Magnus lift (directed upwards) and Magnus side force (directed sideways). The existing throwing devices allow objects to be thrown linearly. The Magnus effect may be ignored with these devices. Plane fitting to object positions as described in the subsection 4.2.3 showed that although the flying ball is rotating, no curving effect of the Magnus side force was detected.

2.1.2 Arm ballistics and aerodynamics of gliders

The flight of bullets and projectiles is studied by the projectile ballistic. The trajectories of shot projectiles were studied earlier than the trajectories of other thrown bodies; the first studies were done in the 16th century by Niccolo Tartaglia. It was in his work that projectile flight lead curve was investigated and did not consist of two straight lines. Since then our knowledge of the properties of flying projectiles has greatly increased. A large part of common ballistics has come from studying arm projectiles and their trajectories. The development of indirect fire and then the use of ballistic rockets in the 20th century has increased our knowledge of ballistic curves. However, modern arm ballistics are only slightly useful for modeling the flight of objects in throw-catch routes for several reasons:

1. The projectile is thrown from the gun at supersonic velocities of more than $500m/s$. The physics of supersonic motion are completely different from those of subsonic motion (e.g. the air drag is proportional to the first power of velocity instead of the second) [Kar54, pp. 108-147].
2. The shape of the projectile is constructed in such a way that it has a stable trajectory. In the TbT application, the shape of the object can vary.
3. The high accuracy of projectile impact is derived from the high accuracy of the gun provided by its specific design.
4. Applied gun ballistics does not take the observation of flying projectile into account when defining trajectory. It is common to allow the development of a probabilistic model of the impact point based on the accuracy of the gun. Nonetheless, forecasts in robotic catching need to be exact.

At any rate ballistic rockets and their tracking are a well-investigated area of research, and some aspects that may be particularly useful for TbT are discussed in section 2.2.

One specific field of research is connected with the ballistics of arrows and javelins. Today arrows and javelins are used in sports rather than as weaponry. Unlike bullets and rockets, arrows and javelins fly at subsonic velocities. The classic sport of archery is direct in that the goal is to hit the target with an arrow. The trajectory is not considered to be a curve but rather a straight line. Research in archery concentrates on defining the drag coefficients at various velocities and the critical Re for arrows. The standard velocity of an arrow is about 60 m/s [Gof13, p. 143]. Hence, the flight of an arrow is not very similar to the flight of a TbT object.

In comparison, sport javelins are thrown indirectly at an elevation of about 30 degrees. Only if the javelin touches the gound with the tip first, are the resulted counted [3]. The javelin is thrown at a velocity of around 30 m/s. The best sportsmen are able to throw it up to 100 meters [3]. A sports javelin is 2.2-2.7 meters long and has a mass of around 0.6-0.8 kg [Gof13, p. 143], which means it is bigger than TbT objects. With a maximum shaft diameter of 2.5-3 cm, it is an oblong object. The javelin's flight is tied to the pitching moment as the center of mass is not the same as the center of pressure [Hub87]. Also, the javelin vibrates during flight which influences the drag [Hub89]. These factors show the dissimilarities between a javelin and a compact object for use in transportation by throwing.

The ballistic flight of planes and gliders is also a well-investigated area of aerodynamics. The development of such machines began with the gliders proposed and constructed by George Cayley in 1809-1853. from 1809-1853. Modern gliders vary from very small objects that weigh several kilograms to orbital reusable launchers (Space Shuttles and Buran Spacecrafts), which use the glider principle for landing. Gliders and planes are different from everyday objects that can be transported by throwing and catching in the following ways:

1. The planes are constructed in such a way that they have high lift. The main influence (of what) is connected to specifically-shaped wings to achieve a high value of lift. Normally TbT objects have no wings and fly through the air without strong lift.
2. A plane is large, meaning the air flow near the flying plane is completely different from that of small objects.
3. Gliders use thermals to prolong their flights. In industrial environments thermals play no role.
4. Planes have engines on board while a thrown object has no impulse income during flight.
5. The velocities of planes are usually much higher and often supersonic.

2.1.3 Research on the aerodynamics of sport balls and similar objects

An exploration of smaller and more commonplace objects is also taking place as the need arises in application. A large part of this research revolves around sports. Aerodynamic conditions similar to transport-by-throwing can be found in sports that involve throwing. This is because thrown objects have similar characteristics, e.g. balls used in tennis, table tennis, golf, baseball, squash, etc. These studies are done to define the significant parameters of the ball that allow an increase in the quality of produced balls and to examine the claims made by the companies selling the balls [Ala10]. An exploration of the aerodynamic forces influencing the behavior of the balls can be done by observing them as they are freely thrown into a wind tunnel [Dav49] or by accurately measuring the forces with the help of mounting equipment (if the object in the wind tunnel is mounted) [Ala10].

Research on the flight of sports balls is usually done in conditions similar to those that would occur during a game. These conditions may differ dramatically from TbT conditions. For example, during a game a tennis ball flies at a velocity of several tens of meters per second and for a distance of more than twenty meters. Golf balls fly even further and at higher velocities. Also sports balls are often thrown with spin; thus, the Magnus effect must be taken into account.

The exploration of tennis ball aerodynamics began with Newton in 1672 and continued in a number of scientific works. A summary of these works is given in [Meh08]. The study of the aerodynamics of tennis balls has led to a number of results:

- The values of the drag and lift coefficients were estimated empirically with respect to spin and for zero spin [Ste88, Meh08]. It was found that the drag coefficient of a rotating ball increases due to the centrifugal extension of the fuzzy covering. The air drag coefficient usually varies from 0.55 to 0.65 [Meh08].
- It was shown that the Reynolds number does not have a significant influence on the drag coefficient of the flying ball [Ste88, Meh08].

- A tennis ball quickly reaches a semi-steady aerodynamic state after leaving the rocket: at the distance of approximately 10 of its diameters [Coo00].
- The fuzz on the ball increases the drag coefficient in comparison to that of a smooth sphere [Cha00].
- The seam on the ball does not significantly affect its trajectory [Meh08].
- Increasing the size of the ball does not increase the drag coefficient very much. The increment of drag force is proportional to the increment of effective square as it is defined by the equation 2.2 [Meh08].
- If the ball is worn its drag coefficient is lower [Goo00].
- The influence of viscosity is approximately 50 times less than the influence of pressure drag and can be ignored [Ach72, Meh08].
- Lift force connected with the Magnus effect appears when the ball is rotating as it is assumed by the theory in [Meh08].

Table tennis balls have not been studied to this extent. An experimental exploration of table tennis balls as aerodynamic objects is considered in [Non10]. The ball was thrown from the catapult at a velocity from 6.2 to 6.5 m/s. The distance of flight in such a setup is about 1.6-1.8 meters which corresponds to the size of the table tennis field. The motion of the ball was observed by a complex set of visual sensors in two areas: the throwing area and the landing area. To examine the properties of the object's flight in the throwing area, high-speed cameras (900 fps achieved) were used. The landing process was recorded by cameras at a lower speed (150 fps). Experiments have shown that the mathematical model, which does not take aerodynamic forces into account, is inapplicable for predicting trajectory. The distances between the calculated and measured positions were up to several tens of centimeters.

For modeling the trajectory with respect to air drag and the Magnus effect, the following equation was used:

$$m * \mathbf{p} = m * \mathbf{g} - \frac{1}{2} * C_d * \rho * A * ||\dot{\mathbf{p}}|| * \dot{\mathbf{p}} + \frac{4}{3} * C_m * \pi * \rho * r^3 * (\omega \times \dot{\mathbf{p}}), \tag{2.14}$$

where m is mass, g is acceleration of gravity, ρ is air density, A is projected area, r is radius, C_D: drag coefficient, C_M is lift coefficient, p is position of the ball, ω is rotational velocity.

The first part of this formula corresponds to the influence of gravity, the second corresponds to air drag and the third model demonstrates the influence of rotational lift caused by the Magnus effect. It was shown that the rotational velocity of the flying ball is more or less stable during flight. The model used showed an average error of 4 cm and 9 cm, both when back spin occurred. In [Non10] this level of accuracy was considered to be sufficient for robotic table tennis, and more precise models were not developed.

A golf ball is dimpled and made of metal. When compared to a tennis ball, it is heavier and has no fuzzy outer layer. The study of golf ball aerodynamics has a long history. It was carefully considered in [Dav49], where the results obtained in an aerodynamic tube are presented. Newer results for golf ball aerodynamics are presented in [Ala10]. These papers primarily study the influence of spin on the object?s trajectory. The results prove similar to the trends showed in the tennis ball exploration. The golf ball has aerodynamic properties similar to the properties of a smooth sphere, but they are not exactly the same. While a tennis ball differs from a smooth sphere due to the fuzz on its surface [Cha00, Coo00] and its elasticity [Coo08], the golf ball differs from a smooth sphere because of its dimples [Ala10].

Studies on the aerodynamics of baseballs also exist. For instance, the dissertation [Ala98] studies the aerodynamic properties of a rotating baseball. An analysis of the experimental data shows that various experiments ([Wat87], [Bri59]) done to define the correspondence between the rotational velocity of the ball and the value of "lift" (i.e. the force related to the Magnus effect) give different results, arguing with each other [Ala98, p. 32]. In experiments conducted in [Ala98], a set of cameras was used to observe the trajectory. Afterwards in offline processing of the video data, the aerodynamic properties of the ball were investigated.

Larger sports balls have also been studied with respect to their aerodynamics: soccer [Asa07, Bar09a, Gof09], basketball [Qin13], volleyball [Asa10], and rugby [Ala08], for example. They are less comparable to potential TbT objects, especially at its current stage of development, due to their size and lower full density, so they will not be spoken about in great detail. These works consider:

- defining the critical Reynolds number [Asa07],
- measuring and calculating the values of the drag coefficient [Asa07, Bar09a, Gof09, Ala10],
- measuring and calculating the values of the Magnus force [Asa07, Gof09, Ala08]
- determining the influence of different seam orientations on velocity direction [Bar09a],
- visualizing the air flow near the flying sphere (a titanium tetrachloride suspension was used and the flight was observed with 4500 fps cameras in [Asa07]),
- comparing the results with smooth spheres [Asa07, Bar09a],
- determining the dependence of aerodynamic parameters on the yaw angle (for rugby balls) [Ala08].

The visualization of air flow in [Asa07] allows the behavior of the Karman vortex street to be defined. These results seem relevant but are highly dependent on the linear size of the object. This method could potentially be used for defining its influence in the future.

An exploration of rugby balls holds additional interest as they are not spherical. However, experiments in [Ala08] are simulated with some shaky assumptions, e.g. a rugby ball was defined as a paraboloid, which is not entirely correct. Also, a rugby ball is too large to be considered at the current stage of TbT development. In [Ala08] it is shown that drag coefficient increases with an increasing yaw angle. The plot of the drag coefficient with respect to drag looks like a sinusoid with the minimum at 0 angle and the maxima at yaw angles of $\frac{\pi}{2}$ and $-\frac{\pi}{2}$.

With the exception of balls used in sports, research on the aerodynamics of non-sports-related objects is done as the need for it appears. For instance, [Luc87] examined the subsonic velocities of parallelepiped-shaped bodies. The influence of the Reynolds number, the angle of attack and the roll-angle was considered. The motivating factor for the use of parallelepiped-shaped containers was because of their potential use in the transportation of radioactive materials in space missions. The static stability tests (the object was mounted in the wind tunnel) and dynamic stability tests (object is mounted but can oscillate) were done. The results are as follows:

- The Reynolds number had little effect, and critical values were not achieved during the experiments with subsonic velocities in static tests.
- An incrementation of the Reynolds number decreases the dynamic stability of the object; it starts to oscillate chaotically.
- A decrease of drag coefficients with an increase of chamfer (edge roundness) was proven.
- An increase in edge roundness increases the drag and the Magnus force for an angle of attack of up to 60 degrees. At higher angles, the effect on drag is the opposite.
- An increase in edge roundness allows drag crisis to move to higher values of *Re*.

There are, of course, other projectiles besides balls in sports. In wind-tunnel experiments, the "drag", "lift" and "side force" coefficients were measured for a discus, a hockey puck and a frisbee [Gof13]. The shuttlecocks used in badminton have also been investigated in sport aerodynamics [Pos09]. Traditional shuttlecocks were made of goose feathers; modern shuttlecocks are plastic. Badminton shuttlecocks are lightweight and have higher windage than tennis balls. In professional competitions, it is thrown at a velocity of up to 70-115 m/s [Pos09,4]- higher than that of any other sports projectile. The final fall velocity of a shuttlecock is around 6,8 m/s [Pos09, Che09]. After being hit by a racket, a shuttlecock will usually completely change its direction of flight after traveling a distance of 20 to 80 cm [Pos09]. With this change, it loses about half of its velocity. The drag coefficients for synthetic shuttlecocks are usually lower than that of feathered [Coo96]. In [Coo99] the significant change of drag coefficient on Re values from 13000 to 200000 was not recognized (for spherical balls such values are over critical threshold).

An analysis of the aerodynamics of small-sized, rigid objects proves the statement made in chapter 1: rules of motion exist, but they are difficult to estimate. Flight under the influence of gravity and aerodynamic forces is more or less a deterministic process that takes place with respect to physical rules. The careful and complex study of the flight properties of a simple-shaped body allow these rules to be defined and the equation coefficients to be calculated. However, this presents unknown issues and effects that can only be detected through experiments. Several experiments in similar conditions (e.g. [Wat87] and [Ala98, p. 104]) have led to various and sometimes mutually exclusive results. Hence, the creation of physical models on which the experiments are to be based does not guarantee that the accuracy of prediction will suffice for transportation-by-throwing applications.

The most significant aerodynamic forces for sport balls are air drag and the Magnus effect. Finding out the influence of spin is more complicated than determining the influence of drag. In most papers on the aerodynamics of sports balls, the balls typically fly with spin. In industrial throwing, it is possible to throw an object with zero spin and consider the flight free from the Magnus effect. In the subsection 4.2.3, it is shown that throwing experiments with a linear throwing device conducted at Vienna University of Technology did not demonstrate significant side force connected with the Magnus effect.

An analysis of the ball aerodynamic models in terms of their applicability in TbT was made in [Pon09]. Three models were analyzed:

1. *Polynomial model*: The motion of the object in each dimension is expressed by a second order polynom:

$$\begin{aligned} x_1 t &= A_1 + B_1 * t + C_1 * t^2, \\ x_2 t &= A_2 + B_2 * t + C_2 * t^2, \\ x_3 t &= A_3 + B_3 * t + C_3 * t^2, \end{aligned} \tag{2.15}$$

where $A_1, A_2, A_3, B_1, B_2, B_3, C_1, C_2, C_3$ are weighting coefficients, which are estimated based on available trajectory measurements. This is the only model of the three that takes the Magnus effect into account [Pon09, p.41].

2. *Gravity-drag model*: Solves the equation 2.10 by iterative calculation. The dependence of the object coordinate on time has no functional description; therefore, fitting the model to data is made by the Monte Carlo generation of flight parameters (initial position and initial velocity vectors, drag factor $k = \frac{1}{2} S C_d \rho$, gravity vector) [Pon09, p.42].

3. *Spatial-separated model*: Model with an assumption about the independence of drag components in various spatial dimensions. Under this assumption, the vector differential equation 2.10 is replaced by a set of three scalar differential equations:

$$\begin{aligned} x_1''(t) =& -k * x_1'(t)^2 + g, \\ x_2''(t) =& -k * x_2'(t)^2, \\ x_3''(t) =& -k * x_3'(t)^2, \end{aligned} \tag{2.16}$$

which has the following solution:

$$\begin{aligned} x_3(t) =& x_1(0) + \frac{1}{k} * \ln \frac{\cosh\left(\sqrt{g * k}(t - t_0)\right)}{\cosh(\sqrt{g * k}(t_0))}, \\ x_2(t) =& x_2(0) + \frac{1}{k} * \ln\left(1 + k * x_2'(0) t\right), \\ x_3(t) =& x_3(0) + \frac{1}{k} * \ln\left(1 + k * x_3'(0) t\right), \end{aligned} \tag{2.17}$$

where $t_0 = \frac{1}{\sqrt{gk}} \arctan(\sqrt{\frac{k}{g}} * x_1'(0))$

A comparison of these methods in [Pon09] demonstrated that the highest accuracy was achieved by the spatial-separated drag model (the upper bands limits for prediction error in 99% of the cases were 44 mm for the physical model, 44 mm for the polynomial model and 29 mm for the spatial-separated physical model [Pon09, p. 61]). [Pon09] showed that the models based on the influence of gravity and air drag are accurate enough to represent and predict the ballistic motion of spherical objects even if the objects are rotating.

EDITED PART STOPPED HERE

2.2 Statistical estimation of ballistic curves

The results discussed in section 2.1 were primarily obtained in wind tunnels using highly precise measuring equipment and well-known calculation mechanisms. When the measurements and calculations are needed for the purposes of tracking and prediction, several additional questions pop up. How can these measurements be taken? How accurate will they be? Is it possible to increase the accuracy by computational means? In section 2.3 the technical and algorithmic issues of observing ballistic trajectories with visual sensors are discussed. This section focuses on the situation where the measured coordinates of the flying object depending on time are already known.

Measurements made by cameras or other sensors are usually erroneous. To decrease the influence of measurement inaccuracy on further calculations, the statistical procedure of estimation is made. The task of estimating the trajectories of projectiles is taken from air defense. The radars used to locate a target moving in the air (including ballistic rockets and missiles) have a certain error of positioning which is minimized by the statistical processing of the data. Mostly estimation is used for tracking in order to define the most probable position of object at the current moment. This is based on a set of current and past measurements. There is no task for the reconstruction of the trajectory curve. The definition of the current physical parameters of the trajectory at the moment may also be included, and these would be velocity, acceleration, spin, etc. These parameters may also be used to predict a future impact point. In [Cha80, Wei80] the estimation done with the (LS) filter was used for angle-only measurements [Cha80], i.e. measuring the angles of incidence with the use of sensors on the earth and Doppler sensors measurements [Wei80]. These works considered the exo-atmospheric flight of projectiles. Hence, drag was ignored, and gravity was a function of height. It showed the applicability of LS, and how the model may be adapted for other models.

In [Cha80, Wei80] the estimation using (LS) filter was used for angle-only measurements [Cha80], i.e. measuring angles of incidence by the sensors on the earth, and Doppler sensors measurements [Wei80]. These works were considering the exo-athmospheric flight of projectiles, hence the drag was neglected and the gravitation was considered as function of height. It has shown applicability of LS and that the model may be adapted for another models.

Use of the Kalman filter was first considered for ballistic tracking in [Jaz70]- nine years after Kalman introduced it [Kal61]. In [Far02] a comparison of the nonlinear filters, e.g. the Extended Kalman filter (EKF), the unscented Kalman filter (UKF), the Particle Filter, and the statistical linearization (CADET), was made. There the task of estimating the radar-observed, reentry trajectory is considered. Although the problem was considered for intercontinental missile defense and aging satellite tracking, the motion model was quite universal. Gravity and drag were considered to be significant forces while other factors were ignored. It was assumed that the ballistic coefficient C_b of the projectile from equation (2.7) is known beforehand. The results of the EKF were better than those of other estimation mechanisms. To validate the statistical accuracy, the Cramer-Rao Lower Bound (CRLB) was derived for ballistic tracking. Estimators were compared within the Monte-Carlo simulation. All four filters showed statistical efficiency. Later in [Ris03], the work was extended to objects with an unknown C_b. In this case the best results came from the UKF.

There are a number of other works that deal with the tracking of ballistic objects with statistical estimation. The following examples are given:

- In [She09] KF estimation based on GPS-positioning is examined. An evaluation of the approach was made using program simulation.
- In [Rav10] impact point prediction is made based on radar measurements. The flight of the object in the atmosphere (i.e. with valid drag force) is studied. The multiple model filter is introduced. The influence of gravity, drag and spin is included in the motion model. The simulation is used to validate results.
- In [Yua12, Yua14] a multiple model approach is also introduced. The thrusting and ballistic parts of trajectory are considered. Hence, the main forces considered are gravity, drag, thrusting acceleration [Yua12], and wind [Yua14]. Each model examines a certain value of

the drag coefficient. The likelihood of each filter is calculated, and the result with the maximum likelihood is taken as an output. The evaluation was done using real measurements data.

- In [Yao08] the KF algorithm is used, and the attenuating memory algorithm is proposed that allows the divergence to be decreased. The results were validated by the simulation.
- In [Kra05] the concept of neural EKF was used (i.e. the neural network trained by EKF algorithm). This model was also validated by simulation.
- In [Zar00] an examination of a spiraling ballistic missile is examined. The linear and extended KF were applied.

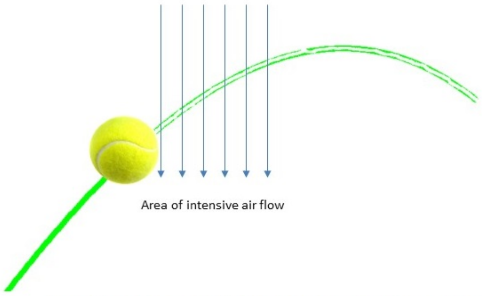
The aforementioned works targeted the prediction of ballistic trajectory based on erroneous measurements. The following sections show that in robotic catching with predictions based on physics similar mechanisms are often used to decrease errors (e.g. EKF in [Fre01, Bar08], LS in [Pon09]). However, these approaches applied for long-term prediction are lacking due to the inaccuracy of the linear and polynomial physical models. More complicated models are hard to embed into the KF framework and need a large amount of memory and time resources.

Model inaccuracy is often considered by the estimator as process noise. Here process noise is a real coordinate change that is not expected by the model. For instance, if the object flies through the area with intense air flow, it changes its trajectory, which the gravity-drag model did not expect. Once the object has left the area, the model become useful again (figure 2.2). Here the case of a very powerful wind is considered for visualization. In an actual robotic environment, the influence of process noise is little, and a failure to catch is mostly connected with measurement noise and the imperfections of the robotic system [Kim14]. Process noise must be distinguished from measurement noise, which means there was an error in measuring the coordinate and does not mean there was a real deviation from the model.

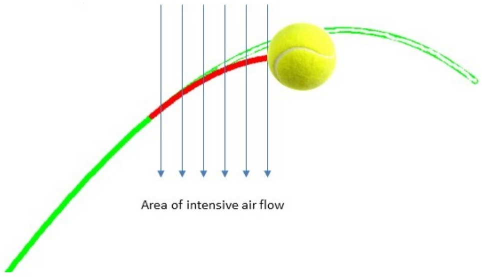
If the model used for prediction is not accurate enough, deviations in the trajectory due to inaccuracy may be considered to be a result of the influence of process noise. This is not entirely correct from an analytical point-of-view but may be computationally useful. For long-term prediction, such an approach would lead error of estimation to increase as distance increases. On the other hand, for short-term prediction even very simple physical models are accurate enough (e.g. [Ima04]). This leads to the idea that estimation mechanisms for physical models may be useful for decreasing the influence of measurement error for short distances. A simplified model may be used at the measurement processing stage to decrease the influence of errors but may not be used at the long-term prediction stage.

Due to such potential, use of the works in the approximation of the trajectory based on erroneous measurements becomes interesting for the approach. This task becomes necessary when the source of the projectile must be defined based on measurements. In this case the process noise is usually very little (significant process noise makes it impossible to define the source point accurately), and more precise models of motion are used.

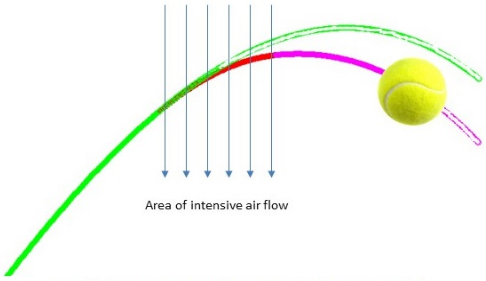
The need to determine the source of the projectile appears in a number of fields, including in sniper positioning during counterterrorism operations [Win12]. This positioning is based on radar measurements estimated by the least squares method. In this case even a simple linear model (without taking in mind any physical forces influencing the bullet) was useful to achieve suitable accuracy.



a - The ball is moving under the influence of gravity and drag along the green trajectory



b - The wind force ball trajectory to deviate, which is not expected by the gravity-drag model



c - Now the ball move according to the gravity-drag model again but the trajectory will be another

Figure 2.2: Example of process noise: deviation of object trajectory in the area of intensive air flow.

More complicated models are needed to define the source points of projectiles with a curved trajectory, e.g. howitzer missiles. [Zho07] examines the rising part of the trajectory, which allows drag term to be ignored as drag is directed downwards as gravity. The Kalman filter is used to estimate the parameters of the trajectory model. To estimate the source, the trajectory is extrapolated based on these parameters using the Runge-Kutta algorithm. The simulation showed that the errors of estimating parameters by the KF are more significant than the errors of extrapolation. Earlier in [Nel05], nonlinear regression with an LS estimator was discussed for the same task. The influence of drag is also not considered here. In [Cha14], the trajectory model is simplified and inserted into the EKF estimator. Attitude angles are considered as a trajectory domain. In [Ben08] the maximum likelihood estimation (MLE) is used to create a model of the trajectory that allows both the launch and destination point to be defined.

Research on the statistical estimation of ballistic curves primarily examine several different types of statistical filters: the Kalman filter with its extensions (extended and unscented Kalman filters), the maximum likelihood estimation, and the least square estimation. The applicability of these estimators depends on the character of the process (?). Kalman filters need more accurate noise models, while curve fitting based on least squares needs an accurate process model [Ril02, p. 124]. These state-of-the-art implementations must be carefully considered. Many of them have been proven based on the simulation, and in the simulation simplified and often corresponding to the estimator models of the process are used. In reality there is no confidence that these models are accurate. The aerodynamics of a projectile thrown by an arm are different from those of mechanically thrown objects.

2.3 Visual tracking of moving objects

In section 2.2 the term "object tracking" was used several times to describe the process of sensors observing the motion of an object in time. However, section 2.2 concentrated primarily on the mathematical aspects of processing erroneous coordinates and decreasing the influence of mathematically-modeled errors of observation. In this section, the technical and algorithmic issues of object tracking are considered.

Object tracking in general include the process of observing and positioning of the mowing object by a set of sensors. Tracking include the observation of certain area in space by one or more sensors. Usually the video tracking is considered if the video sensors (cameras) are used to observe the area. However sensors used for tracking could be various: infra-red, radio-location, etc. The visual sensor may be extended by additional sensors e.g. light barriers [Bar09]. Also exist a number of devices that are produced with such extension already done.

The aim of the camera setup is to provide the best placement for the sensors. This would allow the mechanical structure of the three-dimensional environment to be reconstructed. The visual measurement that allows the reconstruction of the 3D scene is called the range image. Various equipment compositions may be used. The three most widespread tools for capturing stereo images are as follows:

1. *Stereo pair*: This setup consists of two digital cameras with a synchronized frame-rate. If it is known how a certain point in space is projected onto both images (i.e. what the pixel coordinates of this point $\{u_l, v_l\}$ and $\{u_r, v_r l\}$in these images are) the position of this point in space can be defined (Figure 2.3; object spatial coordinates are expressed in the system connected with the optical center of the left camera). Each pixel in the image corresponds to a certain ray in space proceeding from the optical center of the camera. The point in space where the corresponding rays from two cameras cross is the target point. The mathematical procedure of reconstructing spatial coordinates from pixel coordinates is discussed more precisely in further chapter of the thesis in section 3.1.3.

2. *Structured light scanner* (Bor08, p. 222): The geometrical principle of structured light range measurement is similar to stereo pair. Camera is extended by light source (projector) which is projecting light patterns on the scene. As the light form the light source include patterns it is called *structured light*. A certain point from the light pattern correspond to a ray in space proceeding from the projector. If we know the pixel position of a certain point on the image from the camera and what point of the pattern it is reflected, its 3D position may be calculated as a cross of the camera ray (corresponding to the pixel position) and projector ray (corresponding to certain point from the pattern).

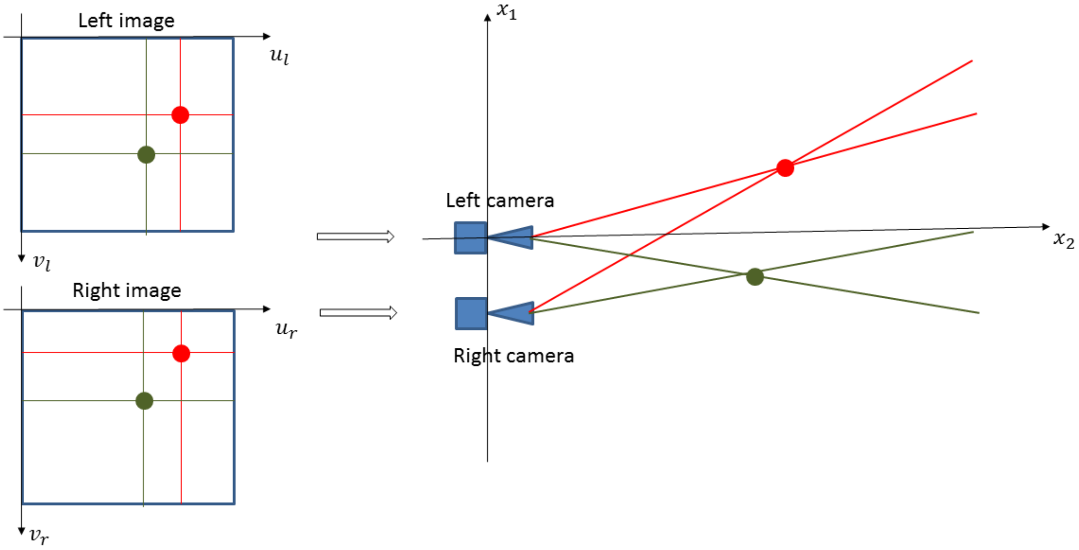


Figure 2.3: Defining the position of the point in the scene based on its known pixel position in two images.

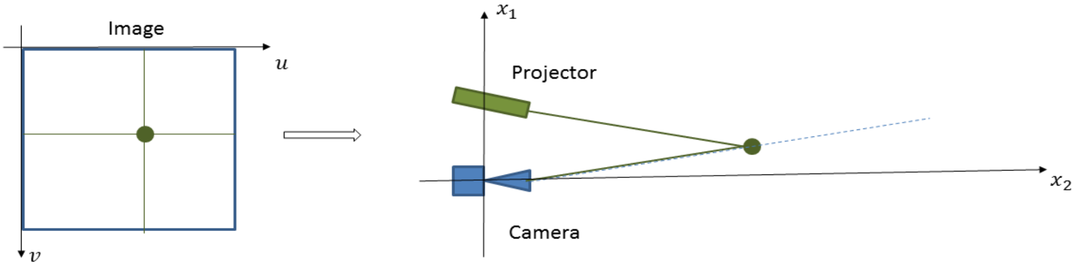


Figure 2.4: Defining 3D position of the point based on structured light.

3. *Time-of-Flight (ToF) camera.* One digital camera is extended by time of flight sensor. The basic principle of ToF sensing is similar to radio-location. The laser sends rays out around itself and then defines the distance from the object based on the time the rays are received once sent back.

In all three methods, digital video cameras are an important part of the sensing equipment. Cameras usually need a *calibration*, which is the procedure of estimating various parameters of the camera needed for stereo reconstruction. Various calibration methods exist. One of the most popular is based on the use of a chessboard with cells of a very precise size [6, Zha00]. The size of the cells must be known prior to calibration. The chessboard is photographed in various positions in the camera's field of view. Posterior processing of the image set from a single camera allow the intrinsic parameters of the camera to be determined. These represent its optical properties: focal length, principal point, and the distortion coefficient. In the stereo vision system, synchronized images of the chessboard from both cameras are made. After each camera has been calibrated on time, the stereo calibration is made, which integrates the data from calibrating both cameras and allows the extrinsic parameters of the system to be defined- the position and direction of one camera with respect to another. With the extrinsic and intrinsic parameters of the system, stereo triangulation becomes possible.

2.3.1 Motion capture with object-integrated sensors

Before discussing range imaging, it is necessary to mention that other means for determining the position of an object exist. For instance, sensors may be put on the flying object that transmit signals about the object?s location. This positioning technique was used in [Kim14] where Optitrack Markers by Natural Point [12] were used for motion capture. In that paper complex-shaped, rigid bodies were tracked in flight: bottles, tennis rackets, hammers, etc. As the coordinates of three points are sufficient to define an object?s position and orientation in 3D space, three markers were mounted on each object before the experiments. The accuracy of the positioning is not discussed in [Kim14], however it was sufficient for the robotic catching application developed within this work and will be discussed in more detail in subsections 2.4.1 and 2.5.2. In [Bir11] marker-based tracking was used to validate the tracker based on stereo vision.

The main advantage of marker-based tracking consists of the easily achievable higher frame-rate (e.g. 240 fps in [Kim14]). When one considers its use in industrial transportation applications, it has a disadvantage in that it requires additional equipment, energy and time to place sensors on each object and then take them off. This can dramatically increase the complexity of the production process. Hence, observing the object with the use of an external vision system is likely preferable in industrial transportation.

2.3.2 Structured light and time-of-flight range measurement

Structured light range scanners may be based on projecting colored dots [Dav96], black-and-white or colored stripes [Boy87], a white grid [Pro96], or dynamical light patterns [Bit76, Cas98, Hor99] in the scene. The projected patterns may be regular or pseudo-random. Commonly, it is not possible to know which point of the pattern corresponds to the known point of projection if the pattern is regular. However, the shape of the projection allows the shape of the scene to be determined. The use of regular patterns (stripes) allows the shape of the object to be defined with great accuracy; however, it is not intended to define the absolute coordinates. For example, the DLP projector in [Hal01] could measure the range with an accuracy of 0,1 mm on the distance up to 10 cm. This does not mean that the object has to be, at most, 10 cm away from the front of the projector. The variety of object points from each other must exceed this value (as SLS measures the respective proportions between various parts of the object). These systems are not useful for positioning airborne objects (except in the estimation of rotation vectors) as the airborne object is far away from any other objects in the scene. In [Wan08] the mechanism of defining the absolute range is discussed. Light patterns are projected onto the scene via the reflection from a rotating mirror for this purpose. Mirror rotation provides the dynamics of lightning patterns hence this method is also useful but only for static scenes.

The pseudo-random light pattern allows absolute range to be measured; however, the accuracy of this measurement is usually lower than for techniques based on regular patterns. Today, the most widespread SL camera is the Microsoft Kinect [5]. The high use of this device has increased the interest in SLS sensors, especially in object tracking. However, the positioning accuracy of the Kinect is not very high. The errors in the positioning of the object are increasing from around 2 mm at a distance of one meter to 7 cm at a distance of five meters. At three meters, it was about 2,5 cm. All three values are taken from [Kho12, p. 1451]. The resolution of the Kinect depth images is 640 by 480; however, how this is achieved is not public information, and the real resolution of the Kinect sensors is unknown [Noo11]. The low positioning accuracy of the Kinect sensor could mean that the resolution is achieved by interpolating data from a smaller number of pixels.

The first research that used structured light scanning to observe moving objects was [Hal01]. In it the special boundary codes were applied that allowed the recognition of frame-to-frame correspondence. The system was able to create a 3D model of a miniature elephant at a working volume of 6-by-12-by-16 cm. However, the positioning of isolated objects in 3D space was not considered in [Hal01]. Due to hardware limitations, the system was only able to reconstruct slowly moving objects (1 cm/s in the example). In [Ada04, Ada05], a structured light technique was applied to control the robotic manipulator. [Ada04] concentrated mostly on the use of structured light patterns while [Ada05] described the tracking procedure and searched for inter-frame correspondence. The set of points is projected into the scene, and each point may have one of seven colors. The color codes of the points are coded for a pseudo-random sequence to avoid repetition in the point patterns. In [Suk12, Kim13] the depth estimation of moving objects using a structured light technique is investigated. The CAD models of objects are also used as a prior input; known detail size with the pattern projections allows the distance to the objects in 3D space to be determined. The object positioning error reached several millimeters (4.72 maximum).

With the exception of these works, there is little literature on the use of common structured light technique for positioning the object within a 3D environment. In general, the use of structured light for tracking fast moving objects is not well investigated. At the current stage of range imaging technology, stereo vision is preferable for transport-by-throwing applications.

Accurate ToF systems are usually much more expensive than Kinect or stereo setups due to the high cost of high-resolution laser range finders. The accuracy of ToF positioning is limited by the low resolution of available depth sensors. In [Pia10], a sensor with a depth resolution of 320x200 pixels was considered worldwide to be the one with the highest resolution. Up until now, it has not been clear if other ToF cameras with a higher resolution exist. The new version of Kinect, released at the end of 2013, is based on the ToF principle. The resolution of the depth images from this device is 512 x 424 pixels; however, this may also be achieved by interpolation. Non-ToF cameras with much higher resolution are easily available. For example, IDS UI-3370CP, used for tracking in experiments described in section 3.1 has a resolution of 2048 by 2048 pixels [10].

Additional calibration of a ToF camera can increase its accuracy or at least enable it to be estimated. In [Fuc08] a method of additional calibration for ToF sensors to accurately position the camera in space. The nominal precision of the robot position control was 1 mm for translation and 0.1 degree for rotation. A precision of 1 mm with standard deviation of 5 mm was achieved during the calibration. In [Noo11] the calibration procedure for Kinect-based tracker is also applied. The errors of positioning were from 1 to 6 mms depending on the frame. An analysis of ToF positioning accuracy and calibration is given in the Piatti dissertation [Pia10]. The results were applied to two ToF cameras: the SR4000 (10 m range, 176 x 144 pixel resolution for depth image [Pia10], available at [8] for $4772) and the PMDCamCube3.0 (7.5 m range, 200 x 200 pixels depth image [9]). The discrepancy of positioning was up to 15 mm. After calibration it decreased to 3 mm in the range area of 1.5-4 m.

Research on ToF and SLS object tracking is widespread in areas where accurate positioning is noncompulsory, such as people-tracking within human-robot interaction [Swa08, Gan10, Jia14], the indoor self-localization of mobile robots [Ita12] or tracking mini-aircraft [Jur12]. In [Swa08] the SR300 ToF camera was used for the detection and tracking of humans. Positioning errors of several cm were achieved, satisfying the requirements of the research project. [Gan10] studied the same problem. Later in [Jia14], the tracking multiple people was done with the use of multiple ceiling-mounted ToF sensors. Both works scope on estimating human pose instead of motion models and positioning accuracy. In [Par11] the system based on Kinect sensors was trained to estimate the location of texture-less objects (small armillary sphere, miniature building, toy ship, etc.). Projection errors are given; however, the measurement units are unclear (mm, cm, rad, respective size of object, etc.). Also the time expenses are given. The process of initial estimation took less than 30 ms and the process of estimation based on previous frames took less than 15 ms (the Kinect nominal frame-rate is 25 fps which leads to a 40 ms delay between frames).

Robot positioning is another application where vision is used and where ToF sensors have been applied [Nak11]. In this task the accuracy of positioning may be easily determined by comparing the data with the robot control logs. In [Nak11] (tracking record available at [11]) detection of the object and the association between depth and color pixels is done for objects that have a higher color contrast than that of the scene. A particle filter was used for estimation. Unfortunately, the numerical error results are not given, but the figures [Nak11, pp. 787-788] and record [11] suggest that they achieved at least several mm.

The tracking of spherical objects was investigated in [Sir12, Jur12]. In [Jur12] the tracking of a flying quadrotor with a spherical shape for the propeller base was developed. [Sir12] is particularly interesting as ToF-tracking in it is used to grasp a ball with a 7 DoF industrial robotic manipulator (a 7DoF robotic arm is also used in the TU Wien TbT project). Linear KF was adapted for trajectory estimation (constant velocity model, change of velocity is considered to be process noise). The robot manipulator picks from either a static ball or a ball from a mobile robot moving along a straight line. The robot was able to perform the task at a maximum of 20 seconds.

ToF cameras were first applied to detect thrown objects in [Opr13]. Two setups were examined: a throw where the object?s energy and direction come from a human hand and a drop where a person releases the object and lets it fall. The solution was tested on 50 Kinect records (25 throws and 25 drops). The task was not to locate flying objects but to detect when an object was thrown or dropped.

Electro-magnetic waves as well as ultrasonic acoustic waves can be used for ToF sensing. [Ita12] can be used as an example of acoustic ToF tracking. In it ultra-sonic waves (160 kGz frequency) for echo-location were used to navigate a mobile robot. The order of accuracy was 1 cm [Ita12]. The long length of acoustic waves (2 mm for 160kGz) restricts the accuracy of echo-location positioning [Ita12]. Also, the low speed of sound restricts the frame rate and accuracy of locating fast-moving objects.

ToF tracking could be a promising technology for the future, but for the time being it does not have high resolution necessary for systems based on stereo vision.

2.3.3 Monocular vision

A stereo pair is a specific setup that guarantees that point pixel positions can define the 3D position of an object. However, this does not mean that it is impossible to determine an object's location with a single camera. Use of additional information in specific cases could allow an object's position to be defined based in the images from one camera. An object tracker for a TbT application based on single camera systems is studied in [Bar08]. The camera was extended by additional sensors, but the object coordinates were extracted from the images only. This setup does not allow the full model of 3D scene to be extracted. However, information on the 3D structure of the scene is not really necessary to track an airborne object especially when the tracker has prior information about the object and environment.

In [Bar08, Bar11], one camera extended by light barriers was used. The barriers measured the launching velocity of the ball. Two light barriers were positioned with d = 100 mm (distance) between them. The first barrier stayed in 100-200 mm forward the object acceleration endpoint. The launching velocity is calculated using the equation:

$$v_{0z} = \frac{d}{t_2 - t_1} = 0,1 * (t_2 - t_1)^{-1}, \tag{2.18}$$

where t_1 and t_2 are the time of interception with the first and second light barriers respectively. These time marks are made with an accuracy of 0.5 ms, which is the time needed for the barrier's reaction. The object position in the image allows the ray to be defined in the world coordinate system starting with the camera sensor on which the object lies (same as in figure 2.3). The actual position of the object in z-direction (i.e. perpendicular direction to light barriers) is calculated by numerically solving the following differential equation:

$$\frac{d^2 z}{dt^2} = -\frac{C_d * \rho * A * (\frac{dz}{dt})^2}{2 * m}, \tag{2.19}$$

where C_d is drag coefficient, rho is air density A is frontal surface of the ball, m is mass of the ball. The Runge-Kutta method can help approximate the value of $z(t)$ with the use of this formula. The estimated ball position for the moment of time of t_1 is determined as interception of ray b extracted from the image and plane $z_1 = z(t_1)$ (figure 2.5). In other words, the assumption about the shape of the trajectory allows it to be reconstructed based on the monocular image sequence.

This method for ballistic forecasting was useful in the conditions of the experiment and allowed a prediction error of less than 12,5 mm in 66% of throws to be achieved. Later on in [Rib09], an estimation of 3D ballistic trajectory was investigated as a local optimization task. In this work the gravity-only model was applied. The lack of trajectory triangulation is because estimating ball position is based on information about ball aerodynamics. As discussed in section 2.1, the gravity-drag aerodynamic model expressed by equation 2.10 is not very accurate even for simply shaped, spherical objects like a tennis ball, and it is better to avoid its use in order to achieve higher accuracy. The motion model expressed by 2.19 is a simplification of the model expressed by 2.10: The drag is assumed to be independent for each spatial dimension, which is not correct. On the contrary, the financial advantage of using one camera instead of two is not a dramatic one; the price of a few digital camera is much less than that of the 6DoF robot manipulators used for catching. As this work aims to develop a prediction model without hard relation (?) to exact physical knowledge, the trajectory triangulation would be an invalid tracking method. However, the method for velocity measurement described in [Bar08, p. 894] could be useful for extracting information about launching parameters.

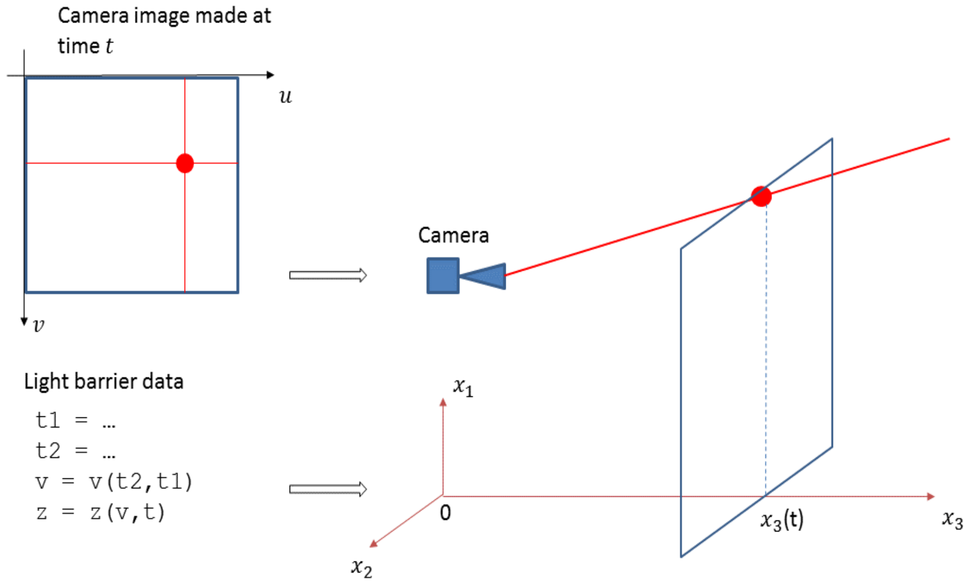


Figure 2.5: Defining ball position using the method from [Bar08]. Light barriers measure the time when the ball reaches them. The difference between these times allows the object?s horizontal velocity to be defined. This velocity is forecast in order to define the distance reached by the object at any moment in time, fitting this distance to the ray defined by the object pixel position allow the 3D position of the object to be determined.

2.3.4 Stereo vision

Vision systems may consist of one camera, two cameras or possibly more. A vision system with one camera must be extended by additional devices (light barriers, ToF sensors, etc.) for achieving the range of necessary images (see subsection 2.3.2). Two cameras are enough to extract coordinate information by stereo triangulation without the use of additional sensors. Most of state-of-the-art applications of TbT and robotic catching use stereo triangulation for object tracking [Hov91, All91, Ril02, Nam03, Sca05, Bae10, Pon11].

The stereo triangulation algorithm does not need aerodynamic models to reconstruct object position. It needs information about the relative location of the cameras (extrinsic parameters) and about the intrinsic optical parameters of each camera: focal length, principal point, distortion coefficients, etc. This information is obtained during stereo calibration. The stereo calibration is done based on the calibration of single cameras as previously mentioned. If the relative location of the cameras has not changed, the stereo triangulation become possible after the calibration [6].

The accuracy of object positioning via stereo triangulation depends on the resolution and other parameters of the cameras. The definition of accuracy for concrete setup is not well-formulated in the state-of-the-art works on stereo vision. As few rare examples of such works, e.g. by [Lee02, Liu06], should be mentioned. [Lee02] deals with the question of how to localize cameras in such a way that the observed surface is maximized and reconstruction errors are minimized. A multiple stereo pair system for scene observation is discussed, but the accuracy of a single stereo pair is also considered. Lee et. al specifies the following sources of positioning inaccuracy [Lee02, pp. 222-223] in stereo triangulation systems:

- *Quantization errors*: One pixel in the image represents a certain area of the scene. With an increase in the distance from the camera to the background, the size of this area also increases. Within the one-pixel area, it is not possible to position the point accurately. A visualization of the quantization errors is given in the figure 2.6. It is possible to define intervals Δ_1 and Δ_2 on which the boundaries between neighbor pixels are lying, but the error of one pixel will correspond to the error of interval size $\Delta_1 + \Delta_2$ in 3D space.

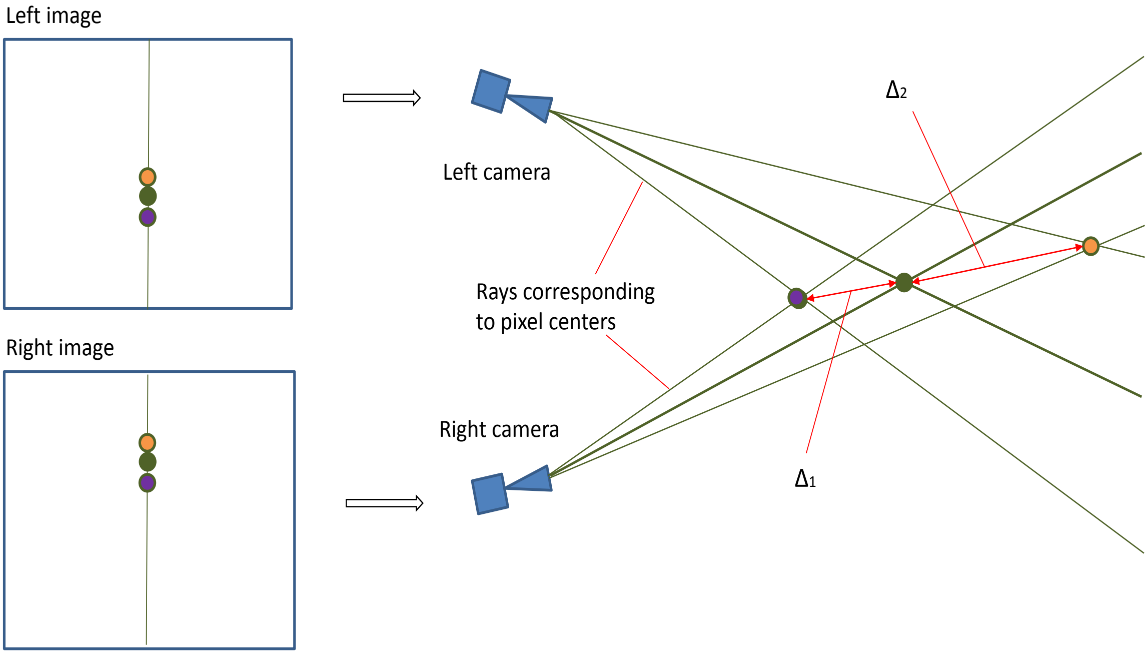


Figure 2.6: Quantization accuracy in stereo vision; Δ_1 and Δ_2 show the range where the border between pixels may be positioned.

- *Image processing errors*: They are connected with the incorrect positioning of feature points in the image. In [Lee02] these errors are primarily considered to be errors in stereo matching. Stereo matching is the process of defining points on the left and right images that correspond to one another. This is one of the main tasks in processing stereo images, and it often leads to errors, especially with occlusions, when dealing with texture-less objects and asymmetric lightning. In object tracking, stereo matching is dissimilar to the task of scene reconstruction. The position of the object's center in each image may be determined separately, but this determination may be erroneous.
- *Calibration errors*: These come from quantization errors during the calibration process. This leads to inaccuracy in the defined extrinsic and intrinsic parameters of the camera system.

The quantization errors were estimated to be highly reliable in the corresponding camera arrangement (check my correction), while image processing and calibration errors have a negligible relation in it. The arrangement was examined and optimized within the camera simulation. The task of optimization was to find the optimal maximization stereo vision area of the two cameras while minimizing the quantization errors. During the simulation, a 20% decrease in quantization errors was achieved.

Later in [Liu06], the question of the corresponding camera location in the stereo pair is discussed. This work explores the following setup. The distance between the camera and the observed object is fixed, and the angle between the cameras' optical axes α and height of the baseline above the object h vary (the geometrical mean of these parameters is shown in figure 2.7). The relation of positioning accuracy to the values of h and α is estimated. The definition of this task could be disputable as it is not exactly clear what parameter h really represents. Changing h without changing the distance to the object means the stereo pair would be rotated around the object in the vertical plane. From the physical point of view, this means that different angles between the stereo system and gravitation vector, but the optical sense of this rotation is unclear.

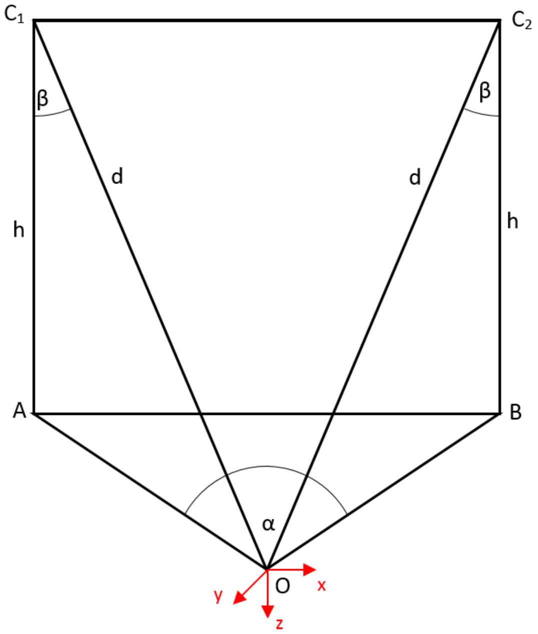


Figure 2.7: The model of the camera setup geometry used in [Liu06]. The observed object is positioned at point O. C_1O and C_2O are the optical axes of the right and left camera respectively. AO and BO are their projections on the horizontal plane and α is the angle between AO and BO

A description of the procedures of accuracy definition that were applied in the current research is given in sections 3.2 and 3.3. Further increase of cameras quantity does not change the principle of triangulation, however it can:

- Improve the accuracy of positioning as the object position may be better estimated comparing the results from different stereo-pairs. The case of such method is use of trinocular vision system.
- Enlarge the size of observed scene as various subareas of the scene might be observed by different stereo pairs.

Trinocular stereo systems mean that the images from three cameras can be compared. They use special triangulation techniques that allow an increase in accuracy. In the binocular system, the 3D position is ideally a cross of two rays in space or realistically the center of the shortest possible distance between the points on these rays. Trinocular vision means that the 3D point is obtained where three rays cross. This means that the third camera gives extra information which allows the binocular stereo reconstruction to be validated. For example, the final estimated position of the point in 3D space may be determined to be a centroid of reconstruction results from three possible camera pairs in the trinocular setup. The accuracy of these systems was studied in [Chi95]. The work concentrates on the inherent issues in stereo matching and some particular features of a trinocular system are discussed, e.g. projection location constraints. No comparison with binocular systems was made in the paper [Chi95].

If multiple stereo pairs are used for observing the scene, the question of the optimal positioning of the sensors becomes relevant. It is studied, for example, to determine security camera placement [Mor10], road traffic management [Cho11] and also for industrial environments. The task is to cover the requested area with a minimum number of cameras as efficiently as possible. Also, the positioning of light sources can be added to the list of existing problems [Gar11].

In TbT the application of the area of an object's trajectory can be divided into subareas to be observed by various stereo pairs. One pair observes the initial part of the trajectory to provide input for prediction, while another pair observes the gripper's workspace for possible adjustments and to identify whether a catch was successful or not. The common tracking tasks in such environments mean the "transition" of the object from one observing unit to another. [Liu09, Liu10] studied the situation where the object is detected by one camera and the second camera must be chosen to collaborate in order to construct a stereo pair.

[Lee02] has been previously mentioned concerning the accuracy of stereo pairs. In general, this work studies the aspects of a multi-camera system constructed to be a set of stereo pairs. The arrangement was examined and optimized within the camera simulation. The task of optimization was to find the optimal maximization of the stereo vision area of two cameras while minimizing errors. This task was solved for the given shape of a scene.

Section 3.2 demonstrates that at short distances (up to 1.5 meters) a single stereo pair is sufficient for tracking the flying body. An investigation into the accuracy of this positioning did not show a significant correlation from the object location within the whole area of the object trajectory [Pon15]. The accuracy results were found to be sufficient for the application. In theory it would be interesting to apply the trinocular setup or other setups that would afford a more accurate estimation of 3D position than binocular stereo. The reason why they were not considered in the current research is due to the large amount of time to increase the camera number. In chapter 5 it is shown that finding the ball's center in the image is the step of the tracking and prediction algorithm that needs the most time, even if it is implemented on a specific fast hardware platform (GPU or FPGA implementation). Moreover, when it comes to the enlargement of transportation distances, the use of multi-camera systems seems promising.

The stereo sensors can be mounted not only in static way or onto the object, but they can also move through the observed scene. This technique is often used for static scenes when the need arises to fully reconstruct the shape of a 3D object. The use of moving cameras to track a moving object was introduced in [Fed90] for one-DoF movement tasks. A two-DoF tracker was developed by Mukai and Ishikawa [Muk94]. [1] The sensors are positioned on the robotic arm which performs

---

[1] Masatoshi Ishikawa later on took part in the research on robotic catching, described in subsections 2.4.1 and 2.5.1, however in [Muk94] airborne objects are not considered.

the movement close to the object. For an airborne object, this solution?s challenge is to address the speed of flight; incredibly fast manipulators are needed. Another disadvantage is that error of reconstruction is increased by the inaccuracy of camera self-positioning. Because of this, the static camera setup is preferable.

2.4 Robotic catching and transportation by throwing

The ability to throw and catch rigid objects is something almost all humans share. The same concept as applied to the actions of robotic manipulators was first introduced as a way of making robots more similar to humans in their abilities. Though humans do not use physical or mathematical models to forecast the trajectory of the ball, most works in the area of robotic catching use analytical techniques that are based on these models. The aim of this section is to give an overview of the recent developments to the systems that enable robotic throwing and catching as well as related tasks (i.e. track airborne objects and forecast the trajectories). This overview includes research from the Transport-by-Throwing project in addition to unrelated research. Prediction aspects are briefly discussed in this section as issues of catching movements and grasping principles. A more precise review of how to solve the challenge of prediction in these works is given in section 2.5.

2.4.1 Throwing and catching

The goal of robotic throwing is to provide launching parameters that allow the object to reach a certain area (gripper workspace in the case of the throw-catch application). The issue of minimizing damages might also be considered [Miy10]. A number of works exist where throwing without catching is considered and analyzed [Kob11, Nem11, Zha12, Kan12]. There are some examples where potential industrial applications put objects into storage pallets [Miy10], but most of the works investigate throwing as an issue in common robotics development. In various concepts, throwing techniques were developed for 1-DoF manipulators [Miy10, Kob11], 6-DoF robots [Nem11, Zha12], and an arm for a humanoid robot [Kan12]. When the task of throwing the object towards the static target (e.g. the workspace of the robotic catcher) is examined, the 1D catching device sufficiently provides a specific range of possible trajectories. The regulated parameter of a 1D throw is its launching impulse, while the direction of throw is determined by the throwing device?s setup. Robotic throwers with a higher number of DoFs are interesting because in this case the manipulator may perform some additional actions with the object prior to the throw, e.g. it may pick up objects from storage. A 1DoF catapult can throw and nothing more; the object must be put into its cup by another manipulator or by a human.

The simple grasping and picking up of static objects was one of the first abilities robots were able to achieve. Some time later, the task of matching the manipulator with the moving object was investigated. In these early works, the two-dimensional (e.g. a ball rolling on a planar surface in [And85]) and then three-dimensional trajectories of objects were considered. The main difference between early 2D catching and catching airborne objects consisted in their low velocities. The robot used in [All93] grasped miniature trains moving along a round train track at velocities between 10-30 cm/s. The robot in [Lin89] was able to catch the object moving around the sinusoida, but it took about 2.25 seconds to perform the catch. The robot in [And85] was able to catch most of the balls with a velocity of 0.75 m/s and some balls with a velocity of 1.3 m/s.

The flight of the thrown object is high-speed 3D process in comparison with relatively slow 2D motion discussed above. The catcher must be fast enough to perform the appropriate catching movement (especially in soft catching). This includes both mechanical performance (ability of joints to provide fast movements) and algorithmic performance (real-time tracking on high frame-rates, fast prediction and fast path-planning for the gripper). The catch may be passive (minimizes the force of impact with the gripper) or active (an intensive grasping movement to the object).

The robotic catching of airborne objects was introduced more than twenty years ago in [Hov91]. The concept of "vision-motion" coordination (i.e. to adjust the robot?s motion based on the data from a stereo vision system) was developed, and parabolic models were applied for trajectory prediction. This concept was demonstrated using a 4-DoF manipulator with 4.2 m^3 workspace and a maximum velocity of the end effector equal to 2 m/s. In the experiments, the rate of successful catches was 75-80%. Data on the throwing setup is not provided.

Robotic catching was also considered in [Nis97]. In this work, the task was first delegated to humanoid robot with the aim of implementing ?human skills on a humanoid". Visual feedback is not taken into consideration and the robot's motion is defined using only predicted ball trajectory In other words, there is no adjustment to the gripper?s motion to correspond to the new vision data once the catching movement has begun. Predictions are based on information about the initial part of the trajectory, but after the initial sequence the ball is not tracked anymore. A basket with a diameter of 120 mm was attached to the humanoid arm to procure the ball. Only the arm is active during the catching movement and the robot itself is motionless. Two CCD cameras mounted on the head of the humanoid are integrated into the stereo pair. Balls that were vertically dropped were investigated as well as balls that were thrown. Prediction was not needed to catch the object dropped vertically by a basket as the trajectory of the ball is determined by the first measurement and the interception velocity is not critical for hard catching. For the thrown ball, the prediction is implemented via parabola fitting. The rate of success is not mentioned, but inserting visual feedback into a control scheme was suggested to increase the rate of success.

In another research project by the Tokyo university, a number of articles were published that investigate the robotic grasping process [Ish96, Nam99, Nam03a, Nam03b, Ima04, Sen04, Shi05, Fur06]. Within this project the catching with the use of a grasping movement (i.e. the process of closing a robot's fingers around the object) was considered. The philosophy is the opposite of that of [Nis97] where movement was determined based simply on prediction with no consideration of actual visual feedback. In this work the grasping is determined solely based on visual feedback with no consideration for prediction.

In [Ish96] a framework for fast target tracking was introduced. The hardware setup and algorithms in this system are based on the direct connection of sensor photovoltaics to processor units and the high parallelized processing of photo-sensor information instead of the traditional capture of an entire image from the camera. This allows system feedback to be provided in 1 ms (with a time delay of 20 ms between observer and actuator). Video processing algorithms are based on the property of high speed video in that sequential frames have very little differences between them. The target object (considered to be monochrome) is extracted from the image by comparing the current frame with a dilated target pattern from a previous frame (the dilation procedure is shown in figure 2.8). Also the situations of target visual collisions with the scene objects and their separations are recognized and computed.

The grasping technique (applied to slowly moving objects) based on the tracking mechanism from [Ish96] was introduced and expanded upon within the work in [Nam99]. The catching was provided by a 7-DoF robotic arm with a multi-fingered robotic hand as an end effector (there

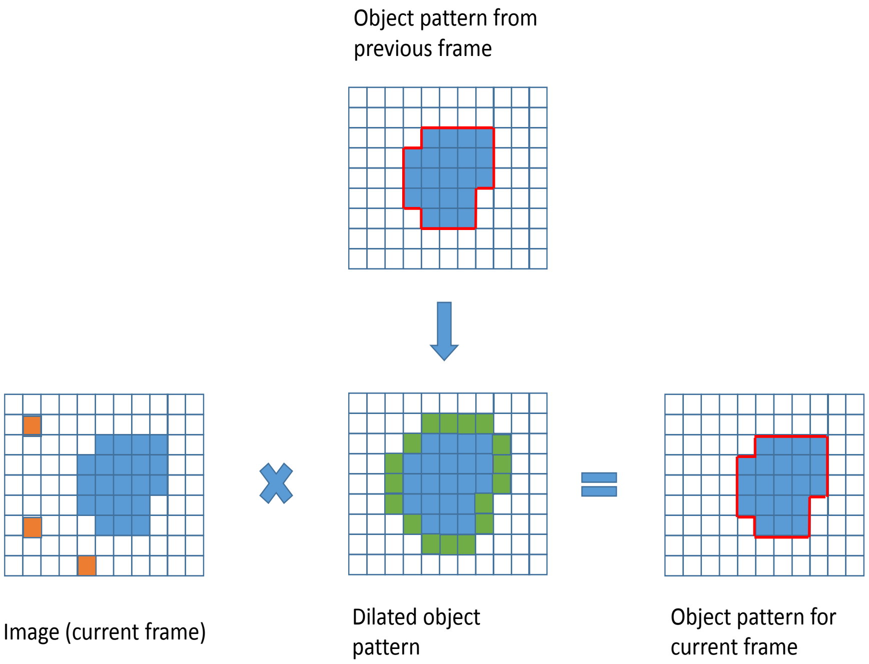

Object pattern from previous frame
Image (current frame)
Dilated object pattern
Object pattern for current frame

Figure 2.8: Object tracking in [Ish96].

were 4 fingers and 14 joints, one finger is the "thumb", i.e. opposite of the others). The sensor platform can also move. It had two DoFs- the pan and the tilt, which enable movement according to the changing position of the object. The tracking concept from [Ish96] was implemented on a specific vision chip, SPE-256. The control subsystem consists of 7 DSP working in sync. They control the arm, the hand and the camera base. The tracking algorithm is extended by the 3D-reconstruction and pose estimation for the object. The prediction stage is ignored due to high speed of system reaction. At each moment the new command to the arm is defined in such a way that the distance is minimized between the hand and the object. The aim of the hand control is to cover the object with the fingers whenever possible. In the experiments described, the grasped object was moved by a human hand.

The grasping of free-flying objects based on the same principles was introduced in [Nam03a] where the robot motion parameters (i.e. the desired trajectory of the end effector) are defined directly from the observed 3D coordinates of the object without taking the velocity and intermediate step of trajectory prediction into consideration. The relation between the input and output parameters was defined after optimizing the coefficients of the 4-order polynomials. The concept was applied to a 4-axis robotic arm (only 3 axes were used in the catching movement) which was able to achieve a velocity of 5 m/s and acceleration of 30 m/s^2. The catching motion was declared successful (and the recorded frame sequence of the successful catch is given on [Nam03a, p. 2405]), but no data on the rate of success is given.

In [Nam03b] a new multi-fingered hand that is useful for catching was developed. It achieves a pressure force of up to 47 newtons and a finger velocity m/s which allows the hand to close in 70-120 ms.Experiments in active catching using this hand and the recently developed high-speed vision system are discussed more precisely in [Ima04] (the previous work examined passive catching). Active catching allows the object in movement to be grasped with a higher velocity than the maximum velocity of the robot fingers. In this work, how to catch a falling object (i.e. velocity of the object is directed downwards) is considered. Successful catches were achieved for balls and cylinders.

In [Sen04, Shi05, Fur06], the concept was extended to more complex manipulation tasks than catching, for instance batting [Sen04] and dribbling the ball [Shi05] and then grasping the cylindrical object again [Fur06]. The whole active catching project [Ish96, Nam99, Nam03a, Nam03b, Ima04, Sen04, Shi05, Fur06] includes a high number of novel robotic and machine vision concepts like the direct interaction of a processor with elements of the sensor matrix, the direct interaction of tracker and robotic controller, active grasping, etc. However, with TbT in mind, it has some disadvantages. First the situation where any movement of the gripper towards the object is good for catching is considered. This seems useful at a short distance with a very fast robot manipulator, but at longer distances of several meters, the curve character of a projectile trajectory must be taken into account. The trajectory path in the gripper workspace must be predefined otherwise the gripper may not manage to catch the object if it flies to far from its initial position. This means that the concept may be useful at a terminal stage when the object is near to the gripper.

The active catching principle when the grasping device is moving towards the flying object at maximum velocity does not minimize the impact force, so the fragile object may be damaged during catching. Active catching may be applied for strong objects and strong grippers that cannot be damaged after interception at velocities of several tens of m/s.

The next significant research project concerning in-flight catching was done by a group from the Institute of Robotics and Mechatronics at German Aerospace Center [Fre01, Bae10, Bir11, Bae11]. In the work, the scale of catching was larger than in the previous project; throws of several meters were initially investigated. Due to long distance from the gripper workspace and the very fast processing, the catching was not interactive but pre-defined. The prediction was applied for the predefinition.

In [Fre01] an extended Kalman filter (EKF) was used to estimate ball position and forecast. Objects were thrown with a nominal velocity of 7 m/s and at an angle of 45 degrees to the horizon. The object was airborne for 0,8-1,0 s and covered a distance of about 5 m in that time. A stereo system with a 1 m vertical baseline was used to observe the ball trajectory (shown in figure 2.9). The rotation angle of the camera system was equal to 90 degrees. The precision of the ball's positioning with the described setup was about 3 cm. Object recognition was done based on a comparison of the current frame with a reference image. The difference (i.e. differences between current image and reference images) image is divided into shapes. The region with the best fit with the object pattern is considered to be the location of the object in the image. The frame-rate of the camera system was standard (25 fps), which allowed 50 images per second to be processed.

The prediction algorithm was based on the gravity-drag model expressed by the equation 2.10, which is solved numerically by the Euler integration. This motion model was inserted into EKF. The catching point is generated when the EKF reports on the sufficiency of the achieved covariance. The catching point should not be close to the workspace boundary or to the robot base. It is then given to the robot control in order to define the catching trajectory of the gripper (motion planning is based on the inverse kinematics interpolation). The whole data processing cycle (image extraction, stereo reconstruction, prediction and robot control operation) took 75 ms.

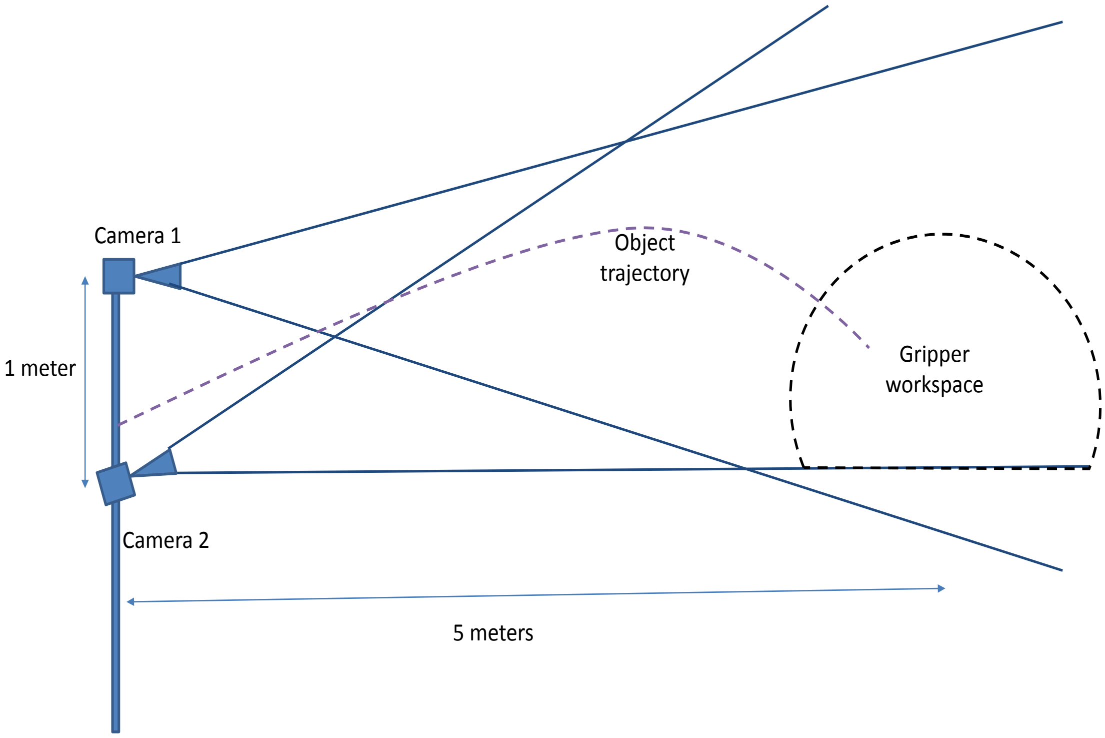


Figure 2.9: Observation setup in [Fre01].

The small basket was mounted on the end effector of the robot to procure the balls. In catching experiments no throwing device was used. The ball was thrown by humans. This means that the throwing point is not fixed in space, and there are no nominal values for the angle and velocity of the throw. About 67% of the throws were successful in the catching experiments.

Further development of the catcher was presented in [Bae10]. In the paper, a 4-fingered robotic hand with a total of 12 DoF was used for grasping. The task of defining the catching movement is examined as a complex task for non-linear optimization. It is solved directly from the prediction results without specifying the intermediate steps of choosing a catching point and defining the catching configuration.

The geometrical setup and prediction algorithm are the same as those in [Fre01]. The precision of prediction of 2 cm, which is enough for a robotic hand to successful grasp, according to [Bae10, p. 2595]), was typically achieved 100 ms prior to the moment of catching. Due to this small time interval, the catching movement must begin prior to the final prediction result and must then be recalculated based on the new prediction data. Since the optimization process is computationally expensive, it was implemented on 32 CPU cores. Optimization stops after 60 ms of a search for the best solution. The rate of success was increased to more than 80%.

In [Bir11] the concept was extended for the parallel tracking and catching of multiple flying balls, and the same research was already briefly introduced in [Bae11]. The balls are caught by the humanoid robot Rollin' Justin introduced in [Bor09]. The cameras are mounted on to the head of the humanoid. Two flying balls were caught at the same time by the humanoid. In [Bir11] a tracking concept based on circle detection was applied for ball catching. It was introduced in [Bir09] in an application for the surveillance of sport matches. After image analysis, all of the circles are inserted into a multi-hypothesis tracker based on the Kalman filter. The rate of success was again about 80%. Later in [Bir11a], a new tracking algorithm based on the Byesian probabilistic models was introduced which allowed tracking to be performed with similar accuracy and 55% quicker. In [Bae11a] several improvements to the whole system are presented. The tracking concept from [Bir11a] is put to practice. Other improvements mainly address the challenges of a situation where "everything is moving", including the camera system mounted on the head of moving humanoid.

Within the humanoid catching project [Fre01, Bir09, Bae10, Bae11, Bir11, Bir11a, Bae11a], important results are obtained. The accuracy and performance of tracking and prediction seems to come close to the constraints of gravity-drag physical models. This project also deals with many specific aspects not considered at the current stage of the TbT project like the ability of one system to catch multiple balls instantaneously, the application of humanoid robotic systems for this task, and the situation where the gripper base and vision system are moving. These accomplishments could possibly be useful in the future development of transportation principles.

Another ball-catching application for humanoid robots was described in [Ril02]. The aim was to investigate human-like behavior of the robot while catching the ball. The catching motion converges with the catching motion of the human hand introduced in [Kaj99]. A baseball glove was used to collect the ball. The stereo vision system provided 60 fps of position measurements. The hand of the thrower, in this case a human, as well as the ball is tracked. Prediction was done via parabola fitting. The peak velocities of the end effector during the catching were usually between 1.0 and 2.5 m/s. The catching motion was usually completed at 350-850 milliseconds.

In [Mor04] a catching strategy using a monocular vision system is introduced. The proposed approach for ball tracking is called Gaining Angle of Gaze (GAG). This strategy is similar to the baseball player catching strategy described in [Cha68]. The fisheye camera is mounted directly on the manipulator hand. While tracking every second, the instructions for the manipulator are defined in such a way the angle of gaze is kept within a prior specified constant range. This means that there are two main stages of the catching movement, e.g. reaching the target angle and then keeping it constant. Thus, the question of defining ball position in 3D space is not taken into consideration. The catching motion is stopped when the ball leaves the camera's field of view. At that moment the gripper is ready to catch the ball. Hence, GAG-based catching is one of the methods that does not deal with prediction in [Ima04]. These methods require the robot's response to the visual information to be incredibly fast. However, the robot in [Mor04] was able to catch the ball with the end effector moving at a relatively low velocity of 0.6 m/s.

In [Her09] the monocular vision system (70-fps-camera) was used to track the ballistic object (a flipped coin) in order to catch it. This work claimed to be the first to apply monocular positioning for flying objects, but [Bar08] had already been published. A less complicated model for estimation than the one in [Bar08] was applied, e.g. parabola fitting with recursive least squares estimation. The catching movement was performed by a 6-DoF robotic arm with a speed of up to 1.0 m/s. This is much less than the object's velocity, so hard catching was used. The parabola is fitted to the projection model, i.e. it is used to define the 3D coordinates of the flying body. The accuracy of prediction is increases over time, so arm movement is adjusted according to the new data from the predictor.

The research work in [Smi07] concentrates on developing software and the hardware platform that allows the fast catching movement to be performed. The catching was considered to be tele-operated, but an autonomous ball-catching was also implemented as a benchmark. A parabolic model was applied for prediction, which is based on the trajectory measurements done by the 50 fps cameras. The time range of 300 ms is reserved for data processing. A total of 200 ms of those are needed to capture 10 frames, and the other 100 ms are needed for coordinate acquisition, forecasting and generating the motion of the gripper. A total of 500 ms are reserved for performing the catching movement. A 0.6 by 0.6 m square was used for the gripper workspace. The aim was to make the catching device that could perform "a movement of worth case", moving from the lower-left to the upper-right corner of the workspace (i.e. a movement against gravity) in 500 ms (i.e. reaching the distance of about 90 cm in this time). The color information was used to detect the ball, and to do this the image was translated from RGB to HSV color space. The entire frame was not processed by the recognition algorithm, just a small subwindow where the ball can appear. The position of the subwindow in the image is determined by the position of the ball on the previous frame of the sequence. In the catching experiments, 25 of 32 thrown balls were caught successfully. Color-based recognition with a parabolic model was also applied later in [Kao13], but the linear Kalman filter was used for statistical estimation instead of least squares.

Parabolic fitting using recursive LS was also used for prediction in [Bat10]. The setup allow catching objects (in the experiments basketball thrown by the human was used) in non-prehensile way by balancing the object on the planar surface. A circular plate with a radius of 17 cm is used as an end effector mounted on the 6-DoF robot. The success rate was approximately 35%, but this result is incomparable with other success rate data, e.g. in [Hov91] or [Bae10], as the catching task for non-prehensile catching based on balancing is much more difficult than that for grasping.

In [Kim12, Kim14] a 7-DoF KUKA LWR 4+ arm was used to catch free-flying objects of various shapes: a hammer, bottle, tennis racket, and an oblong box. Of the captures, 52 of 71 (73.2%) were successful. This work is interesting because of the specific prediction concept used, so it is discussed more precisely in subsection 2.5.2.

2.4.2 Transport by throwing

As previously mentioned, the method of transportation by throwing was introduced by Frank in 2006. Since then work on TbT has been developed by two scientific groups in Reinhold-Wuerth University in Kunzelsau ([Fra06, Fra08, Fra11]) and at the Institute of Computer Technology, Vienna University of Technology ([Pon11, Pon13, Mir13]).

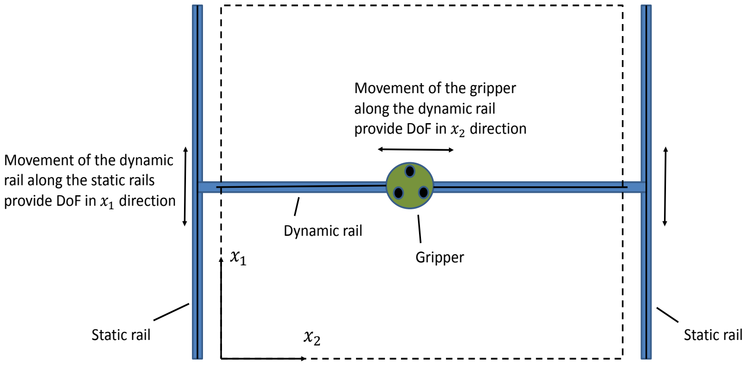

Movement of the gripper along the dynamic rail provide DoF in x_2 direction
Movement of the dynamic rail along the static rails provide DoF in x_1 direction
Dynamic rail
Gripper
x_1
x_2
Static rail
Static rail

Figure 2.10: The 2-DoF gantry manipulator used in [Fra07].

At the initial stage of the research project, the catching setup based on a 2-DoF gantry robot was applied [Fra06, Fra07, Bar08, Fra08, Fra08a, Bar09]. These robots are able to move within the Cartesian square and are often used as loading and unloading machines in the industrial environment [Fra07, Fra08a]. The scheme of the 2DoF Cartesian gantry robot is shown in the figure. The size of the workspace is (horizontal) 100 cm by (vertical) 80 cm. The maximum acceleration and speed of the end effector are 25 m/s2 and 4 m/s respectively at a payload of 5 kg. The gripper has three grasping jaws that close when the object touches the center of the gripper. The closure may be performed with the use of the object's kinetic energy (the upper part of the figure 2.11) or the kinetic energy of a previously tightened spring (the lower part of the figure 2.11). These grippers are able to close within 4.5-14 ms and 5 ms respectively. The closing time for the first gripper relies on the mass and velocity of the object. If its kinetic energy is high, it will close quickly. The respective relation for the spring-based gripper was not detected. Therefore, the spring-based gripper could be preferable for lightweight objects, while heavier objects could be effectively caught using their own kinetic energy. The monocular physics-related technique described in [Bar08] and discussed in subsection 2.3.3 was applied to track the ball and forecast its trajectory.

The setup was as follows; the objects with a weight up to 60 g were thrown with a velocity of up to 10 m/s at a distance up to 3 meters. Therefore, the time needed for object transportation was about 300 ms. Two control strategies were proposed: ”simplified” (an assumption about the linear increase of prediction accuracy is made) and ”realistic” (exponential increase). Both strategies provide the smooth movement of the end effector towards the adjusted catching point. In the results of the Barteit dissertation [Bar11, p.125], 60% of the balls thrown by a human were caught.

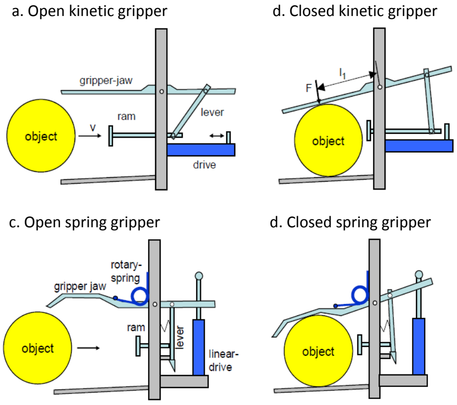

a. Open kinetic gripper
d. Closed kinetic gripper
c. Open spring gripper
d. Closed spring gripper

Figure 2.11: Grasping principle from [Fra07]

At both the Vienna University of Technologies and Heilbronn University, further development of the research project took place. The work of the Heilbronn group is summarized in [Fra09, Fra11, Fra11a Fra12]. In these works, the concept was extended to cylinder-shaped objects. In [Fra09] the specific throwing setup for these objects is introduced, and the aerodynamic properties of airborne cylinders are studied. The throwing experiments have shown that the angle of attack for a linearly thrown cylinder is close to zero during the flight, i.e. the orientation of the object matches the current direction of flight (figure 2.12, a). The motion model for the cylinder was created with respect to gravity and air drag. As the angle of attack is equal to zero, the lift influence of the air was ignored in the motion model. However, a comparison of the motion model with the measurements showed that after 3 m the trajectory of the object varies from the theoretical values for 22-36 mm. The authors propose that these deviations could be due to the influence of lift. When the open pipe was thrown instead of the cylinder, it began to tumble after a time (figure 2.12,b). When the cylinder was thrown with a spin of 35 revolutions per second (this idea was inspired by the use of rifling to stabilize missiles), it does not keep the angle of attack along the trajectory but keeps its orientation in the world coordinate system (figure 2.12, c). This behavior corresponds to the property of the flywheel to keep the constant orientation of the rotational axis in space (also used in gyroscopes).

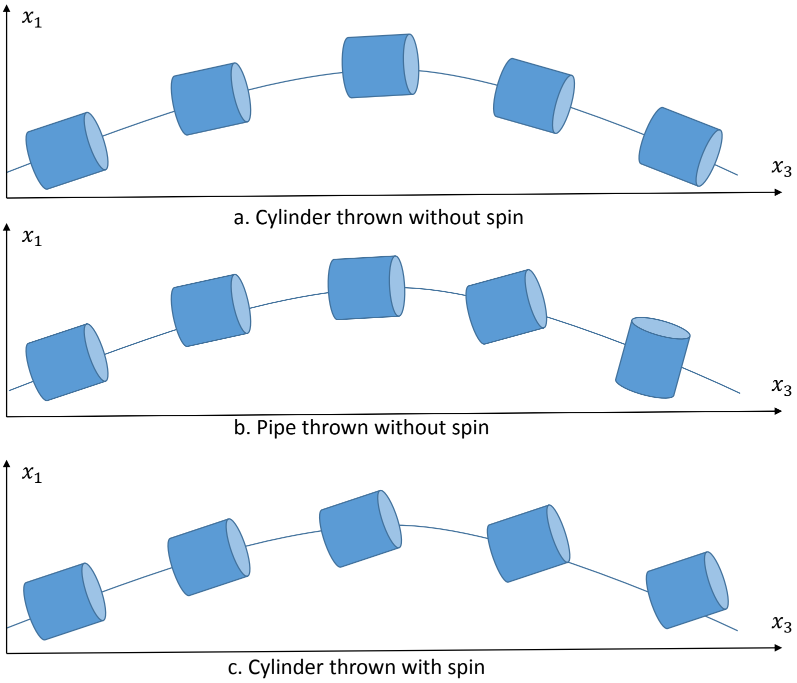

x_1
x_3
a. Cylinder thrown without spin
x_1
x_3
b. Pipe thrown without spin
x_1
x_3
c. Cylinder thrown with spin

Figure 2.12: The dynamic of the orientation for a flying cylinder and pipe depending on the conditions of flight.

In [Fra12] a setup with a 1-DoF linear throwing device and a 1-DoF rotary catching device was used for cylinder transportation. No sensor system was used at all; the thrower provided the sufficient level of accuracy for successful catching. Both robots are controlled via the common control system, and in the time after the throw, the catching movement is actuated. In experiments this system was able to catch all 50 of the 50 thrown objects. The authors admit that this accuracy is only possible for cylindrical objects because of their high aerodynamic stability. Catching less stable objects would require the use of visual feedback [Fra12, p. 5269].

2.4.3 Discussion

An overview of the existing concepts in robotic catching is given in the table 2.1. The rate of success is written in % and, where given, in absolute numbers of thrown and caught balls. These numbers are not entirely correct in the way they compare algorithms because they depend on the setup (e.g. it is harder to catch the ball via grasping than via collecting into basket), but they do help rate the abilities of the existing catchers. The only catcher that overcame the limit of an 80% success rate was in [Fra12] where all thrown objects were caught. The drawback was that the cylindrical objects thrown without spin had very high aerodynamic stability in comparison with the other objects. The authors of [Kim12] assume that the application of the concept to less stable objects will require the use of object tracking in order to adjust the catching movement.

Table 2.1: Comparison of existing approaches to robotic throwing and catching

#	Institution	Sources	Rate of success	Focus and contribution	Objects
1	MIT	[Hov91]	75-80%	First robotic catcher, parabolic fitting	small balls
2	Tokyo University	[Nis97]	Not stated	First humanoid catcher	small balls
3	Tokyo University	[Nam03a]	Not stated	Visual feedback in 1 ms; grasping the flying objects	small balls
4	German Aerospace Center	[Fre01] [Bae10] [Bir11]	67-80%	KF estimation, modeling air drag, catching multiple balls simultaneously	small balls
5	Advanced Telecommunication Research Institute, Kyoto	[Ril02]	Not stated	Tracking the thrower's hand, convergence with human catching strategy	small balls
6	Tokyo University	[Mor04]	Not stated	Catching based on angle of gaze information only	small balls
7	Tohoku University	[Her09]	Not stated	Parabola fitting to the monocular measurements	flipping coins
8	Royal Institute of Technology, Stockholm	[Smi07]	78% (25/32)	Tele-operated ball catching, development of a specific fast catching arm, ball recognition based on HSV color information,	small balls
9	Munich University of Technology	[Bat10]	35%	Non-prehensile catching	basket balls
10	Swiss Federal Institute of Technology, Lausanne	[Kim14]	73% (52/71)	Catching objects with complicated shape, learning-based definition of object acceleration, marker-based tracking	hammers, bottles, tennis rackets
11	Heilbronn University, Vienna University of Technology	[Fra06] [Fra07] [Bar08] [Fra08a]	60%	Industrial transportation by throwing, monocular tracking with relation on physical model, catching by 2DoF gantry cartesian robot	small balls
12	Heilbronn University	[Fra12]	100 % (50/50)	Cylinder-shaped objects, precision throwing	cylinders

It can be seen that the majority of solutions use small balls as an object. The term "small balls" here describes a ball similar to the size of the tennis ball or baseball (in comparison to bigger balls used in soccer and basketball). This term is used because the articles are unclear as to whether a tennis ball was used; however, it can be seen from the images and characteristics of the balls that they were approximately the same size as a tennis ball.

A comparison of the various approaches to object tracking is given in table 2.2. All of the aspects connected with 3D reconstruction are mentioned in the column "sensing solution" as 3D reconstruction relies on a sensor setup. The column "processing solution" corresponds to how the task of recognizing the object in the image is solved. It is not clear how the coordinates of the objects are extracted from the images in some of the solutions [Hov91, Nis97, Mor04]. It may be assumed that similar methods, like motion detection or the Hough transformation, are applied for this task. The division of the camera field of view (FoV) into subfields (Area of Interest, AoI) with smaller pixel sizes is applied in a number of works, e.g. [Bar08, Bat10], in order to reduce calculations. The subfield may be static if it is known that the object cannot appear at some part of the FOV. In [Bar08] this approach allowed the available frame rate to be increased from 87 to 215 fps. The subfield may also be dynamic. In this case the position of the AoI in the image plane is adjusted according to the new position of the object. For example, the vision system from [Bat10] operates with 180 x 180 pixel windows instead of an entire 1280 x 1024 image.

Table 2.2: Comparison of existing approaches to object tracking for robotic catching

#	Sources	Sensing solution	Processing solution	framerate
1	[Hov91]	Stereo vision	Not clarified	30 fps
2	[Nis97]	Stereo vision (unsynchronized), monocular range estimation based on object size	Not clarified	30 fps
3	[Ish96] [Nam99] [Nam03a]	Single camera with direct connection of photodetectors with processing unit	Motion detection on high-speed image sequence	1000 fps
4	[Fre01] [Bir11]	Stereo vision	Comparison with reference image, Hough transform	50 fps
5	[Ril02]	Color stereo vision	Calculating 3D color centroid	60 fps
6	[Mor04]	Monocular vision (No 3D reconstruction at all)	Extracting angle of gaze information	30 fps
7	[Her09]	Monocular vision with parabolic motion model	Not clarified	70 fps
8	[Smi07] [Bat10]	Color stereo vision	Ball recognition based on HSV color information	150 fps
9	[Bar08]	Monocular vision with ballistic motion model	Hough transform	215 fps

The radical approach to increasing the frame rate is proposed in [Ish96], where the bottleneck of transmitting the image from the sensor to controller is eliminated. Each sensor element is directly connected to the specific processing unit. This increased the actual frame rate to 1000 fps. This approach needs a specific hardware solution for the vision system, which would enable parallel processing on a mass scale. Commercially available digital cameras do not allow a direct connection to photo detectors but transmit the sensor data as images. In any event, even relatively slow cameras with 50-70 fps provide useful information about object trajectory [Fre01, Ril02, Her09].

Obviously, there is no reason for the real time application to increase the frame rate if the interframe timeout is less than the period needed to extract the object spatial coordinate from the frame. The Hough transform circle-detection is a well-known technique, which is intended for recognizing spherical objects on images and determining their center positions. Color-based and motion-based techniques only allow specific regions in the image to be recognized. The advantage of motion-based and color-based is the lower computational expense. On the other hand, the results from [Bar08] show that Hough transform could work correct even at the frame rate of 215 fps.

Table 2.3: Comparison of existing approaches to object catching

#	Sources	Catching device	End-effector	Type of catching	Details of catching strategy
1	[Hov91]	4-DoF arm	Not clarified	passive, soft	
2	[Nis97]	5-DoF arm	basket	passive, hard	
3	[Nam03a]	4-DoF arm	Robotic hand with 3 fingers (each with 3 joints)	active, hard	Direct motion definition from vision data
4	[Fre01]	7-DoF arm	basket	passive, hard	
5	[Bae10] [Bae11]	7-DoF arm	Hand with 4 3-DoF fingers	active, hard	
6	[Ril02]	Humanoid	Baseball glove	passive, soft	Learning human-like movements
7	[Mor04]	6-DoF arm	basket	passive, hard	Keeping constant angle of gaze to the object from the point of view on the end-effector
8	[Smi07]	high-speed 6-DoF arm	basket	passive, hard	motion of the end-effector within the planar window
9	[Bat10]	6-DoF arm	planar surface	nonprehensile, soft	
10	[Kim14]	7-DoF arm	hand with 4 4-DoF fingers	active, soft	
11	[Fra07] [Fra08a]	Gantry 2-DoF Cartesian robot	Mechanical gripper	active, hard	
13	[Fra12]	1-DoF rotary arm	Pneumatic gripper	active, soft	

A comparison of various solutions for performing the catching movement is given in table 2.3. The humanoid from [Ril02] has a 4DoF on each arm and a 2DoF on its torso. The catching strategies are classified according to two aspects. First, catching may be hard or soft as specified in subsection 1.2.3. Second, catching may be passive. The end effector has no moving parts. It is designed in such a way that after interception, the object is stored in it, e.g. the end effector is a mounted basket in [Nis97]. Catching may also be active. The end effector has moving parts which grasp the object at the moment of interception. This grasping may be actuated by the robot controller as in [Nam03a], performed mechanically according to the design of the end effector as in [Fra07]), or non-prehensile as solely implemented in [Bat10], where the object was collected on the planar surface of the end effector which was achieved by balancing. The control of multi-joint arms is implemented by means of inverse kinematics in most cases. For soft catching this meant the calculation of the joint positions and rotational velocities, which provided the target position and the velocity of the end effector. If hard catching is applied, the task is simplified. Then there is no need to adjust the velocity of the end effector.

In most of the works, multi-DoF robotic arms are applied for catching. These arms have a 3D workspace and are able to provide soft catching. The use of actively grasping or passively collecting the object into a cup is depends on the robot?s abilities. If there is a task where a robot needs to catch a ball like a human would, the catching robotic hand is applied. The ability to grasp with a hand is a very specific and complicated robotic task, which requires a high degree of precise predictions (2-3 cm according to [Bae10]). Passive catching also allows the object to be collected and the object trajectory to be estimated. In general, controlled active catching requires an additional compositional unit of the system, e.g. the hand in addition to the throwing device, the robot and the cameras. It also requires an additional algorithmic unit, e.g. a controller for the grasping movement in addition to an object tracker, a predictor, and a robotic path planner. Passive catching is sufficient to provide successful catching and an initial validation for other units.

2.5 Trajectory prediction in robotic catching and transport-by-throwing

This section discusses the prediction task and how is it solved within the research work described in section 2.4. Information on the location point of catching within the state-of-the-art approaches is obtained in the following three ways:

1. *Accurate adjustment of the throwing device*: Differences between the various trajectories lie within the allowed error of catching. Thus, neither prediction nor observation of the trajectory by the tracking system is needed [Fra12]. These systems are discussed in subsection 2.5.1.

2. *Direct mapping* from the vision system to motion control without the intermediate step of prediction and the definition of the catching point [Nam03a, Mor04].

3. *Long-term prediction*: when the part of the trajectory within the gripper workspace is estimated based on the initial part of the trajectory: Most of these predictors use the predefined physical model of object flight, e.g. [Hov91, Nis97, Fre01, Ril02, Bar08, Her09] (summarized in subsection 2.5.2),). However, in some solutions the learning operation is

inserted [Kim12]. In subsection 2.5.4, an introduction to the nearest neighbor prediction is given. The development of the nearest neighbors approach for trajectory prediction is one of the main contributions of this thesis; therefore, arguments for its use are also given in this section.

2.5.1 Catching without prediction

In a throw the challenge primarily lies in transporting the object to a certain predefined area. In the case of non-predictive transport-by-throwing, the object reaching this area must guarantee a successful catch. Throwing objects is a specific robotic task developed in a number of solutions, e.g. [Miy10, Kob11, Nem11, Zha12, Kan12]. Most of these works did not concentrate on providing stable object transportation by throwing. In [Zha12] the main scope lies in specific path-planning for the robot, and throwing is considered to be one of possible applications. A similar task was considered in [Kan12] for a humanoid robot with a high number of DoFs. Therefore, in most of these, it is only stated that ”the object was thrown towards the target by the throwing system”, and there was no mention of how accurate the throw was although the applicability of the throwing strategy in industrial applications is given.

The accuracy aspect was considered in [Miy10] and [Nem11]. In the first work, a rotational 1-DoF lever is used as a throwing device. The trajectory is changed by changing the position of the object on a lever. The error norm was 3 mm after 21 throws. The distance of the throw is not given; however, from the illustration it can be seen that it is no more than several tens of centimeters. In [Nem11] 5 cm precision in throwing balls into a basket was achieved. The errors were blamed on vision inaccuracy. The purpose of this work was to develop the new technique for learning motions, and throwing balls into a basket was one of the examples.

Within the TbT project, precision throwing is developed by Thorsten Frank, et al [Fra09, Fra11, Fra11a, Fra12]. This is the only research project founded in the literature, which includes the deep exploration of transportation by high-precision throwing. In [Fra11a] a 1-DoF linear throwing device from [Fra09] is replaced by a specific 2DoF robot. The cup with the object lies on the end of a lever that is 50 cm length. The throw is implemented by the rotary movement of this lever. The lever can achieve a specific angular velocity, and if it decelerates from this value, the launch takes place. The launching velocity of the object is set by the angular velocity of the lever at the moment of deceleration, and the angle of the throw depends on the orientation of the lever at the moment of deceleration. The maximum available launching velocity is 25 m/s, though in the experiments the velocity was usually between 8 and 12 m/s. The minimum velocity of 8 m/s provides the centrifugal force, which is more powerful than gravity, so the ball does not fall out of the cup before the launch. The definition of the launching position in order to reach a specific destination area is based on the gravity-drag motion model specified in [Chu03]. In the experiments, the robot is able to throw the balls to holes of a specific size. The size was not clarified in the paper, but from the illustration it seems as thought they were several tens of cm in diameter.

In [Fra12] a 1DoF device was used to throw cylindrical objects. The model of object flight used for the adjustment of the throwing device was developed in [Fra10]. The orientation of the cylinder follows its velocity vector due to the effect of shoulder stabilization (see figure 2.12,a) so a cylinder is a stable aerodynamic object that can be considered as a point-mass. The flight of small and heavy cylinders may be accurately represented using even parabolic models [Fra10]. The prediction of lightweight cylinders of a bigger volume requires air drag to be taken into account. In fact, the task of adjusting the launching parameters in [Fra10, Fra12] is inverse to trajectory prediction. There the planned future trajectory is used as an input to calculate the required launching parameters but not vice versa. The advantage is that if the model is a little bit inaccurate the launching parameters can be tweaked to remove systematic error. Also, complex calculations (e.g. the use of Levenberg-Marquardt optimization in [Fra10]) are allowed because they are not performed in real time during catching. The gravity-drag model with an iterative calculation was used for trajectory modeling in [Fra12]. The high stability allowed 100% success to be provided in catching after 50 throws.

The applicability of precision throwing is limited by several factors. First of all, the shape of thrown object must provide its aerodynamic stability. The design of the throwing device must minimize deviations. The method is very sensitive to the respective location of the throwing device and the destination point. For each of these positions, a new hard adjustment for the thrower and catcher must be made. If soft catching is used (the only known implementation of precision throwing with soft catching is [Fra12]), then precise knowledge about the time of the object's flight is required. Due to these factors, the use of visual feedback would be preferable for objects of a normal shape and size.

In vision-based catchers, the prediction of object trajectory may be skipped if the robot's reaction to the visual information is considered to be instantaneous. In [Mor04] catching is realized even without a 3D reconstruction of object position. The camera is mounted on the gripper, and the movement strategy consists of keeping the constant angle of gaze of the object in the image. This strategy does not take the absolute velocity of flying object into account, so it is not applicable for soft catching.

If the 3D reconstruction is applied, the strategy may consist of moving the robotic hand towards the measured position of the object. In [Nam99] this strategy was applied to an object moved by a human hand and for a flying object in [Nam03a]. This approach is useful when the process noise is high, so there is no point in a long-term forecast. However, the process noise for a free-flying object is relatively little. It is not possible for the robot to react quickly if the velocity of the object and the distance towards it exceeds the velocity limits. The maximum velocity of the KUKA LWR+ end effector is about half that of the object's velocity, so it cannot react instantaneously in its movement.

2.5.2 Prediction using physical models

A comparison of existing prediction techniques is presented in the table 2.4. There the abbreviation LS refers to the Least Squares estimation, and the abbreviation EKF refers to Extended Kalman Filter. In the third column, the rows with the term "gravity" or "gravity and drag" mean that the respective motion models take respective forces into account. The term "SVR with RBF kernel" (Support Vector Regression and Radial Basis Function respectively) describes the learning based method of estimating accelerations in [Kim14]. Acceleration forecasting mean that the acceleration vector is considered to be constant:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix} = const, \tag{2.20}$$

where $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ is a vector of tangential acceleration and $\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix}$ is a vector of angular acceleration of a rotating body.

Table 2.4: A comparison of existing approaches to object trajectory prediction based on mathematical or physical models

#	Sources	Estimator	Motion model
1	[Hov91]	LS	parabolic
2	[Nis97]	LS	parabolic
3	[Fre01]	EKF	gravity and drag
4	[Ril02]	LS	parabolic
5	[Her09]	LS	gravity
7	[Bat10]	LS	gravity
9	[Bar08]	EKF	gravity and drag
10	[Pon09]	LS	gravity and drag (axial separated)
11	[Kim12]	SVR with RBF kernel	Acceleration forecasting

The parabolic model is in fact constant acceleration forecasting. Acceleration in this case is considered to be equal to $\mathbf{g}$ [Bat10] or found by fitting the parabola to the measurements [Hov91]. Angular accelerations are ignored in [Hov91, Nis97, Bat10] as point-symmetric bodies are considered, so orientation forecasting is not required. In [Her09] it is also ignored as the object, a coin, is very small and may be considered to be a point-mass.

The parabolic model allows the object's trajectory to be defined based on three measured points even without knowing the value of g. With the assumption that g is known, it may be done based on 2 points. Usually, the higher number of measurements is used in order to improve accuracy by the statistical estimation. The value of g is obtained by parabola fitting in [Hov91, Ril02]. In fact, the task is to approximate the data with a second-order polynom.

Making g unknown allows the influence of the air drag to be included in the model. In the phase of ascent, the influence of drag is co-directional with gravity. In the phase of descent, its influence is in opposition to gravity. The polynomial model makes the assumption that the overall acceleration of the object is constant. This assumption may be valid if the object generally has no significant change in vertical velocity in the part of the trajectory considered. Either that or it at least does not change its direction; the object must either ascend or descend throughout the flight. The most energy effective and flexible way to provide object transportation by throwing is to throw the object upwards and to catch it during descent. If the object is thrown downwards, it limits the maximum distance of the throw and can restrict the environment's organization. This is because the thrower must be located above the catcher. If the object is caught during its ascent, this means that there is excessive energy consumption because the actual launching velocity is much higher than what is needed to reach the destination. The gripper in this case must be located above the thrower. In the case of an ascending throw and a descending catch, the thrower may be higher, lower, or the same height as the catcher. For these flights the polynomial models lead to high systematic errors [1].

As it was discussed in section 2.5.1, inserting drag into a model of object motion leads to the differential equation 2.10that has no analytical solution. The prediction of a trajectory based on this model may be done by an iterative calculation of the coordinates as in [Fre01, Bar09] or by using the simple assumption about the independence of the motion in various coordinates as in [Pon09]. The value of the drag coefficient is required when using these models and must be known. As discussed in section 2.1, determining this coefficient could be a significant challenge even for a simple-shaped body. For objects of a general and not-so-aerodynamically-stable shape this determination becomes a real problem, especially because the coefficient depends on object orientation to air stream. At the current stage of research, primarily simple-shaped bodies are used, but further development implies objects with various shapes must be thrown. Due to this, it is better not to rely on drag coefficients.

The polynomial motion models, as opposed to those discussed above, do not require the physical coefficients to be known (k as well as g). The motion of the object is expressed by 2.15 where all the coefficients are obtained by fitting to data. This model allows the influence of gravity to be included as well as all the aerodynamic factors on the object. The final impact of these factors is assumed to be quadratic, which is also a simplification of the assumptions. In [Pon09] this model demonstrated its accuracy, but the accurate estimation of weighting coefficients for 2.15 requires computationally expensive multi-dimensional optimization.

Statistical filters, especially the extended Kalman filter ? EKF, are used to decrease error influence in [Fre01, Bar09] and least square (LS) fitting in [Hov91, Pon09]. For filtering, a working model of the process is required. It might not be very accurate, and in this case the systematic error component of prediction grows with an increase in distance. In the LS estimator, the best fit for all the measured points is found. In comparison, EKF is a Markovian estimator, i.e. the forecast is done based on the last estimate of the object's position and velocities and the last measurement of its position. The previous measurement's effect on the result is only a source of the last estimate. Therefore, errors connected to model inaccuracy may be a result of the process noise.

2.5.3 Neural network prediction

Neural networks allow learning by example without an accurate model of the process and consider unknown nonlinear dependencies in the training set. For forecasting the time-series data, various time-delays neural networks (TDNN) may be used. TDNN was proposed in [Sej87]. Neural networks have several properties, which make them useful and promising for TbT application:

- No exact prior models of the process are needed. The network creates the predicting model itself during the learning process.

- Neural networks can easily deal with nonlinear processes.
- Although a learning process may take a while, the trained predictors can work very quickly, especially if they are implemented in the hardware.

The typical architecture of the TDNN is shown in the figure 2.13. The network input consists of the reference of measurements from various times provided by the digital delay line. The time delay line consists of delay elements Z^{-1}. Each element provides the delay by a certain time unit τ hence a reference of objects coordinates $\{X(t_0), X(t_0-\tau), X(t_0-2\tau), ..., X(t_0-n\tau)\}$ is given to the network input. The network should predict the value of these coordinates $X(t_0+t_1)$ in a fixed period of time t_1 after the current moment.

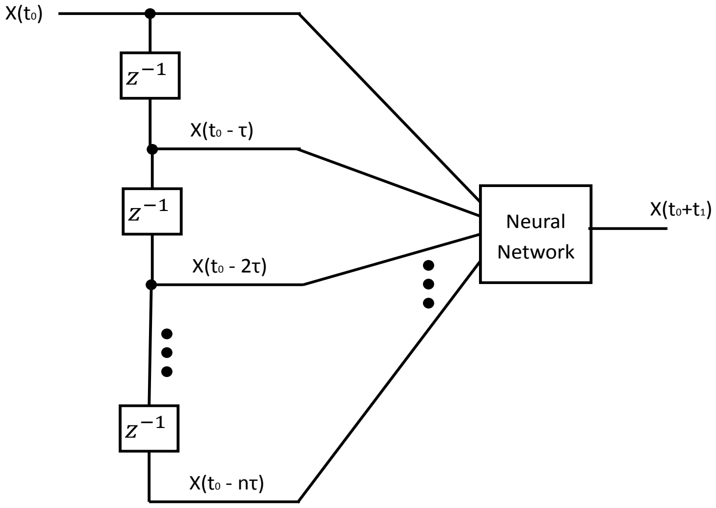


Figure 2.13: The typical architecture of a Neural Network Predictor.

If the object's trajectory is observed, the input will be a sequence of measurements extracted from the observer's frames [Mir13]. τ equal to the observer inter-frame period or to the observer's frame rate at the power of 1. t_1 is equal to the time it takes for the object to reach the gripper's workspace.

The neural network trajectory predictor was developed and simulated by the author of the thesis in 2013 [Mir13]. As the initializing parameters were unknown, forward networks were used in the simulation feed. The simulation software and the prediction model were made using MATLAB. Two types of predictors were proposed. The first predictor consists of three separate networks, each with one dynamical input and one output. Each of the networks predict one of the three spatial coordinates. The second predictor includes one network with three references of inputs and three outputs, predicting the values of all three coordinates [Mir13].

Increasing the number of layers to more than two, causes processing time to be increased and does not really increase accuracy. Hence, for a time-critical trajectory prediction application, a neural network with one hidden layer was proposed. The number of neurons that provide the best accuracy for each network was defined in preliminary experiments: 6 neurons for predicting the value in x_1, 2 neurons for predicting the value in x_2, 6 neurons for predicting the value in x_3, and

4 neurons for complex prediction. The standard deviation for the spatial separated prediction was from 26 mm for a highly precise simulated throw up to 74 mm. For a throw of low precision, the mean error for complex prediction was from 24 to 89 mm respectively [Mir13].

The results have also demonstrated some additional aspects of NNTP [Mir13]:

- When HPT is used, the Neural Network comes up with results that can be used in the work of a real catching system.
- More precise throwing causes more precise catching.
- Spatial-separated predictor showed better accuracy than complex predictor.
- The use of more dispersed learning samples provides better training.
- The more careful the measurement of the target parts of the learning trajectories is, the better the training can be.

Processing 100 frames took about 0.7 ms for a complex neural network and about 1.2 ms for a spatial-separated model even within the relatively slow MATLAB application. Such parameters allow the prediction to be recalculated after each new frame. The error size could be rejected by the accurate adjustment of the throwing device, but the disadvantage of the NN is that it is, in fact, a black box. It is hard to interpret the results as NN creates the output function itself based on the training datasets. In this case it will be hard to adapt the prediction mechanism to the changing parameters, e.g. frame rate of the measuring system, shape of the object, etc. A simple and interpretable solution is desirable for the initial construction of the flexible sample-based predictor. One of the simplest and most interpretable methods for sample-based decision-making is k Nearest Neighbors (kNN).

2.5.4 Premises for use of nearest neighbors algorithm

The algorithm k Nearest Neighbors (kNN) was initially invented as a technique for pattern classification [Cov67]. Since the 1960s, it has developed into a large family of various algorithms used for various tasks [Bha10]. One of these tasks is to forecast time series (TS). Formerly, use of kNN algorithm for forecasting was studied in [Yak87]. Since then, it has been applied for various tasks that have to do with time series, e.g. weather forecasting [Acc03], predicting road traffic [Guo12] and electricity demand [Alq13, Fuj14], and electrocardiogram analysis [Fae08], etc.

For the nearest neighbors algorithm, the dataset of samples is collected prior to solving the task. The classification task means that the need to associate the input data vector with a specific category exists (the set of categories is finite). For the kNN classification, the dataset has the view $\{\mathbf{X}_1, Y_(1)\}, \{\mathbf{X}_2, Y_(2)\}, ..., \{\mathbf{X}_n, Y_(n)\}$ where $\mathbf{X}_i$ is the input data vector and $Y(i)$ is a category taken from $C = \{Y_1, Y_2, ..., Y_k\}$. When the classification of the current sample $\mathbf{X}_c$ is done the task is to define $Y(c)$. If one nearest neighbor (1-NN) rule is used, vector X_i from the dataset is taken, which is the most similar to X_c (it is called ”nearest neighbor”), and its category $Y(i)$ is associated with the current input. ”Nearness” or ”Similarity” here means that the distance (Euclidean, Manhattan, Mahalanobis, etc.) from X_c to X_i is smaller than that to any other vector from the dataset. If the number of taken nearest neighbors is more than one, each of them votes for its category, and the category with the maximum number of votes is associated with X_c.

The Weighted k Nearest Neighbors (WKNN) classifier was proposed in [Bai78]. In this algorithm the vote of each neighbor's weight depends on the distance to X_c.

The kNN regression algorithm differs from the kNN classifier in the following way: Y here is taken not from the finite set of categories but from the infinite set of real numbers. It could be a scalar or a vector $\mathbf{Y}$. The value of $\mathbf{Y}_c$ in kNN regression is calculated as a mean of neighbors' target values:

$$\mathbf{Y}_c = \frac{1}{k}\sum_{j=1}^{k}\mathbf{Y}_j. \tag{2.21}$$

In WKNN regression weighting coefficients are added into the model:

$$\mathbf{Y}_c = \sum_{j=1}^{k} w_j \mathbf{Y}_j. \tag{2.22}$$

Typically WKNN considers all neighbors instead of just the closest, but the weighting coefficients may be applied for a finite subset of examples also. In [Bai78] all the samples from the dataset vote for their categories. In other implementations of the algorithm, only a finite number of nearest neighbors is taken in mind. This allows computational expenses to be decreased.

Time series forecasting is partially a case of regression. Here $\mathbf{Y}$ and $\mathbf{X}$ represent the same parameters but $\mathbf{X}$ corresponds to its values measured in the past, and $\mathbf{Y}$ corresponds to its unknown future values.

In comparison with other techniques, kNN is very promising. The main adbvantages of the kNN are listed below:

1. *Independence from physical model*: The kNN is purely experience-based approach. It does not require any knowledge about the rules of objects' motion. This is its advantage in comparison with analythical methods (polynomial fitting, physical modelling, Kalman filters)

2. *Similarity to human learning*: The training principle is similar to how humans learn to catch balls. They collect the knowledge by remembering their previous trials and decide how to catch the ball by remembering the similar examples from the previous experience [Gil99]. Humans are better ball-catchers than any existing robotic system, therefore converging their strategies is a promising way for improving the system.

3. *Simplicity of the learning process*: In common version of the algorithm previous samples are just stored in the database. In the version described in the next chapters some transformations are added into the learning algorithm, however the main principle stay the same: learning is collecting previous examples in the memory. This is opposite to other intelligent algorithms (e.g. neural networks) where learning means adjusting parameters of a complex model to the sample data. This is a reason for the next advantage.

4. *Good interpretability*: It is easy to determine the reason for the decision made by the predictor. "we do this forecast because our trajectory C is similar to trajectories of A and B from the database". This reasoning is attractive for the task of trajectory forecasting. The process of an object's flight is deterministic, and in similar conditions the trajectories look similar.

5. *Freedom from overfitting*: The problem of overfitting occurs in statistical (e.g. polynomial models or Kalman filters) or learning-based methods (neural networks) where any motion models are created. These models have certain parameters, which are adjusted to the sample data. If sample data are erroneous the random deviations of the parameters may be detected as deterministic. In this case prediction model works good with the samples from the learning sampling, but gives high errors while predicting the new trajectories. As no specific process model is used in kNN, it is free from such a problem.

6. *Simplicity of the forecasting operation*: In case of one nearest neighbor the value of the forecast is simply equal to the corresponding value of the neighboring trajectory. In case of higher k forecasting include the operation of addition. No complicated mathematical operation like curve fitting is required. Humans also do not make complicated calculations in order to catch the ball, so it is possible to avoid complicated models and forecast the trajectories well.

The main issue in implementing an NN-based predictor is due to the requirement that the current trajectory must be compared with all the trajectories from the dataset. This means that if the dataset is large, the volume of computations will be incredibly large. This means the memory requirement for processing the datasets would also be quite large. To overcome these issues, various improvements to the basic algorithm are made in [Bha09]. In [Bha09] the approaches primarily deal with classification instead of regression. All of the versions of the algorithm are divided into structured and structureless techniques. In the structureless techniques, the database is considered to be an integral set, and the same procedure of distance calculation is applied to all samples. The size of the dataset is decreased by eliminating the samples that are overly similar to other samples, and this means the potential information that they store is already contained in other samples [Chi79, Alp97, Ang05]. In structured techniques the dataset is transformed so that the algorithm can be executed more quickly. Usually the set is partitioned using of some tree-based partitioning techniques [Spr91]. The leaves of the tree contain information about neighboring samples and internal nodes are used to guide a quick search for the nearest neighbors. Most of the tree-based partitioning techniques deal with leaves formed by the member of specific class, and are therefore inapplicable for regression and time series forecasting. The tree structure is discrete, so it may lead to discretization errors.

In general, the kNN algorithm is a promising technique for trajectory forecast. What it lacks due to the increase in the volume of calculations is eclipsed by the specific advances it demonstrated during the research. A more detailed description of the development of these changes is given in chapter 4.

3 Observing object trajectory

The final evaluation of the prediction algorithm is done based on catching experiments. The percentage of successful catches shows the usability of the whole approach, but this does not tell us anything about the quality of the separate parts of the system: observer, predictor, path-planner, etc. This information may be obtained in a numerical simulation of the process if it represents the process more-or-less accurately, or by comparing the results with available ground truth data, i.e. the data that is measured with high precision and is independent of the prediction results. Therefore, information on the process of the flight and its observation is required.

This chapter concentrates on an exploration of observer accuracy. The experiments that are described aim to determine what will be the measurement error's influence on the predictor and how this influence can be decreased. The prediction itself is examined in chapter 4. Another significant aspect of this chapter is the description of the trajectory datasets. Learning-based prediction requires a set of example trajectory measurements. This set is acquired during the throwing experiments described in this chapter.

The throwing experiments are described in section 3.1. This section mainly contains the technical information about the throwing device, the cameras and the algorithms that allow the three-dimensional (3D) model of the trajectory to be obtained from the camera measurements. This information is the base with which an exploration of the accuracy in further sections will be done. In section 3.2 , the positioning accuracy for a static ball on a single image pair is discussed. In section 3.3, the accuracy model is extended for the tracking of airborne balls.

3.1 Throwing experiments

Throwing experiments have two main goals. They provide a dataset of sample trajectories for predictor learning and allow the properties of the observed trajectories to be explored, most importantly the observer accuracy. Experiments were conducted based on the available setup at the Institute of Computer Technologies at Technische Universitaet Wien. This setup consists of the throwing device and two digital cameras (IDS uEye UI-3370CP with 2048 x 2048 resolution) integrated into a stereo pair. Both cameras and the throwing device are controlled from a desktop personal computer (PC). The control of the cameras and the throwing device is performed via a universal serial bus (USB) and an RS-232 interface respectively. The cameras are triggered by the single generator and are synchronized. The throws were done by the device described in subsection 3.1.1. Subsections 3.1.2 and 3.1.3 describe how the flight is observed by the vision system and how the reference of the object 3D positions is extracted from the camera images. Subsection 3.1.4 contains an overview of the acquired sample trajectories.

3.1.1 Throwing

The throwing device was constructed by Martin Pongratz and Gilbert Markum [17, 18]. A photo of it is shown in figure 3.1. The energy of throw is provided by the tension of two springs, which pull the catapult cup with the ball. The tension is provided by the electromechanical drive which is connected to the cup by the solenoid magnet. The throw takes place if the controller switches to the solenoid. It releases the catapult cup. The motor may adjust the launching position to reach the needed values of tension.

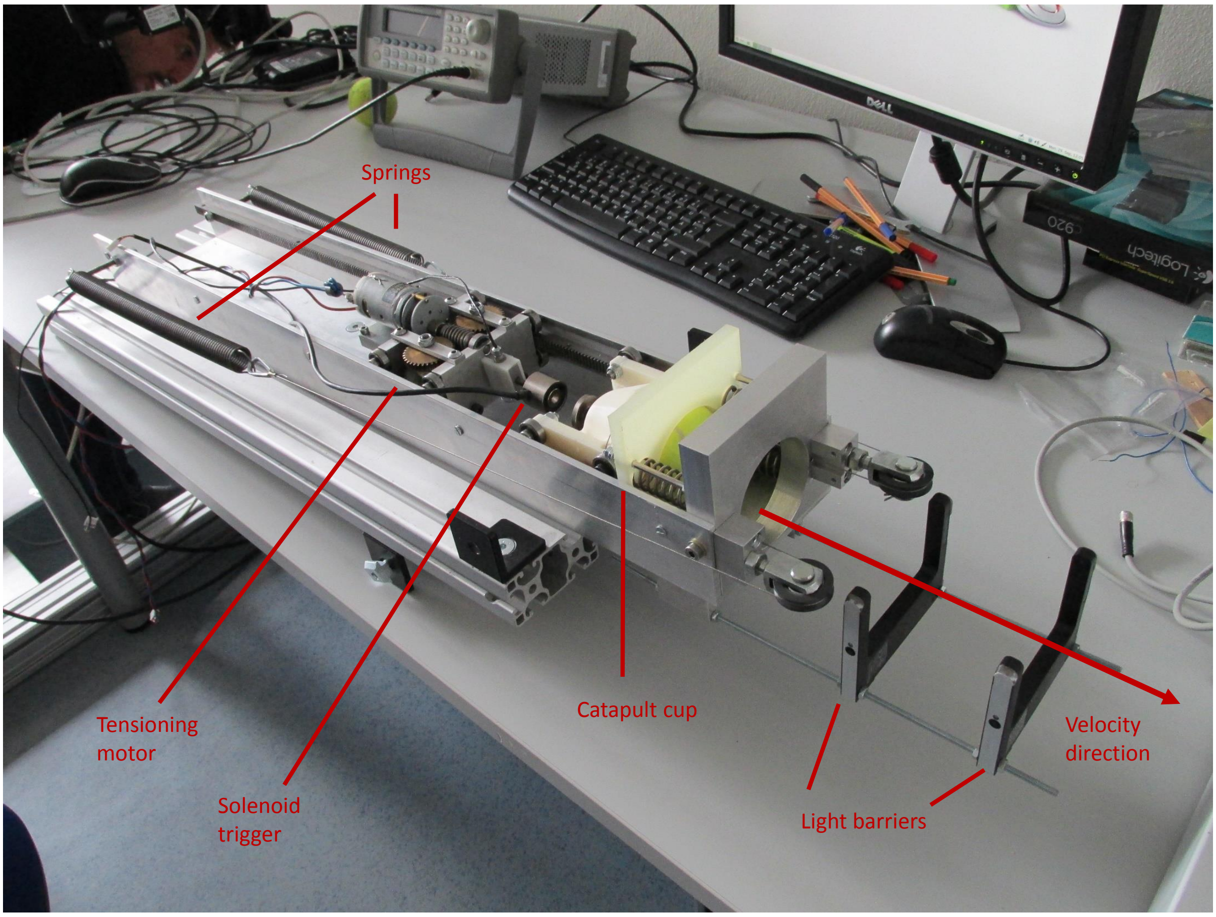


Figure 3.1: Throwing device used in the experiments.

To explore the trajectory, it is valuable to know what the velocity of the thrown ball was. If the vector of the throwing velocity of the object is the same every time, the trajectory of the object will not vary from time to time according to the motion models 2.10 and 2.8. TA real thrower deviates in the throwing velocity, therefore, the actual trajectory may vary from time to time. The variation in the measured launching velocity can give a perception about the variation of the trajectory. Also this variation can be used as a base for modeling the trajectory deviations within the process simulation.

The velocity is obviously dependent on the spring?s tension. At the moment of release, the potential energy of the tightened spring E_p transforms to the kinetic energy of the flying object E_k. The relationship between the energy, spring tension and object velocity is summarized in the equations 3.1 and 3.2 that are simple school physics.

$$E_p = \frac{kd^2}{2}. \tag{3.1}$$

$$E_k = \frac{mv^2}{2}. \tag{3.2}$$

Here k is the restitution coefficient, d is the linear extension of the spring, F is tension force, m is object mass, v is the value of object velocity. The tension force is equal to the product of linear extension and restitution coefficient according to Hooke's law:

$$F = kd. \tag{3.3}$$

As a combination of equations 3.1 and 3.3, the relationship of the spring's energy and tension force is:

$$E_p = \frac{F^2}{2k}. \tag{3.4}$$

As the values of E_p and E_k are ideally equal, according to the energy conservation law, the known value of the force allow the launching velocity to be estimated:

$$v = \frac{F}{\sqrt{km}}. \tag{3.5}$$

In reality, the value of v will be a little bit lower due to energy losses. In the calibration experiments, the object was thrown with a specific force, and light barriers were used to measure the launching velocity that corresponds to this force [Mar16]. The following empirical dependence between tension force (in newtons) and launching velocity (in meters per second) was determined in these experiments [Mar16]:

$$v = 0,1179F - 3,759. \tag{3.6}$$

The opposite equation may be used to define tension force from the nominal velocity [Mar16]:

$$F = 8,48v + 31,92. \tag{3.7}$$

The maximum nominal throwing velocity of the machine (achieved with maximum tension of the springs) is about 10 m/s and corresponds to 115 N according to equation 3.7.

Two light barriers are positioned in front of the cup in 10 cm one after another. They signalize when the object y through them. Time difference between interceptions of the ball with light barriers allow estimating the actual launching velocity of the object (see equation 2.18). A comparison of the velocity values measured by the light barriers system with the nominal tension values shows that deviations in launching velocity achieve 0.5 m/s, e.g. when the nominal velocity was set to 4.5 m/s, the actual values measured by the light barriers were distributed in a range from 4.2 to 5.1 m/s. A comparison of the measured and nominal throwing velocities based on the throwing experiments is given in table 3.1 and figure 3.2. In figure 3.2, the histograms showing the deviation of the measured velocity from the nominal value are plotted, and in table 3.1, the numerical values for the systematic error component and the standard deviation are given.

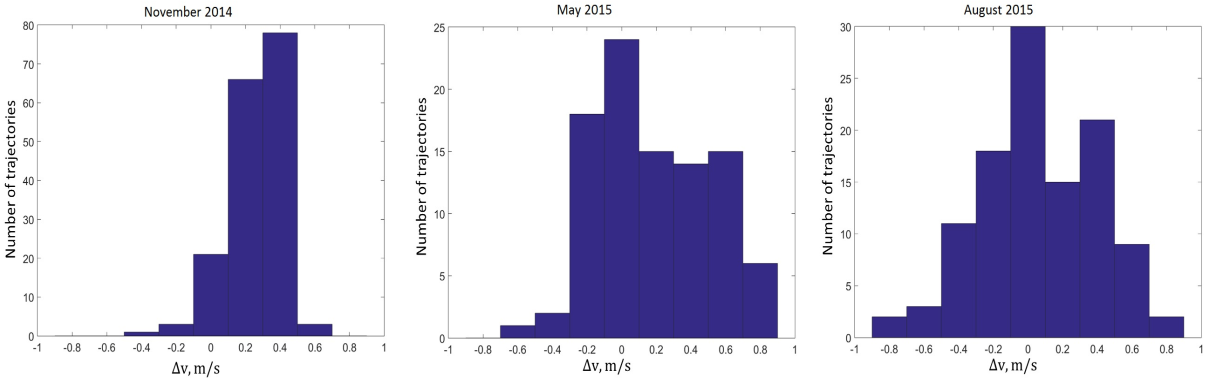


Figure 3.2: The histogram of throwing deviations for three series of experiments.

Table 3.1: The deviations of the throwing velocities for the three series of experiments.

Dataset	November 2014	May 2015	August 2015
Number of samples	169	95	111
Systematic error	-0.25 m/s	-0.19 m/s	-0.06 m/s
Standard deviation	0.16 m/s	0.33 m/s	0.33 m/s

These results show that both the systematic and random component of the errors achieve several tens of cm/s. A systematic error may be connected with the change of spring elastic properties with time and calibration errors. It may be seen that the systematic error component becomes smaller with time, but it in fact seems to be an accidental correspondence. The standard deviation in second and third series become worse. The reason could be that the adjustment of the mechanical components of the device become worse with time (e.g. the springs are deformed with time). Last set of the experiments made after creating all the datasets showed that finally the catapult precision became extremely bad: the actual velocity was achieving more than 6.5 m/s while the nominal one was set to 5 m/s. Before collecting each new dataset, the throwing device was greased and adjusted. Note that this data shows the deviation of the velocity component in the direction perpendicular to the planes of light barriers. Improvements to the quality of the catapult is not the goal of this thesis. This is particularly due to the fact that in further development of the system it could be replaced by a robotic thrower. This information is of note to understand the deviations in available throwing devices. The frontal light barrier on the throwing device is used to detect the moment the throw has begun. Interception of the ball with it is triggering the camera system.

3.1.2 Tracking

The task of the observer is to acquire reference images that include the flying ball and to extract a reference of object positions in space from these images. At the learning stage, this reference is required as a sample of the trajectory. When the transportation system is functioning, the reference of the 3D coordinates at the beginning of the trajectory are used as an input for predicting further values. The process of image acquisition is the scope of this subsection, and the extraction of the 3D coordinates is examined in the next. The main issues of tracking are the need to detect the object in the image and the hard time constraints, e.g. the current frame must be processed before the next one arrives.

Two monochrome cameras with a resolution of 2048 x 2048 pixels (IDS uEye UI-3370CP) are used to observe the flight. They are connected to a common desktop personal computer (PC) via the USB port. The image processing application from [Goe15] was used in the system. Processing 2048 x 2048 images takes some time. This limits the available frame rate to 80 fps [Goe15, p.36]. Time expenses are due to data transmission via the USB cable and the complexity of the image processing algorithms. The influence of both of these factors is strongly related to the size of the image. To increase the frame rate limitations, just a small area of interest (AOI) is processed by the algorithm not the entire image.

There are basically two AOI: the camera AOI (C-AOI) and the algorithm AOI (A-AOI). The C-AOI is a part of a camera sensor that is used to capture the image, and the A-AOI is the part of the acquired image where the search for the image of the ball is done. Due to the limitations of the camera hardware, the C-AOI could not be changed during tracking; it can only be set prior to the procedure. Another hardware limitation is that the borders of the C-AOI may only be set in one dimension of the image. In another it should cover all 2048 pixels. The size of the C-AOI is 800 x 2048 pixels [Goe15, p.36]. The 800 x 2048 pixel images are sent from the camera to PC at the available frame rate of 110 fps.

The A-AOI is a part of the C-AOI. It is a sub field in the camera image, which is processed by the circle detection algorithm. The size of A-AOI is 300 x 300 pixels [Goe15, p. 37]. The initial position of the A-AOI in the image is set in such way that it covers the area of the second light barrier, i.e. the ball will reach it right when the throw has begun. After processing each new frame, the position of the A-AOI in the image is recalculated according to the new position of the ball. This hierarchical use of the AOI concept allows the frame rate to be kept at 110 fps. The illustration of the C-AOI and moving A-AOI is given in figure 3.3.

The new position of the A-AOI in the image is defined from the previous one. For this purpose, the fast algorithm of ball positioning in the image is applied. A sample image of the scene is made prior to the throw. After receiving the new frame, the fast tracking algorithm calculates the difference between the last image and the part of sample image that corresponds to the current A-AOI. The centroid of a difference image is assumed to be the pixel position of the ball center. It is inserted into a linear Kalman filter which determines the most probable ball position in the next image. This position is set as the center for the new A-AOI. Ideally the difference image is equal to the ball's appearance, but in reality this is not the case due to the flickering of the lightning system and the influence of the background objects. Various objects in the scene have different brightness values compared to the ball. When the ball is flying behind objects that have a similar level of brightness, work of the algorithm is distorted. The fast tracking algorithm is accurate enough to avoid dropping the ball out of the AOI, but it cannot provide accurate data about the ball's position in the image.

Despite that, background subtraction is a helpful stage in image processing. It decreases the influence of the background objects on the accuracy of the ball positioning. How an image looks after background subtraction is shown in figure 3.4.

Accurate ball positioning in the image is implemented in two steps. First of all, the Canny edge detection algorithm [Can86] is used to get a so-called ?edge image?. Edges are points in the images, where the intensity (lightness of the pixel) changes drastically. In a simple edge image, each pixel contains a zero if there is no edge at a point or a one if there is an edge at the point.

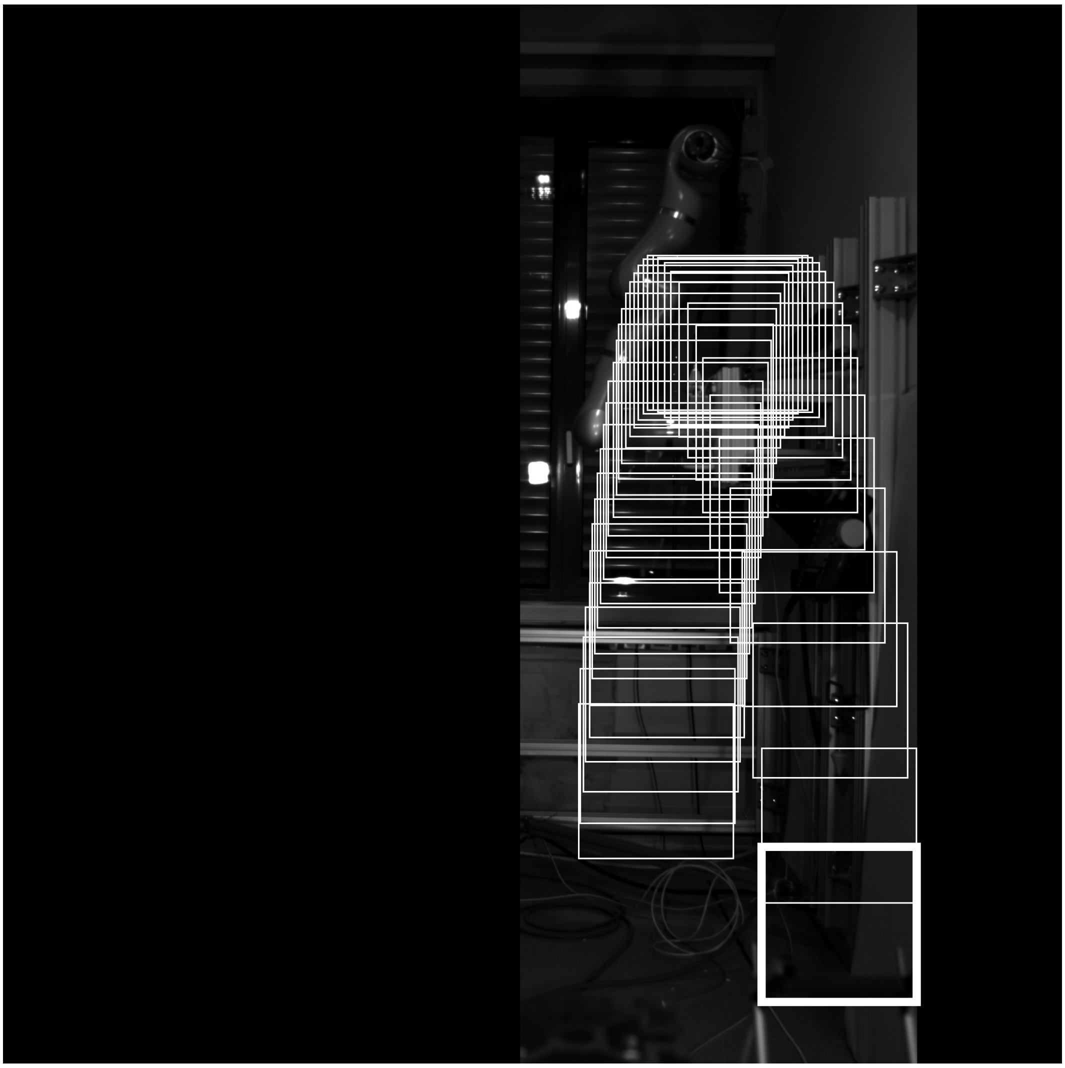


Figure 3.3: Illustration of the AOI usage; the black areas in the figure correspond to the parts of the camera sensor matrix, which are not used for image acquisition; the rectangle drawn with bold lines shows the initial position of the A-AOI in the image; it lies strait above the frontal light-barrier; rectangles drawn with thin lines show the reference positions of the AOI re-calculated during the flight, the position of the AOI in the image first ascends with the ascending motion of the ball and then descends with the descending motion of the ball.

In advanced edge images, the value in the edge pixel shows the direction of edge to be normal at this point. Edges may correspond to the contour of the objects. In the case of ball tracking, they show the circle corresponding to the contour of the ball. Example edge images are shown in figure 3.5. In the first row of results, the Canny edge detection is applied to the input image. In the second row, it is applied to the difference image. It can be seen that background subtraction decreases the number of noise edges (i.e. edges, which do not correspond to the ball contour), however it adds noise inside the ball.

The second step consists of positioning the ball center inn the edge image. Each edge point shows the possible line that could include the circle radius. Therefore, the three points of ideally accurate edge image (i.e. an edge image, which precisely represents the circle contour and nothing more) are enough to define the circle center position in the image. A real edge image is distorted by the

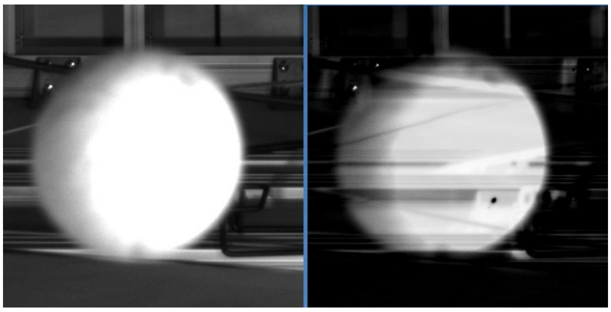


Figure 3.4: Results of the background subtraction: the original image is on the left, and the image after subtraction is on the right. It can be seen that background objects are still visible in the image to the right. This is an effect of the blinking illumination (outside the ball's image) and is the result of subtracting it from the monochrome ball surface (inside the ball's image). The circular image is in fact the difference between the ball's intensity and the background covered by the ball.

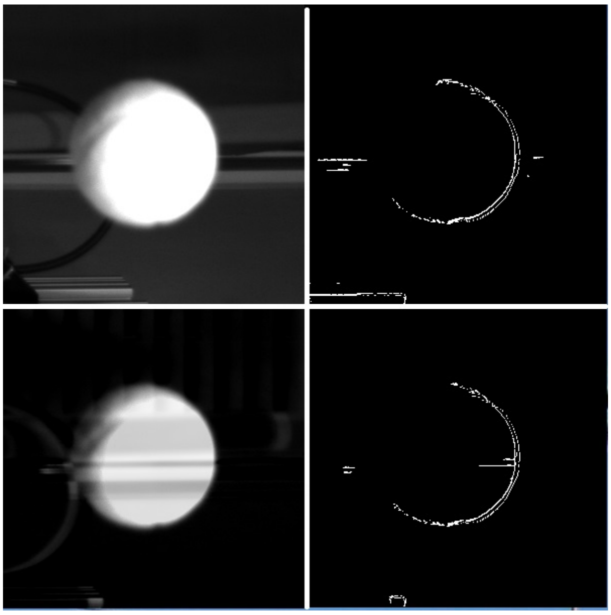


Figure 3.5: Example results of the edge detection for the same image with (higher row) and without (lower row) background subtraction.

edges of background objects, ball texture and the shadows of the object. Specific algorithms are intended to extract circle centers from such noisy images.

A Hough transform (proposed in [Hou62]) is a common approach for extracting features from a large array of noisy data (particularly for extracting shapes from the images). In this approach the required feature (a circle) is defined by the minimum set of parameters (for the circle these are pixel coordinates of the center and radius $\{\begin{pmatrix} w_1(0) \\ w_2(0) \end{pmatrix}, r\}$). These three parameters are forming the 3D space, which is called the accumulator [Hou62]. The accumulator is discretized i.e. it is divided into cells of a $\{\begin{pmatrix} w_1(1) \\ w_2(1) \end{pmatrix}, \begin{pmatrix} w_1(2) \\ w_2(2) \end{pmatrix}, \begin{pmatrix} w_1(3) \\ w_2(3) \end{pmatrix}\}$ to the circle with the specific values of $\{\begin{pmatrix} w_1(0) \\ w_2(0) \end{pmatrix}, r\}$.

$$\{\begin{pmatrix} w_1(1) \\ w_2(1) \end{pmatrix}, \begin{pmatrix} w_1(2) \\ w_2(2) \end{pmatrix}, \begin{pmatrix} w_1(3) \\ w_2(3) \end{pmatrix}\} \mapsto \{\begin{pmatrix} w_1(0) \\ w_2(0) \end{pmatrix}, r\}. \tag{3.8}$$

Each possible combination of three points votes for the specific cell of the grid. This means that each cell has a specific rating, which is set to zero at the beginning. When certain set $\{\begin{pmatrix} w_1(i) \\ w_2(i) \end{pmatrix}, \begin{pmatrix} w_1(j) \\ w_2(j) \end{pmatrix}, \begin{pmatrix} w_1(k) \\ w_2(k) \end{pmatrix}\} \mapsto \{\begin{pmatrix} w_1(m) \\ w_2(m) \end{pmatrix}, r\}$ the rating of the accumulator cell including the point $\{\begin{pmatrix} w_1(m) \\ w_2(m) \end{pmatrix}, r\}$ is incremented. After searching through all possible triplets of the points at the center of the cell with the highest rating is associated with the parameters of the circle. If there are multiple circles in the image, the cells with a rating higher than the specified threshold value are chosen.

The application of Hough transform to ball tracking, proposed in [Kim75, Sca05] and applied in [Bar09, Pon09], is a little bit different from the algorithm described above. It uses three specific features: the assumption that the ball's contour is the only circle in the image or at least the strongest one, the fact that the value of r is not required for further processing and information about edge normal that is included in the edge image. Each edge normal corresponds to one possible line on which the center of the ball is lying. The accumulator is 2D instead of 3D, and the size of the cell grid is equal to the size of the edge image. In fact, the accumulator itself is a grayscale image. The values of the accumulator pixels are set to zero in the beginning. Then all edge pixels are processed iteratively. For each edge pixel, the corresponding edge normal is mapped onto the accumulator image. The pixels of the accumulator image that are covered by the normal, increment their intensity. After processing all edge pixels, the accumulator image is smoothed by the Gaussian filter. This allows the influence of errors to be decreased. Then the point with the highest intensity on the accumulator image is assumed to be the center of the ball in the camera image. An example of the sequence of the corresponding input image, edge image accumulator image and ball center projection for 2D Hough transform is shown in figure 3.6.

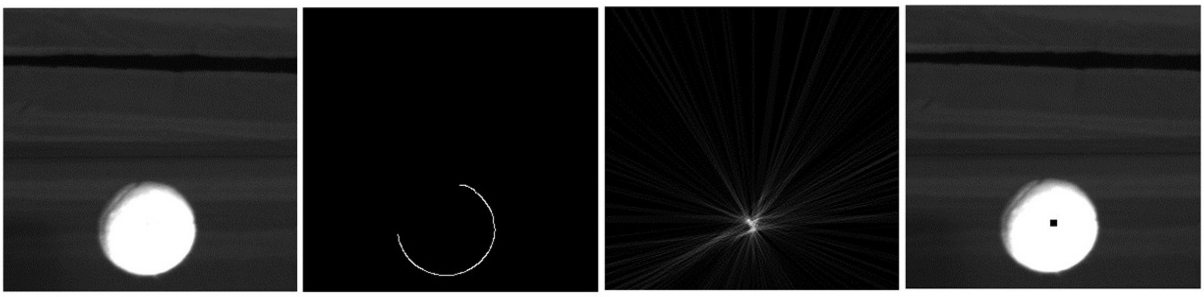


Figure 3.6: An example of 2D Hough circle recognition. From left to the right the following images are shown: a basic image, an edge image, an accumulator image and a basic image with a center of the ball projection.

The advantage of 2D Hough ball detection when compared with the classical Hough circle is the lower amount of calculations. The accumulator space is also 2D instead of 3D, and the loop includes the processing of each edge pixel instead of each possible triplet of edge pixels. However, it is unfortunately less robust and does not allow the radius to be estimated although that may be done afterwards. Because of the complicated scene, the 3D Hough was applied instead of the 2D.

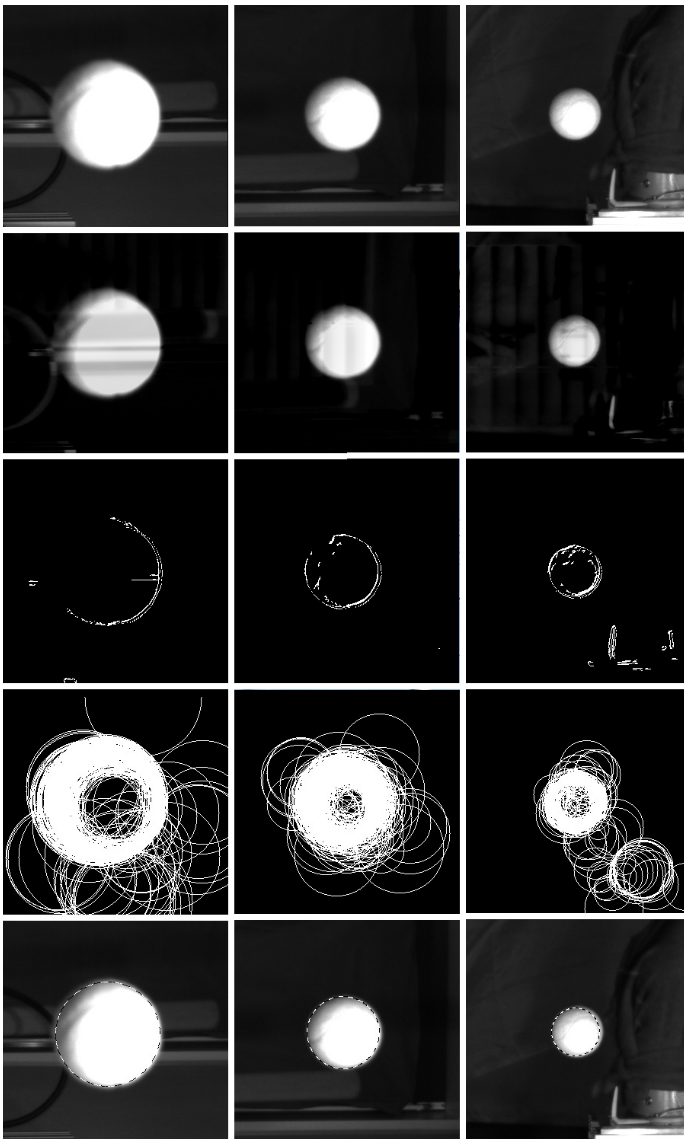


Figure 3.7: RANSAC ball recognition for three sample images shown in the first row. The second row shows the results of background subtraction. The third row consists of the edge images. The fourth row shows the projections of randomly generated circles onto the image plane. Finally, in the fifth row the chosen circles are projected onto the initial images.

RANSAC is a common technique for fast random-based fitting in the large dataset. It was proposed in [Fis81]. Its application to ball recognition is also based on mapping as in equation 3.8. Here the triplets of edge points are chosen at random, and the hypothetical circle is constructed according to this mapping. All other edge points are fit to this circle. If the fit is bad, the described operation is repeated until a sufficient fit is obtained. This method does not guarantee the best fit, but in practice it usually requires less computations than Hough. The visualization of full reference of RANSAC ball recognition is given in figure 3.7. Each of three columns represent a process of center extraction from the single image: image taken just after the throw (left column), image taken in the middle of the trajectory (central column) and image taken in the catching area (right column) In the first row images themselves are presented, then goes the results of background subtraction (second row), the results of edge detection (third row), the set of RANSAC-generated hypothetical circles (fourth row). In the last row the chosen circles are projected onto the initial images.

Both the edge detection and ball center positioning (either Hough-based or RANSAC-based) steps are computationally expensive but are easy to parallelize; therefore, they are implemented in the graphic processing unit (GPU) using C++ CUDA library by Goetzinger [Goe15]. In the experiments that test the performance, the maximum execution time for the sequence of image processing operations (including background subtraction, canny edge detection and either Hough or RANSAC circle detection) does not, in most cases, exclude 10 ms [Goe15, pp. 77-78]. An analysis of the accuracy characteristics for the image processing algorithm is done in section 3.3.

3.1.3 Triangulation

When the ball's center coordinates are extracted from both images, stereo vision is then applied to determine its 3D position. The operation of determining the object?s spatial position from its pixel positions in both images is called stereo triangulation.[1] This operation uses the camera?s intrinsic parameters (focal length, distortion coefficients, etc.) and information about the relative location of the cameras (distance and rotation angles) as necessary inputs for stereo reconstruction. The values of these parameters are obtained prior to the observation during the calibration procedure. Stereo triangulation is a reversal of the operation of point projection in the image. According to the pinhole camera model from the geometrical optic, the position of the 3D point in the image plane is expressed by the following system of equations [Sze09, p.59]:

$$\begin{aligned} u_1 &= \frac{f}{x_3} * x_1, \\ u_2 &= \frac{f}{x_3} * x_2, \end{aligned} \tag{3.9}$$

where f is camera focal length, $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ is a vector of 3D point coordinates in the camera coordinate system and $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ is a vector of plane coordinates of point projection on the image plane (note that this is not equal the pixel coordinates of the point). In homogeneous coordinate system it is expressed by the following matrix equation [Sze09, p.50]:

$$\begin{pmatrix} u_1 \\ u_2 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} * \frac{1}{x_3}. \tag{3.10}$$

This means that if f and $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ of the point is known, the coordinates of its projection in the image plane may be found. The calculation of the pixel coordinates of the points projection is made using the following equation below [6, 19].

---

[1] In common stereo vision, the challenge lies in determining the correspondence between pixels to the left and right in the image. In the current application, this step is unnecessary because the ball's center positions are already determined separately in each image.

$$\begin{pmatrix} u_1 \\ u_2 \\ 1 \end{pmatrix} = P * F * W * \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{pmatrix}. \tag{3.11}$$

Here $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ is a vector of point coordinates in a world 3D coordinate system, P is 3 by 3 matrix for transformation from pixel coordinates to geometric coordinates of the image plane. The value of P is determined by the linear size of the pixel element on the sensor and the coefficients that express the radial distortion of the camera lens system. F is a 3 by 4 projection matrix similar to the matrix from equation 3.10 and W is a 4 by 4 matrix for transforming the 3D coordinates from the system in which they are initially defined to the system connected with the camera's optical center. These matrices are determined by the following parameters [6]:

- The focal length of the camera f,
- The position of the pixel coordinates origin in the image plane $\mathbf{C} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$
- The linear size of the pixel element in the image sensor $\mathbf{D} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$,
- The coefficient vector expressing the distortion of the camera lens system $\mathbf{K} = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \end{pmatrix}$,
- The angle α expressing the skew of the pixel element if this element is a parallelogram but not a rectangle.

All of these parameters are determined via the camera calibration procedure. Their effect on the transformation matrices is not discussed here due to its complexity. It is only necessary to mention that when these parameters are known the calculation of the point projection in the image can be made [6].

$$\{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, f, \mathbf{C}, \mathbf{D}, \mathbf{K}, \alpha \} \mapsto \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}. \tag{3.12}$$

Here $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \mathbf{W}$ is a pixel coordinate vector of the point projection on the image. Reverse mapping from known $\{\mathbf{W}, f, \mathbf{C}, \mathbf{D}, \mathbf{K}, \alpha\}$ to $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ is not possible as the specific point on the projection plane corresponds to an infinite number of points in 3D space lying on a single ray that start from the projection point and include the optical center of the camera as discussed in subsection 2.3.4. If the coordinates of the point projection onto the images from two different cameras are available, then the following mapping is possible [6]:

$$\{\mathbf{W}(1), f(1), \mathbf{C}(1), \mathbf{D}(1), \mathbf{K}(1), \alpha(1), \mathbf{W}(2), f(2), \mathbf{C}(1), \mathbf{D}(1), \mathbf{K}(1), \alpha(2), \mathbf{T}, O\} \mapsto \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{3.13}$$

Here $\mathbf{T} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$ is a translation vector that expresses the location of the second camera optical center in the coordinate system connected with the first camera optical center. O is a 3 x 3 matrix that expresses the rotation of the second camera's optical axis compared to the first camera's optical axis. The other items on the left side correspond to the respective calibration parameters of the first and the second cameras. The values of $\mathbf{T}$ and O are obtained during the stereo calibration.

The range is mainly obtained based on the range-disparity equation [Liu06]:

$$x_3 = \frac{f * b}{d}. \tag{3.14}$$

Here b is a baseline of the camera system equal to the norm of $\mathbf{T}$), f is the focal length of the first camera, d is the disparity between the point positions in the two images. The disparity in this equation is expressed in image plane coordinates, i.e. it is equal to the difference between the point vectors in the planes of two images:

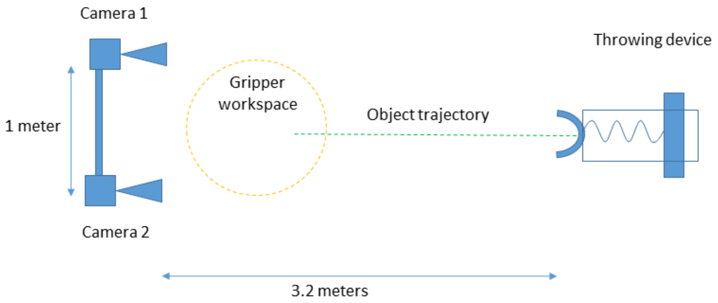
$$d = \sqrt{(u_1(1) - u_1(2))^2 + (u_2(1) - u_2(2))^2}. \tag{3.15}$$

Note that $\begin{pmatrix} u_1(1) \\ u_2(1) \end{pmatrix}$ in this equation is expressed in the coordinate system connected with the first camera and $\begin{pmatrix} u_1(2) \\ u_2(2) \end{pmatrix}$ is expressed in the coordinate system connected with the second camera.

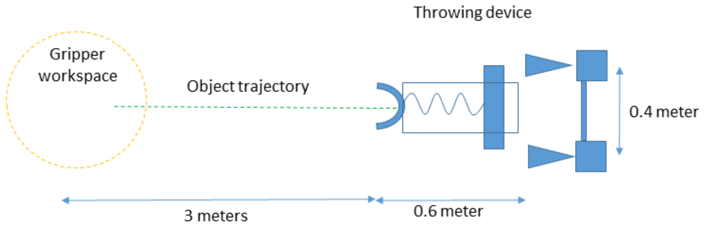
The calibration of the camera system (i.e. obtaining the values of $\mathbf{W}(1)$, $f(1)$, $\mathbf{C}(1)$, $\mathbf{D}(1)$, $\mathbf{K}(1)$, $\alpha(1)$, $\mathbf{W}(2)$, $f(2)$, $\mathbf{C}(1)$, $\mathbf{D}(1)$, $\mathbf{K}(1)$, $\alpha(2)$, $\mathbf{T}$, O) may be done in MATLAB using the Bouguet camera calibration toolbox [6] or the Mathworks camera calibrator [14]. The Mathworks application is faster and easier to use. The calibrator's disadvantage is that the code of computations is hidden. The Bouguet code is free, so it was a base for the C++ function that implements the stereo triangulation. The calibration for the experiments was done with the Mathworks calibrator, and the results were used as the setting parameters for the C++ triangulation function.

3.1.4 Acquired datasets

Several separate series of throwing experiments were done in order to collect trajectories to explore their properties and to learn about the predictor. Based on the experiments, three main datasets were collected. These three series were already mentioned in table 3.1 as they allowed the thrower's accuracy to be estimated. The first dataset with 169 trajectories was obtained in December 2014 with cameras positioned opposite of the throwing device (figure 3.8, a). The nominal throwing velocity was set to 4.5 m/s, and the measured values varied from 4.3 to 5.1 m/s. Further exploration of the accuracy showed that the positioning errors are critical if the cameras are far from the object (subsection 3.3.2). Therefore, several more throwing experiments were performed in another setup. The cameras were positioned in such a way that the throwing device was between them (figure 3.8.b). The cameras' optical axes are nearly horizontal and nearly parallel to one other.



a. First throwing setup



b. Further throwing setup

Figure 3.8: The relative locations of the cameras and the throwing device.

The second dataset based on the second setup was acquired in May, 2015, and consisted of 95 trajectories. The trajectories were recorded with several different launching velocities (4, 4.25, 4.5, 4.75 and 5 m/s). It was assumed that the velocity of 4.5 m/s was used in catching experiments, and the other values were only for learning[2]. The range of trajectories in the learning set should be wider than in real conditions in order to avoid the fringe effect. The overall range of the measured launching velocities is 3.7 to 5.3 m=s. The third dataset was acquired in August, 2015, with the same setup. A more advanced algorithm was applied for image processing. In the second setup, the images were processed based on the algorithm 2D Hough transformation, and the second setup image was processed with a fast and more accurate GPU implementation from [Goe15]. An exploration of the observer's accuracy was done using the second dataset, but the results of this exploration were then replaced by the same results from the third dataset. The third dataset is primarily investigated in further sections. The number of trajectories acquired for this set with various nominal velocities is listed in table 3.2. Another dataset was obtained in September, 2015, in order to define how the robot's motion inside the field of view influence the trajectory's capture. The results of this experiment are briefly discussed at the beginning of section 3.3.

---

[2] These value of throwing velocity was chosen as it provide the most suitable trajectory for catching in the experimental setup [Pon16]

Table 3.2: The parameters of the acquired datasets.

Nominal throwing velocity, m/s	# of samples
4.00	22
4.25	22
4.50	22
4.75	23
5.00	22
Overall	111

The trajectory representation in the database consists of several data units. The main unit is a reference of the object coordinates measured in the camera coordinate system. It is a matrix with 100 rows (each row corresponds to one acquired frame) and 3 columns (each column corresponds to one spatial dimension in a camera-related coordinate system). If there were no coordinate measurements in a frame (e.g. because the ball was out of the field of view at that moment), the values in this row were set to zero. Further tracking of the ball after the 100th frame was not kept because in those cases the ball was already out of the robot's workspace at that moment. In some trajectories with low throwing velocities, it has already rebounded from the floor. Besides the coordinate references, some additional parameters of each trajectory are stored in the database. These are mainly associated with trajectory pre-processing, e.g. the positions of the C-AOI and A-AOI in an image plane, timestamps, the pixel coordinates of the ball center, etc. These parameters also include the nominal and measured values of the launching velocity.

3.2 Accuracy of positioning static object

This section and the following one investigate the accuracy of the 3D ball?s coordinate extraction from the images. This information is crucial for the further development of the prediction algorithm. This accuracy will now be examined in the situation where a ball is static in space. Section 3.3 extends the accuracy model to the flying ball. The reason for investigating static accuracy is because the static ball may be very precisely positioned in space. The measured position of the ball may be compared with its actual position set by the mounting equipment. This comparison allows the accuracy to be explored, but it does not take specific factors into account that influence the position of the moving objects, e.g. motion blur, errors in background subtraction, etc. An analysis of these factors is given in the next section.

[Lee02] defines the following three sources of errors in stereo positioning:

- *Calibration errors* (CE): They are connected with the incorrect estimation of the intrinsic and extrinsic parameters during calibration. For example, if the linear size of the support object used in calibration (e.g. the grid size of a chessboard used in the Zhang calibration [6, 14]) is measured inaccurately, it may distort the linear distances in a reconstructed scene.
- *Quantization errors* (QE). One pixel corresponds to a certain area of an image. It is not possible to determine precisely where the point actually is in that area. When the

triangulation is done it is assumed that the point position corresponds to the pixel center. The difference between pixel centers shows the variation of point coordinates.

- *Image processing errors* (IPE). In [Lee09] the errors in stereo matching, i.e. the procedure of determining the pairs of pixels on left and right side of the images, are in this group. As previously mentioned, there is no specific step for stereo matching in the algorithm. The pixel positions of the ball?s center in both images are determined by the Hough or RANSAC circle extractor. The algorithms for edge detection and circle extraction do not provide 100% accurate pixel positions of the centers. Therefore, IPE are considered to be errors in these algorithms.

A set of experiments was performed in order to define the accuracy of the vision system [Pon15]. The aim of experimental setup was to create an environment that allowed object coordinates measured by the camera system to be compared with the information about their real values. This is provided by the mounting equipment that allowed the object to be accurately positioned in space.

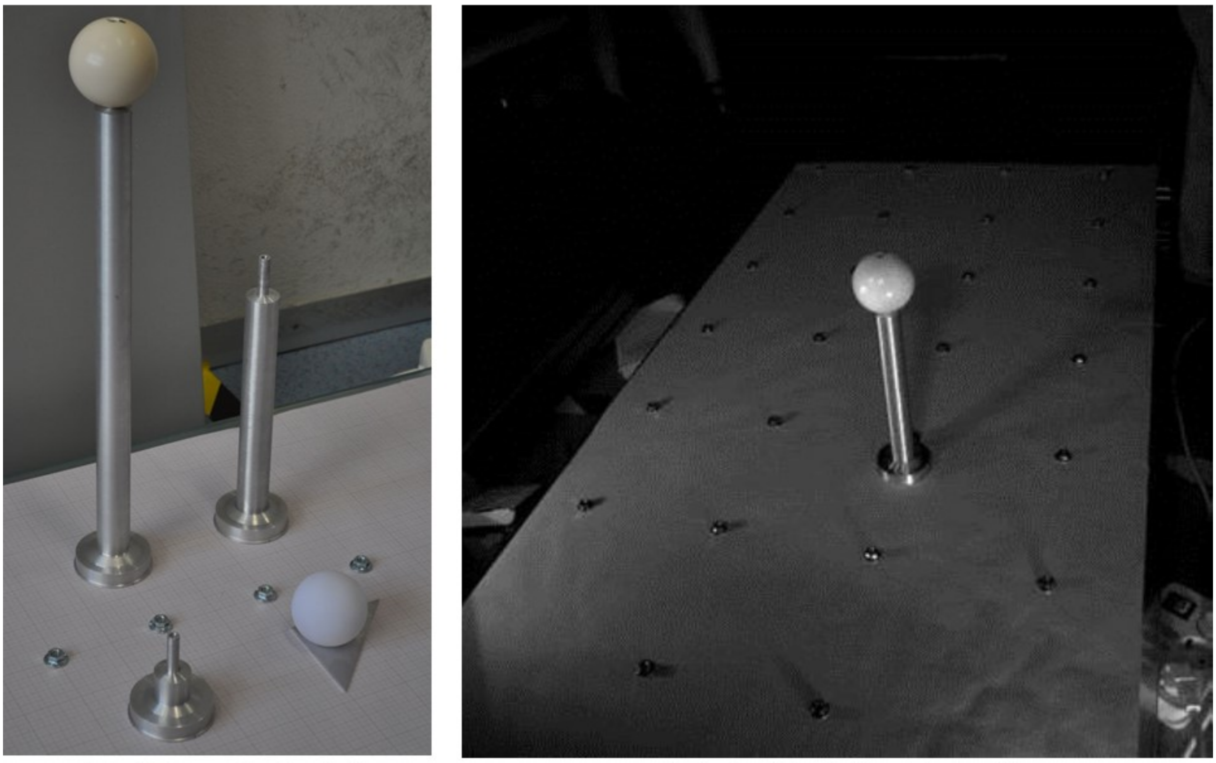
The base for the mounting is a horizontal planar plate made from glass and pasted on millimeter paper. The spherical objects are mounted on the pedestals at a specific height (Figure 3.9, a). These pedestals allow the ground-truth position of the sphere to be defined in a vertical dimension. The ground-truth accuracy in two horizontal dimensions is provided by positioning the pedestals on millimeter paper (Figure 3.9, b). Small cracks at the base of the pedestals allow it to be adjusted with the millimeter grid. The millimeter plate was positioned under the area of the object's flight. Therefore, the setup allows the accuracy of the object's positioning in the area of flight to be defined (Figure 3.10). The camera positions were similar to the positions shown in figure 3.8, b.

The origin point of the stereo camera coordinate (SCC) system was put at the center of the baseline. Prior to estimating the accuracy of the ball's positioning, the relative location of the mounting equipment in the camera coordinate system was defined. In order to get the position of the ground plane with the millimeter paper in the SCC system, a number of points on the grid are taken. Their pixel positions in the images are taken manually. Then their SCC are calculated by stereo triangulation. The plane fitted to these positions represents the position of the millimeter plate in SCC. The quality of fit primarily allows the influence of QE to be estimated. A total of 18 points were picked from the distance of 0.57 to 1.72 meters from the baseline. At the furthest distances, the nodes of the millimeter grid are not visible in the images, and at the closest distance, there are no points visible to both cameras. The distance from the points to the fitted plane does not exceed two millimeters, and the standard deviation achieves 0.7 mm.

Other errors extracted from this data are errors in the estimation of the distances between the points of interest by stereo triangulation. The points were taken at a distance of 200 mm from each other and were taken for the plane estimation. In the first computational experiment, it was revealed that the distance $\hat{d}$ estimated by stereo triangulation has a systematic error component compared to the real distance d:

$$\hat{d} = 0.997 * d. \tag{3.16}$$

In other words, when the actual distance is 200 mm, the triangulation is 199.3 mm, for 600 mm it is 597.9, etc. An analysis of the calibration data showed that the reason for such errors lies in the



a. Pedestals for mounting the object

b. Positioning of the object on millimeter plate

Figure 3.9: The setup for the accurate positioning of the spheres in space. The spherical objects are mounted on the pedestals with precise height (subfigure a). The pedestal in the right bottom corner of figure a. is intended for the self-lighting sphere. Accurate positioning in the vertical dimension is provided by the pedestals while accurate positioning in two horizontal dimensions is provided by the millimeter plate. Subfigure b show an example of accurate object positioning in space: the pedestal is put into specific position on the plate. In other cases the objects on various pedestals were put into various positions.

incorrect measurement of the grid size on the calibration checkerboard. An error of approximately 50 microns led to this distortion. After the second measurement of the checkerboard with a high-precision caliper, there was no significant systematic error component that was found in the distance estimation. The random error component achieves 0.5 mm, which is seemingly connected with QE.

After this preliminary stage, the sphere-positioning experiments were conducted. At the first stage, the spheres were illuminated by four 500 W halogen flood lights. These were the same flood lights that were used for illumination in the throwing experiments. An example image made at this stage is shown in figure 3.9. The standard deviations of the ball positioning with Hough transformation achieved 3.6 mm. Mostly, the errors were about 1.2 mm. Only in two points, which lie in front of the baseline, the error achieved values of more than 5 mm. (Clarify and split up the sentence into smaller sentences). It is possible that the reason for such a large value could the asymmetrical lighting that distorts the results of the edge detection and circle extraction.

To determine the influence of the lighting conditions, the second stage of experiments was con-

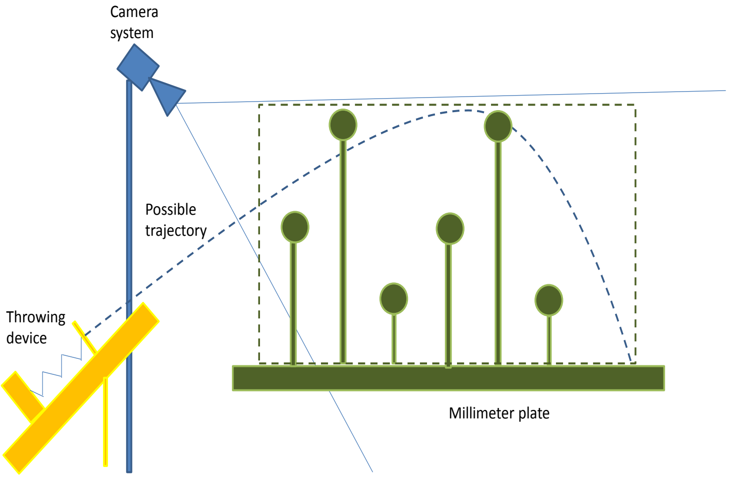


Figure 3.10: Positioning of spheres in the area of observing trajectory.

ducted. The glowing sphere without external illumination was used here. An example pair of images captured in such conditions is shown in figure 3.11. The value of standard deviation for the glowing spheres achieved 2.3 mm. Except for the external-lighting set, no drastic increase in error was detected near the baseline. As the tennis ball is not self-lit, an external lighting setup has to be used. To avoid lighting asymmetry, it is recommended that the throwing position not be put closer than 0.6 meters to the baseline, which is the distance, where the asymmetric lighting appears, and to use dispersed light sources.

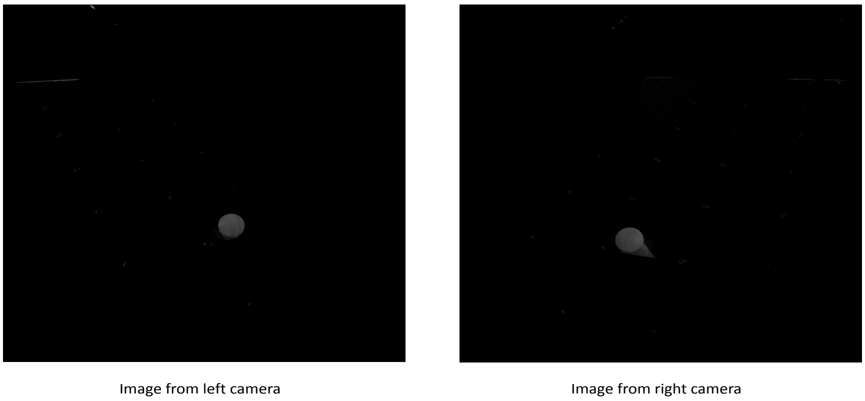


Image from left camera

Image from right camera

Figure 3.11: Image pair for acquired position of self-lightning sphere.

Therefore, the positioning of the computationally extracted ball centers is more erroneous than the positioning of the manually extracted grid points. This increase in errors is obviously connected to the image processing errors at the stage of the pixel coordinate extraction of the ball's center. These experiments ignore several factors, e.g. the motion blur when the object is flying, errors of background subtraction while tracking the moving object and the increase of quantization errors at distances of more than 1.72 m. It was not possible to define the pixel positions of the point of origin at such distances. An exploration of these factors and the overall accuracy of the positioning of the flying balls is given in section 3.3.

3.3 Accuracy of positioning the flying object

The experiments described in the previous section mainly deal with the theoretical perception of the accuracy of the vision system. Now the errors detected in a real situation will be analyzed. This analysis is challenging as there is no ground truth data about the real object's position at each moment in time. Many errors may be detected as they distort the smoothness of trajectory curve, as discussed in section 2.1. This curve cannot be accurately defined analytically, but it is smooth. Another way to estimate the errors is by fitting the datapoints to the simplified motion models. This may not be incredibly accurate as these models do not exactly represent object motion. However, if these models are generally more precise than the vision system, the quality of fit may deliver information on the observer's accuracy.

The errors of the separate processing steps may be detected by a visual analysis of the intermediate images. The quality of edge detection may be evaluated by comparing the edge image with the edges of the ball in the original images. The quality of the RANSAC or Hough circle extraction may be evaluated by projecting the circles that were found on the initial images. For example, the visual analysis of the images from figure 3.7 show that the edge detection algorithm has some noise, but the RANSAC estimation delivers reasonable results. The disadvantage of this visual analysis is that it is done by a human and cannot deliver unbiased information. However, it does allow some obvious tracking errors to be detected.

One other way of providing more accurate data for comparison and validation is specific to RANSAC. As RANSAC does not provide the same results for various runs, multiple runs give multiple hypotheses about the ball?s center position. A correct statistical estimate based on these hypotheses is more accurate than the result of a single RANSAC run. The results of multiple measurements are noisy and are not supported by the ground-truth motion model and prior statistical knowledge, e.g. probability density function. According to [Kay93, Hla12 p.11-12], the least squares (LS) estimation should be used in such conditions. The least squares estimate for the static parameter with unknown random noise is equal to the mean of measurement results [Hla12 p.7]. In the current estimator, the mean is replaced by median. The median and mean estimation give similar results, but the median is more robust to outliers. The median of 1000 RANSAC runs was used while forming the database, and any further increase of runs number does not change the results of the median estimation. The use of such an estimate at the prediction stage is not possible due to the high volume of calculations. The GPU, which is able to perform a single RANSAC circle extraction in real time (i.e. less than 9 ms for two images and less than 1 second for the whole trajectory), took about 10 minutes to run RANSAC 1000 times.

For moving objects, the full error of positioning may be divided into two components. Full error e specified in mm or other length indices is the distance between the estimated object coordinates $\hat{\mathbf{X}}(t)$ at time t, and real object location at this time $\mathbf{X}(t)$. The space error component e_s is the distance between $\hat{\mathbf{X}}(t)$, and $\mathbf{X}(t+\tau)$, which is the nearest point to $\hat{\mathbf{X}}(t)$ on the real trajectory of the object. The time error component is equal to τ. This is time shift between t and the time moment, when the real position of the object is closest to $\hat{\mathbf{X}}(t)$. τ is specified in ms or other time indices. Time error also can be defined as length of trajectory part between $\mathbf{X}(t+\tau)$ and $\mathbf{X}(t)$. The relationship between e_s, e and τ is illustrated on figure 3.12. The division of errors

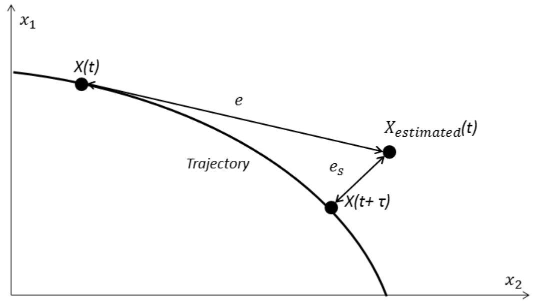


Figure 3.12: Time and space components of the positioning error.

to time and spatial components is mainly important for prediction. As shown in chapter 5, the time error component is less crucial for the gripper than for the spatial error component.

No motion blur is visible to the human eye in the image. However, this does not guarantee that blur has no influence on accuracy. This effect takes place when the object moves significantly during the time of exposure. The exposure time of the cameras in the throwing setup was set to 1 ms. The maximum velocity of the ball is 5 m/s. This means that at the time of exposure, it reaches about 5 mm. The blur is mainly due to the motion of the object in a perpendicular direction to the image plane. The ascending and descending motion of the ball after the throw is the primary component. As the ball is thrown with an angle of $\frac{\pi}{3}$ radians to the horizon, the velocity of the ascending motion is equal to $v_a = v * \sin\frac{\pi}{3} = 5 * \frac{\sqrt{3}}{2} = 4.33(m/s)$ which means that the ball is moving at $4mm$ during exposure time in a vertical direction during exposure time. This is much less than the size of the ball, which has a diameter of more than 65 mm [15]. As blur is connected with object motion along the trajectory, the influence of motion blue is only part of the time component of the error. This influence is constrained by the exposure time, which is 10 times less than the frame rate of the system. Hence, the specific influence of motion blur was ignored in when it comes to further error explorations.

One more specific factor that has an influence on the quality of tracking is robot motion within the catching area. In the fourth set of the experiments, a number of trajectories were acquired while the robot was moving along the catching area. In this situation, the robot's appearance is not eliminated by the background subtraction algorithm and the robot's contour exists on the edge image. The tracking trials with the moving robot demonstrated that the end effector is recognized as an object in most of the images. Therefore, in the current setup, tracking of the object at the final stage of the trajectory (after the 70th frame) is useless if the robot if moving. Therefore, the catching motion inspired by the predictor data may not be supported by the visual feedback once the ball reaches the catching area. The trajectory of the caught ball is not used for additional learning. The learning samplings are provided by the throwing experiments without catching.

3.3.1 Influence of background subtraction

An analysis of the tracking results showed that background subtraction is necessary for the trajectory to be reconstructed correctly. If the background subtraction is not applied, the deviations of the measured values when the distance is more than 1.5 meters increase enormously. The effect is illustrated in figure 3.13. The plots for the same trajectory extracted by the RANSAC algorithm with and without background subtraction are given. It can be seen that up to a distance of approximately 1.5 meters, the measurements nearly coincide and appear as a second-order curve. The measurements with background subtraction (blue circles) keep this appearance afterwards, but the appearance of another reference (red dots) become chaotic. This behavior is typical for most of the trajectories in the dataset. The numerical estimate for the errors are given in table 3.3.

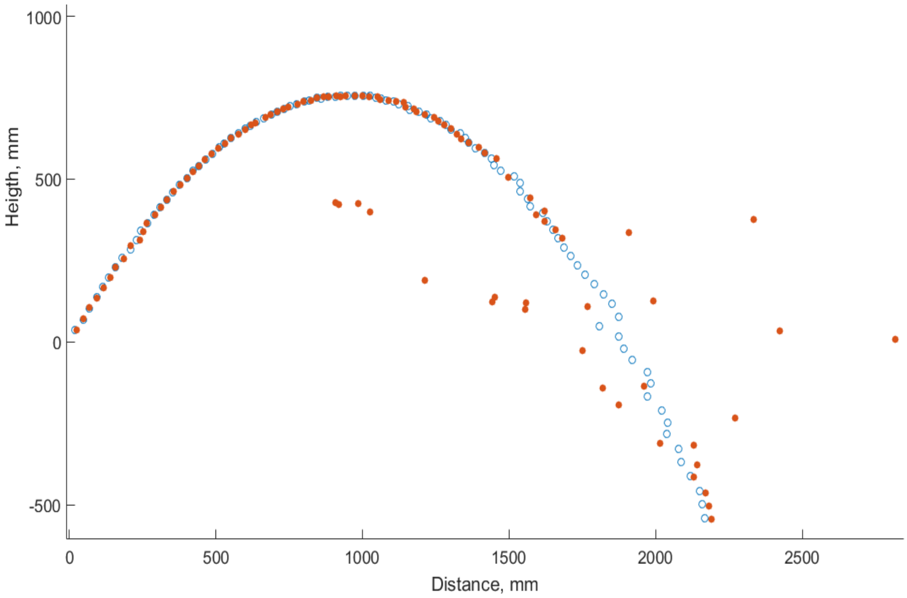


Figure 3.13: The plot of the same trajectory measured with (blue circles) and without (red dots) background subtraction.

Here the coordinates extracted by the single RANSAC run are compared to the results of median estimation for 1000 runs. The differences are considered to be ”errors?. In fact, these numbers are not equal to errors of positioning, but they can be used to perceive the dispersion of measurements. Based on these differences, an estimate of the standard deviation is calculated for each frame since the ball was thrown. In the table the frames are united in blocks of 5 frames in order to save space. The standard deviations are summarized based on all 111 trajectories acquired in the third series of experiments. It can be seen that the parameter starts to grow enormously after the 65th frame, and this growth is much more dramatic for the version of the algorithm without background subtraction. The reason for this increased stability at the beginning is that for initial frames, the size of the ball is bigger and almost covers the image entirely (compare the first and third column in figure 3.7). Therefore the background edges make smaller distortion on the results of edge detection.

t can be seen that even for the version with background subtraction, the standard deviation after the 70th frame achieves very high values. The standard deviation may be not the best parameter as it has low robustness to outliers. It is for this reason that statisticians prefer to reject outliers from the set before calculating the standard deviation. Thus, in the columns on the right side of the table, the median differences for the same blocks are given instead. The results of the median look similar to the ones for standard deviations, but they are more detailed. For the algorithm without background subtraction, the median error lies within the 3σ interval for static spheres, estimated to be 6.75 mm (see subsection 3.2), until the 60th frame. For the algorithm with background subtraction, this property is kept till the 80th frame. In the version without background subtraction, a median difference of more than 20 cm is achieved after the 75th frame.

This means that most of the frames are outliers in this area. Thus, the measurements made without background subtraction are practically useless.

Table 3.3: A comparison of the differences between the measured 3D positions based on a single RANSAC run and the median of 1000 RANSAC runs for the versions of the algorithm taken with and without background subtraction.

Frame number	Standard deviation without background subtraction, mm	Standard deviation with background subtraction, mm	Median error without background subtraction, mm	Median error with background subtraction, mm
1...5	7.9	6.4	1.6	0.8
6...10	4.0	1.9	1.8	0.9
11...15	3.7	2.1	2.0	1.1
16...20	2.8	1.9	1.6	0.5
21...25	2.1	1.4	1.8	0.3
26...30	4.0	2.2	2.2	0.5
31...35	22.1	17.2	2.9	2.2
36...40	10.9	4.0	3.4	0.4
41...45	24.8	3.2	3.7	0.4
46...50	29.8	3.9	4.5	0.7
61...65	63.9	14.8	5.5	1.0
66...70	187.4	41.3	10.5	4.2
71...75	305.6	138.5	20.4	5.4
76...80	520.4	242.2	208.2	7.3
81...85	897.0	229.3	171.9	8.0
86...90	1361.6	197.5	163.9	8.4
91...95	1450.0	212.1	176.1	9.3
96...100	1272.6	106.63	146.8	10.1

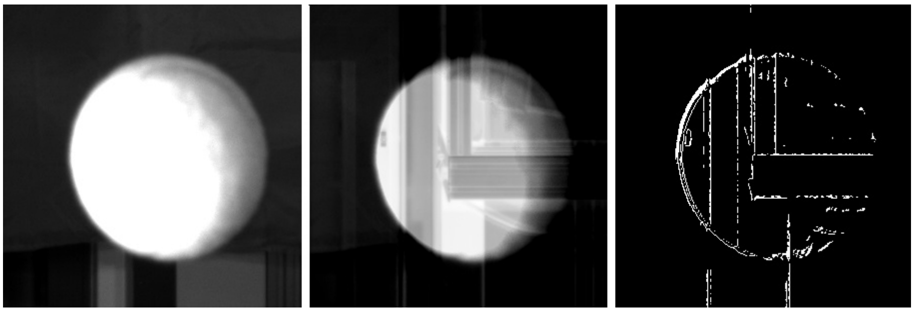


Figure 3.14: The bad influence of the background subtraction at the initial stage of flight. The ball practically covers the entire input image, so background objects are printed on the subtraction image, and their edges may be seen on the edge image.

In addition, the bad influence of the background subtraction procedure takes place in the initial part of the trajectory. Right after the throw, the image of the ball practically covers the entire A-AOI. Because of this, nearly all background objects are printed on the subtraction image (figure

3.14). Their edges may be stronger than the edges of the ball itself, and in this case, they distort the coordinate extraction. However, this distortion is much weaker than in the previous case. This effect stops after the 4th or 5th frame for measured trajectories. This issue can be overcome by only applying background subtraction after the 5th frame.

3.3.2 Errors in range measurement over long distances

The values of the standard deviations and median difference in the final part of the trajectory after approximately the 65th frame are higher than at the previous stage. An analysis of these errors' sources and how to reduce their influence is made in this subsection. The position measurements may be divided into inliers and outliers. The outliers are defined as measurements that are completely useless, even harmful, to trajectory reconstruction. Inliers could be erroneous, but they help an estimate to be improved. Obviously, it is not possible to decide with 100% confidence whether a measurement is an inlier or outlier. The huge difference between the standard deviation and median error at the end of the trajectory shows that the outliers have a significant influence. The task of outlier detection is mainly discussed in chapter 4 as this task is solved within a reference of coordinate transformations. In this subsection, the inliers measured with errors are investigated.

Plotting the trajectory shows the specific property of these errors. In figure 3.15 three plots are shown for an example trajectory, e.g. the relationship between the height of the object and the distance from the camera (top plot), the relationship between the frame number and the height (bottom-left), and the relationship between the frame number and the distance (bottom-right). It is easily seen that the first and the third plot seem distorted to the right, and the height reference kept the appearance of a smooth second-order curve. In other words, the errors are mainly localized in the range dimension.

The reason for error localization in one dimension is that when the distance to the object is greater than the baseline, the difference in one pixel is much more significant in the range dimension. An illustration of this property is given in figure 3.16.If the value of the full error is equal to the length of P_1P_2, the values of range and side components of the error ((δd) and (δh) respectively) will be defined by the angle β between the baseline and the ray that is defined by the pixel position of the point in the image from the second camera.

$$\begin{aligned}\Delta h =& P_1P_2 * \cos\beta,\\ \Delta d =& P_1P_2 * \sin\beta.\end{aligned} \tag{3.17}$$

As the distance from the baseline increases, the value of β grows, meaning the sine increases and the cosine of this angle decreases. Therefore, over long distances, errors in the range dimension achieve the most significant values.

Table 3.4: A comparison of the difference between measured 3D positions based on single RANSAC run and the median of 1000 RANSAC for three spatial dimensions in camera coordinate system.

	Standard deviation, mm			Median error, mm		
Frame number	height	side	range	height	side	range
1...5	1.9	6.0	1.6	0.3	0.1	0.7
6...10	0.4	0.3	1.9	0.2	0.1	0.9
11...15	0.3	0.3	2.1	0.21	0.1	1.0
16...20	0.5	0.4	1.8	0.1	0.1	0.1
21...25	0.3	0.3	2.4	0.2	0.0	0.0
26...30	0.7	0.3	2.1	0.3	0.0	0.1
31...35	7.3	2.9	15.3	0.3	0.1	2.0
36...40	1.0	0.4	3.9	0.3	0.0	0.1
41...45	1.1	0.5	3.1	0.3	0.0	0.1
46...50	1.6	0.5	3.6	0.4	0.0	0.1
61...65	1.2	0.5	14.8	0.4	0.0	0.2
66...70	6.0	2.5	40.8	0.5	0.0	4.1
71...75	29.7	22.8	133.5	0.6	0.2	5.3
76...80	49.5	40.0	234.1	0.9	1.0	7.0
81...85	42.3	38.1	222.3	1.0	1.0	7.7
86...90	33.0	30.0	192.6	1.0	0.9	8.2
96...100	13.7	11.7	207.5	1.2	1.0	9.1

Due to this, experiments show that the value of error over longer distances is significant for tracking in terms of range dimension. The only way this issue can be overcome is by increasing the number of cameras used for tracking, but this could be expensive. The experimental setup at the Institute of Computer Technology includes a single pair of cameras. In this work, the tracking of the object by the single stereo pair is assumed, and the challenge of erroneous measurement is overcome by the algorithmic means and specific camera alignment.

The cameras should be aligned in such a way that the influence of long-distance errors on the quality of system function be minimized. The following question should be answered. In which part of the trajectory is accurate positioning most important? In the first setup, the cameras were positioned opposite the throwing device. In this situation the first frames are erroneous. It was only possible to position the ball accurately at the 10th or 12th frames. In further experiments, the cameras were positioned to the side of throwing device. With this alignment, the positioning at the beginning part of the trajectory is rather good, but the final part of the trajectory is measured with higher errors. The high accuracy at the initial part of trajectory at the initial stage and the lower accuracy at the final stage seem preferable than vice versa according to the following factor. The initial part of the trajectory is used for a prediction made in real time. The final part of the trajectory is not processed in real time. In real transportation conditions, the ball would already be in the gripper workspace. This is used only for learning; therefore, its accuracy may be improved using an offline statistical technique, e.g. the voting of 1000 RANSACs or model fitting to the data. Accurate initial positioning is required to measure the launching parameters: velocity, angle of throw, position at the first frame, etc. One more factor is in the catching of the ball, the measurement of the ball position in the final area will not be accurate anyway. The robot

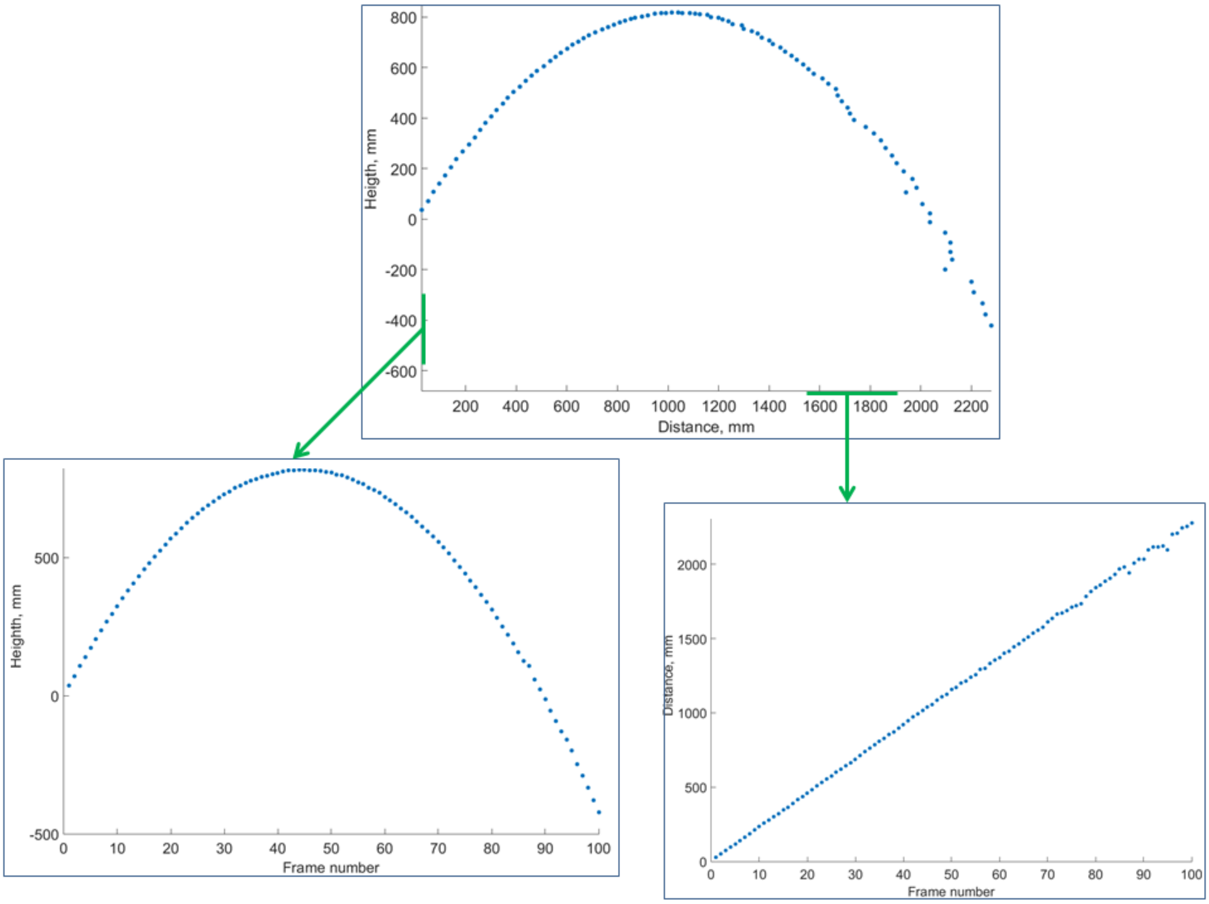


Figure 3.15: The smoothness of the measured trajectory could make an impression on the accuracy of positioning. For the most part, the trajectory is smooth, but in the final part, the positions are oscillating. The correspondence between the position of the ball in a specific dimension and the frame number shows that the oscillations are mainly localized in the range dimension.

moving in the FOV makes an extreme distortion from the algorithm functioning (see subsection 3.3.2). Due to these factors, the camera location on the side of the thrower is more likely than the opposite.

It is also possible to position the cameras in another way, i.e. with a bigger baseline or not parallel to trajectory direction. As a consequence of these changes, the high error is not localized in the dimension that coincides with the direction of object motion. In fact, this localization is very useful for error correction. The object moves at a nearly constant velocity, and the motion could be approximated by the second order polynom (see the final paragraph of this section). Motion in this direction provides a longer distance than in other dimensions, so the distance measurements have high values in comparison with the size of the errors. Because of this, the camera alignment that is nearly parallel to the motion direction optical axis and short baseline is kept.

The increase in errors further from the cameras is associated with the quantization aspects, but the outliers in this area are mostly associated with the errors in image processing. It is mainly these errors that are connected with the influence of background objects on the edge detection algorithm. Some examples are shown in figure 3.17.

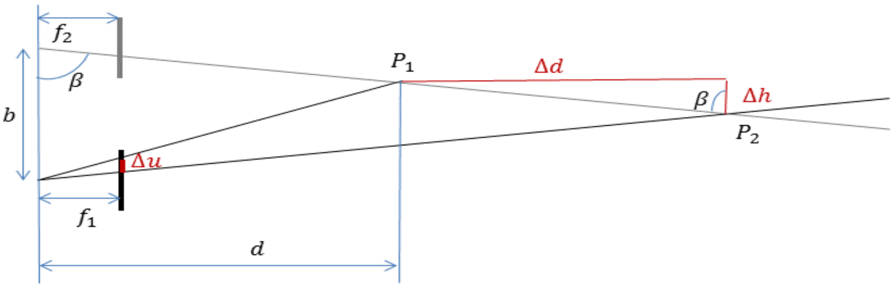


Figure 3.16: Influence of pixel error δu on the errors of 3D positioning in range dimension (δd) and perpendicular dimension (δh).

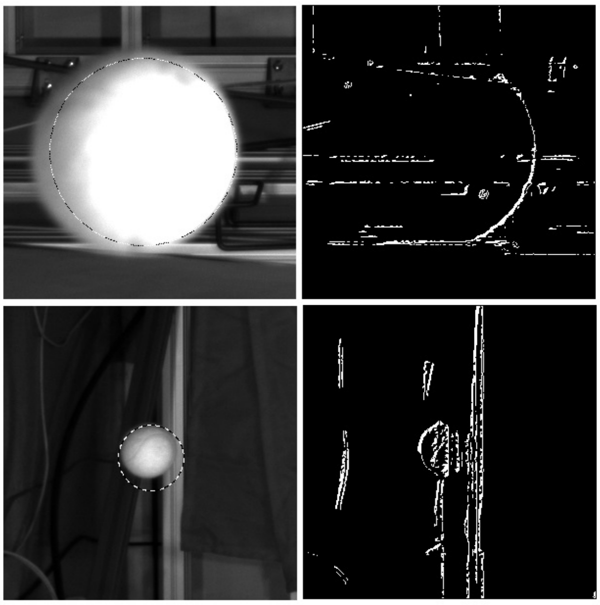


Figure 3.17: Incorrect edge detection may influence the circle extraction. In the left column, the extracted circles are mapped onto the input images. In the right column, the corresponding edge images are shown.

This aspect motivates the use of a specific covering for background objects to decrease their brightness in the images. The robot details in particular were covered by the black material. However, in a real factory environment, the background could have more contrast and that should be taken into account. Also, the robot cannot be covered to the point that it cannot move. These actions decrease the influence of the errors, but bad positioning and outlier detection still remain. An example of the erroneous trajectories extracted by 1000 RANSAC runs are illustrated in figure 3.18. These erroneous cases are atypical. Most of the trajectories have a smoother, basic look, but erroneous cases still exist. Multiple runs of RANSAC cannot eliminate such errors because the RANSAC is based on the Canny edge detector, which is a deterministic algorithm. If the edge detector works erroneously, these errors cannot be eliminated by RANSAC in many cases.

This influence may be decreased by fitting a curve to the trajectory data. The fitting requires an accurate model of the object's flight. The use of such models is not likely due to the mission of this dissertation, which is to provide a prediction method that does not require accurate modeling. Therefore, the careful use of curve fitting is proposed. The only aim of fitting is to improve the

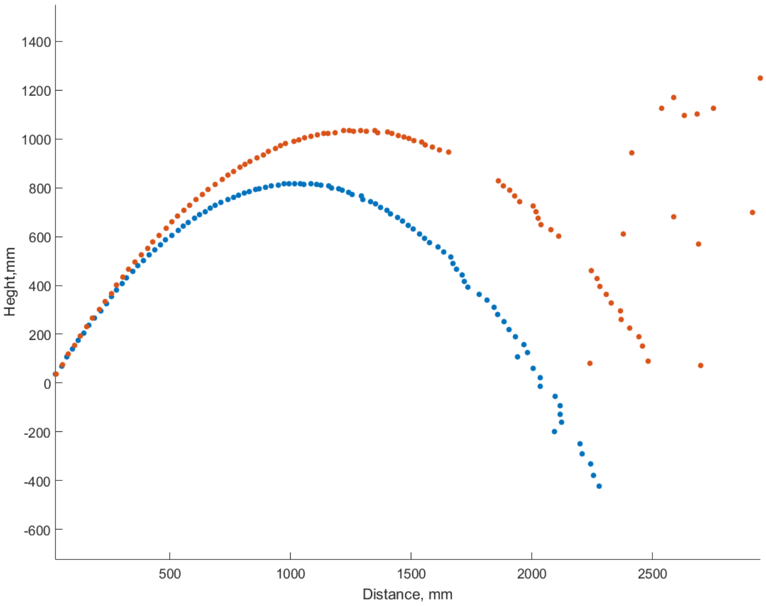


Figure 3.18: Example plots of trajectories measured by 1000 RANSAC run erroneously. The red dots represent the trajectory with outliers, and blue dots represent the trajectory with poor smoothness.

accuracy of the data in the final part of the trajectory, which is used for forming the learning dataset. Therefore, the fitting result only replaces the measured one in this final part beginning from the 60th frame. The fitting is made only for the range dimension; the separate processing of each dimension was already considered in [Pon09]). The curve of the second order is fitted to the data in the range dimension:

$$x_3 = p_1 * n^2 + p_2 * n + p_3, \tag{3.18}$$

where n is frame number p_1, p_2 and p_3 are the coefficients obtained by the fitting operation. The plot of the measured and fitted references of the object range is shown in figure 3.19. From the visual point of view this correspondence seems likely. An evaluation of its suitability for trajectory prediction is given in chapter 4.

3.3.3 Measuring object's velocity

Previous subsections were considering the accuracy of measuring object position in space. Measured positions may be used to estimate the velocity and the acceleration of the flying body. The velocity vector $\mathbf{V}$ of the object at frame number n is estimated using the following formula:

$$\mathbf{V}(n) = \frac{\mathbf{X}(n) - \mathbf{X}(n-1)}{\tau} = (\mathbf{X}(n) - \mathbf{X}(n-1)) * f, \tag{3.19}$$

where τ is interframe period and f is framerate. Both interframe period and framerate are constant parameters used to express the velocity in the international system of units, so it may be said that the velocity is equal to the difference of object coordinates on the neighboring frames.

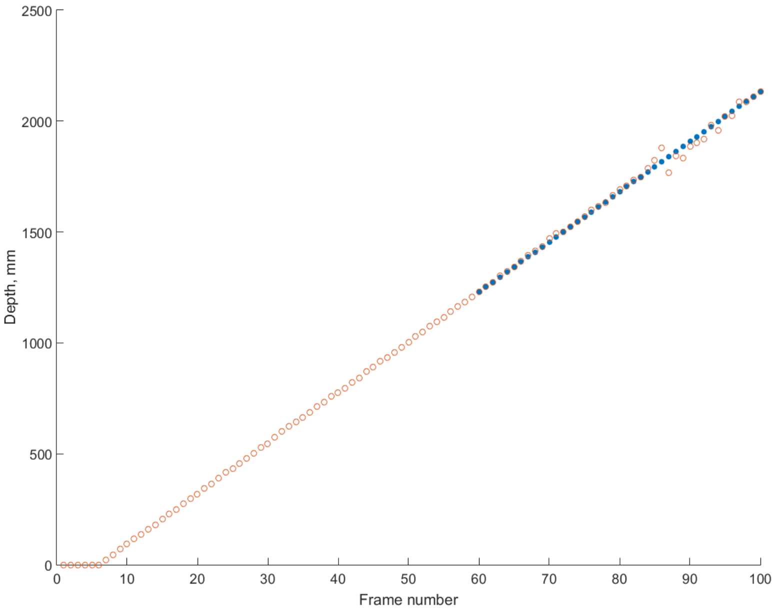


Figure 3.19: Reference of the measured (red circles) and fitted (blue dots) values of object range.

The absolute error of measuring these errors have the same scale, but the relative errors are much bigger. Let estimate the errors of velocity estimation based on errors of position estimation defined in previous subsections. The standard deviations object's positioning may be seen in table 3.3. For the most frames from 1 to 60 the standard deviation varies from 1.4 to 6.4 mm. Let us take 5 mm error for better interpretability (the aim of this section is to prove that velocity measurements are not accurate enough, so it is correct to take accuracy values a little bit better than in reality). If velocity of the object is 5 m/s and the framerate is 100 fps, the object move on 5 cm between two frames. The absolute error of 5 mm on the distance of 5 cm correspond to the relative error of 10%, or to the absolute error of 0.5 m/s in estimating object's velocity. According to well-known 3σ-rule [Hla12] the deviation does not exceed three times the standard deviation with 99% confidence. In the considered case it means that the ground-truth value of the velocity lies somewhere between 3.5 and 6.5 m/s.

This interval cover completely the interval of throwing velocities from 4.2 to 5.1 m/s measured in section 3.1. So the accuracy of measuring speed is bad due to higher sensitive to measurement errors. The same absolute errors on low scale have stronger influence than on high scale. This correlate with well-known rule: the derivatives are more sensitive to measurement errors, than their parent functions. Acceleration, which is the derivative of velocity, is even more sensitive. Therefore in the algorithm development made in next chapters, short-term velocities and accelerations are out of consideration.

3.3.4 Summary

The previous subsections have the following output:

1. The camera position to the the side of the throwing device with optical axis co-directional with nominal trajectory is preferable.

2. The background subtraction is necessary for accurate tracking.
3. The final part of the trajectory in this setup is measured with errors that are higher than the errors of static positioning. These errors are localized in the range of spatial dimension.
4. In forming the database, the influence of the errors is decreased by calculating the median of 1000 RANSAC runs and fitting the polynomial model to the positions in the range dimension.

The RANSAC circle extractor is chosen instead of Hough. The comparison in [Goe15, pp. 70 to 82] showed that RANSAC is faster in most cases, and its accuracy is no worse than for Hough. An analysis of the trajectory data showed that Hough transform has the same trouble with noisy edge images as RANSAC. Here RANSAC is chosen because it may provide multiple outputs for the same input that can be compared in order to get a more accurate value.

4 Algorithm for trajectory prediction

In the current chapter, the task of constructing a useful model for predicting the trajectory of the thrown object is discussed. The Two Nearest Neighbors (2NN) method is applied and developed for the prediction. The kNN trajectory forecasting operation is investigated in section 4.3. Two additions are proposed in order to improve the efficiency and speed of the algorithm: the reference of coordinate transformations (section 4.2) and a comparison of the current trajectory just with a small subset of the entire dataset (section 4.4). The development and initial validation of the proposed ideas are made with the use of a simplified motion simulation and the results of the throwing experiments. Therefore, section 4.1 gives a short introduction to the simulation environment. The results of the experiments based on this simulation are also presented in [Mir14, Mir15].

According to the task's definition given in section 1.4 the predictor has the following input: a reference of estimated measurements of the current trajectory of the thrown object $\mathbf{X}_C = \begin{pmatrix} x_{c1}(0) \\ x_{c2}(0) \\ x_{c3}(0) \end{pmatrix}, \begin{pmatrix} x_{c1}(1) \\ x_{c2}(1) \\ x_{c3}(1) \end{pmatrix}, \begin{pmatrix} x_{c1}(2) \\ x_{c2}(2) \\ x_{c3}(2) \end{pmatrix}, ..., \begin{pmatrix} x_{c1}(t) \\ x_{c2}(t) \\ x_{c3}(t) \end{pmatrix}$, and a large but finite set of previous trajectories both measured in the measurement area and the gripping area $L = \{(\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Y}_2), \ldots, (\mathbf{X}_m, \mathbf{Y}_m)\}$. Each position measurement $X(t)$ consist of a timing index $t = 1, 2, \ldots, n$and the measured values of the coordinates $x_{c1}(t), x_{c2}(t), x_{c3}(t)$ in the Cartesian coordinate system connected with the optical center of the left camera. This position of the coordinate system is set with the stereo triangulation operation [6,14]. For each trajectory k from the dataset corresponding to each moment in time t after release when the measurement was made, we had the corresponding point $\mathbf{X}_k(t)$. It is assumed that the measurements with the same time index t were made at the same period of time after release for all the trajectories. The task of the predictor is to estimate the value of $\mathbf{Y}_C$ using this input. As pointed out in subsection 2.5.4 the basic method for solving this task is the k Nearest Neighbors (kNN) algorithm.

4.1 Means of initial validation

This small section aims to describe the means used in the following sections for initial validation of the prediction algorithm. This validation is made in order to completely eliminate useless methods and theoretically compare the basic ideas. The ideas not rejected by this validation are then implemented and evaluated according to chapter 5. The initial validation is based on two options: the simulation of the object flight based on simplified motion models and the application of the algorithms to the acquired datasets from subsection 3.1.4. In general, a simplified flight simulation is used to check the algorithm's usability in principle, i.e. in ideal conditions, and the application to real flight data is to show the practical applicability of the concepts.

The physical models of projectile flight are either complicated or inaccurate (see section 2.1). Simulation based on the simplified models helps determine whether or not the concepts are completely useless, but to be sure of their use in experiments, there is a need for real data. The simulation of the flight is based on the gravity-drag model expressed by the differential equation 2.10. As this equation cannot be solved analytically, the object's motion is calculated iteratively, applying the following procedure.

The first iteration is set by the release point. The simulated throwing device deviates in its release velocity v in various dimensions. It was assumed that these deviations have random errors with a normal distribution around the nominal values with standard deviations of 0.1 m/s for the velocity values in each direction. This is slightly more optimistic than the 0.16 and 0.33 m/s measured for the throwing device in table 3.1. Such an assumption is allowed as the simulation is ideally intended for algorithm validation.

Once the object is thrown, the new values of the object?s position $\mathbf{X} = \begin{pmatrix} x_1(i) \\ x_2(i) \\ x_3(i) \end{pmatrix}$ and velocity $\mathbf{v} = \begin{pmatrix} v_1(i) \\ v_2(i) \\ v_3(i) \end{pmatrix}$ are calculated using the following operation:

$$\begin{pmatrix} a_1(i) \\ a_2(i) \\ a_3(i) \end{pmatrix} = \begin{pmatrix} g \\ 0 \\ 0 \end{pmatrix} - k * \begin{pmatrix} v_1(i-1) \\ v_2(i-1) \\ v_3(i-1) \end{pmatrix} * \sqrt{v_1^2(i-1) + v_2^2(i-1) + v_3^2(i-1)}, \tag{4.1}$$

$$\begin{pmatrix} v_1(i) \\ v_2(i) \\ v_3(i) \end{pmatrix} = \begin{pmatrix} v_1(i-1) \\ v_2(i-1) \\ v_3(i-1) \end{pmatrix} + \begin{pmatrix} a_1(i) \\ a_2(i) \\ a_3(i) \end{pmatrix} * \Delta t, \tag{4.2}$$

$$\begin{pmatrix} x_1(i) \\ x_2(i) \\ x_3(i) \end{pmatrix} = \begin{pmatrix} x_1(i-1) \\ x_2(i-1) \\ x_3(i-1) \end{pmatrix} + \frac{1}{2} * (\begin{pmatrix} v_1(i-1) \\ v_2(i-1) \\ v_3(i-1) \end{pmatrix} + \begin{pmatrix} v_1(i) \\ v_2(i) \\ v_3(i) \end{pmatrix}) * \Delta t. \tag{4.3}$$

This operation is an iterative solution for the differential equation 2.10. Here $\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ is the vector of object acceleration, i is the number of iterations, k is drag coefficient from equation 2.5, Δt is the difference in time between two iterations. $\mathbf{X}$ is expressed in the coordinate system where the first dimension is collinear with gravity direction and the third dimension is collinear with a nominal direction of throw. The value of k was calculated according to equation 2.6 using the values of drag coefficients for the tennis ball in the standard conditions [Ala10]. The value of Δt is 1 ms (chosen for better interpretability). The application of the model with these values of Δt and k is not significantly different (more than one millimeter) than the results from [2]. The simulated frame rate of the observer was set to 100 fps. This value is close to 110 fps of the real observation system. Thus, for simplicity and interpretability, it is best to use the integer number of milliseconds as an inter frame period. As the step of position recalculation is 1 ms, each 10th calculated position is measured.

This model allows the algorithm to be evaluated by an absolutely accurate observer. If there is a need to simulate the erroneous observer, the errors might be set to be randomly distributed around the simulator-generated values with a set value of the standard deviation. The process simulation was mainly applied for the validation of the kNN forecasting operation (section 4.3) and the subset allocation (section 4.4). The sequence of coordinate transformations (section 4.2) was validated based on datasets acquired during the throwing experiments (subsection 3.1.4).

4.2 Coordinate transformations

Predictions are based on comparing the current trajectory with trajectories from the database. The aim of such a comparison is to find the trajectory that is as similarly shaped as the current one as possible. However, the spatial coordinates $\mathbf{X}_c$ of the object in the camera coordinate system (these positions are returned by the stereo triangulation operation) are dependent not only on the shape of trajectory but also on relative position of the camera, the object and the horizontal direction of throw. As shown in section 4.3 the difference metric for the trajectories is a mean Euclidean distance between the corresponding points. If the launching points for two different throws have a certain Euclidean distance d between each other, the difference between them will be nearly equal to $d_0 + d$ where d_0 is the difference in a hypothetical case when these trajectories might have the same launching points. If two trajectories of a similar shape have the same launching point and various horizontal directions of throw, the distance between the coordinates of the corresponding points will increase in time. Even if the similarity of the shape of these trajectories is detected, the kNN forecast based on camera coordinates will be erroneous. An effects are illustrated in figure 4.1.

It is possible to adjust the throwing conditions in such a way that the relationship between the thrower and the cameras will be constant. It is also possible to minimize the variance of horizontal direction of throw, which cannot be fully eliminated due to the deviation of the throwing device (see subsection 4.2.3). This approach requires the collection of large databases in order to take various possible horizontal directions of throw into account and for the same setup to be kept up in a learning stage and functioning stage. In fact, for each new system configuration, the it is required that the trajectories be learned anew. This decreases an ease of system reconfiguration, which is one of the main advantages of TbT (section 1.2).

The aim of the coordinate transformation reference is to provide a coordinate system for storing and forecasting the trajectory that allows them to be compared based only on the properties of shape (independent of the azimuth of throw and the spatial position of the launching point) and the correct forecast of the trajectory based on kNN method.

4.2.1 Overview

The reference consists of three transformations. First of all, the gravity-related coordinates $\mathbf{X}_g = \begin{pmatrix} x_{g1} \\ x_{g2} \\ x_{g3} \end{pmatrix}$ are defined. The coordinate axis x_{g1} is collinear with gravity direction. The aim of this transformation is to localize gravity in one spatial dimension and simplify further

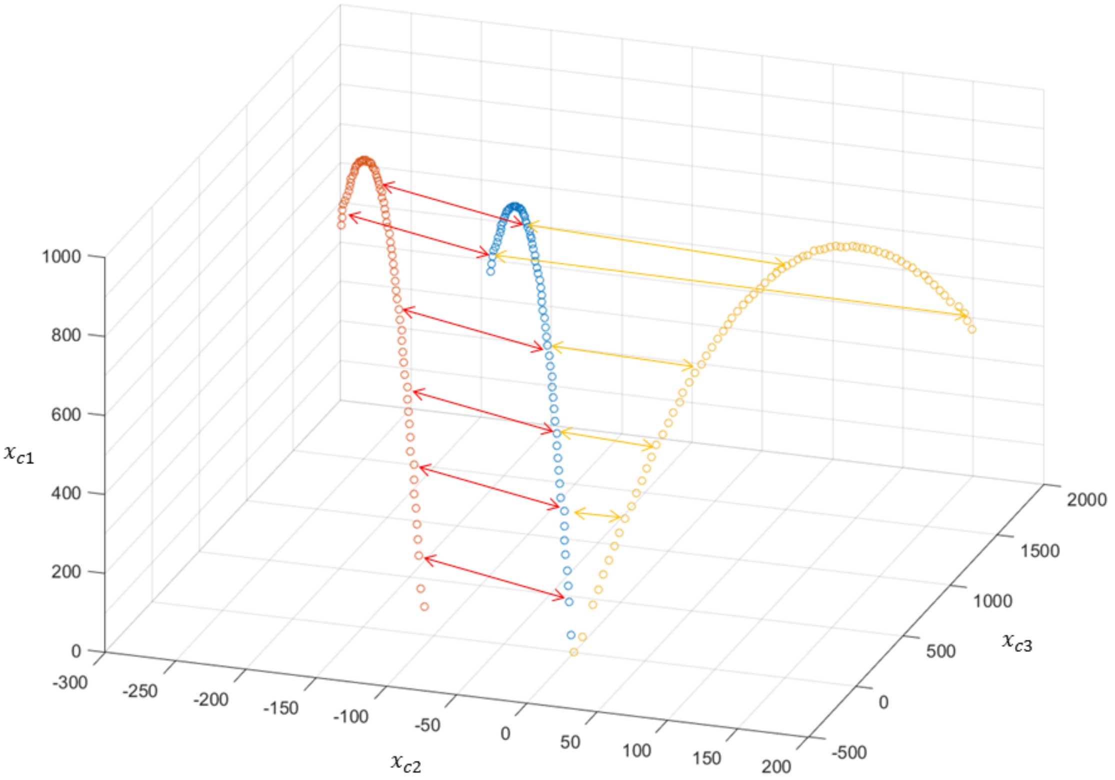


Figure 4.1: The Euclidean distance between points from various trajectories with the same timestamp in camera coordinates depends on the position of the launching point. All three plotted trajectories have exactly the same shape, but the red one has a different launching point and the yellow one has different azimuth of throw compared to the two others. Therefore, the Euclidean distance is high.

calculations (subsection 4.2.2). Secondly, the gravity-related coordinates are projected onto 2D Plane-of-Flight (PoF) $\mathbf{X}_p = \begin{pmatrix} x_{p1} \\ x_{p2} \end{pmatrix}$ (subsection 4.2.3).This transformation provides invariance of coordinates to the horizontal direction of the throw. Finally, one of the measured points of the trajectory is picked as a zero point for the coordinates $\mathbf{X}_z = \begin{pmatrix} x_{z1} \\ x_{z2} \end{pmatrix}$ (subsection 4.2.4). When the coordinates of the zero-point are subtracted, the trajectory invariance to the relative positions of the throwing device and the camera system is provided. The relationship between these coordinate systems is shown in figure 4.5. The object coordinates prediction is made within the zero-point coordinate system. The results of the prediction may then be transformed back to a camera-related coordinate system or to the robot-related coordinate system, which is useful for defining the catching movement. These back-transformations are discussed in subsection 4.2.5. All coordinate transformations are based on projective geometry. The main principles of projective geometry are briefly described below.

The coordinate transformation from a 3D coordinate system a to 3D coordinate system b is performed by multiplying the transformation matrix on the point coordinates in a homogeneous coordinate system:

$$\begin{pmatrix} x_{b1} \\ x_{b2} \\ x_{b3} \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & x_{01} \\ p_{21} & p_{22} & p_{23} & x_{02} \\ p_{31} & p_{32} & p_{33} & x_{03} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x_{a1} \\ x_{a2} \\ x_{a3} \\ 1 \end{pmatrix}, \tag{4.4}$$

or in a more compact version:

$$\mathbf{X}_b = \mathbf{P}_{AB} * \mathbf{X}_a. \tag{4.5}$$

Here $\mathbf{X}_0 = \begin{pmatrix} x_{01} \\ x_{02} \\ x_{03} \end{pmatrix}$ is the position of the origin for coordinate system a in the coordinate system b; $P_{rot} = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}$ is a rotation matrix. The vectors $\mathbf{P}_1 = \begin{pmatrix} p_{11} \\ p_{21} \\ p_{31} \end{pmatrix}$, $\mathbf{P}_2 = \begin{pmatrix} p_{12} \\ p_{22} \\ p_{32} \end{pmatrix}$ and $\mathbf{P}_3 = \begin{pmatrix} p_{13} \\ p_{23} \\ p_{33} \end{pmatrix}$ are equal to normal vectors collinear with the direction of coordinate axes from the a coordinate system in the b coordinate system. If the values of two vectors from $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$ are known, then this is enough to define the third one because it is equal to their cross product:

$$\mathbf{P}_3 = \mathbf{P}_1 \times \mathbf{P}_2. \tag{4.6}$$

This statement is proved by the following factors. If the vector triplet $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$ is right, then the direction of $\mathbf{P}_3$ is collinear with the product of $\mathbf{P}_1$ and $\mathbf{P}_2$. The length of the product vector is equal to:

$$|\mathbf{P}_1 \times \mathbf{P}_2| = |\mathbf{P}_1| * |\mathbf{P}_2| * \sin\frac{\pi}{2} = 1 * 1 * 1 = 1 = |\mathbf{P}_3|. \tag{4.7}$$

A reverse transformation of coordinates from A to B is achieved by solving 4.4 as a system of linear equations with $\{x_{a1}, x_{a2}, x_{a3}\}$being unknown. Another way of calculating $\mathbf{X}_a$ from $\mathbf{X}_b$ and P_{AB} is multiplying the inverse of P_{AB} on $\mathbf{X}_b$:

$$\mathbf{X}_a = P_{AB}^{-1} * \mathbf{X}_b. \tag{4.8}$$

In other words, the transition matrix for a reverse transformation is calculated as the inverse of the basic transition matrix:

$$P_{BA} = P_{AB}^{-1}. \tag{4.9}$$

Coordinate transformation based on an inverse transition matrix in further subsections is more common than it being based on a direct transition matrix. The reason for this is that the new coordinate system is more often defined by the position of its origin and coordinate axes in the old coordinate system rather than vice versa.

When defining the new coordinate system, the task of plane fitting or plane definition often arises. In subsection 4.2.2 it arises as a definition of a horizontal plane based on a normal vector. In subsection 4.2.3 it arises as a plane fitting to the measured coordinates. The standard equation that expresses the plane in 3D space looks like this:

$$A * x_1 + B * x_2 + C * x_3 + D = 0. \tag{4.10}$$

Here A, B, C, D are the parameter coefficients of the plane. Obviously the values of A, B, C, D may vary in expressing the same plane while the proportion $A : B : C : D = const$. Here and below the case is considered when $A^2 + B^2 + C^2 = 1$. The vector $\mathbf{P} = \begin{pmatrix} A \\ B \\ C \end{pmatrix}$ is perpendicular to the target plane. If it is a normal vector to the plane (i.e. $A^2 + B^2 + C^2 = 1$) the equation may be transformed to the following:

$$Ax_1 + Bx_2 + Cx_3 - Ax_{01} - Bx_{02} - Cx_{01} = 0. \tag{4.11}$$

Here $X_0 = (x_1(0); x_2(0); x_3(0))$ is a certain point of the target plane for which D from the equation 4.10 is equal to $-Ax_1(X_0) - Bx_2(X_0) - Cx_3(X_0)$. This expression is the most useful for transformations of the coordinate systems. It enables plane definition based on the known values of the normal vector $\mathbf{P}$ and the origin point X_0 that lies on the plane.

Another way to express the value of one coordinate as dependent on two other coordinates is:

$$x_2 = a * x_1 + b * x_3 + c. \tag{4.12}$$

Where $a = A/B$, $b = C/B$, $c = \frac{-Ax_{01} - Bx_{02} - Cx_{03}}{B}$. This expression of planes is used in the MATLAB curve fitting toolbox. The transition to the standard form of plane equation may be done by defining the values of $\{A, B, C, D\}$ using the following formulas:

$$\begin{aligned} B &= \frac{-1}{\sqrt{a^2 + b^2 + 1}}, \\ A &= \frac{a}{\sqrt{a^2 + b^2 + 1}}, \\ C &= \frac{b}{\sqrt{a^2 + b^2 + 1}}, \\ D &= \frac{c}{\sqrt{a^2 + b^2 + 1}}. \end{aligned} \tag{4.13}$$

The expression 4.11 can be defined with these values.

4.2.2 Gravity-related coordinate system

Many of previously mentioned equations (e.g. 3.18, 4.1, 4.2, 4.3) assume the first dimension of the coordinate system is collinear with gravity, and the third is collinear with the horizontal direction of the object motion. The calculation of physical-based motion models in such a system is easier because there is no need to take gravity into account as a significant force in all of the dimensions besides the first one. For the current algorithm, the main reason for defining the gravity coordinate system is that it is simpler to distinguish the plane-of-flight (see section 4.2.3)from it rather than from the camera coordinate system.

In the experimental setup, gravity's direction is determined based on the heavy load suspended on the long nail. If two distant points, G_1 and G_2 , on the nail are marked, the direction of gravity may be determined as a vector from the upper point to the lower point. To define the origin of the gravity coordinate system, two points, L_1 and L_2 are picked on the light barrier that is mounted on the throwing device. The origin of the coordinate system is put in the middle between them, so it is near the launching point for the ball. x_3 and x_2 lie within the plane perpendicular to x_1. The direction of x_2 is set by the projection of L_1L_2 on this plane. In this case x_3 is near the nominal direction of throw. The illustration of this determination is given in figure 4.2.

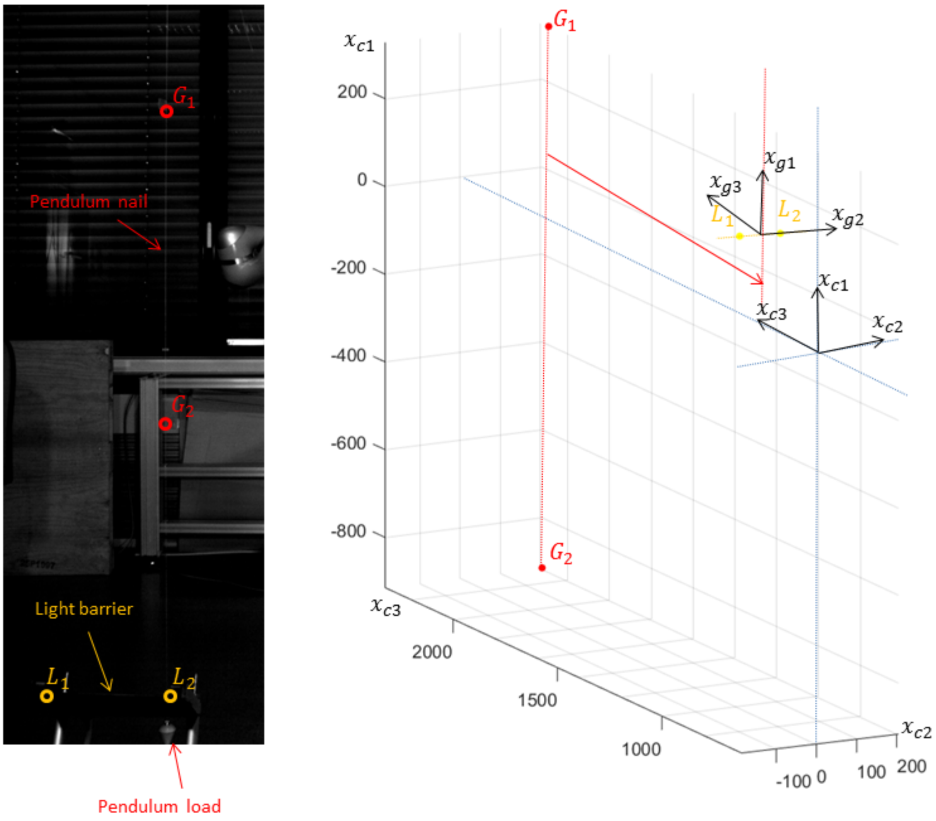


Figure 4.2: Determining the gravity coordinate systems based on camera images. The pendulum with a heavy load is hanged within the camera's FOV. When the pendulum is static, the direction of the nail is collinear with gravity. The gravity vector is determined based on the two red points, G_1 and G_2, on the nail. The origin of the coordinate system is defined based on the two yellow points, L_1 and L_2, on the light barrier. On the left side, the part of the calibration image with the highlighted support points G_1, G_2, L_1, and L_2 is shown. On the right part of the figure, these points are projected into the 3D camera coordinate system, and the position of the gravity coordinate system is shown.

The gravity coordinate system is determined based on knowing the 4 points $\{G_1, G_2, L_1, L_2\}$ in the following way.

- The origin point X_0 is defined as the middle point on the segment L_1L_2

$$\mathbf{X}_0 = \frac{1}{2} * (\mathbf{L}_1 + \mathbf{L}_2). \tag{4.14}$$

- The coordinate axis x_1 is defined as a normalized vector G_2G_1.

$$\mathbf{P}_1 = \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \frac{\mathbf{G}_1 - \mathbf{G}_2}{|\mathbf{G}_1 - \mathbf{G}_2|}. \tag{4.15}$$

  The horizontal plane may be defined using the equation 4.11 based on parameters obtained in 4.14 and 4.15.

- The coordinate axis x_2 is defined as a normalized projection of the vector L_1L_2 on the horizontal plane, which is expressed by 4.11. As the origin of the coordinate system is lying on L_1L_2 it is enough to determine the projection $\Lambda = \begin{pmatrix} x_{\Lambda 1} \\ x_{\Lambda 2} \\ x_{\Lambda 3} \end{pmatrix}$ of the point L_1. This determination is made in two steps. In the first step, the line that is normal to the plane (i.e. collinear with plane normal) and include the point L_1 is determined. The canonical line equation in space has the following appearance; in fact, this is a system of two equations expressed by one formula with two equals signs:

$$\frac{x_1 - x_{01}}{A} = \frac{x_2 - x_{02}}{B} = \frac{x_3 - x_{03}}{C}. \tag{4.16}$$

  In the second step, the system of the double line equation 4.16 and plane equation 4.11 is solved in order to define the point where the line intersects the plane:

$$\begin{aligned} \frac{1}{A} * x_{\Lambda 1} - \frac{1}{B} * x_{\Lambda 2} + 0 * x_{\Lambda 3} &= \frac{x_{01}}{A} - \frac{x_{02}}{B}, \\ 0 * x_{\Lambda 1} + \frac{1}{B} * x_{\Lambda 2} - \frac{1}{C} * x_{\Lambda 3} &= \frac{x_{02}}{B} - \frac{x_{03}}{C}, \\ A * x_{\Lambda 1} + B * x_{\Lambda 2} + C * x_{\Lambda 2} &= Ax_{01} + Bx_{02} + Cx_{01}. \end{aligned} \tag{4.17}$$

- Coordinate axis x_3 is defined as a cross product of x_1 and x_2.

4.2.3 2D representation for 3D coordinates

When the object is airborne, gravity is directed downwards, and air drag is directed backwards. Hence, if the Magnus effect and the asymmetry of a body does not have much influence (for non-rotating bodies like parallelepipeds), it can be assumed that the common force influencing the thrown object lies within a vertical plane. This is the so-called Plane-of-Flight (PoF), which is defined by two vectors: gravity $\mathbf{g}$ and the horizontal projection of the object velocity $\mathbf{v}_{hor}$. If all forces influencing the body are lying within such a plane and the velocity vector is also lying on this plane, the future motion of the object under the influence of these forces will also take place within this plane. In other words, the trajectory of the body is in this case planar. This circumstance lead to the idea of calculating the trajectory in a 2D coordinate system connected with PoF. This transformation has the following potential advantages:

- It provides the invariance to the horizontal direction of throw. The nearest neighbors prediction is based on searching the database for the trajectories that are similar to the current one. When a comparison of two throws with a different horizontal direction is done in a camera-related or gravity-related 3D coordinate system, the similarity of their shapes cannot be detected due to the large distances between points of the same timestamp. In planar coordinates, the trajectories of a similar shape are recognized as similar.
- Decreasing the dimensionality of the space mean decreasing the order of the computations. The solving task in such a system is potentially more simple and more robust to errors.
- Fitting the plane to measurement data allows the detection of outliers in the trajectory measurement. If the points are measured accurately, they would fit well to the PoF. The points with a bad fit are considered to be outliers and not included in further calculations.

The most useful representation of the plane for the fitting task is 4.12 because only three parameters are used (instead of six parameters in 4.11 and four parameters in 4.10)), and the values of these parameters are unique and independent of one another (instead of 4.10 where the values of the parameters may vary if the proportion between them is constant). Therefore, plane fitting here is a search in 3D parameter space. The task of fitting the PoF could become even simpler if we assume that the PoF is vertical in the original coordinate system. For instance, if the point $Q = \begin{pmatrix} x_{q1} \\ x_{q2} \\ x_{q3} \end{pmatrix}$ lies on the PoF, then the point $Q' = \begin{pmatrix} q \\ x_{q2} \\ x_{q3} \end{pmatrix}$ would lie on the PoF for any real q) the value of a in equation 4.12 is equal to zero and the plane is represented by:

$$x_2 = b * x_3 + c. \tag{4.18}$$

In this case the fitting task is a search in 2D parameter space. In fact, this is a line fitting within the plane expressed by x_3 and x_2. Here parameter b is equal to the tangent of the angle α between PoF and x_3 axis, while parameter c represent the point on the axis x_2 where it intersect the PoF (figure 4.3).

This simplification is the main motivation in determining the gravity-related coordinates in subsection 4.2.2 As in gravity-related coordinates, the PoF is vertical. If we operate with object coordinates in the camera coordinate system, the direction of the gravitational force that corresponds to x_1 direction in prior described calculations is not clear, and neither is the launching point that corresponds to the base of coordinate system. Defining the parameters of the Plane-of-Flight is in this case a task of plane fitting in 3D space. Figure 4.3 shows how this plane is positioned in the gravity coordinate system. It can be easily seen that if position of the PoF is determined by the angle α that is between the horizontal projection of the object velocity and x_1-axis, then it is an azimuth of throw. The PoF coordinate system is defined by the axis x_1 and axis x_4 which is collinear with the direction of throw.

Each point on the PoF corresponds to a point in 3D-space, and its 3D coordinates $\mathbf{X}_g = \begin{pmatrix} x_{g1} \\ x_{g2} \\ x_{g3} \end{pmatrix}$ may be calculated from plane coordinates $\mathbf{X}_p = \begin{pmatrix} x_{p1} \\ x_{p3} \end{pmatrix}$ using simple transformations:

$$\begin{aligned} x_{g1}(t) &= x_{p1}(t), \\ x_{g2}(t) &= (x_{p3}(t) - c) * \cos\alpha, \\ x_{g3}(t) &= (x_{p3}(t) - c) * \sin\alpha. \end{aligned} \tag{4.19}$$

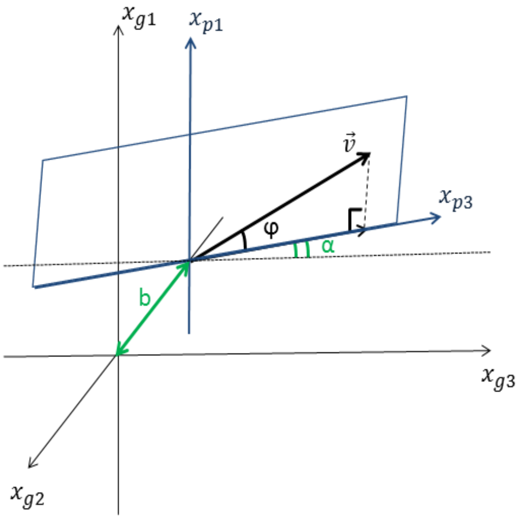


Figure 4.3: The plane of flight in a gravity-related coordinate system.

If the point lies in the PoF and its 3D-coordinates are known, it is not a problem to transform it into PoF coordinates.

$$x_{p3}(t) = \frac{x_{g3}}{\cos\alpha} + c. \tag{4.20}$$

The challenge in this approach is that for transformation α needs to be known. In fact, the value of the angle itself is not used in transformation equations. The needed parameters are $sin\alpha$ and $cos\alpha$. The tangent of α may be taken as a parameter b from equation 4.18. The $sin\alpha$ and $cos\alpha$ may be calculated from the tangent using these standard trigonometric equations:

$$\cos\alpha = (\tan^2\alpha + 1)^{-\frac{1}{2}} = \frac{1}{\sqrt{b^2+1}}, \tag{4.21}$$

$$\sin\alpha = \tan\alpha * \cos\alpha = \frac{b}{\sqrt{b^2+1}}. \tag{4.22}$$

Let $q_1 = \frac{1}{\cos\alpha}$ and $q_2 = \sin\alpha$ be the intermediate constants used in the transformation process of transforming coordinates from 3D to PoF with respect to these rules then it will look like::

$$\begin{aligned} q_1 =& \sqrt{b^2+1}, \\ x_{p3} =& q_1 * (x_{g3} - c), \\ x_{p1} =& x_{g1}. \end{aligned} \tag{4.23}$$

The process of transforming coordinates back from PoF to 3D looks like this:

$$\begin{aligned} q_2 =& \frac{\hat{b}}{q_1}, \\ x_{g3}(t) =& \frac{x_{p3}(t)}{q_1}, \\ x_{g2}(t) =& c * x_4(t), \\ x_{g1} =& x_{p1}. \end{aligned} \tag{4.24}$$

To make the transformation reference uniform, the PoF-coordinates of the object are considered to be a 3D vector. The coordinate values are stored in x_{p1} (height) and x_{p3} (distance). The x_{p2} coordinate is dummy (?). In the trajectory forecasting operation, it is assumed that its value is equal to zero every time. If the transformation process is done in the matrix form (i.e. it corresponds to equation 4.4), the parameters of the transformation matrix are defined in the following way.

- The origin point X_0 is defined as an intersection point between the PoF and x_{g2} axis.
$$\mathbf{X}_0 = \begin{pmatrix} 0 \\ c \\ 0 \end{pmatrix}. \tag{4.25}$$
- The coordinate axis x_{p1} is equal to x_{g1}.
$$\mathbf{P}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{4.26}$$
- The coordinate axis x_2 is defined as being collinear with the PoF projection on the horizontal plane:
$$P_2 = \begin{pmatrix} 0 \\ \sin\alpha \\ \cos\alpha \end{pmatrix}. \tag{4.27}$$
- The coordinate axis x_3 is defined as a cross product of x_1 and x_2.

The value of x_{p2} is equal to zero for all the points that lie on the PoF. For other points the value of this coordinate is equal to the distance from the PoF. Due to the observation errors, the measured values of the coordinates are commonly not lying in one plane, and an analysis of the x_{p2} allows the quality of the fit to be determined.

Previously in this subsection, it was assumed that the values of α and c are already known. However, in reality these values are defined by the deviation of the throwing device. As there are no means to measure this deviation directly, the values of α and c should be estimated by fitting the plane to the measurement data. This estimation may be implemented via various statistical methods. The method for estimating the PoF parameters should have the following properties [Mir15]:

- *Accuracy of fit*: Accuracy determination is a challenging task as no ground-truth data about object coordinates is available. If the points are translated to the estimated PoF and then back to 3D, their coordinates will be different from their original values. It is difficult to tell if this variation is a result of algorithm inaccuracy or the correction of measurement errors. According to [Pon15], there is no significant systematic error component in positioning spheres, and the deviations are random in appearance. The normal appearance of the difference (e.g. growing difference with time) means that the values of a and c are estimated with errors.
- *Robustness*: The algorithm must work correctly even if there are a few points with coordinates that are totally incorrect.

- *Stability*: During the object's flight, the values of α and c are recalculated after each new received frame. If the PoF estimation mechanism is likely, they must not vary strongly after each recalculation but should be adjusted to a certain value. The estimated values of the parameters should especially be the same when the whole trajectory $\{X_g(1), X_g(2), ..., X_g(n)\}$ is used for estimation and when only the part of the input $\{X_g(1), X_g(2), ..., X_g(m)\}$ is known. This requirement is connected to the circumstances of prediction. The predictor has only to compare the current trajectory measured in the initial part and the trajectories from the database that are fully tracked.
- *Performance*: As the application is in real time, all calculations must be done within the fixed period of time.

Several estimation methods were applied for determining α and b: least squares (LS), robust least squares (RLS), random sample consensus (RANSAC), mean calculation, and median calculation. Commonly, the implementation of the LS estimation is the following [Hla12]. The measurement vector $\mathbf{X}$ relies on the state vector $\mathbf{\Theta}$ as seen in the following equation:

$$\mathbf{X} = H * \mathbf{\Theta} + \mathbf{U}, \tag{4.28}$$

where H is a matrix that defines the relation between $\mathbf{X}$ and $\mathbf{\Theta}$ and $\mathbf{U}$ is measurement noise (i.e. the error of measurement). H has $m \times n$ size where m is the number of measured values in $\mathbf{X}$ and n is the number of parameters, describing system state in $\mathbf{\Theta}$. If the value of $\mathbf{X}$ is known, the least squares estimation of the system state $\mathbf{\Theta}$ may be obtained using the following equation [Hla12]:

$$\mathbf{\Theta} = H^{\#} * \mathbf{X}, \tag{4.29}$$

where $H^{\#}$ expresses the pseudo-inversion of matrix H.

To express the polynomial dependencies, the matrix equation is transformed to another view.

$$P * \mathbf{\Theta} = \mathbf{1}, \tag{4.30}$$

where P is $m \times n$ coordinate measurement matrix (m is the number of measurements, and n is the dimensionality of the space in case of linear dependency) $\mathbf{\Theta}$ is a vector of polynomial coefficients and $\mathbf{1}$ is an m-size vector of ones. In the case of line fitting in the $x_{g2}Ox_{g3}$ coordinate plane, this equation has the following appearance:

$$\begin{pmatrix} x_{g2}(1) & x_{g3}(1) \\ x_{g2}(2) & x_{g3}(2) \\ \vdots & \vdots \\ x_{g2}(m) & x_{g3}(m) \end{pmatrix} * \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{4.31}$$

The equation of the target line looks like this:

$$A * x_1 + B * x_2 = 1. \tag{4.32}$$

It can be easily transformed to the dependence of x_2 from x_3

$$x_1 = b * x_2 + c. \tag{4.33}$$

where $b = A^-1$ and $c = -B * b$.

The least squares estimation of A and B is obtained using the pseudo-inversion operation:

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} x_{g2}(1) & x_{g3}(1) \\ x_{g2}(2) & x_{g3}(2) \\ \vdots & \vdots \\ x_{g2}(m) & x_{g3}(m) \end{pmatrix}^{\#} * \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{4.34}$$

LS showed high speed (execution in 1-2 ms even in the MATLAB environment, which is relatively slow) and sufficient stability. It works accurately enough on the trajectories that do not contain outliers. Errors seem to be randomly distributed around zero and do not exclude several millimeters in most cases. The application of LS to the acquired data showed that LS is not robust. Outliers may lead to errors of up to several tens of cm.

The application of RLS [13] improves tolerance towards outliers. A robust least squares estimation from the curve fitting toolbox in MATLAB was applied. The likelihood that it was able to fit the points to the line in a 2D coordinate system was high. The value of x_2 for the points on trajectory does not usually exceed 1 mm. The work of the MATLAB fitting function is rather slow. It takes about 20 ms to calculate the value of α and b. This performance is insufficient because the camera frame-rate is 110 fps, meaning that obtaining the prediction information after every new frame all data processing cycle (object positioning on both images, stereo triangulation, transformation into PoF coordinates, prediction and transformation back to 3D) must not exceed 9 ms. Of course more low level implementation increases the speed.

The RANSAC PoF estimation is an application of the generalized RANSAC method to the line fitting task [Mir15]. Two points are picked randomly from the trajectory. Information from two points is enough to get a correct solution to 4.34. This solution gives a hypothetical PoF. After that it is checked to see how well the points fit to this plane. If the fit is good, the hypothesis has been proven. Otherwise, another pair of points is used. The RANSAC implementation was more robust than LS, and the accuracy was rather good. RANSAC implementation showed better robustness than LS and rather good accuracy.

RANSAC picks the points randomly, allowing even large amounts of data to be quickly processed. In trajectory forecasting this amount is relatively small (less than 100 frames). Therefore, a deterministic way of picking the pairs of points is proposed as it is not so calculation-expensive in this case [Mir15]. Let $\{X(1), X(2), ..., X(m)\}$ be a part of the trajectory available at the current moment (after m or $m+1$ frames, here m is even). $m/2$ pairs of points are taken: $\{X(1), X(m/2+1)\}$, $\{X(2), X(m/2+2)\}$, ..., $\{X(m/2), X(m)\}$. This allows $m/2$ hypotheses to be made about the position of the PoF (i.e. values of a and b). The simplest method to determine the PoF in such a situation is to compute the means of α and b. Somewhat unexpectedly, the accuracy and stability of such an estimation is no worse than for LS. Double estimation is made to improve robustness. After the first run, points with a bad fit to the plane are removed from the set. Then the estimation is repeated. The robustness of this algorithm is no worse than for RLS; however, it also cannot deal with the trajectories where the rate of outliers is high (the systematic error reaches several centimeters).

If the median values of α and b are used instead of the mean values, the robustness of the estimation increases. The median is more robust with outliers because in a sorted list of values, they will either be in the first or in the last position, and the middle part of the list consists of inliers. Therefore, the median calculation is chosen as a method for estimating PoF. It is accurate in that errors seem to be randomly distributed around the value zero and they are usually less then 1 mm. Only in a few the cases, did the difference exclude 2 mm. It is also robust in that the accuracy stays the same even when the number of outliers is about 20% of the whole dataset and stable because the values of a and c do not vary more than in second digit depending on is $X(1:n)$ or only $X(1:m)$). It is also faster than any other method. The execution in MATLAB takes about 1 ms to process a trajectory consisting of 80 points [Mir15].

A comparison of the accuracy for the six proposed methods is shown in table 4.1. The standard deviation for the fitted points from the plane was measured based on the fourth dataset from table 3.2. It includes 111 trajectories with 100 points each, meaning there are 11100 points in total. The standard deviations for the six proposed estimation methods are listed in the table. In the left column, the standard deviation of x_2 (i.e. the distance from the points to the fitted plane) is listed. This parameter allows the accuracy of PoF transformation to be estimated for the fixed amount of data. The parameters in the two columns on the right allow the stability of the fit to be estimated. They show the standard deviations for the parameters of the PoF (α and c) estimated with varying amounts of available frames. The amount of frames was changed in the following way. In the first iteration, the first 40 frames were used for estimation. In the second one, 41 were used. In the last iteration, information from all 100 frames was taken. From the data in the table, it appears that the median estimation is the most accurate method.

Table 4.1: Standard deviation for plane fitting using various proposed methods.

Method	$\sigma(x_{p2})$, mm	$\sigma(\alpha)$, $10^{-3}rad$	$\sigma(c)$, mm
LS	6.50	218.8	365.1
RLS	0.95	3.3	1.3
RANSAC	0.93	3.7	2.0
mean	3.67	4.1	2.0
median	0.85	3.2	1.1

The accuracy and stability of the median PoF estimation also proves that the PoF model is valid, and there is no significant side-directed force influencing the thrown object. If the horizontal projection of the trajectory is not a straight line, errors would be dependent on the time, and the parameters of the PoF would vary depending on m. As median fit does not include such an effect, the influence of side force is considered to be negligible. These results show that PoF transformation is valid for spherical objects thrown by linear launching device [Mir15].

The value of α demonstrates the deviations of the throwing device in terms of horizontal direction of throw. The histogram of α variations from the mean value is shown in figure 4.4. As the value of angle is low, it is nearly equal to the value of the tangent. Note that the angle $30 * 10^{-3}rad$ is significant. It corresponds to the shift of 6 cm at the distance of 2 meters.

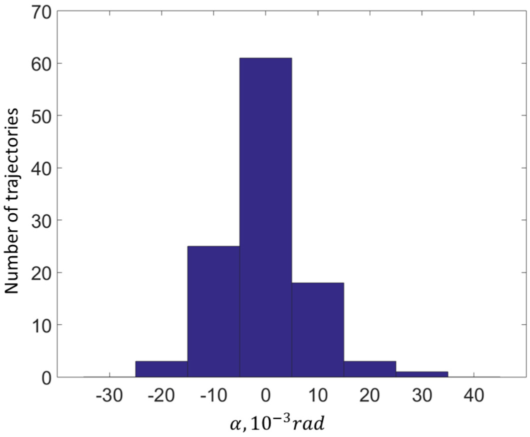


Figure 4.4: The histogram of α values for the trajectories from the dataset.

4.2.4 Invariance to release point

The PoF representation provide independence of trajectories comparison to the direction of throw. One more aspect is the dependence of the comparison to the position of the launching point in the coordinate system. If the trajectories have the same shape but the position of the release points for them is different, the euclidean distance between them will be large and they will not be recognized as similar. For the correct comparison of the trajectories they must have the common point(point with a certain timestamp z such that the coordinates of the ball for all the trajectories are equal in this point). For the purpose of interpretability it is good that the coordinate of this common zero-point are equal to $\begin{pmatrix}0\\0\\0\end{pmatrix}$, i.e. that the zero-point is equal to the origin point of the coordinate system.

The origin point of gravity-related and PoF-related coordinate systems are defined in such way that they are near to the point in space where the ball intersect the light-barrier. When it happens the camera system is triggered and the tracking process is started. I.e. in ideal case the position of the ball at $t = 0$ is equal to $\begin{pmatrix}0\\0\\0\end{pmatrix}$ and $z = 1$. In reality that does not happen: the position of the object on the first frame differ up to several millimeters from zero. Wrong estimation of the launching point may lead to the wrong results in trajectory comparison. The idea is that it is possible to translate the trajectory to the new coordinate system where the zero-point is more likely than in X_g or X_p. For this purpose certain point of the trajectory is chosen as a zero-point and its coordinates are subtracted from next points on the trajectory. In other words coordinate transformation consist in subtracting the coordinates of the zero-point.

$$\begin{aligned} x_{z1}(i) =& x_{p1}(i+z) - x_{p1}(z), \\ x_{z2}(i) =& 0, \\ x_{z3}(i) =& x_{p3}(i+z) - x_{p3}(z). \end{aligned} \tag{4.35}$$

where x_{z1}, x_{z2} and x_{z3} are object coordinates in zero-point coordinate system. The subtraction operation is removed from the second row as we deal with 2D PoF coordinates and the second coordinate is not used in further calculations (however it is kept in order to provide compatibility with 3D transformation matrices).

The matrix coordinate transformation formula in this case will have the following view:

$$\begin{pmatrix} x_{z1}(i) \\ x_{z2}(i) \\ x_{z3}(i) \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -x_{p1}(z) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -x_{p3}(z) \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} * \begin{pmatrix} x_{p1}(i+z) \\ x_{p2}(i+z) \\ x_{p3}(i+z) \\ 1 \end{pmatrix} \tag{4.36}$$

The question is how to choose the zero-point in the right way. It is important to find such a value of z for which the coordinates of ball center are measured with sufficient accuracy. On the other hand the zero-point should lie in the beginning of trajectory. The comparison of trajectories is less erroneous when the difference between them is big (it should be higher than the possible estimation errors). The difference between measured points is high if they lie far from the zero-point. Therefore it is better if the zero-point is near to the launching point. On the other hand visual analysis from subsection 3.3.1 showed that the measurement of object position is not very accurate on first five frames. Therefore the 7th frame of the video sequence is proposed to be the zero-point.

The geometrical relations between gravity-related, PoF-related and zero-point-related coordinate systems illustrated on figure 4.5.

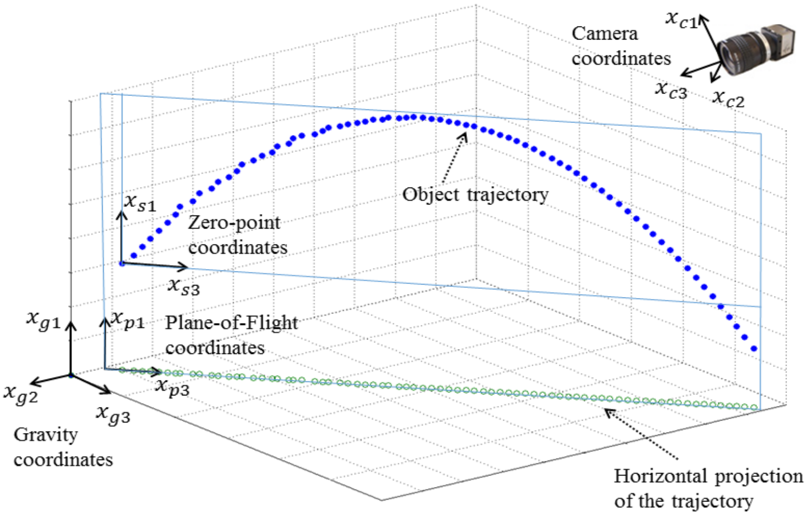


Figure 4.5: Relative location of camera-related, gravity-related, PoF-related and zeropoint-related coordinate systems.

4.2.5 Reverse transform and robot coordinate system

Coordinate transformations described in previous subsections have an aim to improve the efficiency of the forecasting algorithm. The forecasting operation is made in the zero-point coordinate system and the result of this operation is also expressed in this system. However the robot controller deal with world coordinates therefore the reverse coordinate transformation of the prediction results is required.

In previous subsections the following transformation matrices were determined: P_{cg} from camera-related to gravity-related system (equations 4.14, 4.15, 4.16, 4.17), P_{gp} from gravity-related to PoF-related system (equations 4.25, 4.26, 4.27), P_{pz} from PoF-related to zero-point-related system (equation 4.36). The coordinate transformations are made by multiplication of inverses of these matrices on the coordinates. It mean that reverse transformation is made by multiplying these matrices on the object coordinates:

$$\begin{pmatrix} x_{c1}(i) \\ x_{c2}(i) \\ x_{c3}(i) \end{pmatrix} = P_{cg} * P_{gp} * P_{pz} * \begin{pmatrix} x_{z1}(i-z) \\ x_{z2}(i-z) \\ x_{z3}(i-z) \end{pmatrix}. \tag{4.37}$$

Robot controller use a specific coordinate system connected with a base of the robot arm. The matrix P_{rc}, which connect this system with camera coordinate system may be obtained during the additional calibration (considered in chapter 5). The equation for transforming the final coordinates to the robot system will have the following view:

$$\begin{pmatrix} x_{r1}(i) \\ x_{r2}(i) \\ x_{r3}(i) \end{pmatrix} = P_{rc} * P_{cg} * P_{gp} * P_{pz} * \begin{pmatrix} x_{c1}(i-z) \\ x_{c2}(i-z) \\ x_{c3}(i-z) \end{pmatrix}, \tag{4.38}$$

where $\begin{pmatrix} x_{r1}(i) \\ x_{r2}(i) \\ x_{r3}(i) \end{pmatrix}$ is a vector of object coordinates in robot-related coordinate system. Both P_{cg} and P_{rc} are defined prior to the throw. It allow determining the matrix for transition from gravity to robot coordinate system:

$$P_{rg} = P_{rc} * P_{cg}. \tag{4.39}$$

Coordinate transformation from zero-point-related to robot-related system in this case will have the following view:

$$\begin{pmatrix} x_{r1}(i) \\ x_{r2}(i) \\ x_{r3}(i) \end{pmatrix} = P_{rg} * P_{gp} * P_{pz} * \begin{pmatrix} x_{c1}(i-z) \\ x_{c2}(i-z) \\ x_{c3}(i-z) \end{pmatrix}, \tag{4.40}$$

4.3 Predictor

The kNN principle is proposed as a basic method for trajectory prediction in subsection 2.5.4. This section tell how this method is adapted for this task. Subsection 4.3.1 describes the calculating the reference of object future positions based on corresponding positions of two nearest neighbors (which are already known). Subsection 4.3.2 consider the question of how to define, which trajectories from the database are the nearest neighbors.

4.3.1 Forecasting operation

Consider simple application of kNN for time series forecasting. Let $\mathbf{X}_C$ be a measured part of the current trajectory and a set of input-output pairs $L = \{(\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Y}_2), \ldots, (\mathbf{X}_m, \mathbf{Y}_m)\}$ be a dataset of previous sample trajectories ($\mathbf{Y}_c$ is a part of the trajectory which must be predicted). The predictor compares $\mathbf{X}_C$ with inputs from dataset and search for an example $\mathbf{X}_s$, which is the most similar to $\mathbf{X}_C$ ("nearest"). The final part of this nearest trajectory $\mathbf{Y}_s$ is taken as a predicted part of the current trajectory. This is 1-nearest neighbor prediction.

$$\begin{aligned} s =& arg \min_{i=1}^{m} |X_c - X_i|, \\ Y_c =& Y_s. \end{aligned} \tag{4.41}$$

The question of how to determine the similarity of trajectories is considered in the next subsection. For this subsection let us assume that there is a mechanism allowing such a determination. Consider now forecasting based on two nearest neighbors. Two trajectories from the dataset are taken, which are the most similar to the current one, and the forecast is calculated as a mean of their $\mathbf{Y}$:

$$\mathbf{Y}_c = \frac{\mathbf{Y}_1 + \mathbf{Y}_2}{2}. \tag{4.42}$$

This operation can be easily adapted to the arbitrary number k of taken nearest trajectories:

$$\mathbf{Y}_c = \frac{1}{k} * \sum_{j=1}^{k} \mathbf{Y}_j. \tag{4.43}$$

Thus simple k-NN predictor calculate the output value as a mean of outputs of k examples taken from the dataset, which inputs are the nearest to the current input.

Weighted k-NN is a modified version of such approach. The goal of this modification is to consider not only the fact, that "the neighbor is near", but also numerical distance between current example and its neighbors. The output value of the predictor in such case is calculated, using the following formula:

$$\mathbf{Y}_c = \sum_{j=1}^{k} w_j * \mathbf{Y}_j. \tag{4.44}$$

Here $\sum\limits_{j=1}^{n} w_j = 1$. The values of $\{w_1, w_2, \ldots, w_k\}$ are defined with respect to the distance from the corresponding neighbor to the current example. The way of weighting the trajectories for the case of two neighbors is shown below.

On the figure 4.6 the plot for three trajectories in the zero-point coordinate system is shown. C is the current trajectory, A and B are its nearest neighbors, taken from the dataset. We know measurement area trajectory X_A, X_B and gripping area trajectory Y_A, Y_B for A and B, and measurement area trajectory X_C for C. It is assumed that at any time moment the object from trajectory A is higher and farther than the object from trajectory C, while the object from trajectory C is higher and farther than the object from trajectory B. Other cases are discussed in subsection 4.3.2 and section 4.4. 2-nearest neighbors forecasting[1] is applied. For each measured coordinate of the object C in MAT we know the corresponding (i.e. measured at the same period after the throw) coordinates of A and B. We can calculate distances between these three points for each t in both dimensions:

---

[1] Usually it is not recommended to use even number of neighbors for classification tasks. Reason is that it can cause uncertainty. Consider simple situation. We have k taken nearest neighbors. If $\frac{k}{2}$ of them belong to class A and $\frac{k}{2}$ of them belongs to class B, the simple kNN classifier can not classify current example to A or B. In the case of trajectory prediction considered above it is not a classification task. The output value is not taken as a result of neighbor voting, but calculated numerically on its base. Hence taking even number of nearest neighbors is correct for trajectory prediction and does not cause any failures. Use of alternative values of k for trajectory prediction is briefly discussed in section 5.1. Experiments there showed that $k = 2$ provide the best prediction accuracy

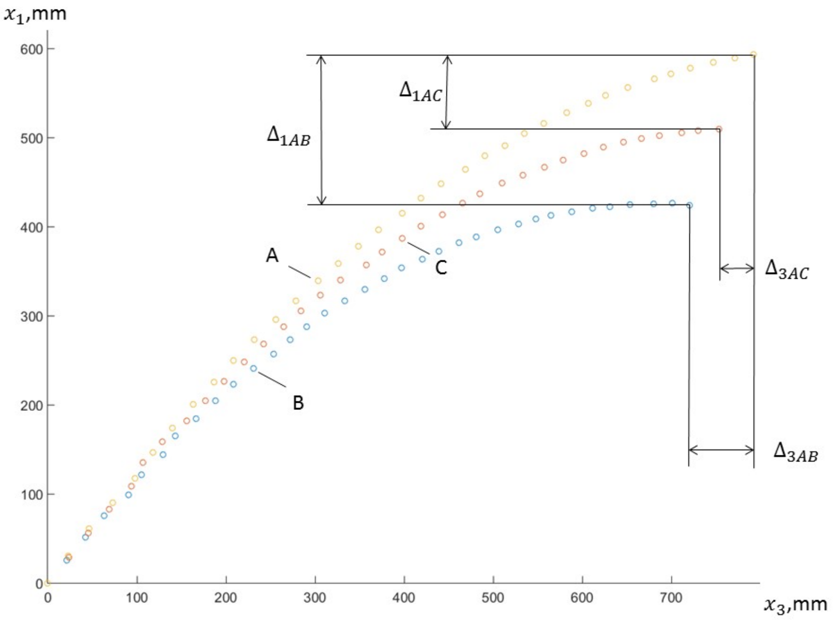


Figure 4.6: Distances between the corresponding points on the current trajectory C and trajectories A and B from the dataset

$$\begin{aligned}\Delta_{1AC}(t) &= x_{1A}(t) - x_{1C}(t),\\ \Delta_{1AB}(t) &= x_{1A}(t) - x_{1B}(t),\\ \Delta_{3AC}(t) &= x_{3A}(t) - x_{3C}(t),\\ \Delta_{3AB}(t) &= x_{3A}(t) - x_{3B}(t),\end{aligned} \tag{4.45}$$

and then proportions

$$\begin{aligned}d_1(t) &= \frac{\Delta_{1AC}(t)}{\Delta_{1AB}(t)},\\ d_1(t) &= \frac{\Delta_{1AC}(t)}{\Delta_{1AB}(t)}.\end{aligned} \tag{4.46}$$

The proportion vector $\mathbf{D}(t) = \begin{bmatrix} d_1(t) \\ d_2(t) \end{bmatrix}$ is calculated for each available measurement, and then its mean value $\mathbf{D} = \{\hat{d_x}; \hat{d_z}\}$ is estimated.

$$\mathbf{D} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \frac{1}{n} * \sum_{t=1}^{n} \begin{bmatrix} d_1(t) \\ d_2(t) \end{bmatrix}. \tag{4.47}$$

Now if we know measurement results for $\mathbf{Y}_1$ and $\mathbf{Y}_2$ and need to define value of coordinates for $\mathbf{Y}_c$, the following formula may be used for any t in catching area:

$$\begin{aligned}\hat{x_{1C}}(t) &= x_{1B}(t) + \hat{d_1} * (x_{1A}(t) - x_{1B}(t)),\\ \hat{x_{3A}}(t) &= x_{3B}(t) + \hat{d_2} * (x_{3A}(t) - x_{3B}(t)).\end{aligned} \tag{4.48}$$

This equation can be converted to the template equation of WKNN prediction 2.22 in the following way:

$$\begin{aligned} x\hat{1C}(t) &= \hat{d_1} * x_{1A}(t) + (1 - \hat{d_1}) * (x_{1B}(t)), \\ x\hat{3C}(t) &= (1 - \hat{d_2}) * x_{3A}(t) + \hat{d_2} * (x_{3B}(t)). \end{aligned} \tag{4.49}$$

If the simple kNN is used instead of WKNN both weights in this formula are replaced by $\frac{1}{2}$:

$$\begin{aligned} x\hat{1C}(t) &= \frac{1}{2} * x_{1A}(t) + \frac{1}{2} * (x_{1B}(t)), \\ x\hat{3C}(t) &= \frac{1}{2} * x_{3A}(t) + \frac{1}{2} * (x_{3B}(t)). \end{aligned} \tag{4.50}$$

If the C lies between B and A the values of d_1 and d_2 lie between 0 and 1. Therefore the weights of the neighbors also lie between 0 and 1.

The process of calculation of future position based on two known neighbors using equations 4.45, 4.46, 4.49 or equation 4.50 is defined here as "k-NN forecasting operation" (KFO). The input for KFO include $\mathbf{X}_C$, $\mathbf{X}_A$, $\mathbf{Y}_A$, $\mathbf{X}_B$, $\mathbf{Y}_B$. Estimate of $\mathbf{Y}_c$ is an output of the operation.

4.3.2 Search for nearest neighbors

KFO does not include definition of what trajectories in $\mathbf{L}$ are $(\mathbf{X}_A; \mathbf{Y}_A)$ and $(\mathbf{X}_B; \mathbf{Y}_B)$. Complete prediction operation consist of defining the nearest neighbors $(\mathbf{X}_A; \mathbf{Y}_A)$ and $(\mathbf{X}_B; \mathbf{Y}_B)$ among a number of trajectories in the dataset $\mathbf{L}$ (k-NN search operation, KSO) and then applying KFO to them. The similarity of trajectories here is defined as a similarity of object coordinates in the points with the same timestamp. Therefore the euclidean distance between the corresponding points shows, how far the trajectories lie from each other. The euclidean distance is calculated by the equation, well-known from the school geometry:

$$d_{ij} = d_i(t) = \sqrt{(x_{1i}(t) - x_{1c}(t))^2 + (x_{3i}(t) - x_{3c}(t))^2}. \tag{4.51}$$

Here i is an index of the trajectory in the dataset, $\begin{pmatrix} x_{1c} \\ x_{2c} \\ x_{3c} \end{pmatrix}$ is a vector of coordinates from the current trajectory and $\begin{pmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \end{pmatrix}$ is a corresponding vector from the trajectory, which is considered as a possible neighbor. The meaning of j is described in the next paragraph. Δ_i is defined as a mean Euclidean difference between the corresponding points:

$$\Delta_i = \frac{1}{n} \sum_{j=1}^{n} |d_{ij}|. \tag{4.52}$$

Here n is a number of points used for distance calculation and j is an index of the such points in the list. The trajectory with the minimum mean distance is chosen as a nearest neighbor.

$$s = arg \min_{i=1}^{m} |\frac{1}{n} \sum_{t=1}^{n} |d_i(t)||. \tag{4.53}$$

Here s is an index of the nearest neighbor in the trajectory database. The task of distance calculation here is easier, than in many other classification, regression and forecasting task, where the k-NN is used, because all used dimensions have the same scale.

Note that j in equation 4.52 is not equal to the timestamp t of the measurement. Timestamps show the number of frames left after the launch by the moment of the current measurement. j has no such meaning: the points for estimating the distance may be taken from the whole available part of the trajectory but there is no requirement that all available frames must be used. E.g. if some frames are lost by the observer, they are not used for distance calculation. Let the used frames list (UFL) be a set of timestamps for the frames, which are used in distance calculation. E.g. if the UFL $U = \{24, 26, 27\}$, it means that the distance between the trajectories will be calculated in the following way:

$$\Delta_i = \frac{1}{3}(d_i(24) + d_i(26) + d_i(27)).$$

The mapping from t to j in this case is the following:

$$\begin{aligned}(t = 24) \mapsto &(j = 1),\\ (t = 26) \mapsto &(j = 2),\\ (t = 27) \mapsto &(j = 3).\end{aligned}$$

Note that correct comparison of the distances between two pairs of trajectories is possible only if the UFL for calculating these distances were the same. The distance between the corresponding points $X_c(t)$ and $X_i(t)$ grow with the growth of t and for the same pair of trajectories in the common case the distance calculated with $U = \{31, 32, 33\}$ would be higher than the distance calculated with $U = \{1, 2, 3\}$. This property is proved by the throwing experiments. On figure 4.7 the histogram of such distances is given. It represent the mean distances between the points with respective frame number. These values were obtained based on 111 trajectories from the dataset of "August-2015" (see subsection 3.1.4). It may be seen that the mean distance grow nearly linear with the growth of the frame number.

What part of available frames should be inserted into the UFL? On one hand more items used for search is better for the quality of the estimation. On the other hand the accuracy of proportion estimation is growing with the grow of timestamps: the distances are higher. Therefore the UFL in various version of the algorithm has the following view.

$$U = l, l + 1, l + 2, ..., n. \tag{4.54}$$

Here n is a timestamp of the last available frame by the current moment. l is a certain number (obviously $l \in [k, n]$). To define the most appropriate UFL specific computational experiments was conducted (described in chapter 5). These experiments showed that the change of l has a moderate influence on the prediction accuracy. In the conditions of the experiments $l = 15$ showed the best results, therefore it was taken as a nominal value for the implementation.

4.4 Allocating the subset of neighbors

Previous subsections consider the evaluation of the algorithm under some specific assumptions. The main such assumptions are the following:

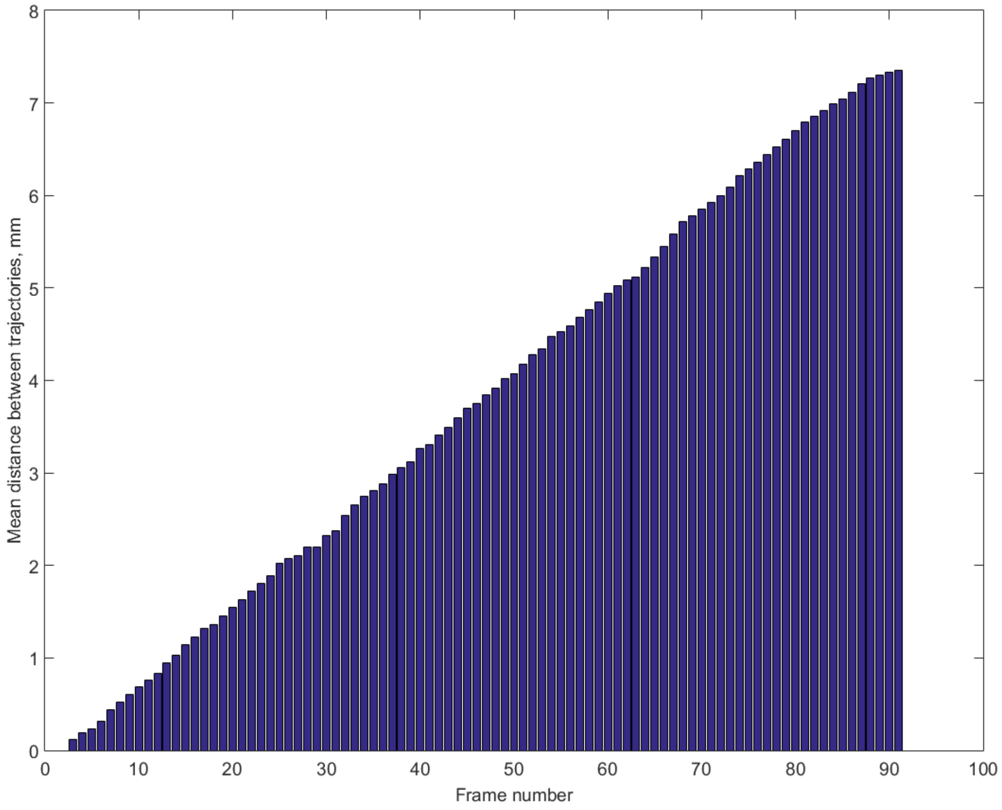


Figure 4.7: Mean distance between the points from the database according to their frame number.

1. Possibility to define nearest neighbors such that the current trajectory lie between them in both dimensions at the whole duration of flight;
2. Stability of the proportions d_1 and d_2 from equation 4.46 in time.

The rule that the current trajectory must lie between two nearest neighbors in both dimensions may be simply inserted into the KSO. For this purpose the search for higher and lower nearest neighbor is made separately. The simple heuristic for search is the following. For each trajectory $\mathbf{X}_i$ for each timestamp t it is checked if the object lies above the object from the current trajectory $\mathbf{X}_c$. If $x_{1c}(t) > x_{1i}(t)$ for most of the timestamps trajectory $\mathbf{X}_i$ is added to the list of lower trajectories.

Stability of the proportions d_1 and d_2 in time is not an obvious truth. Process simulation showed that in common case (i.e. when three random trajectories are taken from the set) these proportions are not stable and the object, which was between two others at the beginning of flight, could be above or below each other at the end (i.e. the trajectories are crossing, figure 4.8).

Different results may be seen when the specific adjustment of the throwing parameters is made in the simulation. Throwing velocity may be set in two ways: in Cartesian coordinates or in polar coordinates. In both domains launching velocity consists of two parameters: value v and elevation ϕ in polar domain or horizontal projection v_3 and vertical projection v_1 in Cartesian

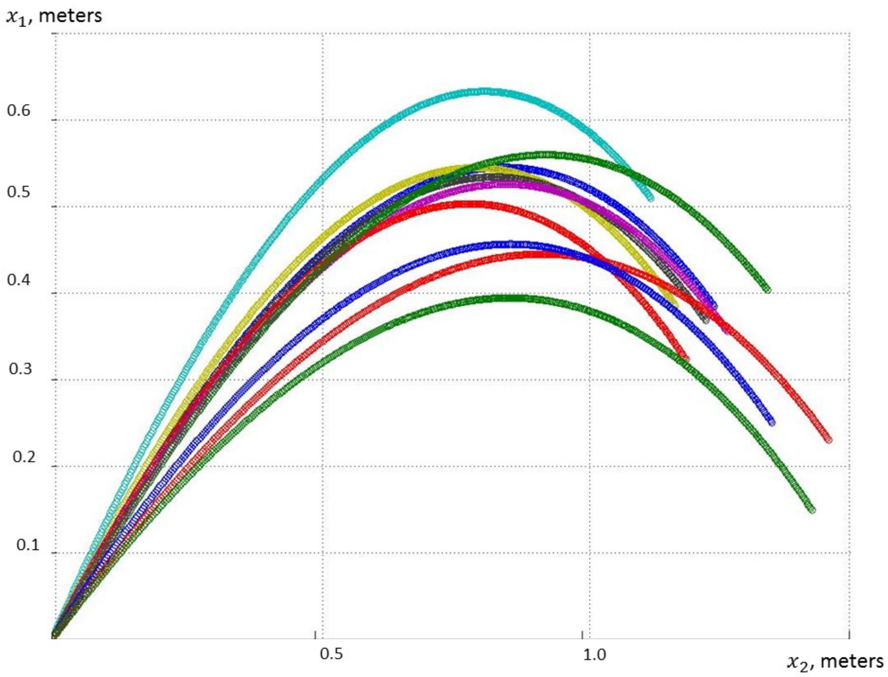


Figure 4.8: Plane-of-Flight projections of trajectories randomly generated in the simulation environment. It may be seen that the trajectories are crossing.

domain. The shape of the trajectory may be defined by both of the parameters of any domain. In common case (figure 4.8) of the simulation the values of launching parameters were set to be random normally distributed around the nominal values. Standard deviations of these parameters were set to the same value (0.1 m/s for both horizontal and vertical velocity components). This lead to crossing trajectories, which may be seen on figure 4.8.

Another situation may be seen when the standard deviation for one launching parameter is set to be much smaller than for another one. Example plots for ten trajectories generated such asymmetric setting of the launching parameters are presented on figure 4.9 (parameters are set in polar domain; standard deviation for launching velocity set to $0.01 m/s$, standard deviation for elevation of throw is set to $0.1 rad$), figure 4.10 (parameters are set in polar domain; standard deviation for launching velocity set to $0.1 m/s$, standard deviation for elevation of throw is set to $0.01 rad$), figure 4.11 (parameters are set in Cartesian domain; standard deviation for the vertical velocity set to $0.01 m/s$, standard deviation for the horizontal velocity is set to $0.1 m/s$), and figure 4.12 (parameters are set in Cartesian domain; standard deviation for the vertical velocity set to $0.1 m/s$, standard deviation for the horizontal velocity is set to $0.01 m/s$). It may be seen that these plots have more regular appearance than random plots from figure 4.8. The trajectories from each of these four plots have similar appearance to each other (e.g. the last points of the trajectories on figure 4.12 have nearly the same value of the horizontal coordinate).

This is just an intuitive perception got from the plots. For the predictor accuracy it is important to know the following: are the proportions d_1 and d_2 for the trajectories generated with this setup stable in time or not? If one trajectory lie between two others in the beginning, will it lie in between in the end? Numerical experiments with the simulation showed the following answers on these questions:

1. *Throwing velocity is set in polar domain with high deviation of the angle and low deviation*

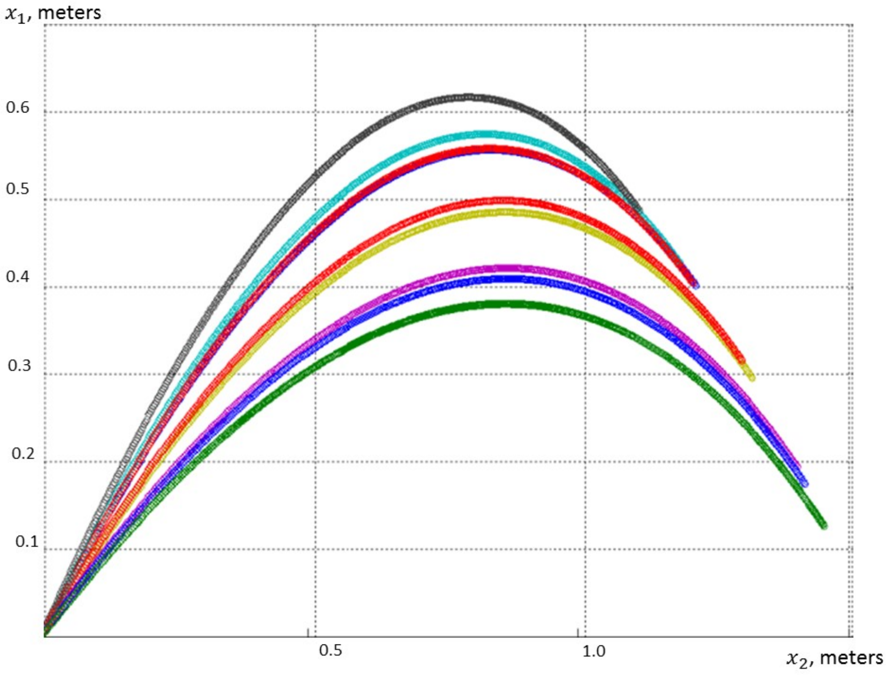


Figure 4.9: Plane-of-Flight projections of trajectories belonging to one cluster with respect to v.

of the value: The proportion is kept stable (it changes in the second order only). If one trajectory is between two others in both dimensions in the beginning, it will lie in between in both dimension in the end.

2. *Throwing velocity is set in polar domain with low deviation of the angle and high deviation of the value*: The deviations of the proportions (and crosses of the trajectories) take place in very beginning of the flight. After first 60 milliseconds proportions become stable and no more crosses are detected.

3. *Throwing velocity is set in Cartesian domain with low deviation of the vertical projection and high deviation of the horizontal projection*: proportion is not stable and trajectories are often crossing in both dimensions.

4. *Throwing velocity is set in Cartesian domain with high deviation of the vertical projection and low deviation of the horizontal projection*: for the vertical dimension proportion is kept stable and trajectories do not cross. In horizontal dimension this requirement is not satisfied however it is more important that the corresponding coordinates in horizontal dimension are nearly the same for all trajectories.

v, ϕ and v_1 are changing in time, hence, it can not be said that "for this trajectory it $v = 5,6$ and for this trajectory $v = 5,4$". We can talk only about the value of these parameters at the certain time moment. Due to this fact it is hard to estimate the value of these parameters based on erroneous measurements. On the other hand the horizontal velocity does not change much with time. The mean value of this parameter may be estimated as a result of dividing $x_3(t)$ by t.

Main advantages of k-NN in comparison with other sampling-based techniques (e.g. Neural Networks and Support Vector Machines) are simplicity of implementation and good interpretability: predictor made decision based on concrete examples. Main disadvantages are connected with the

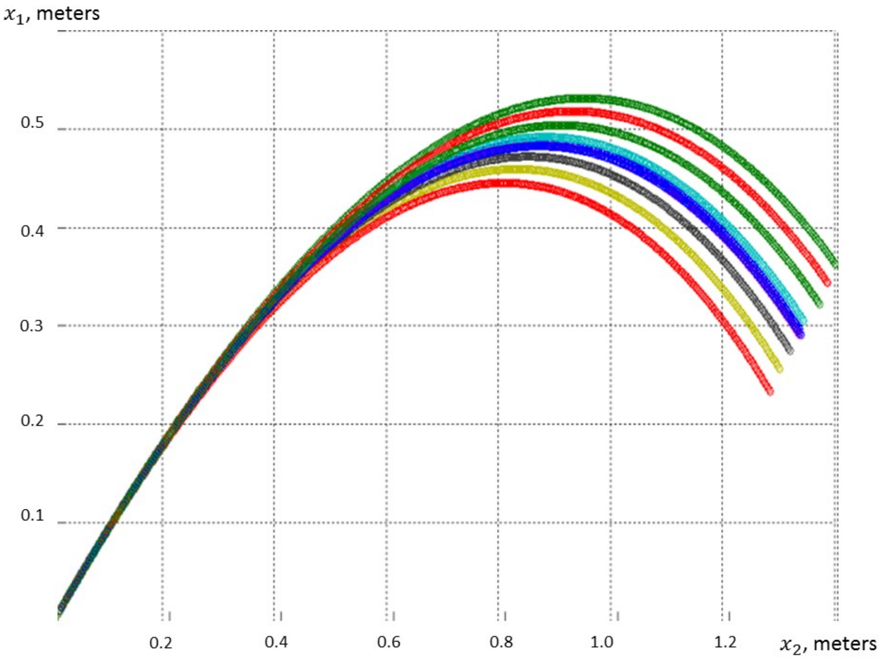


Figure 4.10: Plane-of-Flight projections of trajectories belonging to one cluster with respect to ϕ.

fact, that we does not modify our learning sampling to more simplified model but store them all in the database. When we apply k-NN forecasting we should compare our current trajectory with all trajectories from the dataset. This circumstance produces high memory and time costs of k-NN with huge databases. Speed of processing is very important for time-critical projectile prediction task. Hence comparison of the current trajectory with high number of samples is unlikely.

Here the approach is proposed based on comparing the current trajectory not with all trajectories from the dataset but only with a relatively small subset. It allow decreasing the volume of computations: even large databases may be processed relatively fast. More important aspect of this proposal is that proportion stability assumption is true for the trajectories inside the subset. This is achieved by making one of the velocity parameters nearly the same for all trajectories from the subset. In other words the basic principle of the subset allocation is the following: the current trajectory is compared only with trajectories from the dataset such that the value of certain velocity parameter p for these trajectories is nearly equal to the corresponding parameter of the current trajectory. What parameter should be taken as a base for such allocation? Let consider parameters listed above as a potential p:

1. *Velocity value and elevation in polar domain*: Both these parameters are not likely as they are hard to be estimated. The light barriers provide the measurement of the velocity projection on the direction perpendicular to them but this is not a value of the velocity. Accurate measurement of the velocity value and elevation on the short term is not possible as the accuracy of stereo positioning is not enough for this task (see section 3.3.3).

2. *Vertical projection of the velocity*: this velocity is not likely as it does not provide proportion stability.

3. *Horizontal projection of the velocity*: This parameter provide high stability of the proportion in vertical coordinates and it is relatively easy to estimate. Horizontal velocity does not

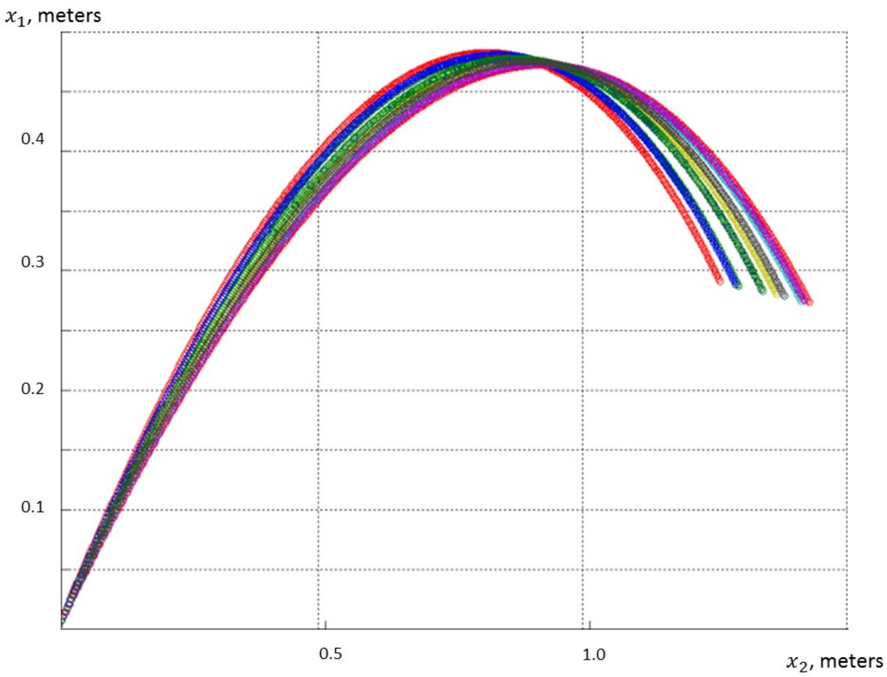


Figure 4.11: Plane-of-Flight projections of trajectories belonging to one cluster with respect to v_1.

> change significantly in time therefore calculation of the mean horizontal velocity after a relatively long period of time may be used to estimate the horizontal velocity at the moment of throw. Another advantage of taking the horizontal velocity as a sorting parameter is that all trajectories have nearly the same value of x_3 at the corresponding time moments. Therefore by estimating the value of the horizontal velocity we can roughly estimate the time, when the ball reach gripper workspace. The process of estimating the horizontal velocity is described below.

According to the definition of mean horizontal velocity it is estimated as a result of dividing object horizontal path by the time left after the throw. In zero-point coordinate system it corresponds to the division of the coordinate x_3 by the time t left after the zero-point:

$$\hat{v_3}(t) = \frac{x_3(t)}{t} = \frac{x_3(t)}{n * \tau}. \tag{4.55}$$

Here τ is length of the inter-frame period and n is number of frames left.

Absolute errors of stereo positioning achieve several millimeters (section 3.3), however for the velocity estimation relative errors are critical. If the value of x_3 is 10 mm and absolute error is 1 mm, the relative error is obviously around 10%. If absolute error is the same and x_3 is equal to 100 mm the relative error would be around 1%. This example shows that the accuracy of velocity estimation improves with grow of the value of x_3. On the other hand this operation must not be done too late: the system need some time to perform the prediction and to execute the catching movement. In the basic experimental setup frames with numbers from 35 to 39 were used for calculating mean horizontal velocity [Mir15]. Taking these frames allow robust estimation of the horizontal velocity which may be seen on figure 4.13 taken from [Mir15].

This figure represent the result of sorting trajectory database with respect to v_3. Trajectories with low values of v_3 were put to the beginning of the list while trajectories with high values were

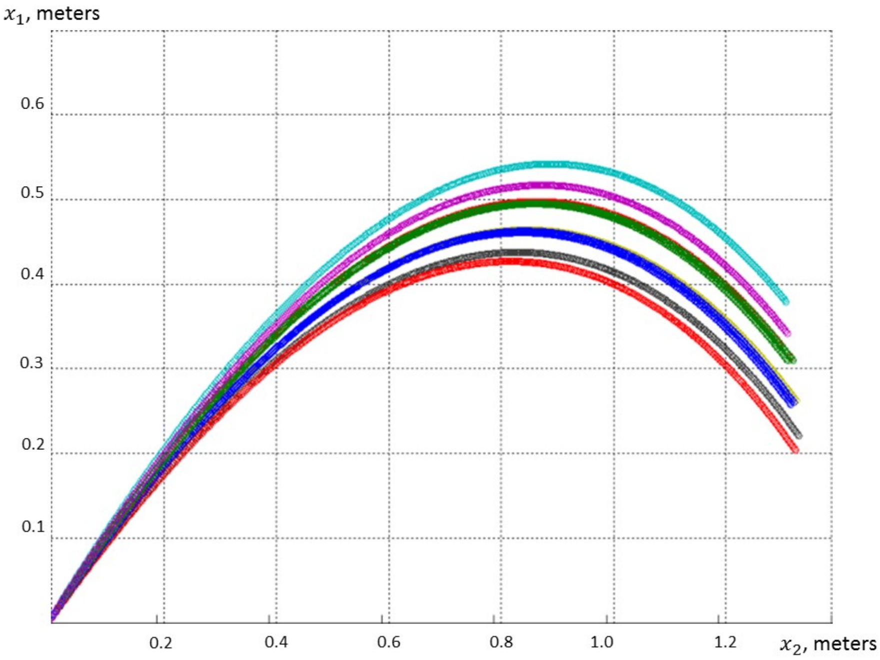


Figure 4.12: Plane-of-Flight projections of trajectories belonging to one cluster with respect to v_3.

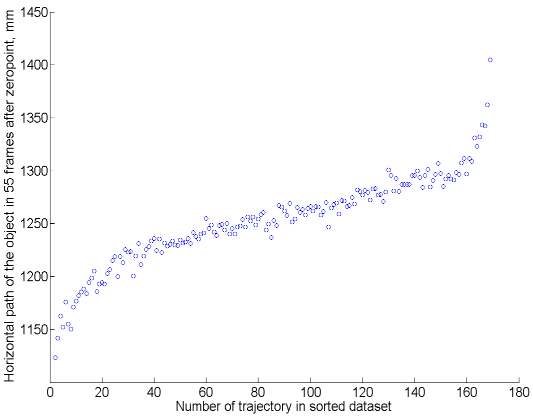


Figure 4.13: Horizontal coordinate of the ball in 0.5 second after the throw for all trajectories depending on the sequential number in the sorted dataset. Picture taken from [Mir15]

put to the end. On the figure the value of the horizontal coordinate for the frame number 55 is plotted with respect to the sequential number of the trajectory in the sorted dataset. It may be seen that for neighboring trajectories the value of x_3 does not differ more than for 2 cm (at least for for trajectories with numbers from 20 to 160; trajectories in the beginning and in the end of the list differ more due to fringe effect, these are trajectories captured with exceptional values of throwing force, they are used only for learning).

The operation of sorting the dataset with respect to v_3 described in the previous paragraph is applied to the trajectory database on the learning stage. When the algorithm predict the current trajectory it works with the database, which is already sorted. It estimate the value of v_3 and compare it only with trajectories from the dataset which have similar value of v_3. The structure of learning and prediction algorithms is described in section 4.5.

As it was mentioned above the proportion d_3 is not kept stable in time for the datasets with similar horizontal velocity. However due to similar values of $x_3(t)$ for all dataset members the weighted nearest neighbors principle here could be replaced by the simple nearest neighbor principle:

$$\hat{x_{C3}}(t) = \frac{x_{A3}(t) + x_{B3}(t)}{2}. \tag{4.56}$$

If we assume that the values of $x_{C3}(t)$,$x_{A3}(t)$ and $x_{B3}(t)$ are nearly the same at any moment and the distance between corresponding points on the trajectories is minimal then this operation allow accurate forecast of future coordinates in x_3 dimensions. Experiments described in section 5.1 showed that simple nearest neighbors prediction is more accurate than weighted nearest neighbors in general.

Comparing the current trajectory with a small subset of the entire dataset reduce the time of calculations T. If KSO process the entire dataset T in simplified way may be defined as

$$T = M * \tau_d + \tau_{KFO}, \tag{4.57}$$

where M is overall number of trajectories in the dataset, τ_d is a time of calculating the distance between two trajectories and its comparison with the current smallest distance (i.e. $n * \tau_d$ is time of applying KSO), τ_{KFO} is a time of applying KFO.

With use of subset allocation it become equal to

$$T = \tau_{all} + m * \tau_d + \tau_{KFO}, \tag{4.58}$$

where τ_{all} is the time of subset allocation and m is number of examples in the subset. Obviously subset allocation is useful if it significantly decrease the time expenses:

$$\tau_{all} + m * \tau_d << M * \tau_d, \tag{4.59}$$

$$\tau_{all} << (M - m) * \tau_d. \tag{4.60}$$

The number of trajectories in the cluster affect on the accuracy and speed of prediction. In section 5.2 it is shown that the best results were achieved when the cluster size was equal to approximately 1/8 of the whole dataset.

4.5 Predictor summary

This section summarize the algorithm development made in sections 4.2, 4.3 and 4.4. In fact predictor consist of two algorithms: learning algorithm and prediction algorithm. Prediction algorithm consist of operations, which are performed in order to forecast the current trajectory. It is called after receiving each new frame. Learning algorithm consist of operations, which are done with the trajectory dataset prior to starting the work of the system. It has an aim to reduce the computations which are done during the prediction. As the prediction is real-time all computations, which may be done prior starting the system are done prior starting the system. This section does not describe the process of extracting 3D coordinates of the object from the video stream. It is already considered in chapter 3. Here it is assumed that the predictor get the sequence of measured object coordinates in camera coordinate system from the beginning till the last available frame as an input, while the learning algorithm get a set of trajectories observed in the camera coordinate system.

Two main tasks are solved on the learning stage: transform all the sample trajectories to the universal zero-point coordinate system and sort them with respect to the estimated horizontal velocity. Sorting trajectories has an aim to speed-up the cluster allocation procedure on the prediction stage. The structure of the learning algorithm is shown on figure 4.14.

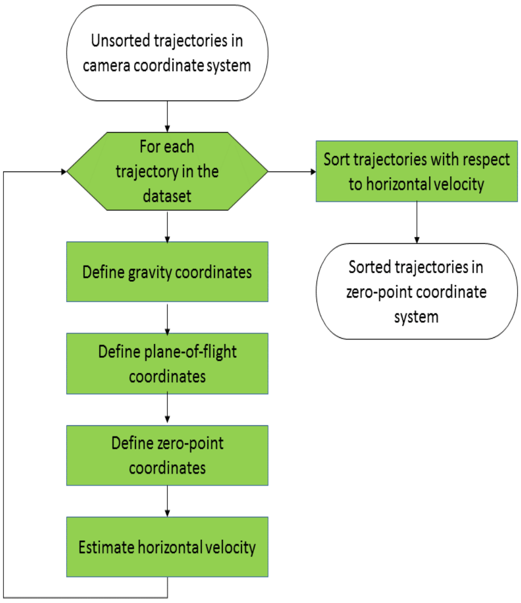

Unsorted trajectories in camera coordinate system
For each trajectory in the dataset
Define gravity coordinates
Define plane-of-flight coordinates
Define zero-point coordinates
Estimate horizontal velocity
Sort trajectories with respect to horizontal velocity
Sorted trajectories in zero-point coordinate system

Figure 4.14: The structure of the learning algorithm.

First of all for each trajectory in the set the reference of coordinate transformations is executed: each trajectory is sequentially translated to gravity coordinate system (section 4.2.2), plane-of-flight coordinate system (section 4.2.3) and zero-point coordinate system (section 4.2.4). When the trajectory is already presented in zero-point system the value of the mean horizontal velocity is estimated (section 4.4). After that all trajectories in the dataset are sorted with respect to mean horizontal velocity. In the final database one trajectory is indicated by i, v_i, X_i where i is a sequential number of the trajectory in the set, v_i is the value of the estimated mean horizontal velocity, and X_i is the matrix showing the dynamics of object coordinates in time.

The structure of the predictor is shown in the figure 4.15. For better visualization the functional blocks are put into fields corresponding to the coordinate system where they are executed. First of all the reference of coordinate transformations is performed for the measured part of the trajectory. Then the mean horizontal velocity is estimated. It may be seen that the same operations are done with each sample trajectory on the learning stage. After that the cluster of similar trajectories is allocated from the dataset. For this purpose the sample trajectory X_j is found, which have the value of the estimated mean horizontal velocity, which is the most similar with the corresponding parameter of the current trajectory. Then the range of sample trajectories around X_j with sequential numbers from $j - q$ to $j + q$ are put into the cluster of similar trajectories. q is a natural number, choice of the value of q is discussed in section 5.2. After cluster allocation the operations of search for nearest neighbors within the cluster (section 4.3.2) and forecasting (section 4.3.1) are performed. The second operation return the predicted

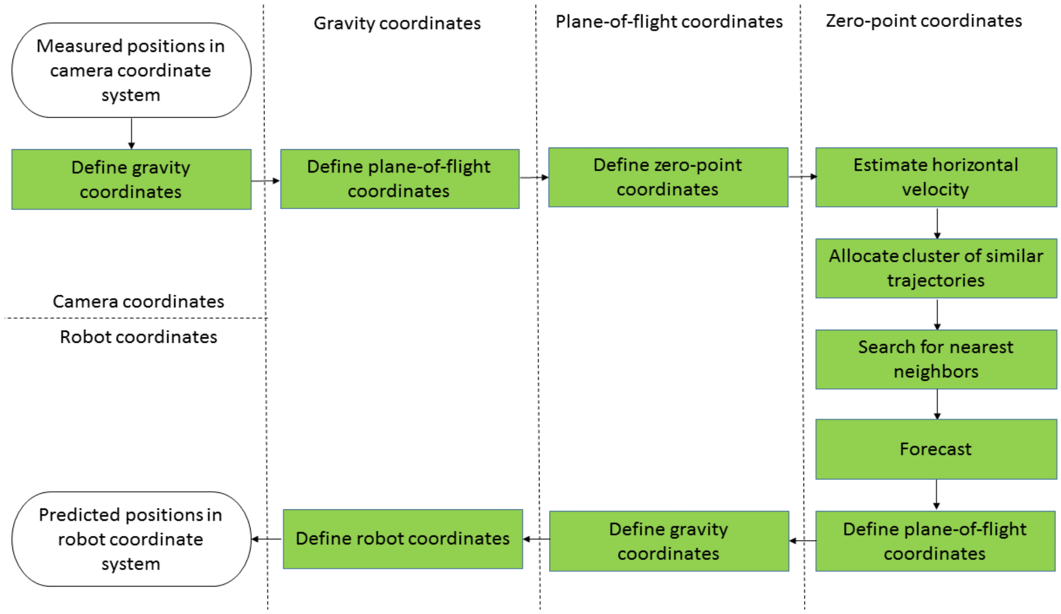

Gravity coordinates
Plane-of-flight coordinates
Zero-point coordinates
Measured positions in camera coordinate system
Define gravity coordinates
Define plane-of-flight coordinates
Define zero-point coordinates
Estimate horizontal velocity
Allocate cluster of similar trajectories
Search for nearest neighbors
Forecast
Camera coordinates
Robot coordinates
Predicted positions in robot coordinate system
Define robot coordinates
Define gravity coordinates
Define plane-of-flight coordinates

Figure 4.15: The structure of the prediction algorithm.

reference of object positions in zero-point coordinate system. This reference is transformed back to gravity coordinate system and then to robot coordinate system (section 4.2.5). Predicted object coordinates in the robot coordinate system are given to the output of the predictor. This data is then used for determining the catching movement.

5 Implementation and experiments

In the chapter 4 the concept of the prediction algorithm was proposed and developed. This chapter tell how this algorithm is applied to the real system for objects' transportation by throwing and catching. Two questions are mainly under the scope: how the algorithm is implemented and how good does it work?

Section 5.1 explain the numerical experiments with sample trajectories stored in the dataset. These trajectories was observed by the vision system in throwing experiments as it is explained in section 3.1. The aim of the numerical experiment is to validate the accuracy of the algorithm and to find the best settings. Section 5.2 describe technical aspects of integrating the algorithm into the transportation system. The version with the parameters determined in section 5.1 is implemented on C++ and integrated into software complex for ball tracking. Final examination of the algorithm is described in section 5.3. Experiments showed that the system with integrated predictor is able to catch thrown balls successfully.

5.1 Numerical experiments with the dataset

The experiments described in these section mainly consist in predicting the object's positions for the trajectories, which were observed in catching experiments (section 3.1). Comparison of the prediction results with the actual positions of the ball observed by the vision system is used to estimate accuracy of prediction. This an advantage of the numerical experiments in comparison with catching experiments. In case of catching experiments accurate tracking is not possible anymore, when the ball is inside the robot's workspace: the robot is moving, which distort the accuracy of stereo positioning. So the only possible output from the catching experiments is that the robot has caught the ball or did not. Numerical experiments allow prediction accuracy to be estimated instead.

k Nearest Neighbors algorithm use the set of previously observed trajectories as a base for prediction. Two type of trajectory datasets were used in the experiments:

- *Artificial dataset*: The trajectories in such a set are created via simulating ballistic motion of the tennis ball. As the tennis ball is a well-studied aerodynamic object, such a simulation is relatively accurate (see section 4.1). Artificial generation of the set allow minimizing time expenses, which are needed to perform catching experiments for creating large-size dataset of real throws. Although the dataset is artificially generated in the experiments it was used to predict the real trajectories observed in throwing experiments.

- *Acquired dataset*: A set of real trajectories acquired by the stereo camera system. The trajectories to predict are also taken from this set. Leave-one-out method is applied: each trajectory from the set is predicted using all other trajectories except itself. In other words the trajectory, which is predicted currently, is removed from the dataset before the prediction and returned there afterwards. I.e. an incorrect situation when one trajectory is used to predict itself is avoided.

The question to discuss is what number of available frames is used for prediction. According to the algorithm settings discussed in sections 4.2.3 and 4.3.1 prediction is done the first time after receiving the frame number 40. In theory accuracy of prediction should improve with increasing number of available frames. Actual influence of the frame number on the prediction accuracy is discussed in the end of this section. The algorithm described in chapter 4 has a number of parameters. Choice of their value effect on the quality of prediction. The main varying settings of the algorithm are listed below:

1. *The method of estimating the plane-of-flight*: Four main methods were proposed in section 4.2.3 (least squares, robust least squares, random sample consensus, sample mean, sample median). Sample median was finally chosen as the most robust, accurate and fast method. Although the choice of the method is done based on numerical experiments it is described in section 4.2.3 instead of this section because of strong relation to the algorithm development.

2. *The parameter for sorting trajectories in the dataset*: Four main parameters were proposed in section 4.4 (velocity scalar v, elevation of throw ϕ, vertical velocity v_1, horizontal velocity v_3). Horizontal velocity was finally chosen as a clustering. The choice of the clustering parameter is described in section 4.4 because of strong relation to the algorithm development.

3. *Weighting coefficients*: these are the numbers w_1 and w_2 with possible values from 0 to 1 from equation 2.22. Two ways of defining these coefficients are proposed in section 4.3: setting them both to 0.5 (simple nearest neighbors, equation 4.50) or defining the values with respect to distance proportion (weighted nearest neighbors, equation 4.49).

4. *Size of the cluster c*: On one hand reducing the size of the cluster means decreasing the volume of computations (less trajectories to compare with the current one). On the other hand it may decrease the accuracy: if the size of the cluster is small the nearest trajectories may be thrown out of the cluster. In this case they are not recognized as nearest neighbors.

5. *Number of taken nearest neighbors k*: In sections 4.3 and 4.4 it was assumed that $k = 2$, one of taken nearest neighbors lies above the current trajectory, and another lies below. Assumption that this choice of k is the most accurate need a proof.

5.1.1 Simple and weighted nearest neighbours

On the first step of numerical experiments the accuracy for the basic proposed setup was examined. Prediction is made based on first 40 frames after the throw. Cluster size is set to 25 trajectories and k was set to 2 with one neighbor above and one neighbor below the trajectory. Versions with simple and weighted nearest neighbors forecasting were validated. Prediction is done based on the same dataset using leave-one-out method. Example of the successful prediction results are plotted in figure 5.1. 150 trajectories were predicted in such a way. The results are given in table 5.1.

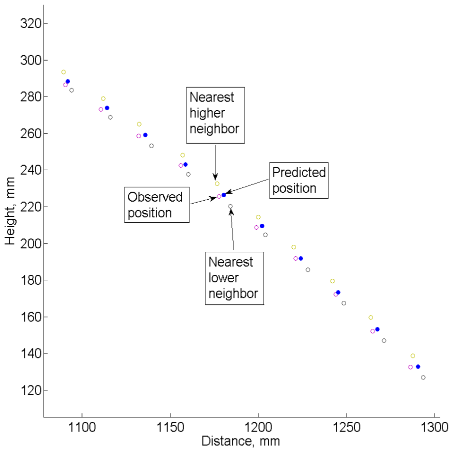


Figure 5.1: Reference positions of the ball from the current trajectory, its lower neighbor, and its higher neighbor; filled circles show the predicted positions of the object.

Table 5.1: Accuracy of the prediction

Parameter	simple kNN	weighted kNN
% of throws with error $< 30mm$	92	85
% of throws with error $< 20mm$	85	73
Median error in mm	10	13

The percentage of trajectories with prediction errors less than the specified thresholds is given in the table. The thresholds of 30 and 20 mm are chosen according to [Bir11] where they are stated as allowing successful catch of the ball with robotic hand. Unexpectedly simple nearest neighbor algorithm showed better accuracy than weighted nearest neighbors. The possible reason is high sensitivity of the proportions to measurement errors. In further experiments simple nearest neighbors prediction was used.

5.1.2 Size of the cluster

Evaluation of the cluster size was done both with artificial dataset and acquired dataset. The experiments was set with the dataset from August 2016 consisting of 100 sample trajectories and 20 test trajectories (see section 3.1). The radius of the cluster r is equal to $0.5 * (s - 1)$ where s is number of trajectories in the cluster. If the trajectory with the most similar value of v_3 to the current one has sequential number i in the sorted dataset the cluster include trajectories with sequential numbers from $i - r$ to $i + r$. Other settings were the same as in subsection 5.1.1. The median error for various cluster sizes is presented in table 5.2. ∞ here means that no sorting and clustering was applied at all: search for nearest neighbors was made through the entire dataset.

Table 5.2: Median errors of prediction for various sizes of the cluster. ∞ means that no sorting and cluster allocation was made at all. Prediction was done based on acquired dataset of real trajectories.

r	e_{med}, mm
1	25,3
2	14,2
3	13,3
4	12,5
5	11,9
6	10,6
7	10,6
8	7,8
9	7,8
10	7,8
11	6,6
12	6,6
∞	16,7

Results show that the error decrease with the growth of the cluster size. This improvement stops after the $r = 11$ for $r = 12, 13, 14, ...$ the median error keeps the value of 6.6 mm. This means that even with larger size of the cluster the search return nearest neighbors inside the cluster with $r = 11$. When the cluster size become near to the size of the entire dataset size of error increase. This result show that use of sorting improve not only the speed of computation but also prediction accuracy.

Median prediction error might be not the best parameter for evaluating the quality of prediction on relatively tiny datasets. It is more important to know what percentage of trajectories was predicted accurately than what was the prediction error for "mean trajectory". In figure 5.2 it is shown, what number of trajectories was predicted in the most accurate way for various cluster sizes. Prediction in the most accurate way means that for this trajectories there is no cluster size providing better accuracy than the current cluster size. This histogram show the same results as table 5.2: the most effective work of the algorithm is reached at $r = 11$. It is nearly the 1/8 of the whole dataset size (as it includes 100 trajectories).

The results of the similar evaluation with artificially-generated dataset consisting of 2048 trajectories are presented in table 5.3. First column present accuracy information, while the second one present the time expenses. The results show that minimum median error is achieved already when the size of the cluster is equal to 256 trajectories. Time analysis is made for the real-time implementation of the algorithm (this implementation is discussed in the next section). The influence of the dataset size on the speed of processing was not significant for the size of the database from 64 to 2048 trajectories (it was everytime less than 1 ms for cluster with 25 trajectories). However the size of the cluster make a significant influence on the speed of calculations (see table 5.3). It grows with the growth of the cluster size. The framerate of the observer is 100 fps (subsection 3.1.2), so interframe period is equal to 10 ms. If the time of computation is less than 10 ms, it is possible to recompute the forecast after each new frame received. For 256 trajectories in the

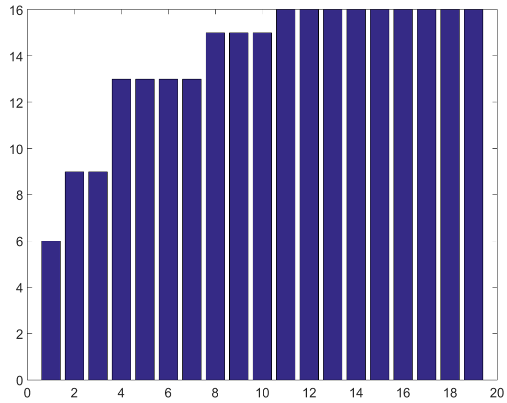


Figure 5.2: Correspondence between number of trajectories, which are predicted with the best accuracy and cluster size. (Draft!)

cluster the upper bound time of computation is equal to 4.3 ms, which is less than 10 ms and therefore sufficient.

Table 5.3: Influence of the cluster size on the accuracy and speed of prediction for the artificially generated database consisting of 2048 samples.

cluster size	median prediction error, mm	time of computation (upper bound for 99,97%), ms
25	14.2	1.4
64	12.5	2.1
128	11.9	2.7
256	10.6	4.3
512	10.6	7.6
1024	10.6	15.2

So in both setups good accuracy with satisfying performance are achieved when the cluster is nearly 8 times smaller than the entire dataset. This is not a universal result. When the trajectories in the dataset are more dispersed (e.g. launching velocity varies from 2 to 7 m/s, but not from 4 to 5 m/s) relative size of the cluster should be smaller. But for the current setups cluster sizes were determined in such a way: 25 trajectories in the set for acquired database and 256 trajectories in the set for artificial database.

5.1.3 Value of k

In the basic setup the value of k was set to 2. This subsection aim to check whether this choice provide the best accuracy or not. Another question is whether picking one higher and one lower nearest trajectory provide better accuracy than just picking two nearest neighbors or not. For this purpose the accuracy of the prediction was checked for various values of k. When k was set to 1 the forecast of the trajectory was just set equal to the found nearest neighbor. When it was set to $2, 3, \ldots$ the forecast is calculated as a mean of respective number of nearest neighbors. The median prediction errors for these versions of the algorithm with various k are listed in table 5.4. Here $k = 2$ mean that simple two nearest neighbors were taken, while $k = 1 + 1$ mean that one higher nearest neighbor and one lower nearest neighbor were taken. Other algorithm settings were the same as in section 5.1.1.

Table 5.4: Median errors of prediction for various numbers of taken neighbors; 1+1 mean that one higher nearest neighbor and one lower nearest neighbor were taken.

k	Median error, mm
1	15,7
2	10,1
1+1	6,6
3	10,6
4	12,9
5	13,0
6	13,8
7	15,6

Results showed that the proposed rule of taking neighbors (2 nearest neighbor, one of which is higher and another one is lower than the currents trajectory) provide the most accurate prediction. Particularly it provides more accurate results then than taking two nearest neighbors without any additional rules. Using one nearest neighbor prediction decrease the accuracy as well as increasing the number of neighbors more than 2.

A set of numerical experiments showed the following algorithm setting that provide the best results in terms of accuracy and sufficient results in terms of performance. The trajectory forecast is calculated as a mean of respective values of higher nearest trajectory from the dataset and lower nearest trajectory from the dataset. The search for these nearest trajectories is made through the cluster with 25 (if acquired dataset with 100 trajectories is used) or 256 (if artificial dataset with 2048 trajectories is used) trajectories. The values for cluster size are not universal: they are good for these specific experimental setup. The implementation of the algorithm was made based on these settings described above.

5.2 Implementation

The algorithm of trajectory prediction is implemented to be used in the automated transport-by-throwing system. This section discuss the details of this implementation. Subsection 5.2.1 discuss how the algorithm is integrated into the experimental hardware and software complex for automated throwing and catching. Subsection 5.2.2 is concentrated on how the low-level implementation of the prediction algorithm look like.

5.2.1 Integration into the transportation system

The final examination of the algorithm is done via the catching experiments. These experiments are performed on the specific experimental software and hardware complex for automated throwing and catching at Technische Universitaet Wien [Pon16]. This system includes:

1. Numerically controlled throwing device (see section 3.1.1),
2. Robotic manipulator KUKA LWR4+ with 7 degrees of freedom, which is used to catch balls [20],
3. Stereo cameras (see subsection 3.1.2),
4. Information processing subsystem.

The structural scheme of the system components is shown in figure 5.3. Communication between various system components is organized as follows [Pon16]. Data processing in executed on two personal computers (PC-1 and PC-2). PC-1 is intended for processing the data from cameras and for prediction, while PC-2 is used for determining the instructions for the gripper. The throwing device is switched from PC-1. When the ball intersect the light-barrier, the notification about this event come to the synchronization generator, which synchronize the work of the entire system based on the robot's cycle time. When the new pair of frames come to PC-1 from the camera, the new position of the ball is extracted from the images, the forecast is updated and the new catching point in space and time is chosen. This information is then sent to PC-2, where the instructions for the robot are updated.

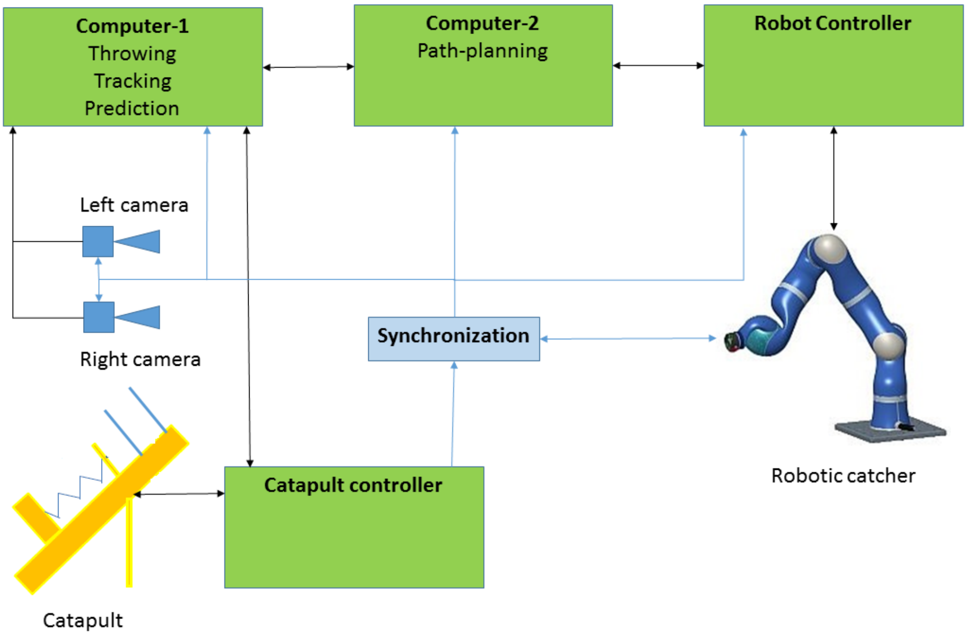


Figure 5.3: Structure of the transport-by-throwing system.

The structural scheme of the entire data processing algorithm is shown in figure 5.4. It get a pair of images from the cameras as an input. As an output it must return the coordinates of the point, where the catch will take place and the time of this event. This data is then transmitted to PC-2. First the pixel coordinates of the ball center are extracted from both images using RANSAC circle recognition (see subsection 3.1.2). This step is computationally expensive, so it is executed in parallel on the graphic processor unit [Goe15]. All other steps are executed sequentially on the central processing unit. 3D coordinates of the ball center are extracted from the pixel coordinates using the operation of stereo triangulation (see subsection 3.1.3). After that the new prediction is made. The result of the prediction is a reference of future positions of the ball's center. One of this positions is picked as a catching position [Pon16] and then transmitted to PC-2.

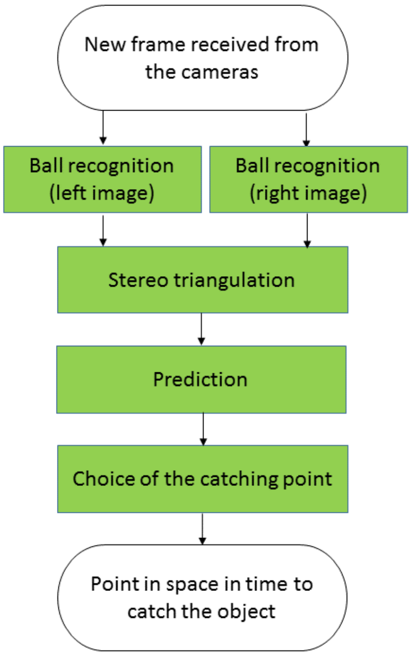

New frame received from the cameras
Ball recognition (left image)
Ball recognition (right image)
Stereo triangulation
Prediction
Choice of the catching point
Point in space in time to catch the object

Figure 5.4: Structure of the software.

This algorithm is implemented as an executable code on C++. The parallelized ball center recognition is implemeted using CUDA library. The predictor is implemented as a C++ function. The details of this implementation are discussed in the next subsection.

5.2.2 Real-time predictor

As it was described in section 4.5 predictor include to different algorithms: prediction algorithm and learning algorithm. Learning algorithm returns the sorted dataset of sample trajectories, which is then used by the prediction algorithm. As the learning algorithm is executed prior to starting the transportation system there is no need to make it real-time. So straightforward MATLAB implementation of the learning algorithm according to section 4.5 was used for creating the sorted dataset.

Prediction algorithm instead of learning algorithm must be real-time and interact with other software components. Therefore it was implemented as a C++ function within the common software project for data processing. Mainly this is a straight implementation of the computational operations described in sections 4.2, 4.3, and 4.4. Three specific issues of the predictor's C++ implementation should be mentioned:

1. *Management of the multiple calls*: The predictor is called every time, when the new frame is received from the vision system, however the first prediction could be made only after receiving the frame number 40 (see subsection 4.2.3). Before it the predictor should only store the received coordinates into the static variables. After the frame number 40 the predictor recalculate the forecast after each new frame.

2. *Management of the dataset*: The dataset of sample trajectories is memory expensive (e.g. dataset with 2048 trajectories is a 2048-by-100-by-3 array with volume of more than 1 megabyte) and the way of initializing it may affect on the time of computation. The fastest way of initializing such an array is inserting it directly into the program code. Therefore the MATLAB array returned by the learning algorithm is automatically transformed into a line of C++ code, which is then inserted into predictor function.

3. *The need of fast performing of two matrix operations*: multiplying 4-by-4 matrix on 4-by-1 vector and defining the value of 4-by-1 vector by known result of multiplication of known 4-by-4 matrix on this vector. These operations are used many time on the reference of coordinate transformations (see section 4.2). Implementation of these operations is discussed below.

In MATLAB environment matrix operations are inserted into the language functionality. In C++ special libraries exist for matrix operations, however the use of these library lead to higher time expenses. Therefore matrix multiplication and the opposite operation were implemented on low level. According to the common rules of matrix multiplication if there is a need to multiply 4-by-4 matrix $B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}$ and 4-by-1 matrix $Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$ the resulting matrix Z will be 4-by-1 $Z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix}$ and the values in four rows of this matrix may be found using the following formula:

$$\begin{aligned}
z_1 &= b_{11} * y_1 + b_{12} * y_2 + b_{13} * y_3 + b_{14} * y_4, \\
z_2 &= b_{21} * y_1 + b_{22} * y_2 + b_{23} * y_3 + b_{24} * y_4, \\
z_3 &= b_{31} * y_1 + b_{32} * y_2 + b_{33} * y_3 + b_{34} * y_4, \\
z_4 &= b_{41} * y_1 + b_{42} * y_2 + b_{43} * y_3 + b_{44} * y_4.
\end{aligned} \tag{5.1}$$

Reverse transformation (defining unknown Y based on known Z) is made by solving the system of linear equations expressed by matrix equation $B * Y = Z$. There is a number of mathematical approaches to solving such a system, here Cramer's rule is chosen due to computational simplicity. In Cramer's method the value of Y is found using the following sequence of computations:

$$\begin{aligned}
\Delta &= \det(B) \\
\Delta_1 &= \det\begin{pmatrix} z_1 & b_{12} & b_{13} & b_{14} \\ z_2 & b_{22} & b_{23} & b_{24} \\ z_3 & b_{32} & b_{33} & b_{34} \\ z_4 & b_{42} & b_{43} & b_{44} \end{pmatrix}, \\
\Delta_2 &= \det\begin{pmatrix} b_{11} & z_1 & b_{13} & b_{14} \\ b_{21} & z_2 & b_{23} & b_{24} \\ b_{31} & z_3 & b_{33} & b_{34} \\ b_{41} & z_4 & b_{43} & b_{44} \end{pmatrix}, \\
\Delta_3 &= \det\begin{pmatrix} b_{11} & b_{12} & z_1 & b_{14} \\ b_{21} & b_{22} & z_2 & b_{24} \\ b_{31} & b_{32} & z_3 & b_{34} \\ b_{41} & b_{42} & z_4 & b_{44} \end{pmatrix}, \\
\Delta_4 &= \det\begin{pmatrix} b_{11} & b_{12} & b_{13} & z_1 \\ b_{21} & b_{22} & b_{23} & z_2 \\ b_{31} & b_{32} & b_{33} & z_3 \\ b_{41} & b_{42} & b_{43} & z_4 \end{pmatrix}.
\end{aligned} \tag{5.2}$$

$$\begin{aligned}
y_1 &= \frac{\Delta_1}{\Delta}, \\
y_2 &= \frac{\Delta_2}{\Delta}, \\
y_3 &= \frac{\Delta_3}{\Delta}, \\
y_4 &= \frac{\Delta_4}{\Delta}.
\end{aligned} \tag{5.3}$$

The performance characteristics for the implemented prediction algorithm are discussed in subsection 5.1.2. The upper bound for the time of execution in 99,7% of runs is equal to 4.3 ms, which is more than 2 times less than the interframe period. It mean that it is possible to recalculate the forecast after each new received frame. When the catching experiments are performed (see the next section), the prediction is everytime done within the interframe period.

5.3 Catching experiments

A set of catching experiments was made to examine the ability of the algorithm to give useful results within the real transport-by-throwing system. The experiments were performed on the experimental transportation system discussed in the previous section. As it was mentioned KUKA LWR 4+ robotic arm is used as a catching device. The gripper is mounted on the arm. It use a butterfly-net principle. The appearance of this end-effector is shown in figure 5.5. The net is mounted on a metal ring with the diameter of 11 cm. As the diameter of the ball is equal to 7 cm such a ring allow prediction error up to 2 cm.

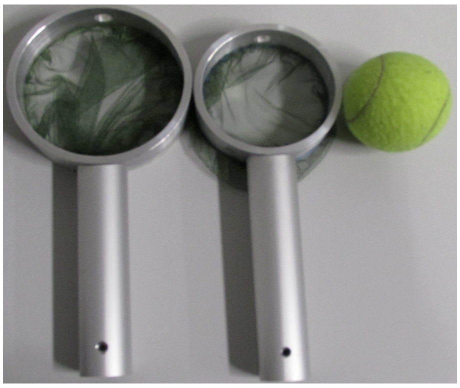


Figure 5.5: Grippers used for ball catching.

A number of throws were made in order to catch the ball. The nominal throwing velocity was set to 5 m/s and the third dataset from September 2015 (see subsection 3.1.4; it also include the ball thrown with nominal velocity of 5 m/s) was used as a basis for prediction. Due to high dispersion of the actual throwing velocities from the nominal values the trajectories often varied from typical appearance and the balls were not caught succesfully. The list of throws sorted with respect to throwing velocities is given in table 5.5. The reference of the object's positions for example successful catch is shown in figure 5.6.

Table 5.5: Results of catching experiments for 20 trials for various velocities between 4.5 and 5.5 m/s.

number	v, m/s	result
1	4.54	lost
2	4.58	lost
3	4.67	lost (rebound from the gantry)
4	4.68	lost (rebound from the gantry)
5	4.68	lost (rebound from the gantry)
6	4.69	lost (rebound from the gantry)
7	4.96	caught
8	4.98	caught
9	5.00	caught
11	5.02	caught
12	5.04	caught
13	5.07	caught
14	5.13	caught
15	5.24	lost (rebound from the net)
16	5.25	lost (rebound from the gantry)
17	5.28	lost
18	5.36	lost
19	5.46	lost
20	5.50	lost

According to the table all balls thrown with velocity near to nominal (from 4.8 to 5.2 m/s) were

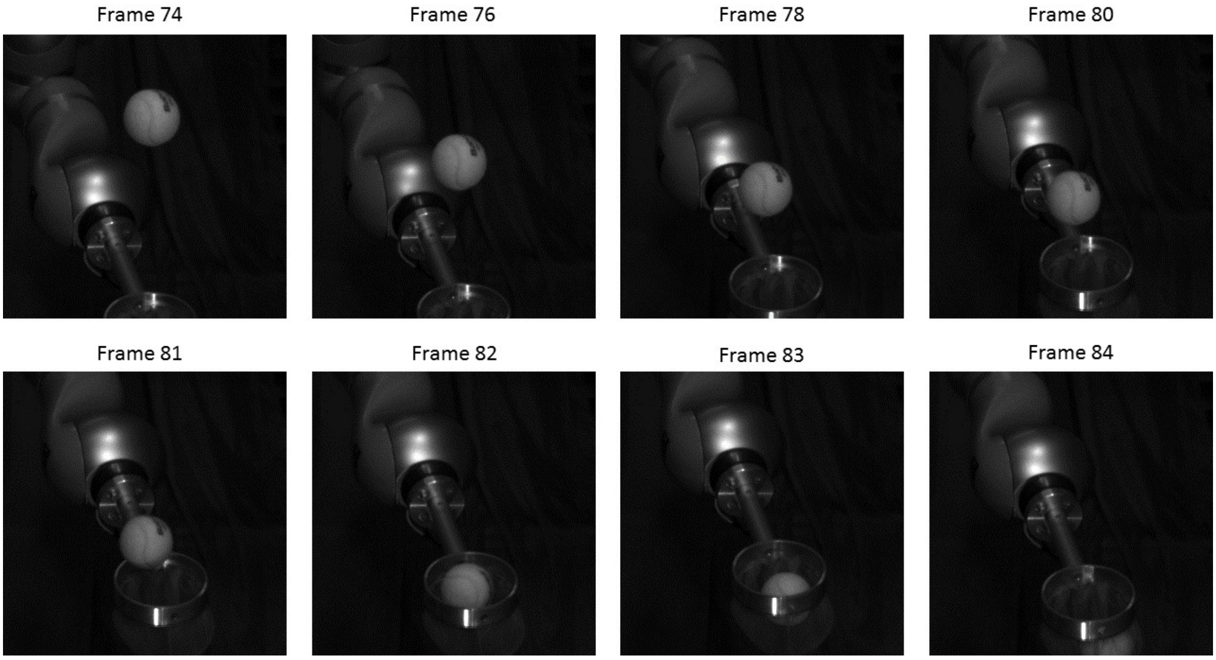


Figure 5.6: A sequence of the camera frames representing the successful catch of the ball.

succesfully caught. Higher variation of the velocity lead to losses of the ball by the mechanism. Note that the dispersion of the throwing velocities in catching experiments was higher than in the learning dataset. Therefore losses of the ball on the extreme velocities are expected. Use of more dispersed datasets should decrease the influence of this effect. The successful catches on the near-to-nominal velocities prove that the algorithm may be successfully applied in transport-by-throwing.

6 Conclusion and Future Work

The research made in the thesis was initially aiming to develop a novel approach for predicting the trajectory of the thrown object. In previous chapters 4 and 5 development and implementation of the new trajectory predictor was described in details. This concluding chapter finalize the research made in this thesis. Section 6.1 summarize main results from the chapters 3, 4 and 5. It tells briefly what is done. Section 6.2 tells what is still to do. It discuss the open issues, which still remain after the research made, and propose the ways of future development.

6.1 Results of research

A sample-based algorithm for predicting the trajectory of a thrown body is proposed, developed and evaluated within this thesis. It allows predicting the future coordinates of the flying body in real time based on measured coordinates at the beginning stage of flight. This algorithm is applied on the robotic system for catching thrown objects. This system includes:

1. Numerically controlled throwing device. Discussion on this device is made in section 3.1.1. It throws the object with specified velocity, however real velocity of throw has random deviations from the nominal setting. The throwing device is supplied by two light barriers, which measure time and actual velocity of throw.
2. Robotic manipulator KUKA LWR4+ with 7 degrees of freedom, which is used to catch balls.
3. Camera subsystem. Real-time tracking of the flying spherical object is done using the stereo system consisting of two cameras. Both cameras have image sensors with 2048 by 2048 pixels. Reconstruction of the object's location in space is done via the stereo triangulation.
4. Information processing subsystem. It includes two personal computers (one is used for processing the data from the cameras and another one is used for defining the commands for the robot) and controller devices, which control the robot and the throwing device.
5. Hardware for synchronization and communication between the components of the system.

During the development of the algorithm the following intermediate tasks were solved:

1. *An analysis of observer's performance and accuracy*: This task is discussed in chapter 3. A set of throwing experiments was made in order to explore the accuracy of tracking and to collect the datasets for learning trajectory prediction. Determining the ball pixel position on the images is done via RANSAC circle detection (section 3.1.2). Determining the ball's 3D position from its pixel positions is done via the stereo triangulation (section 3.1.3). Accuracy analysis showed that the observer is determining the 3D location of the static spherical objects with standard deviation of 2.25 mm (see section 3.2). The influence of the objects on the scene make accurate positioning of the flying ball on the long distances impossible (see subsection 3.3.1). Therefore background subtraction is applied to overcome this influence. With use of subtraction accuracy of positioning for flying objects become similar to the results for static objects. However the errors of the distance from the ball to the cameras are too big, when this distance is more than two meters (subsection 3.3.2). When the cameras are located behind the throwing device this errors are localized in one spatial dimension therefore such camera location is likely. Polynomial fitting in depth dimension is applied in order to eliminate errors on long distances (subsection 3.3.2).

2. *The development of an algorithm for trajectory prediction based on machine learning and the learning algorithm*: k Nearest Neighbours (kNN) are chosen as a basic technique for trajectory prediction (subsection 2.5.4). The prediction is made based on the sample trajectories stored in the dataset. The kNN forecasting is done in two steps: on the first step the search through the dataset is made. This search aims tof found the trajectories in the set that are the most similar (*nearest*) to the current trajectory. The development of the search operation is discussed in subsection 4.3.2. The distance between the corresponding points on the trajectories is used as a metric for determining the similarity of trajectories. As a results of the search two trajectories from the set are returned: nearest higher trajectory and nearest lower trajectory. The second step of the algorithm cinsist in forecasting the current trajectories based on known nearest neighbors (subsection 4.3.1). The future part of the current trajectory is determied a weighted mean of the corresponding parts of the nearest neighbors. The weights of the neighboring trajectories may be defined in two ways. First, in *simple* nearest neighbors the weights of both neighbors are set to 0.5. Secondly, in *weighted* nearest neighbors the weights are defined with respect to the mean distances between the corresponding points on the current trajectory and on its neighbors. Further exploration showed that simple nearest neigbours provide higher accuracy than weighted nearest neighbors.

   Two improvements of the nearest neighbor predictor are made. First, the reference of the coordinate transformations is inserted into the algorithm. The stereo triangulation return the 3D coordinates of the object in the coordinate system connected with the optical center of the left cameras. In section 4.2 it is proposed to translate these coordinates into the new coordinate system (*zero-point coordinate system*), which is defined in the following way. The origin of the coordinate system is put to the point on the trajectory, which is measured after the throw. One coordinate axis is set collinear with the gravity direction, while another one is set collinear with the horizontal projection of the launching velocity. The sample trajectories are translated into this coordinate system and stored in the database in zero-point coordinates. When the prediction of the current trajectory is done, it is first translated to zero-point coordinate system, then predicted, and finally the result is translated back to the world coordinate system. Use of zero-point coordinate system make the algorithm invariant to the spatial position of the launching point, to the horizontal direction of throw and to the location of the cameras. It also simplifies the process of detecting erroneous position

measurements.
The second improvement consist in comparing trajectory with relatively small subset of the large entire dataset (see section 4.4). For this purpose mean horizontal velocity of flight is estimated for each trajectory from the set. After that all trajectories in the set are sorted with respect to this parameter. When the current trajectory is predicted it is compared only with those sample trajectories, which have similar value of the horizontal velocity. This lead to significant decrease of the volume of computations. Also it provide fast search for the subset of trajectories similar to the current one.
Predictor software include two main modules: prediction algorithm and learning algorithm (see section 4.5). Learning algorithm consist of several operations made with the dataset prior to the prediction. These operations are: translation of the sample trajectories into the zero-point coordinate system, estimating mean horizontal velocity for each sample trajectory, and sorting the trajectories in the set with respect to mean horizontal velocity. Prediction algorithm include a set of operations, which are made while forecasting the current trajectory. These operations are: translation of the current trajectory into the zero-point coordinate system, estimating mean horizontal velocity for the current trajectory, allocating the subset of similar trajectories from the entire dataset, search through this subset for two nearest neighbors, forecasting the current trajectory based on found nearest neighbors, and translation of the forecast to robot coordinate system.

3. *The evaluation of the accuracy of the constructed model*: It is made by applying the algorithm to the trajectories observed by the cameras in the throwing experiments. These numerical experiments are described in section 5.1. They are used to determine the algorithm settings, which provide the best accuracy and sufficient performance. The following output is got from this experiments. Simple nearest neighbors show better accuracy than weighted nearest neighbors (subsection 5.1.1). For the current experimental setup the best accuracy is provided when the subset of similar trajectories is approximately 1/8 of the entire dataset (subsection 5.1.2).

4. *The integration of the proposed prediction module into the existing transport-by-throwing system*: The predictor is inserted into the software module for processing the data from the cameras (subsection 5.2.1). The results of prediction are then given to the software module which determines the catching movement of the gripper. Predictor is implemented as a C++ function which is called by the data processing software (subsection 5.2.2). The learning algorithm is implemented as a MATLAB program: it generate the sorted dataset, which is then used by the predictor. The tests of the implemented predictor showed that it is able to perform all the prediction within the time-out period between receiving two frames from the cameras.

5. *The final evaluation of the whole transport-by-throwing route with the integrated learning-based prediction algorithm*: On the final step of the experiments the transportation system was able to catch flying balls based on the prediction results provided by the algorithm (section 5.3). This shows that the algorithm is in principle applicable for the robotic catching.

The result of the entire doctoral research is a novel algorithm for predicting the trajectory of a thrown spherical body. The main differences of this algorithm from the state-of-the-art predictors are the following:

1. No accurate physical model of the flight is needed. The predictor works only based on previous examples.

2. The method does not require massive parallel processing, instantaneous feedback and use of special hardware. All the calculations are made on the standard processor unit in a sufficient time.
3. The method allow to compare current trajectory with high number of trajectories stored in the database and all these trajectories are taken in mind without high computational expenses.
4. The predictor is independent from the position of the launching point, from the horizontal direction of throw and from the location of the observer (see section 4.2).

6.2 Future work and outlook

The main outcome from the thesis consist in proved theoretical applicability of the k nearest neighbors to the trajectory prediction task for the system of material transportation by throwing and catching. This result is to be extended in future by developing the new versions of the algorithm, which may be applied in practical systems for material transportation. The catching experiments from the section 5.3 are made for the relatively specific setup. Future development of the predictor requires coming out of these constraints. The significant constraints of this setup are the following:

- *Relatively small variation of the throwing parameters*: the training set included the throws with launching velocities from 3.7 to 5.2 m/s, mainly 4 to 5 m/s (see section 3.1). This means that the algorithm is able to predict the throws accurately within the specific range of launching velocities. Throwing experiments from section 5.3 showed that the gripper miss the balls thrown with velocities less than 4.7 m/s and more than 5.2 m/s. Extending the work of the algorithm on wider ranges of velocities require collecting larger and more dispersed datasets of trajectories. Exploration of the algorithm accuracy and performance with use of higher ranges of trajectories is a significant direction of future work.
- *Linear throwing of objects*: Other ways of throwing (e.g. throws by a fast-rotating lever) were out of consideration. The linear throwing devices are good as they provide good dynamic stability of flight and allow minimizing spin (e.g. the lever thrower used in experiments in [Pon09] could not throw objects without a spin). However lever-throwing may be performed by the robotic arms of the same type, as used for catching. These transportation systems would be more flexible and universal then systems using the specific throwing devices. The predictability of the trajectories caused by lever-based throws require additional exploration.
- *Unified spherical objects are thrown*: As it was stated in the end of chapter 1 tennis ball was chosen as an experimental object to throw and to catch. This was motivated by the good knowledge about its aerodynamic properties. However real objects, which may be transported by throwing and catching in the industrial environment, have more complicated shape. Predicting trajectories of such bodies creates a number of questions to discuss. First of all the number of parameters to predict increases. of the object. Position of the spherical bodies is determined by three parameters: coordinates of the center-of-mass in 3D space. The plane-of-flight representation of the trajectories reduce the number of dimensions to two. If the object has more complicated shape three more parameters appears, which

express object's orientation in space [Kim12]. These parameters also need and application of the algorithm for predicting them has to be developed. In state-of-the-art literature this task has been considered e.g. in [Fra12] and [Kim12] for other prediction algorithms. One more important aspect of applying the algorithm for prediction of asymmetric bodies is whether the plane-of-flight representation or not? I.e. does the trajectory of the flying object with the specific shape curve to the side or not? For some object trajectories do not curve (e.g. cylinders fron [Fra12]) for some they do. The differentiation of such objects has to be made in further exploration. Even if the object's trajectory curves it may be transformed to the direction-independent representation by aligning one of the coordinate axis to the horizontal direction of throw. The task of defining this direction for the curved trajectory is to be solved in this case.

- *Specific abilities of the catching device*: the gripper based on the butterfly-net principle is used in catching experiments (section 5.3). This shape is chosen as it set hard correspondence between the prediction error and the rate of success. The allowed error was chosen relatively small: 20 mm. There are a number of other gripper constructions, which provide higher tolerance to prediction errors [Nis97, Cig15] however catching with use of active grasping [Nam03a, Fra07, Bae11, Cig15] require development of more complicated catching strategies.
- *The object is thrown in such a way to make the catch easier*: In other words the thrower does want that the catcher catch the ball successfully. This concept is defined by the concept of material transportation by throwing. In the industrial environment the thrower is adjusted in such a way that provide the easiest catching. The strategies when the thrower does not want the ball to be caught could be useful for other applications of trajectory prediction, e.g. robotic table-tennis. These strategies would require
- *Use of the stereo vision system with two cameras*: Increasing the number of cameras may improve the accuracy of the vision system. For example putting the additional pair of caneras observing the gripping area may eliminate the effect of increasing errors in depth direction on long distances discussed in subsection 3.3.2.
- *Sorting trajectories with respect to horizontal velocity*: As it was stated in section 4.4 the search for the nearest neighbors is made within the subset of trajectories, which has the horizontal velocity similar to the current one. Two more parameters were proposed: velocity scalar and the elevation of throw. They were rejected as they are hard to estimate. But hard does not mean impossible. The ways of their estimation may be found in future work. Also some more ways of subset allocation may be found.
- *Using euclidean distance as metric of neareness*: The neighbors with the smallest euclidean distance between the corresponding points was chosen as the nearest (see subsection 4.3.2). Other metrics of defining the nearest neighbors may be found and considered in the future work.

Further development of the algorithm faces with a number of new issues and possible improvements. Some of them are mentioned in the list above. One more possible direction of the further research is estimating the quality of the sample trajectory in the dataset. Quality here means how accurate can we predict other trajectories with use of this trajectory. If the trajectory is measured with errors its use for prediction will be harmful. Better accuracy of prediction with use of artificial datasets (which are free of error) partially prove this statement. However the exploration of this influence is also a part of future work. The method should be developed, which allow detecting the bad trajectories and rejecting them from the dataset.

These were mainly the local details of further development of the proposed algorithm in order to improve its characteristcs. From the global point of view also the ways of further development exist. The algorithm should become more universal and less dependent from the specific setup. As it was already mentioned one of the way is use more dispersed datasets and considering the situation, when the throw is not precised in order to let the catcher grasp the object. In this case the concept may be applied not only in the specific setups. One more direction of development is making the agorithm able to predict the objects of various shapes after a single learning. Above a different situation was considered: in principle the objects may be various, but the single learning train the predictor to forecast only the objects with one specific shape. However humans are able to catch objects of various shapes without training its shape. When they see flying object they are estimating what will be the trajectory. The next steps that increase the universality is developing such skills in automated systems.

Implementing of the transport-by-throwing system in the real industrial environment face us with the number of open issues, technical challenges and tasks. The scientific and engineering aspects of Transport-by-Throwing networks are waiting for development. The process of prediction and catching should be integrated with the control of the object processing by machine tools into the common automated system. In this case it become a part of common automated manufacturing process. The way of such integration depends on the specificity of the production process on "what" and "how" the factory produce. Large scale transportation network gives the new questions and need an additional development of the method. The TbT network may consist of a number of routes where one catching device must catch objects form different sources and one throwing device can throw objects towards several destinations. One example of issue, appearing in such networks is high number of moving objects on the scene. This make tracking the flying object more difficult to implement. In this thesis the background subtraction is used to differentiate object's appearance from the scene. When the background is dynamic this method does not work. This is only example issue, which appears when adapting the system to the complex industrial environment.

As it was mentioned above robotic catching was not appeared as a part of TbT, but TbT use it as a part of object transportation. Initially robotic catching is considered as a step of improving the abilities of robotic systems, of making them able to make human-like actions in physical world. This motivation for robotic catching is still actual. If the robots are more similar to humans the number of new possible applications for them appears. The prediction of the trajectory of the thrown object may be done not only for catching this object in the industrial environment. Already on this stage it may be applied example in robotic table-tennis. Future of the robotics may give as more applications.

Prediction of flying body trajectory is not the only task, for which the proposed forecasting algorithm may be used. There are a lot of existing applications for time-series forecasting. The characteristics of predicting the ballistic trajectory in comparison with other forecasting task (e.g. predicting the situation of the stock market) are:

1. The process is not influent it has the beginning and the end.
2. No accurate analytical model of the process is available but the rules of the process exists. It is not random and under the same conditions take place in the same way.

3. There are several factors influencing on the process but the influence of each factor cannot be accurately separated from the influence of other factors (e.g. separate influence of the throwing velocity and the elevation of throw).
4. The prognosis must be done in a very small period of time.

For the processes that satisfy these conditions, the introduced algorithm could be an effective method for time series forecasting.

Literature

[Acc03] Accadia, C., Mariani, S., Casaioli, M., Lavagnini, A., Speranza, A.: Sensitivity of Precipitation Forecast Skill Scores to Bilinear Interpolation and a Simple Nearest-Neighbor Average Method on High-Resolution Verification Grids, Weather andForecasting, Vol. 18, No. 5, pp. 918 to 932, October 2003.

[Ach72] Achenbach, E.: Experiments on the flow past spheres at very high Reynolds number, Journal of Fluid Mechanics, No. 54, pp. 565 to 575, August 1972.

[Ada04] Adan, A., Molina, F., Morena, L.: Disordered Patterns Projections for 3D Motion Recovering, International Symposium on 3D Data Processing, Visualization and Transmission, Thessaloniki, Greece, pp. 262 to 269, September 2004.

[Ada05] Adan, A., Molina, F., Vazquez, A. S., Morena, L.: 3D Feature Tracking Using a Dynamic Structured Light System, Canadian Conference on Computer and Robot Vision, pp. 168 to 175, May 2005.

[Akh11] Akhter, N: Visual Tracking of Mechanically Thrown Objects with Planar Surfaces, Dissertation, Faculty of Electrical Engineering, Vienna University of Technology, September 2011.

[Akh12] Akhter, N.: Tracking the planar-textured objects: on the way to transport objects in packaging industry by throwing and catching, International Conference on Pattern Recognition Applications and Methods, Vilamoura, Algarve, Portugal, pp. 316 to 321, February 2012

[Ala98] Alaways, L. W.: Aerodynamics of the Curve-Ball: an Investigation of the Effects of Angular Velocity on Baseball Trajectory, Dissertation, Office of Graduate Studies, University of California, Davis, 1998.

[Ala08] Alam, F., Subic, A. J., Watkins, S., Naser, J., Rasul, M.: An experimental and computational study of aerodynamic properties of rugby balls, WSEAS Transactions on Fluid mechanics: Special Issue on Sustainable Energy and Environmental Fluid Mechanics, Vol. 3, No. 3, pp. 279 to 286, March 2008.

[Ala10] Alam, F., Chowdhury, H., Moria, H., Brooy, R. L., Subic, A.: A Comparative Study of Golf Ball Aerodynamics, Australasian Fluid Mechanics Conference, Auckland, New Zealand, December 2010.

[All93] Allen, P. K., Timcenko, A., Yoshimi, B., Michelman, P.: Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System, IEEE Transactions on Robotics and Automation, Vol. 9, No. 2, pp. 152 to 165, April 1993.

[Alp97] Alpaydin, E.: Voting over Multiple Condensed Nearest Neighbors, Springer Artificial Intelligence Review, Vol. 11, No. 1, pp 115 to 132, February 1997.

[Alq13] Al-Qahtani, F. H., Crone, S. F.: Multivariate k-Nearest Neighbour Regression for Time Series data - a novel Algorithm for Forecasting UK Electricity Demand, 2013 International Joint Conference on Neural Networks, Dallas, USA, pp. 1 to 8, August 2013.

[And85] Andersson, R. L.: Real Time Intelligent Visual Control of a Robot, IEEE Workshop on Intelligent Control, New York, USA, pp. 1 to 6, August 1985.

[And88] Andersson, R. L.: Aggressive trajectory generator for a robotic ping-pong player, IEEE International Conference on Robotics and Automation, Philadelphia, USA, Vol. 3, pp. 188 to 193, April 1988.

[And89a] Andersson, R. L.: Understanding and applying a robot ping-pong player's expert controller, IEEE International Conference on Robotics and Automation, Scottsdale, USA, Vol. 3, pp. 1284 to 1289, May 1989.

[And89b] Andersson, R. L.: Dynamic sensing in a ping-pong playing robot, IEEE Transactions on Robotics and Automation, Vol. 5, No. 6, pp. 728 to 739, December 1989.

[Ang05] Angiulli, F.: Fast Condensed Nearest Neighbor Rule, International Conference on Machine Learning, Bonn, Germany, pp. 25 to 32, August 2005.

[Asa07] Asai, T., Seo, K., Kobayashi, O., Sakashita, R.: Fundamental aerodynamics of the soccer ball, Sports Engineering, Vol. 2007, No. 10, pp. 101 to 110, October 2007.

[Asa10] Asai, T., Ito, S., Seo, K., Hitotsubashi, A.: Aerodynamics of a new volleyball, Procedia Engineering, Vol. 8, No. 2, pp. 2493 to 2498, June 2010.

[Avi00] Avidan, S., Shashua, A.: Trajectory triangulation: 3D reconstruction of moving points from a monocular image sequence, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 4, pp. 348 to 357, April 2000.

[Bae10] Baeuml, B., Wimboeck, T., Hirzinger, G.: Kinematically Optimal Catching a Flying Ball with a Hand-Arm-System, IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, pp. 2592 to 2599, October 2010.

[Bae11] Baeuml, B., Schmidt, F., Wimboeck, T., Birbach, O., Dietrich, A., Fuchs, M., Friedl, W., Frese, U., Borst, C., Grebenstein, M., Eiberger, O., Hirzinger, G.: Catching Flying Balls and Preparing Coffee: Humanoid Rollin'Justin Performs Dynamic and Sensitive Tasks, IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 3443 to 3444, May 2011.

[Bae11a] Baeuml, B., Birbach, O., Wimboeck, T., Frese, U., Dietrich, A., Hirzinger, G.: Catching Flying Balls with a Mobile Humanoid: System Overview and Design Considerations, IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, pp. 513 to 520, October 2011.

[Bai78] Bailey, T., Jain, A. K.: A note on distance weighted k-nearest neighbor rule, IEEE Transactions Systems Cybernetics, Vol. 8, pp. 311-313, 1978.

[Bar08] Barteit, D., Frank, H., Kupzog, F.: Accurate prediction of interception positions for catching thrown objects in production systems, IEEE International Conference on Industrial Informatics. Daejeon, Korea, pp. 893 to 898, July 2008.

[Bar09] Barteit, D., Frank, H., Pongratz, M., Kupzog, F.: Measuring the Intersection of a Thrown Object with a Vertical Plane, IEEE International Conference on Industrial Informatics, Cardiff, UK, pp. 680 to 685, June 2009.

[Bar09a] Barber, S., Chin, S. B., Carre, M. J.: Sports ball aerodynamics: A numerical study of the erratic motion of soccer balls, Computers & Fluids, Vol. 38, pp. 1091 to 1100, November 2008.

[Bar11] Barteit, D.: Tracking of Thrown Objects, Dissertation, Faculty of Electrical Engineering, Vienna University of Technology, December 2011.

[Bat10] Batz, G., Yaqub, A., Wu, H., Kuehnlenz, K., Wollherr, D., Buss, M.: Dynamic Manipulation: Nonprehensile Ball Catching, Mediterranean Conference on Control & Automation, Marrakech, Morocco, pp. 365 to 370, June 2010.

[Bea10] Beale, D., Iravani, P., Hall, P.: Statistical Visual-Dynamic Model for Hand-Eye Coordination, IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, pp. 3931 to 3936, October 2010.

[Ben08] Benavoli, A., Farina, A., Ortenzi, L.: MLE in presence of equality and inequality nonlinear constraints for the ballistic target problem, Radar Conference, Rome, Italy, pp. 1 to 6, May 2008.

[Bha10] Bhatia, N., Vandana: Survey of Nearest Neighbor Techniques, International Journal of Computer Science and Information Security, Vol. 8, No. 2, February 2010.

[Bir09] Birbach, O., Frese, U.: A Multiple Hypothesis Approach for a Ball Tracking System, Spinger Lecture Notes in Computer Science, Vol. 5815, pp. 435 to 444, October 2009.

[Bir11] Birbach, O., Frese, U., Baeuml, B.: Realtime Perception for Catching a Flying Ball with a Mobile Humanoid, IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 5955 to 5962, October 2010.

[Bir11a] Birbach, O., Frese, U.: Estimation and Prediction of Multiple Flying Balls Using Probability Hypothesis Density Filtering, IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, USA, pp. 3426 to 3433, September 2011.

[Bit76] Bitner, J. R., Erlich, G, Reingold, E. M.: Efficient Generation of the Binary Reflected Gray Code and its Applications, Communications of the Association for Computer Machinery, Vol. 19, No. 9, pp. 517 to 521, September 1976.

[Bor08] Borko, F.: Encyclopedia of Multimedia, 2nd Edition, Springer Science + Business Media LLC, 1000 pgs, 2008.

[Bor09] Borst, C., Wimboek, T., Schmidt, F., Fuchs, M., Brunner, B., Zacharias, F., Giordano, P. R., Konietschke, R., Sepp, W., Fuchs, S., Rink, C., Albu-Schaeffer, A., Hirzinger, G.: Rollin' Justin - Mobile Platform with Variable Base, IEEE International Conference on Robotics and Automation, Kobe, Japan, pp. 1597 to 1598, May 2009.

[Boy87] Boyer, K. L., Kak, A. C.: Color-Encoded Structured Light for Rapid Active Ranging, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. I, pp. 14 to 28, January 1987.

[Bri59] Briggs, L. J.: Effect of Spin and Speed on Lateral Deflection (Curve) of a Baseball and Magnus Effect for Smooth Spheres, American Journal on Physics, No.27, pp. 589 to 596, March 1959.

[Can86] Canny, J.: A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, November 1986.

[Cas98] Caspi, D., Kiryari, N.: Range Imaging with Adaptive Color Structured Light, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 5, pp. 470 to 480, May 1998.

[Cha68] Chapman, S.: Catching a Baseball, American Journal of Physics, Vol. 36, No. 10, pp. 868 to 870, October 1968.

[Cha80] Chang, C. B.: Ballistic Trajectory Estimation with Angle-only Measurements, IEEE Transactions on Automatic Control, Vol. 25, No. 3, pp. 474 to 480, June 1980.

[Cha00] Chadwick, S. G., Haake, S. J.: The drag coefficient of tennis balls, International Conference on the Engineering of Sport, Sydney, Australia, pp. 169 to 176, June 2000.

[Cha14] Changey, S., Pecheur, E., Brunner, T.: Attitude Estimation of a projectile using Magnetometers and Accelerometers, IEEE/ION Position, Location and Navigation Symposium, Monterey, USA, pp. 1168 to 1173, May 2014.

[Che09] Chen, L. M., Chen Y. J.: A study of shuttlecock's trajectory in badminton, Journal on Sports Science and Medicine, Vol. 8, pp. 657 to 662, January 2009.

[Chi79] Chitananda G. K., Krishna G.: The Condensed Nearest Neighbor Rule Using the Concept of Mutual Nearest Neighborhood, IEEE Transactions on Information Technology, Vol. 25, No. 4, pp. 488 to 490, July 1979.

[Chi95] Chiou, R. N., Chen, C. H., Hung, K. C., Lee, J. Y.: The Optimal Camera Geometry and Performance Analysis of a Trinocular Vision System, IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, No. 8, August 1995.

[Cho11] Choe, T. E., Rasheed, Z., Taylor, G., Haering, N.: Globally Optimal Target Tracking in Real Time using Max-Flow Network, IEEE International Conference on Computer Vision, Barselona, Spain, pp. 1855 to 1862, November 2011.

[Chu03] Chudinov, P. S.: An optimal angle of launching a point mass in a medium with quadratic drag force, Indian Journal on Physics, Vol. 77B, pp. 465 to 468, December 2003.

[Cig15] Cigliano, P., Lippiello, V., Ruggiero, F., and B. Siciliano, B.: Robotic Ball Catching with an Eye-in-Hand Single-Camera System, IEEE Transactions on Control System Technology, Vol. 23, No. 5, pp. 1657-1671, May2015.

[Coo96] Cooke, A. J., Shuttlecock design and development, in Haake, S.: the Engineering of Sport, Sheffield, Tailor & Francis.

[Coo99] Cooke, A. J., Shuttlecock aerodynamic, Sport Engineering, Vol. 2, pp. 85 to 96, January 1999

[Coo00] Cooke, A. J.: An Overview of Tennis Ball Aerodynamics, Sports Engineering, No. 3 2000, pp.123 to 129, February 2000.

[Cov67] Cover, T. M., Hart, P. E.: Nearest Neighbor Pattern Classification, IEEE Transactions in Information Theory, Vol. IT-13, pp. 21-27, 1967.

[Dav49] Davies, J. M.: The Aerodynamics of Golf Balls, Journal of Applied Physics, Vol. 20, No. 9, pp. 821 to 829, September 1949.

[Dav96] Davies, C.J., Nixon, M.S.: Sensing Surface Discontinuities via Coloured Spots, International Workshop on Image and Signal Processing, Manchester, UK, pp. 573 to 576, November 1996.

[Fae08] Faes, L., Erla, S., Nollo, G: Quantifying the complexity of short-term heart period variability through k nearest neighbor local linear prediction, Conference on Computers in Cardiology, Bologna, Italia, pp. 549 to 552, September 2008.

[Far02] Farina, M., Ristic, B., Benvenutti, D.: Tracking a Ballistic Target: Comparison of Several Nonlinear Filters, IEEE Transactions on Aerospace and Electronic Systems, Vol. 38, No. 3, pp. 854 to 867, July 2002.

[Fed90] Feddema, J. T., Lee, C. S. G.: Adaptive Image Feature Prediction and Control for Visual Tracking with a Hand-Eye Coordinated Camera, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 5, pp. 1172 to 1183, September 1990.

[Fis81] Fischler, M. A., Bolles, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Communications of the Association for Computing Machinery, Vol. 24, No. 6, pp. 381 to 395, June 1981.

[Fra06] Frank, H., Wellerdick-Wojtasik, N., Hagebeuker, B., Novak, G., Mahlknecht, S.: Throwing Objects: a bio-inspired Approach for the Transportation of Parts, IEEE International Conference on Robotics and Biomimetics, Kunming, China, pp. 91 to 96, December 2006.

[Fra07] Frank, H., Barteit, D., Wellerdick-Wojtasik, N., Frank, T., Novak, G., Mahlknecht, S.: Autonomous Mechanical Controlled Grippers for Capturing Flying Objects, IEEE International Conference on Industrial Informatics, Vienna, Austria, pp. 431 to 436, June 2006.

[Fra08] Frank, H.: Design and Simulation of a Numerical Controlled Throwing Device, Second Asia International Conference on Modeling and Simulation AICMS 08, Kuala Lumpur, Malaysia, pp. 777 to 782, May 2008.

[Fra08a] Frank, H., Barteit, D., Meyer, M., Mittnacht, A., Novak, G., Mahlknecht, S.: Optimized Control Methods for Capturing Flying Objects with a Cartesian Robot, IEEE Conference on Robotics, Automation and Mechatronics, Chengdu, China, pp. 160 to 165, September 2008.

[Fra09] Frank, H., Mittnacht, A., Scheiermann, J.: Throwing of Cylinder Shaped Objects, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore, pp. 59 to 64, July 2009.

[Fra10] Frank, T, Schroedter, C., Janoske, U: Holistic Modeling of Trajectories for Cylinder-Shaped Objects, European Symposium on Computer Modelling and Simulation, Pisa, Italy, pp. 223 to 228, November 2010.

[Fra11] Frank, T., Janoske, U., Schroedter, C.: Detection of Position and Orientation of Flying Cylinder Shaped Objects by Distance Sensors, IEEE International Conference on Mechatronics, Istanbul, Turkey, pp. 1623 to 1629, April 2011.

[Fra11a] Frank, H., Frank, T., Mittnacht, A., Sichau, C.: A Bioinspired 2DOF Throwing Robot, IEEE Africon, Livingstone, Zambia, pp. 1 to 6, September 2011.

[Fra12] Frank, T., Janoske, U., Mittnacht, A., Schroedter, C.: Automated Throwing and Capturing of Cylinder-Shaped Objects, IEEE International Conference on Robotic and Automation, Saint Paul, Minnesota, USA, pp. 5264 to 5270, May 2012.

[Fre01] Frese, U., Baeuml, B., Haidacher, S., Schreiber, G., Schaefer, I., Haehnle, M., Hirzinger, G.: On-the-Shelf Vision for a Robotic Ball Catcher, IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, Hawaii, USA, pp. 591 to 596, November 2001.

[Fro84] Frohlich, C.: Aerodynamic drag crisis and its possible effect on the flight of baseballs, American Journal of Physics, Vol. 52, No. 4, pp. 325 to 364, April 1984.

[Fuc08] Fuchs, S., Hirzinger, G.: Extrinsic and Depth Calibration of ToF-cameras, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Ancorage, USA, pp. 1 to 6, June 2008.

[Fuj14] Fujimoto, Y., Sugiura, T., Murata, N.: K-Nearest Neighbor Approach for Forecasting Energy Demands Based on Metric Learning, International Work-Conference on Time Series, Granada, Spain, pp. 1127 to 1137, June 2014.

[Fur06] Furukawa, N., Namiki, A., Taku, S., Ishikawa, M.: Dynamic Regrasping Using a High-speed Multifingered Hand and a High-speed Vision System, IEEE International Conference on Robotics and Automation, Orlando, Florida, USA, pp. 181 to 187, May 2006.

[Gan10] Ganapathi, V., Plagemann, C., Koller, D., Thrun, S.: Real Time Motion Capture Using a Single Time-Of-Flight Camera, IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA, pp. 755 to 762, June 2010.

[Gar11] Garg, R., Indu, S., Chadhury, S.: Camera and Light Source Placement: A Multi-Objective Approach, Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, Hubli, India, pp. 187 to 191, December 2011.

[Gat72] Gates, G.: The reduced nearest neighbor rule, IEEE Transactions on Information Theory, Vol. 18, No. 3, pp. 431 to 433, May 1972.

[Gil99] Gillies, M. F. P., Dodgson, N. A.: Ball Catching: An Example of Psychologically-based Behavioural Animation, Eurographics UK 17th Annual Conference, Cambridge, UK, pp. 229 to 236, April 1999.

[Goe15] Goetzinger, M.: Object Detection and Flightpath Prediction, Diploma Thesis, Faculty of Electrical Engineering, Vienna University of Technology, June 2015.

[Gof09] Goff, J. E., Carre, M. J.: Trajectory analysis of a soccer ball, American Journal of Physics, Vol. 77, pp. 1020 to 1027, November 2009.

[Gof13] Goff, J. E.: A review of recent research into aerodynamics of sport projectiles, Sports Engineering, Vol. 2013, No. 4, pp. 137 to 154, April 2013.

[Gro07] Groover, M. P.: Automation, Production Systems, and Computer-Integrated Manufacturing, Upper Saddle River, NJ, USA : Prentice Hall PTR, 840 pp., 2007.

[Guo12] Guo, F., Short-term traffic prediction under normal and incident conditions using singular spectrum analysis and the k-nearest neighbour method, IET and ITS Conference on Road Transport Information and Control, London, UK, pp. 1-6, September 2012.

[Hal01] Hall-Holt, O., Rusinkiewics, S.: Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects, IEEE International Conference on Computer Vision, Vancouver, Canada, Vol. 2, pp. 359 to 366, July 2001.

[He91] He, X., Benhabib, B., Smith, K. C., Safaee-Rad, R.: Optimal Camera Placement for an Active-Vision System, IEEE international conference on decision aiding for complex systems, Charlottesville, USA, pp. 69 to 74, October 1991.

[Her09] Herrejon, R., Kagami, S., Hashimoto, K.: Position Based Visual Servoing for Catching a 3-D Flying Object Using RLS Trajectory Estimation from a Monocular Image Sequence, IEEE International Conference on Robotics and Biomimetics, Guilin, China, pp. 665 to 670, December 2009.

[Hla12] Hlawatsch, F.: Parameter Estimation Methods: Lecture Notes, Grafisches Zentrum HTU GmbH, Vienna, Austria, March 2012.

[Hor99] Horn, E. and Hiryati, N.: Toward Optimal Structured Light Patterns, Image and Vision Computing, Vol. 17, No. 2, pp. 87 to 97, February 1999.

[Hou62] Hough, P.: A method and means for recognizing complex patterns, U.S. Patent No. 3,069,654, December 1962.

[Hov91] Hove, B., Slotine, J.-J.: Experiments in Robotic Catching, American Control Conference, Boston, USA, pp. 381 to 386, June 1991.

[Hub87] Hubbard, M., Alaways, L. W.: Optimum Release Conditions for the New Rules Javelin, International Journal on Sport Biomechanics, Vol. 3, pp. 207 to 221, January 1987.

[Hub87] Hubbard, M., Bergman, C. D.: Effect of Vibrations on Javelin Lift and Drag, International Journal on Sport Biomechanics, Vol. 5, pp. 207 to 221, January 1989.

[Ima04] Imai, Y., Namiki, A., Hashimoto, K., Ishikawa, M.: Dynamic Active Catching Using a High-speed Multifingered Hand and a High-speed Vision System, IEEE International Conference on Robotics & Automation, New Orlean, USA, pp. 1849 to 1854, April 2004.

[Ish96] Ishii, I., Nakabo, Y., Ishikawa, M.: Target Tracking Algorithm for lms Visual Feedback System Using Massively Parallel Processing, IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, USA, pp. 2309 to 2314, April 1996.

[Ita12] Itagaki, Y., Suzuki, A., Iyota, T.: Indoor Positioning for Moving Objects Using a Hardware Device with Spread Spectrum Ultrasonic Waves, International Conference on Indoor Positioning and Indoor Navigation, Sydney, Australia, pp. 1 to 6, November 2012.

[Jaz70] Jazvinski, A. H.: Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.

[Jia14] Jia, L., Radke, R. J.: Using Time-of-Flight Measurements for Privacy-Preserving Tracking in a Smart Room, IEEE Transactions on Industrial Informatics, Vol. 10, No. 1, pp. 689 to 696, February 2014.

[Jur12] Jurado, F., Palacios, G., Flores, F.: Vision–based Trajectory Tracking on the 3D Virtual Space for a Quadrotor, Electronics, Robotics and Automotive Mechanics Conference, Cuernavaca, Mexico, pp. 31 to 36, November 2012.

[Kaj99] Kajikawa, S., Saito, M., Ohba, K., Inooka, H.: Analysis of Human Arm Movement for Catching a Moving Object, IEEE International Conference on Systems, Man and Cybernetics, Tokyo, Japan, Vol. 2, pp. 698 to 703, October 1999.

[Kal61] Kalman, R. E., Bucy, R. S.: New Results in Linear Filtering and Prediction Theory, Journal on Fluids Engineering, Vol. 83, pp. 95 to 108, March 1961.

[Kan12] Kang, H., Park, F. C.: Humanoid Motion Optimization via Nonlinear Dimension Reduction, IEEE International Conference on Robotics and Automation RiverCentre, Saint Paul, Minnesota, USA, pp. 1444 to 1449, May 2012.

[Kao13] Kao, S.-T., Yang, Z.-Y., Hong, M.-T.: Design and Implementation of a Color-Based Visual Tracking Control System, International Conference on Automatic Control, Sun Moon Lake, Taiwan, pp. 371 to 376, December 2013.

[Kar54] Karman, T. von: Aerodynamics. Selected Topics in the Light of Their Historical Development, Cornell University Press, Ithaca, New York, March 1954.

[Kay93] Kay, S. M.: Fundamentals of Statistical Signal Processing: Estimation Theory, Prentice Hall, Englewood Cliffs, USA, March 1993.

[Kho12] Khoshelham, K., Elberink, S. O.: Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications, Sensors, Vol. 12, No. 2, pp. 1437-1454, February 2012.

[Kim75] Kimme, C., Ballard, D., Sklansky, J.: Finding circles by an array of accumulators, Communications of the ACM, Vol. 18, No. 2, pp. 120 to 122, February 1975.

[Kim12] Kim, S., Billard, A.: Estimating the non-linear dynamics of free-flying objects, Robotics and Autonomous Systems, Vol. 60, pp. 1108-1122, June 2012.

[Kim13] Kim, J., Hung, N. H., Lee, Y., Lee, S.: Structured Light Camera Base 3D Visual Perception and Tracking Application System with Robot Grasping Task, IEEE International Symposium on Assembly and Manufacturing, Xian, China, pp. 187 to 192, August 2013.

[Kim14] Kim, S., Shukla, A., Billard, A.: Catching Objects in Flight, IEEE Transactions on Robotics, Vol. 30, No. 5, pp. 1049 to 1065, May 2014.

[Kob11] Kober, J., Peters, J.: Learning Elementary Movements Jointly with a Higher Level Task, IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, USA, pp. 338 to 343, September 2011.

[Kra05] Kramer, K. A., Stubberud, S. C.: Impact Time and Point Predicted Using a Neural Extended Kalman Filter, International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, pp. 199 to 204, December 2005.

[Lee02] Lee, J.H., Akiyama, T., Hashimoto, H.: Study on Optimal Camera Arrangement for Positioning People in Intelligent Space, IEEE/RSJ international conference on intelligent robots and systems, Lausanne, Switzerland, pp. 220 to 225, October 2002.

[Lia10] Liaw, Y. C., Leou, M. L.: Fast Exact k Nearest Neighbor Search using Orthogonal Search Tree, Pattern Recognition, Vol. 43, No. 6, pp. 2351 to 2358, June 2010.

[Lin89] Lin, Z., Zeman, V., Patel, R. V.: On-line robot trajectory planning for catching a moving object, IEEE International Conference on Robotics and Automation, Scottsdale, USA, Vol. 3, pp. 1726 to 1731, May 1989.

[Liu06] Liu, J., Zhang, Y., Li, Z.: Selection of Cameras Setup Geometry Parameters in Binocular Stereovision, IEEE Conference on Robotics, Automation and Mechatronics, Bangkok, Thailand, pp. 1 to 6, June 2006.

[Liu09] Liu, L., Zhang, X., Ma, H.: Dynamic Node Collaboration for Mobile Target Tracking in Wireless Camera Sensor Networks, IEEE Conference on Computer Communications, Rio de Janeiro, Brazil, pp. 1188 to 1196, April 2009.

[Liu10] Liu, L., Zhang, X., Ma, H.: Optimal Node Selection for Target Localization in Wireless Camera Sensor Networks, IEEE Transactions on vehicular technology, Vol. 59, No. 7, pp. 3562 to 3576, September 2010.

[Luc87] Lucero, E. F., Hagan, J. C., Beyers, M. E.: Subsonic Aerodynamics of Rectangular Parallelepiped Shapes of Fineness Ratio of One-Half, Journal of Spacecrafts and Rockets, Vol. 24, No. 4, pp. 311 to 318, July-August 1987.

[Mao10] Mao, A.: Ball Catching: the Inspiration to Power System Stability Control, A Fast Algorithm for the Generator's Disturbed Trajectory Prediction, IEEE Power Engineering Society General Meeting, Tampa, Florida, USA, pp. 1 to 7, June 2007.

[Mar16] Markum, G.: Object Touchdown Position Prediction, Bachelor Thesis, Faculty of Electrical Engineering, Vienna University of Technology, March 2016.

[Meh08] Mehta, R., Alam, F., Subic, A.: Review of tennis ball aerodynamics, Sports technology review, John Wiley and Sons Asia Pte Ltd, 2008, No. 1, pp. 7 to 16, January 2008.

[Mir13] Mironov, K. V., Pongratz, M.: Applying neural networks for prediction of flying objects trajectory, Vestnik UGATU, Vol. 17, No. 6(59), pp. 33 to 37, December 2013.

[Mir14] Mironov, K. V., Pongratz, M., Dietrich, D.: Predicting the Trajectory of a Flying Body Based on Weighted Nearest Neighbors, International Work-Conference on Time Series, Granada, Spain, pp. 699 to 710, June 2014.

[Mir15] Mironov, K. V., Vladimirova, I. V., Pongratz, M.: Processing and Forecasting the Trajectory of a Thrown Object Measured by the Stereo Vision System, IFAC-PapersOnLine, Vol. 48, No. 11, pp. 28 to 25, June 2015.

[Miy10] Miyashita, H., Yamavaki, T., Yashima, M.: Learning Control Method for Throwing an Object More Accurately with One Degree of Freedom Robot, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Montréal, Canada, pp. 397 to 402, July 2010.

[Mor04] Mori, R., Hashimoto, K., Miyazaki, F.: Tracking and Catching of 3D Flying Target based on GAG Strategy, IEEE International Conference on Robotics 8 Automation, New Orleans, USA, pp. 5189 to 5194, April 2004.

[Mor10] Morsly, Y., Djouadi, M. S., Aouf, N.: On the Best Interceptor Placement for an Optimally Deployed Visual Sensor Network, Istanbul, IEEE international conference on systems, man and cybernetics, Turkey, pp. 43 to 51, October 2010.

[Mue10] Muelling, K., Kober, J., Peters, J.: A Biomimetic Approach to Robot Table Tennis, IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, pp. 1921 to 1926, October 2010.

[Mue11] Mueller, M., Lupashin, S., D'Andrea, R.: Quadrocopter Ball Juggling, IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, USA, pp. 5113 to 5120, September 2011.

[Muk94] Mukai, T., Ishikawa, M.: An Active Sensing Method Using Estimated Errors for Multisensor Fusion Systems, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, USA, pp. 615 to 622, October 1994.

[Mur05] Murphey, T. D., Bernheisel, J., Choi, D., Lynch, K. M.: An Example of Parts Handling and Self-Assembly Using Stable Limit Sets, IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, pp.1624 to 1629, August 2005.

[Nak11] Nakamura, T.: Real-time 3-D Object Tracking Using Kinect Sensor, IEEE International Conference on Robotics and Biomimetics, Phuket, Thailand, pp. 784 to 788, December 2011

[Nam99] Namiki, A., Nakabo, Y., Ishii, I., Ishikawa, M.: High Speed Grasping Using Visual and Force Feedback, IEEE International Conference on Robotics & Automation, Detroit, Michigan, USA, pp. 3195 to 3200, May 1999.

[Nam03a] Namiki, A., Ishikawa, M.: Robotic Catching Using a Direct Mapping from Visual Information to Motor Command, IEEE International Conference on Robotics &Automation, Taipei, Taiwan, pp. 2400 to 2405, September 2003.

[Nam03b] Namiki, A., Imai, Y., Ishikawa, M.: Development of a High-speed Multifingered Hand System and Its Application to Catching, lEEE/RSJ lnternational Conference on Intelligent Robots and Systems, Las Vegas, Nevada, USA, pp. 2666 to 2671, October 2003.

[Nel05] Nelson, E., Pachter, M., Musick, S.: Projectile Launch Point Estimation from Radar Measurements, American Control Conference, Portland, USA, pp. 1275 to 1282, June 2005.

[Nem11] Nemec, B., Vuga, R., Ude, A.: Exploiting Previous Experience to Constrain Robot Sensorimotor Learning, IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, pp. 727 to 732, October 2011.

[Nis97] Nishiwaki, K., Konno, A., Nagashima, K., Inaba, M., Inoue, H.: The Humanoid Saika that Catches a Thrown Ball, IEEE International Workshop on Robot and Human Communication, Sendai, Japan, pp. 94 to 99, October 1997.

[Non10] Nonomura, J., Nakashima, A., Hayakawa, Y.: Analysis of Effects of Rebounds and Aerodynamics for Trajectory of Table Tennis Ball, SICE Annual Conference 2010, Taipei, Taiwan, August 2010.

[Noo11] Noonan, P. J., Cootes, T. F., Hallet, W. A., Hinz, R.: The Design and Initial Calibration of an Optical Tracking System Using the Microsoft Kinect, IEEE Nuclear Science Symposium and Medical Imaging Conference, Valencia, Spain, pp. 3614 to 3617, October 2011.

[Opr13] Oprisescu, S., Florea, L., Ovreiu, E.: Detection of thrown objects using ToF cameras, IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, pp. 83 to 86, September 2013.

[Par11] Park, Y., Lepetit, V., Woo, W.: Texture-Less Object Tracking with Online Training using An RGB-D Camera, IEEE International Symposium on Mixed and Augmented Reality, Bazel, Switzerland, pp. 121 to 126, October 2010.

[Pia10] Piatti, D.: Time-of-Flight cameras: tests, calibration and multi-frame registration for automatic 3D object reconstruction, PhD thesis, Doctoral school of Environment and Territory, Polytechnic University of Turin, December 2010.

[Pon09] Pongratz, M.: Object Touchdown Position Prediction, Diploma Thesis, Faculty of Electrical Engineering, Vienna University of Technology, September 2009.

[Pon10] Pongratz, M., Kupzog, F., Frank, H., Barteit, D.: Transport by Throwing - a bio-inspired Approach, IEEE International Conference on Industrial Informatics, Osaka, Japan, pp. 685 to 689, July 2010.

[Pon11] Pongratz, M., Pollhammer, K., Szep, A.: KOROS Initiative: Automatized Throwing and Cathcing for Material Transportation, ISoLA 2011 Workshops, pp. 136 to 143, 2012.

[Pon13] Pongratz, M., Mironov, K. V., Bauer F.: A soft-catching strategy for transport by throwing and catching, Vestnik UGATU, Vol. 17, No. 6(59), pp. 28 to 32, December 2013.

[Pon15] Pongratz, M., Mironov, K. V.: Accuracy of Positioning Spherical Objects with Stereo Camera System, IEEE International Conference on Industrial Technology, Seville, Spain, pp. 1608 to 1612, March 2015.

[Pon16] Pongratz, M.: Bio-Inspired Transport by Throwing System, Dissertation, Faculty of Electrical Engineering, Vienna University of Technology, January 2016.

[Pos09] Post, S. L., McLachlan, J., Lonas, T., Dancs, J., Knobloch, D., Darrow, C., Sinn, E., Davis, S., Neilly, D., Funk, A., Golz, J., Phelps, A., Goers, B.: Aerodynamics of Badminton Shuttlecock, ASME International Mechanical Engineering Congress & Exposition, Lake Buena Vista, USA, pp. 1 to 6, November 2009.

[Pro96] Proesmans, M., Van Gool, L., Oosterlinck, A.: One-Shot Active 3D Shape Acquisition, Proc. International Conference on Pattern Recognition, Vienna, Austria, Vol. 3, pp. 336 to 340, August 1996

[Qin13] Qin, Y. F., Zhong, H.: Research on Basketball Flight Simulation Based on the Video Data Analysis Technology, Applied Mechanics and Materials, Vol. 380 to 384, pp. 1851 to 1855, August 2013.

[Rao07] Rao, R. V.: Decision Making in the Manufacturing Environment, Springer, London, 373 pp., 2007.

[Rau65] Rauch, H. E., Striebel, C. T., Tung, F.: Maximum likelihood estimates of linear dynamic systems, AIAA Journal, Vol. 3, No. 8, pp. 1445-1450, August 1965.

[Rav10] Ravindra, V. C., Bar-Shalom, Y., Willett, P.: Projectile Identification and Impact Point Prediction, IEEE Transactions on Aerospace and Electronic Systems, Vol. 46, No. 4, pp. 2004 to 2021, October 2010.

[Rib09] Ribnick, E., Atev, S., Papanikolopoulos, N.P.: Estimating 3D Positions and Velocities of Projectiles from Monocular Views, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, No. 5, pp. 938 to 944, May 2009.

[Ril02] Riley, M., Atkeson, C. G.: Robot Catching: Towards Engaging Human-Humanoid Interaction, Autonomous robots, Vol. 12, No. 1, pp. 119 to 128, January 2002.

[Ris03] Ristic, B., Farina, A., Benvenutti, D., Arulampalam, M. S.: Performance bounds and comparison of nonlinear filters for tracking a ballistic object on re-entry, IEE Proceedings on Radar and Sonar Navigation, Vol. 150, No. 3, pp. 65 to 70, April 2003.

[Sca05] Scaramuzza, D., Pagnotelli, S., Valligi, P.: Ball Detection and Predictive Ball Following Based on a Stereoscopic Vision System, IEEE International Conference on Robotics and Automation, Barcelona, Spain, pp. 1573-1578, April 2005.

[Sej87] Sejnovsky T. J., Rosenberg C. R.: Parallel networks that learn to pronounce English text, Complex systems, Vol. 1987, No. 1, pp.145 to168, January 1987.

[Sen04] Senoo, T., Namiki, A., Ishikawa, M.: High-speed Batting Using a Multi-Jointed Manipulator, IEEE International Conference on Robotics and Automation, New Orleans, USA, pp. 1191 to 1196, April 2004.

[She09] Shen, Q., He, X.: GPS Positioning-based Trajectory Parameter Estimation and Ejection Point Self-adapting Control Method, Second International Symposium on Knowledge Acquisition and Modeling, Wuhan, China, pp. 194 to 197, December 2009.

[Shi05] Shiokata, D., Namiki, A., Ishikawa, M.: Robot Dribbling Using a High-speed Multifingered Hand and a High-speed Vision System, IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, pp. 2097 to 2102, August 2005.

[Sir12] Siradjuddin, I., Behera, L., McGinnity, T. M., Coleman, S.: A position based visual tracking system for a 7 DOF robot manipulator using a Kinect camera, IEEE World Congress on Computational Intelligence, Brisbane, Australia, pp. 1 to 7, June 2012.

[Smi07] Smith, C., Christensen, H. I.: Using COTS to Construct a High Performance Robot Arm, International Conference on Robotics and Automation, Roma, Italy, pp. 4056 to 4063, April 2007.

[Spr91] Sproull, R. F.: Refinements to Nearest Neighbor Searching in k-Dimensional Trees, Algorithmica, Vol. 6, No. 1-6, pp. 579 to 589, June 1991.

[Ste88] Stepanek, A., The aerodynamics of tennis balls – the topspin lob, American Journal of physics, No. 56, pp.138 to 142, February 1988.

[Suk12] Sukhan, l., Kyeongdae, Y., Jaewoong, K., Moonju, L.: Surface Patch Primitive Based Object Modeling from CAD Data, Applied Mechanics and Materials, vol. 162, pp. 179-183, March 2012.

[Sul09] Sule, D. R.: Manufacturing Facilities: location, planning, and design. Third edition. CRC Press, Boca Raton, 2009

[Swa08] Swadzba, A., Beuter, N., Schmidt, J., Sagerer, G.: Tracking Objects in 6D for Reconstructing Static Scenes, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Ancorage, USA, pp. 1 to 7, June 2008.

[Sze10] Szeliski, R.: Computer Vision: Algorithms and Applications, Springer, September 2010.

[Wan08] Wang, S., Kim, H., Lin, C.-S., Chen, H.: A Robust Depth Measurement Method with Optimal Trace Tracking of Structured Light Using Dynamic Programming, IEEE International Conference on Industrial Technology, Chengdu, China, pp. 1 to 5, April 2008.

[Wat87] Watts, R. G., Ferrer, R.: The Lateral Force on a spinning sphere: Aerodynamics of a Curveball, American Journal of physics, No. 55, pp. 40 to 43, January 1987.

[Wei80] Weinstein, E., Levanon, N.: Passive Array Tracking of a Continuous Wave Transmitting Projectile, IEEE Transactions on Aerospace and Electronic Systems, Vol. 16, No. 5, pp. 721 to 726, September 1980.

[Win12] Winkler, V., Edrich, M., Ziegler, H. W.: Ka-Band FMCW-Radar for Sniper Detection, International Radar Symposium, Warsaw, Poland, pp. 201 to 204, May 2012.

[Yak87] Yakowitz, S.: Nearest-Neighbour Methods for time series analysis, Journal of Time Series Analyses, Vol. 8, No. 2, pp. 235-247, 1987.

[Yao08] Yao, J., Wang, X., Gao, Y., Wang, Y.: A Method of Identifying Rocket Trajectory Parameters Based on Generalized Kalman Attenuating Memory Algorithm, IEEE Conference on Industrial Electronics and Applications, Singapore, pp. 1204 to 1206, June 2008.

[Yil06] Yilmaz, A., Javed, O., Shah, M., Object Tracking: A Survey, ACM Journal of Computing Surveys, Vol. 38, No. 4, Article 13, pp. 1 to 45, December 2006.

[Yua12] Yuan, T., Bar-Shalom, Y., Wilett, P., Mozeson, E., Pollak, S., Hardiman, D.: A Multiple IMM Estimation Approach with Unbiased Mixing for Thrusting Projectiles, IEEE Transactions on Aerospace and Electronic Systems, Vol. 48, No. 4, pp. 3250 to 3267, October 2012.

[Yua14] Yuan, T., Bar-Shalom, Y., Wilett, P., Hardiman, D.: Impact Point Prediction for Thrusting Projectiles in the Presence of Wind, IEEE Transactions on Aerospace and Electronic Systems, Vol. 50, No. 1, pp. 102 to 119, January 2014.

[Zar00] Zarchan, P.: Tracking and Intercepting Spiraling Ballistic Missiles, Position Location and Navigation Symposium, San Diego, USA, pp. 277 to 284, March 2000.

[Zha00] Zhang, Z.: A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 11, pp. 1330 to 1334, November 2000.

[Zha12] Zhang, Y., Luo, J., Hauser, K.: Sampling-based Motion Planning With Dynamic Intermediate State Objectives: Application to Throwing, IEEE International Conference on Robotics and Automation, Saint Paul, Minnesota, USA, pp. 2551 to 2556, May 2012.

[Zho04] Zhou, Y., Zhang, C.: Tunable Nearest Neighbor Classifier, Pattern recognition, Springer Lecture Notes in Computer Science, Vol. 3175, pp. 204 to 211, September 2004.

[Zho07] Zhou, D.-Q.: Study of Key Techniques Applied in Radars of Locating Enemy Artilleres, International Conference on Microwave and Millimeter Wave Technology, Builin, China, pp. 1 to 4, April 2007.

Internet references

[1] The drag equation, National Aeronautic and Space Association, https://www.grc.nasa.gov/www/k-12/airplane/drageq.html, visited on March 21, 2014

[2] Animation – Der Schraege/schiefe Wurf ohne und mit Luftwiderstand/dynamischem Auftrieb/Magnus-Effekt, M. Tutz, http://www.tutz.ws/JS/Simulation-Schraeger-Wurf-F_L-F_A-F_M.html, visited on July 1, 2013.

[3] Athletics discipline – Javelin – Disciplines – IAAF, International Association of Athletics Federations, http://www.iaaf.org/disciplines/throws/javelin-throw, visited on August 1, 2014.

[4] The record of shuttlecock initial velocity is 414 km/h (in Russian), Badminton Blog, http://badmintonblog.ru/topics/record/, visited on August 4, 2014.

[5] Kinect for Windows features - Microsoft, http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx, visited on September 15, 2014

[6] Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/, visited on September 18, 2014.

[7] Каталог, прайс-лист всех поставляемых изделий Phoenix Contact, Moeller, Rittal, ifm electronic, Socomec (In Russian: Catalog of all available devices with prices: Phoenix Contact, Moeller, Rittal, ifm electronic, Socomec) – DENOL LLC, http://www.denol.ru/sub_cat.php?str_find=+&rests=IFM-electronic&page=6, visited on September 24, 2014

[8] MESA Imaging 3D ToF Camera SR4000 (USB, 10 m Range) – Robot Shop, http://www.robotshop.com/ca/en/mesa-imaging-3d-tof-camera-usb-10m-range.html, visited on September 25, 2014.

[9] CamBoard Apps, PMD [Vision] CamCube 3.0 – PMDtechnologies Gsmbh, http://www.pmdtec.com/news_media/video/camcube.php, visited on September 25, 2014.

[10] UI-3370CP – USB 3 Cameras – CAMERAFINDER – Products, http://en.ids-imaging.com/store/ui-3370cp.html, visited on September 25, 2014.

[11] 3D object tracking by the particle filter and the Kinect – Youtube, https://www.youtube.com/watch?v=2m47m-UNwWc&feature=feedu, visited on September 26, 2014.

[12] Motion Capture Suits & Markers – OptiTrack, Natural Point, Inc., http://www.naturalpoint.com/optitrack/products/suits-markers/, visited on October 3, 2014.

[13] Curve fitting toolbox – MATLAB, http://www.mathworks.com/products/curvefitting/, visited on November 29, 2014.

[14] Camera Calibration – MATLAB & Simulink, http://mathworks.com/help/vision/camera-calibration.html, visited on September 28, 2015.

[15] ITF Tennis – Technical, http://www.itftennis.com/technical/balls/overview.aspx, visited on September 30, 2015.

[16] Laminar Air Flow – Welcome to the Tubus Bauer webpage, http://www.tubus-bauer.de/laminar-air-flow.html, visited on October 6, 2015.

[17] Martin Pongratz | Institute of Computer Technology Wien, https://www.ict.tuwien.ac.at/en/users/pongratz, visited on November 6, 2015.

[18] TISS – Markum Gilbert Harald, https://tiss.tuwien.ac.at/adressbuch/adressbuch/person/231464, visited on November 6, 2015.

[19] Machine Vision – Institute of Automation and Control, , visited on November 10, 2015.

Curriculum Vitae

Konstantin MIRONOV

Date of birth: January 3, 1990
Address in Austria: Ehrensteingasse 3/I/4, 1220 Wien
Address in Russia: ul. Lenina 102-23, 450006 Ufa
mironovconst@gmail.com

EDUCATION

- 2012: MSc Information Security from Ufa State Aviation Technical University, Ufa, Russian Federation.
- 2011: BSc Information Technology from Ufa State Aviation Technical University, Ufa, Russian Federation.

WORK EXPERIENCE

- September 2014 to date: university assistant at Ufa State Aviation Technical University.
- January 2013 to June 2014: junior researcher at Ufa State Aviation Technical University.
- July 2012 to date: doctoral student at Technische Universitaet Wien.

SELECTED PUBLICATIONS

- Pongratz, M., Mironov, K. V., Bauer F.: A soft-catching strategy for transport by throwing and catching, Vestnik UGATU, Vol. 17, No. 6(59), pp. 28 to 32, December 2013.

- Mironov, K. V., Pongratz, M.: Applying neural networks for prediction of flying objects trajectory, Vestnik UGATU, Vol. 17, No. 6(59), pp. 33 to 37, December 2013.

- Mironov, K. V., Pongratz, M., Dietrich, D.: Predicting the Trajectory of a Flying Body Based on Weighted Nearest Neighbors, International Work-Conference on Time Series, Granada, Spain, pp. 699 to 710, June 2014.

- Pongratz, M., Mironov, K. V.: Accuracy of Positioning Spherical Objects with Stereo Camera System, IEEE International Conference on Industrial Technology, Seville, Spain, pp. 1608 to 1612, March 2015.

- Mironov, K. V., Vladimirova, I. V., Pongratz, M.: Processing and Forecasting the Trajectory of a Thrown Object Measured by the Stereo Vision System, IFAC-PapersOnLine, Vol. 48, No. 11, pp. 28 to 25, June 2015.

- Mironov, K. V., Pongratz, M.: Fast kNN-based Prediction for the Trajectory of a Thrown Body, Mediterranean Conference on Control and Automation, Athens, Greece, June 2016 (*submitted*).