AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

INSPECTION OF GLASS CONTAINERS USING MULTI-TARGET PARTICLE TRACKING AND 3D POSITIONING

MIKKEL KRAGH HANSEN, KIM BJERGE
20073811, 20097553

MASTER'S THESIS
IN
TECHNICAL INFORMATION TECHNOLOGY

SUPERVISOR: ASSOC. PROF. PETER AHRENDT
Aarhus University, Department of Engineering

Abstract

When inspecting medical containers of liquids in the pharmaceutical industry, containers that contain either cosmetic defects or foreign particles are rejected. This is done by analyzing two-dimensional side-view images of containers and rejecting these based upon impurities and visible particles. However, a distinction between particles residing inside or outside glass containers is not possible due to the lack of a third dimension, and thus some glass containers with harmless dirt on the outside are unnecessarily discarded. An assessment of the extent of this problems states that roughly 0.5% of the rejected containers are rejected due to dirt on the outside.

Therefore, in this thesis we investigate and propose a method for detecting and positioning particles three-dimensionally based on a sequence of images. We do this by rotating a container with a known angular velocity and record images with known intervals. By tracking the movement of a particle, an estimate of the three-dimensional position can be calculated. A distinction can thus be made whether a particle is located inside or outside a container relative to a measured container radius.

The proposed method has an accuracy of 90% on a test set consisting of 20 containers. It is able to process recorded images in real-time with a camera acquisition rate of 20 frames per second. Classification errors result from tracking failures and extremely small particles overlooked by segmentation. Two advanced tracking algorithms are applied to compensate for problems concerning false detections and occlusion. However, in practice these show no major improvements in performance on the test set. Future work must verify this conclusion based on a larger and more heterogeneous data set consisting of a variety of container types with different thicknesses and particle shapes.

Table of Contents

1

Introduction

When inspecting medical containers of liquids in the pharmaceutical industry, containers that contain either cosmetic defects or foreign particles are rejected. A whole industry is built upon developing automated inspection machines for containers such as ampoules, vials, syringes and cartridges. A major part in the chain of inspection methods concerns machine vision, in which computer vision principles are used to inspect recorded images of the glass containers. Machine vision is used to find both cosmetic defects and foreign particles inside the liquid. This is done by analyzing two-dimensional side-view images of containers and rejecting these based upon impurities. However, a distinction between particles residing inside or outside glass containers is not possible due to the lack of a third dimension, and thus some glass containers with harmless dirt on the outside are unnecessarily discarded. An assessment of the extent of this problems states that roughly 0.5% of the rejected containers are rejected due to dirt on the outside.

Therefore, in this thesis we investigate and propose a method for detecting and positioning particles three-dimensionally based on a sequence of images instead of just a single image. The idea is to rotate a container with a known angular velocity and record images with known intervals. By tracking moving objects from frame to frame the path of movement is analyzed, and an estimate of the three-dimensional position can be calculated. In this way a distinction can be made whether an object is located inside or outside a container.

The thesis is conducted in cooperation with InnoScan A/S - a Danish industrial company specializing in inspection systems for automatic quality control. InnoScan was founded in 1988 based on a research project and is today one of the leading companies in constructing automatic inspection machines for pharmaceutical injectables. With their current system they are able to detect particles in both clear liquids and suspensions. When the viscosity of the liquid is low a method is used where a container is rotated and stopped abruptly to decide whether a particle is tied to the container or moving with the liquid. However, when the viscosity is high this approach fails since the rotating liquid is also stopped abruptly. Therefore, this thesis seeks to position particles independently of liquid viscosity by using a new proposed method.

1.1 Reading Guide

The thesis is structured as follows:

Chapter 1 continues with a problem definition explaining the problem in more details while relating the proposed method to the existing principles behind the inspection systems of InnoScan A/S.
Chapter 2 presents a background section covering related work on 3D imaging in the literature and the work carried out prior to the thesis in order to choose the overall method of positioning particles three-dimensionally.
Chapter 3 describes the experimental setup providing the means for recording image sequences of rotating containers. It also includes a description of a particle simulator and the acquired data set which is hand labeled and split into a training set and a test set.
Chapter 4 presents the first proposed algorithm consisting of particle detection, tracking, positioning and classification.
Chapter 5 discusses a real-time optimization of the first algorithm enabling inspection of up to 11 containers per second.
Chapter 6 seeks to improve the first algorithm by handling problems present in the tracking method. Two advanced tracking algorithms[1] are proposed and applied at our problem using a particle simulator.
Chapter 7 presents the results of the first algorithm as well as the two advanced tracking algorithms applied on the test set.
Chapter 8 discusses future work relevant in order to apply the proposed method in practice.
Chapter 9 finally presents a conclusion of the work as well as the results from the test set.

1.2 Problem Definition

InnoScan A/S is a Danish industrial company constructing and selling high-speed inspection machines for the can-maker and pharmaceutical industry. For the pharmaceutical industry they are using computer vision technology (CVT) to inspect liquids in glass containers such as ampoules, vials, cartridges and syringes for cosmetic defects and foreign particles. The glass containers are transported in a carousel where they are passing two inspection stations that consist of a mirror and line scan camera connected to a computer vision system. Inspection is performed by visually recording rotations of each glass container while it is passing one of the inspection stations at high-speed. The mirror is tracking the horizontal movement of the glass container while the line scan camera unfolds the container surface, allowing imperfections and particles to be detected in two dimensions. However, a distinction between particles residing inside or outside the glass container is not possible due to the lack of a third dimension, and thus some glass containers with harmless dirt on the outside are unnecessarily discarded.

Contribution *The purpose of this thesis is to apply 3D principles upon the image acquisition and analysis of the glass containers, thus possibly providing the ability to distinguish between particles inside and outside the containers.* It is the aim in the future to improve the existing CVT system to achieve an even better False Rejection Rate (FRR) than what is possible today.

[1] Multiple Hypothesis Tracking is discussed and applied by Mikkel Kragh Hansen and the Circular Trajectory Tracker is developed by Kim Bjerge

Goals The main goal of the thesis is ***to investigate and propose an algorithm that can detect and position particles inside or outside glass containers with clear liquid medicine based on a sequence of images.*** Extending this are the following two sub-objectives:

- To optimize the performance and accuracy of a proposed algorithm based on simulation and a test set of containers.
- To investigate a real-time realization of the proposed algorithm based on the requirement of inspecting 11 containers per second.

Approach The thesis work has been divided into six different phases, each building upon and narrowing the results of the previous phase.

Phase 1: Preliminary work and background

The first phase is carried out prior to the actual semester of the thesis. In this phase the contact to InnoScan A/S is initiated and a common understanding of the project and its associated problems is obtained. Possible methodologies are proposed and discussed on joint meetings, and initial experiments are performed on Aarhus University where relevant test equipment is also acquired. The outcome of this phase is the formulation of a problem statement including scenarios, goals and methodologies forming the basis of the thesis work. A number of selected methods will be discussed and compared. This background work is used as a foundation for the proposed method to be investigated.

Phase 2: Experiments

The second phase involves the conduction of a number of experiments. A test set of reference glass containers with liquids of different viscosities and particles will be produced by guidance from InnoScan. Different setups are tested by varying the test units, rotation speeds, illumination and camera acquisition parameters. The purpose of this phase is to narrow down the proposed methods in the first phase to only the most promising one.

Phase 3 : Analysis

In the third phase the proposed method is investigated further and research activities are carried out in order to study and assess related algorithms in other application fields. In this phase a simulated and physical model of the setup is constructed and relevant methods for detecting, tracking and positioning particles are explored.

Phase 4: First Design

The fourth phase deals with the development of a first algorithm combining detection, tracking and positioning of particles three-dimensionally in the glass containers. The phase builds upon the concepts explored throughout the analysis and the images recorded during the experiments.

Phase 5: Advanced Design

The fifth phase involves two subgoals improving both timing performance and accuracy. A real-time system is designed executing the proposed algorithm according to the specified performance requirements. Based on the results of the first proposed algorithm, possible improvements are investigated in order to solve potential problems.

Phase 6: Evaluation

In the last phase the proposed algorithms are evaluated and compared based on an acquired test set. Accuracy and timing performance are recorded and compared to the real-time requirements.

Scope The focus of the thesis will be primarily on image processing, particle tracking and real-time design of the algorithm. Issues regarding the physical camera- and illumination-setup including optics and reflection phenomena will thus be dealt with only to a limited extent. Best practice from InnoScan A/S concerning these practical issues will generally be applied. The investigation is limited only to focus on container rotation ignoring the horizontal movement of the carousel by eliminating the use of the mirror to track movement of containers.

As test units only cartridges and syringes containing transparent fluids will be used. Of these the focus will be on viscous fluids, where the particles do not move but instead tend to be fixed in the fluid when a steady rotation speed is reached. Particles down to 100 μm are examined, and rotation speeds of the container between 500-8000 rpm are carried out. Due to limited research facilities a CCD camera of the type Basler 20 fps is used with a resolution of 1624x1234 pixels. Recordings of 40 containers are obtained for training and final verification of the algorithm.

2

Background

Most machine vision systems have evolved over the last decade based on a single idea about how human vision works [1]. This is also the case for the pharmaceutical industry where certification standards are based on what the human eye is capable of inspecting.

Inspection methods are available today, such as machine vision, that is applied in the manufacturing industry for automated inspection. It is used in many different areas of the manufacturing process like parts inspection, assembly, warehousing and final quality inspection. In the process of manufacturing machine vision is important for the quality control of production, like for instance inspection of pharmaceutical products. Here quality assurance is built in as an integrated part of the manufacturing and production process. Computer vision is used together with advanced optics, mechanical systems and electronics. The goal for these sophisticated systems is to guarantee the product quality based on an automated inspection. Manufactured items are rejected or accepted based on acquired images and computer supported analysis.

The machine vision industry consists of companies that supply technology used in manufacturing as a substitute for the human vision function. This establishment is made up of suppliers that deliver techniques leading to decisions based on the equivalent functionality of human vision, but without any operator intervention. The common technology that serves as the infrastructure in building machine vision systems includes cameras, optics, frame grabbers and computers with image processing and analysis software.

In the pharmaceutical industry manual inspection is still (in some cases) performed for products where an automated machine vision system has not yet been applied successfully. It is one of these challenges we are going to address in this thesis for an automated quality inspection.

2.1 Inspection Today

InnoScan supplies turnkey inspection systems for the pharmaceutical industry that address a single specific application which is the focus of our work [2].

"InnoScan A/S, founded in 1988, specializes in high-tech inspection systems for automatic quality control. The company was founded with the aim of developing the best possible inspection machines for pharmaceutical injectables and has invented a patented method, which enables the customers to assure the quality of their products yet maintaining the highest possible yield." [2]

InnoScan's Computer Vision Technology (CVT) machine (figure 2.1) consists of a mechanical carousel that transports glass containers such as ampoules, vials, cartridges and syringes for

cosmetic defects and foreign particles. The machine is able to suppress the influence of air bubbles and still detect any kind of particles in pharmaceutical liquids contained in glass containers. The CVT machine has a high detection rate (DR) while at the same time retaining a very low false rejection rate (FRR). Two particle inspection stations perform a surface scanning of the glass containers while they are rotating. A mirror is tracking the horizontal movement of the glass container while a line scan camera unfolds the container surface. Images are transmitted to a cluster of computers that contain the image processing and analysis software. The images are processed and analyzed for cosmetic defects like damages to the tip or cap, and the fill level of the container is evaluated. Foreign particles in the liquid like dirt and impurities from the production process are detected, and even scratches and particles on the glass surface can be found.

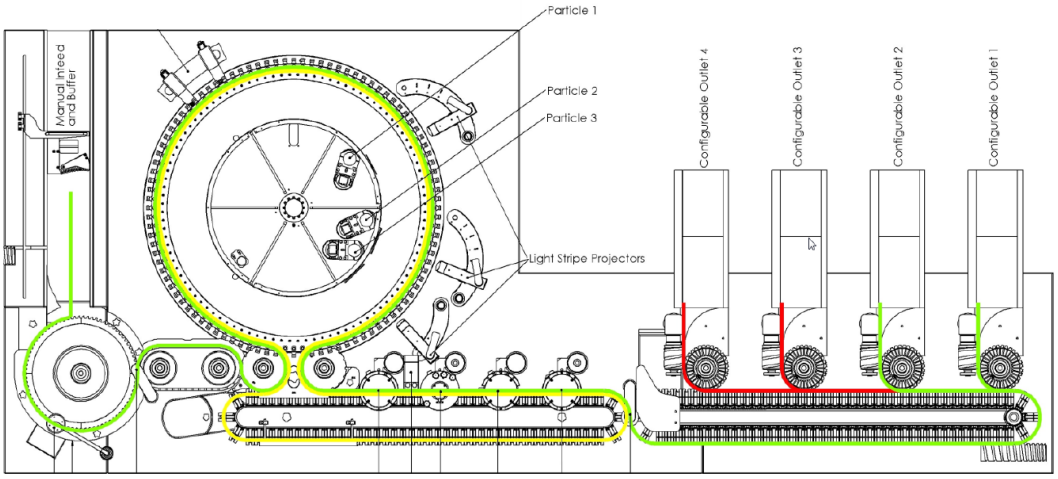


Figure 2.1: Flexible machine layout of a CVT machine including a carousel and 3 particle stations. The actual machine layout varies depending on the specific needs of a customer.

The image is processed through a number of processing steps. First the image is divided into different areas allowing a targeted search for the border of the glass container, the cap of the container, foreign particles and a tracking mark. A cup is supporting the bottom of the glass container where a point is marked for tracking one revolution of the container in the unfolded image. The area to be inspected for foreign particles is divided into zones in which particle blobs can be found. A number of different image processing filters can be configured for each zone: thresholding, vertical or horizontal filtering, edge detection, level separation and morphological operations [3] used for segmentation of particle blobs. The result is a list of properties for every particle blob found in the zone. Depending on an individual threshold for each zone the glass container is rejected if a particle blob exceeds a certain configured size.

The glass containers are rotated to ensure that particles will be moving inside the liquid of the glass container. The rotation speed varies between 500 - 8000 rpm depending on the product. Every 100 ms a new glass container is ready for surface scanning at one of the inspection stations depending on the speed of the carousel. The total handling and image processing time should not exceed this limit in order to fulfill the capacity of the CVT machinery. The sustained maximum capacity of 36.000 items per hour is achieved with a single-in-line machine containing one carousel and two particle stations. However, it is acceptable to have a high latency within the system as long as the capacity requirements are met. The design philosophy is to create a machine where the maximum number of inspections is performed with the lowest number of cameras and the more simple setting of illumination and optics to reduce complexity and increase reliability. Two methods are used for inspection:

Method 1: In this method the glass container is rotating and stopped just before the line scan of the glass surface is initiated. This abrupt deceleration of the rotation introduces a force on the particles floating in the liquid so that they will be moved around. The image is transmitted to the CVT inspection software where it is filtered and segmented searching for particle blobs. Only particles that are moving will be detected and the surface of the glass container will not be scanned. The goal is to find particles that are moving in the liquid. A threshold of the particle blob size will be used for rejection of the container.

Method 2: This method is used for glass containers with liquid of high viscosity. For products of high viscosity it is more difficult to make the particles move within the glass container. Therefore, instead of stopping the rotation before the inspection station, the container continues to rotate during line scan. In this case both floating particles and stuck particles in the glass container will be seen in the unfolded image. By comparing images from two different inspection stations it is possible to identify a particle blob and evaluate the position of it. If the particles are detected at different positions at the two stations the method assumes that the particle is floating. If the positions are the same it assumes that the particle is adhered to the glass surface of the container. However, a distinction between particles residing on the inside or outside of the container is not possible.

There are a number of different types of liquids to be inspected depending on the CVT machine. A CVT machine is constructed and configured to handle only certain types of pharmaceutical products. The products could be clear liquids or suspensions with low or high viscosities. A suspension or slurry is a mixture of a solid and a liquid in which the solid is not soluble in the liquid. Suspensions are typically handled by method 1 with variation of the rotation speed before inspection to ensure proper consistency of the suspension. Light is projected behind the glass container and in front of the camera for inspection of clear liquids. For suspensions the light is projected in front of the container with a small angle relative to the camera pose.

The problem in the current inspection system of InnoScan, however, is that no methods can handle situations where particles are stuck in the same three-dimensional position inside the containers regardless of the rotation speed. This happens for liquids of extremely high viscosities. In such cases, a distinction whether a particle is located inside or outside the container is impossible with just 2D images. Therefore, more advanced computer vision methods incorporating 3D imaging methods must be applied.

2.2 Related Work on 3D Imaging

Industrial quality inspection is an important application domain for 3D computer vision methods [4]. Traditionally, industrial inspection systems primarily rely on two-dimensional detections like the current inspection machines manufactured by InnoScan. These systems typically rely on the detection of point and line features for pose estimation or extraction of blob features from black/white images.In order to detect production faults more reliably and robustly, more advanced vision-based systems employ three-dimensional methods. The automobile industry is one of the important domains where a number of 3D methods have been applied for production inspection. A typical application is to check for imperfections and missing parts in a large work piece comprised of smaller parts such as plugs, cables, screws and covers mounted on a car engine, or three-dimensional reconstruction of a rough metallic surface.

3D imaging sensors are generally operating by projecting (active) or acquiring (passive) electromagnetic energy onto/from an object followed by recording the transmitted or reflected energy.

3D imaging sensors measure the 3D contour of an object. In our case we are looking for particles on the inside and outside of an object. However, most of the 3D imaging sensor do not address the problem of finding a 3D particle position inside a transparent object. Also, the use of high-energy x-rays and laser projections should be avoided when applied at inspection of medical products. Here the risks of heating or even damaging the pharmaceutical products are too high.

In [5] the state-of-the-art and applications of 3D imaging sensors is presented. The most relevant 3D imaging methods will be described and discussed in the following. The methods are evaluated with relevance related to inspection of pharmaceutical liquid container products.

Laser Triangulation. In this category single-point triangulators and laser strips exist. They are all based on the active triangulation principles as described in [6]. Laser triangulation measures the distance to the object surface by active projection and triangulation. The laser source generates a narrow beam and the reflected light is measured using simple geometry. A high accuracy can be achieved in measuring distances and the surface of an object. The application of laser triangulation on our problem is doubtful, especially for finding particles inside the liquid of the glass container. Here reflection is expected to happen on the glass surface and not inside the liquid. We would in general avoid using lasers since the risk of damaging the pharmaceutical liquid is too high.

Structured Light. Commercial three-dimensional surface reconstruction systems for industrial quality inspection purposes are based on active scanning techniques such as projection of coded structured light [7]. In [8] a fast 3D profilometer is presented based upon the projection of a single fringe pattern and camera calibration, enabling a 3D object representation with a lowest uncertainty of 153 μm for an object of size 220x200 mm.

The 3D scanning profilometer is often the chosen solution for most micro medical fluidic applications [9] like determining the surface characteristics such as roughness and micro geometry or the thickness of a transparent coating for medical devices. They capture the topography in a line pattern with a resolution of 0.5 μm and a speed of 200 mm per second.

However, these methods will be hard to apply for inspection of particles inside the glass container since structured light will not be reflected in the same way as on a solid surface. These methods relate to the structure of a surface and not small particles attached to the surface or inside a glass container with liquid.

Stereo Vision. High accuracy is achieved with stereo vision in [10] for computing a depth map. Since stereo vision is a passive method it will not harm the pharmaceutical product in form of heating. Stereo vision has a higher complexity than using a monocular setup but it also requires calibration of the vision setup. A single stereo image is not sufficient to cover the whole surface of the glass container and the method would require further investigation concerning accuracy in finding a particle 3D position inside the glass container. This is one of the proposed methods applied on our problem and further investigated in chapter 2.3.

Time of Flight Surface range measurements can be made directly using the radar time-of-flight principle. The emitter unit generates a laser pulse which impinges onto the target surface. A receiver detects the reflected pulse and the round-trip travel time is measured. A few amplitude and frequency modulated radars have shown promise for close range distance measurements [11]. The application of time-of-flight on our problem is doubtful, since reflections will happen mainly on the surface of the container, thus preventing detection of particles inside the glass container.

Shape from Shading In [12] a general framework for three-dimensional surface reconstruction is presented by fusion of shading and shadow features. Here the well-known method *shape from shading* is combined with shadow features applied to the three-dimensional reconstruction of metal sheet and raw cast iron surfaces in the context of industrial quality inspection. This approach applied to our problem would require a front light and it would only be able to find particles creating a shadow on the surface of the glass container.

Depth from Focus. A monocular pose estimation technique is described for the automobile production environment by Barrois and Wohler [13]. For automobile inspection it is often difficult or even impossible to utilize stereo camera systems, since they have to be recalibrated regularly. A framework is presented combining a number of methods like photometric, polarimetric, edge and defocus approaches. The experimental evaluation of the method achieves a resolution of 0.4 mm.

Especially the depth from defocus method has turned out to be a useful instrument for estimation of object depth even as a real-time sensor technique [14]. Depth from defocus yields a relation between the amount of defocus in the scene and the distance to the camera, allowing estimating a depth value from defocus for each pixel if texture is present. The method requires two pixel-synchronous images acquired with respectively a small ($f/8$) and large aperture ($f/2$).

The depth from defocus method would be interesting to use for inspection of glass containers with liquid for pharmaceutical products since only a single camera is required. However, there are many challenging problems in applying the method. Two similar images could be acquired by recording one image before and one image after one revolution of the glass container assuming that particles for inspection will be visible in the same position. The camera aperture should then be changed while the container is rotating in order to achieve different focuses in the two similar images. If focus is kept the same between acquiring two images of the rotating container, particles on the front side will have a different focus compared to when they are recorded on the back side. In both cases we will have two images where the particle would be in defocus in one of them.

Texture in the image background behind the glass container is hard to achieve with background illumination. One of the challenges is also the size of particles to inspect. Here it is doubtful if a resolution down to 100 μm can be achieved. Due to the number of challenging problems, applying this method on our problem has not been investigated further.

Particle Image Velocimetry. Particle Image Velocimetry (PIV) and Particle Tracking Velocimetry (PTV) are techniques used for estimating the flow velocity of particles in fluids or gases (e.g. air) [15]. Most often so-called tracer particles are used in order to visualize some of the particles, and on the basis of these an estimation of the flow of the whole liquid/gas is found. The PIV setup includes a laser that via optics and a mirror emits a light sheet (plane of light) directed along the direction of flow. This light sheet illuminates some of the particles in the liquid/gas so that they are visible for a camera placed in an angle perpendicular to the laser. The laser is pulsed so that particles can be found in two images recorded immediately after each other. Here cross-correlation can be used to estimate the velocity of different windows (called "interrogation areas") by matching these between the two frames. The typical spatial resolution of PIV is down to 1 mm. However, advances have increased this resolution to detect particles down to 1 μm with Microscopic PIV (μ PIV).

The application of PIV/PTV in this thesis is quite different from the methods described above. In this project we are not interested in estimating the velocity of the individual

particles. Rather we have some knowledge about the (angular) velocity and are interested in estimating the position. Since the particle concentration is extremely low in this problem, PTV is the most similar method. An important parameter in PIV and PTV concerns the duration of the illumination pulse (from the laser). A short pulse is normally needed in order to avoid blurring in the image. In our setup, on the other hand, a long pulse is needed in order to capture and illuminate the potential particles with certainty.

2.3 Proposed Methods

Inspired by the related work presented above, a number of ideas have been proposed, investigated and evaluated in a pre-study project. The purpose has been to choose the most promising method for further in-depth investigation in the thesis. A test setup has been made with a mechanical supporting device, laser and CCD cameras. The mechanical supporting device is attached to a motor with encoders and electronic control. This electronic control is connected to a CAN bus from where a plan can be downloaded that tells how to rotate the glass container. Due to limited research facilities only CCD Basler cameras are used with a resolution of 1624x1234 pixels (20 Hz) and 658x492 pixels (100 Hz). The cameras are connected to a Windows PC with a high speed Ethernet connection with frame grabbers and software for camera adjustment and image and video recording.

In appendix A four different ideas with the purpose of detecting particles inside or outside the glass containers are described. They all address the InnoScan method 2 by recording images while the glass container is rotating. These ideas are evaluated in the pre-project with inputs and comments from the engineers of InnoScan with the purpose of selecting the best suited method.

1) **Line Scan Cameras:** This method concerns using the existing particle stations from the CVT machinery at InnoScan with mirrors and line scan cameras.

2) **Stereo Vision:** This method explores how the principles from stereo vision could be used.

3) **Camera with Laser Sheet:** In this method a laser sheet of light is projected onto the glass container with a CCD camera to record particles passing the laser sheet in a dark room.

4) **Video Sequence of Rotation:** This method records a video sequence of images sampled with a synchronized speed relative to the rotation speed of the glass container. From the video sequence it is our hypothesis that the moving particles can be tracked and that the three-dimensional particle positions can be estimated.

2.4 Selected Method

The question is now: *Which method should we choose to realize and work with in this thesis?* It would be too comprehensive to analyze and implement all the described methods. We would like to work with the method that is the most promising and realistic for InnoScan to integrate into a CVT inspection machine. This should ultimately solve the challenging problem of determining if a particle is located on the inside or outside of a glass surface.

The line scan method would be the easy solution for InnoScan to integrate and does only require new software. The method requires more experiments and a physical model being able to

evaluate whether it can be used. However, in our research facilities, we do not have mirrors and line scan cameras available.

The stereo vision method requires two cameras and precise calibration, in contrast to all other described methods. A single stereo image is not sufficient to cover the whole surface of the glass container. Here we need to add mirrors to follow the horizontal moving glass container, which will be a very challenging mechanical problem to solve.

The laser method also needs a mirror for both the camera and the laser to follow the horizontal movement. It requires a dark room which would be difficult to realize in the CVT inspection machine. Our initial experiments seem only to detect some particles on the surface of the glass container and fails to detect particles in the liquid due to reflection of light.

The last method using a video sequence of a rotating cylinder only needs one camera and a mirror. However, it is a slow method since we need a sequence of images for a number of revolutions. The method is technically challenging where advanced image segmentation, filtering and tracking could be applied to estimate the position of the rotating particles. This is the method we have chosen since it is the most promising but also challenging, agreed upon with both InnoScan and our supervisor.

Some of the major questions for this method are: *How accurate is it possible to estimate the particle position on the inside or outside of the glass container by segmentation, tracking and positioning? Can the image processing algorithm be implemented efficiently to follow the video recording speed?*

3

Experimental Setup

In this chapter we will describe the theory that is essential to understand observations of the experiments we have performed on a selected number of glass containers containing different liquids and particles. For this first prototype a machine vision setup is made. The setup is illustrated in figure 3.1 and 3.2. A glass container with particles is mounted in a mechanical setup, where the container can be rotated with a given frequency. The particles are illuminated with back-light through a diffuser, and a Basler camera with a rectilinear lens is used to record the rotating particles.

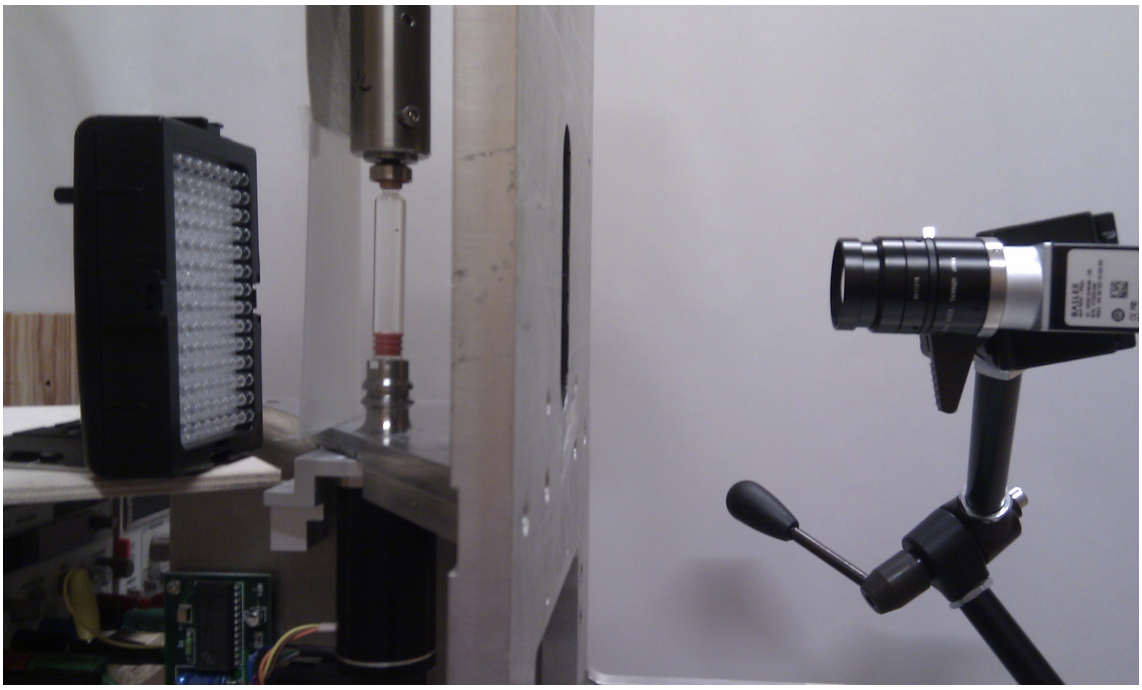


Figure 3.1: Setup for verification of the proposed method. A glass container (cartridge) is mounted in a mechanical supporting device illuminated by a back-light LED-array and diffuser. The camera records a sequence of images while the container is rotating.

The accuracy of the mechanical rotation device will be examined. The dynamic behavior of particles floating inside the liquid of the container will be discussed with the purpose of exploring whether particles are fixed or floating relative to the rotating container. The machine vision system

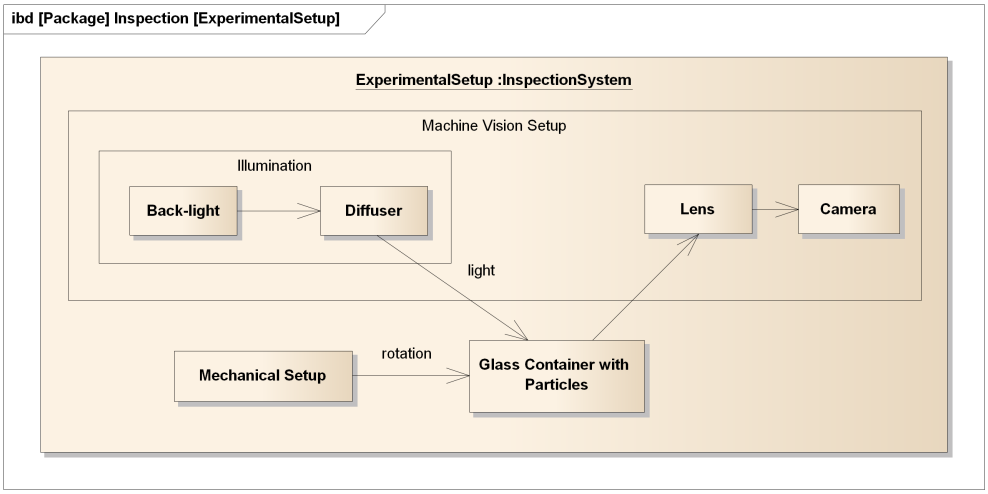


Figure 3.2: Internal block diagram for experimental setup of inspection system.

will be described in detail with the purpose of finding the optimal camera-, lens- and illumination-settings being able to inspect particles on the inside or outside of the glass container. When the glass container is filled with liquid, refraction of background light will have an impact on the observed particles. They will be magnified and only a portion of the particles will be visible on the back side. This observation will be explained based on fundamental optics. Finally test samples of cartridges and syringes with different liquids and particles used in our experiments will be described in detail followed by a presentation of a simulator to produce an artificial data set.

3.1 Mechanical Setup

The glass container is rotating by means of a mechanical supporting device, motor and electronic control as seen in figure 3.1. A microprocessor controls the rotation speed in terms of revolutions per minute and angular acceleration. A plan is downloaded to the microprocessor over a CAN bus interface from a connected central computer. The plan can be written to vary the rotation speed, acceleration and time of rotation- and stop-sequences. An encoder gives feedback to the microprocessor, although this is currently not used in the electronic motor controller to control the speed. The encoder signal could in future designs be used to either inform about the actual rotation speed or add a controlled feedback loop (PID controller) to the motor to ensure a more accurate speed.

We have observed a constant relative error of the actual rotation speed compared with the planned speed in the area of 0.17 %. As an example setting the rotation speed to 1200 or 2400 rpm we measure actual values of 1198 and 2396 rpm. This error would vary between devices in the carousel of the CVT machine. *This apparently small error does have an impact on the accuracy* of the estimated particle positions which is essential in our proposed method. Alternatively the image processing algorithm could account for this tolerance in precision by computing the rotation speed based on a known fixed tracking marker at the bottom of the mechanical supporting device. This method will add even more complex image processing, and it is doubtful if it would be sufficiently accurate. In our experiments we have manually taken into account the constant relative error of the rotation speed.

The glass container does not rotate straight in relation to its center axis. This applies in particular the area at the top or bottom of the rotating container where it is attached to the fixture of the mechanical device. The phenomenon is visible in the captured images as fluctuations of the illuminated glass container at the area of the borders. It also introduces an error in the actual measured 2D particle positions.

3.2 Machine Vision Setup

In this section we will present the theory and practice for creating a machine vision setup that will be able to fulfill the specification for detecting particles inside and outside the glass container. We have used a guide for machine vision made at Aalborg University [16] as a basis for justification and selection of camera, lenses and illumination of the glass container. Our goal is to evaluate the precision that can be achieved with the equipment already available in our research facilities.

Two types of the Basler cameras are available with a Gigabit Ethernet connection to a PC running software for collecting video streams or single images simultaneously from a setup with up to four cameras. It is possible to control parameters of the cameras like exposure time, color format etc. Two types of lenses are available with an adjustable focal length of maximum 16 and 25 mm. In the following, the configuration of the camera, lenses and illumination is described. The refraction of light passing through air, glass and liquid will be explained, which will have an impact on the visible particles, seen inside or on the back side of the container.

3.2.1 Camera

We will focus on area scan cameras that provide a full scan of the object compared to a line scan camera where a single line of pixels is scanned at a high rate up to 18 kHz for the InnoScan CVT machinery. A frame grabber will then compose an image based on pixel lines from the line scan camera. The number of pixels in a line scan is equal to the width of the image. The image height will then depend on the number of frames, and by rotating the container for one whole revolution it can be seen in one image. Line scan cameras are especially suited for inspection on conveyer belts where objects are constantly moving [17]. Line scan cameras are also suited for inspecting circular objects rotating with high speed, where a high resolution is required [18]. This is why InnoScan are using this solution in the CVT machinery today. Also, a line scan camera is more efficient in utilizing the whole image area compared to an area scan camera. If the object is not square-shaped only a part of the image pixels from the area scan will be of interest for image processing. In figure 3.3 a cartridge of 66x10.9 mm is shown. The image is cropped such that only 33% is used for inspection.

A number of factors influence the image quality when using an area scan camera. Some of these are listed in [16]. Image resolution addresses the number of pixels in the horizontal and vertical direction which is purely dependent on the camera sensor. Spatial resolution depends on the optical magnification of the camera and is expressed in mm/pixel. Feature resolution is defined as the minimum particle size that can be detected reliably in the image. In our case feature resolution is important in determining how small particles can be detected.

In the following description we consider particles represented as circular blobs recorded in the camera image. Figure 3.4 illustrates the geometry involved in being able to inspect small particles on the glass surface.

The field of view (FOV) is the area for inspection that the camera needs to acquire. In our case the FOV is the glass container with a diameter in the image of D_c. The depth of field (DOF) is the depth of the scene to be inspected – in this case the diameter of the glass container (D_c).

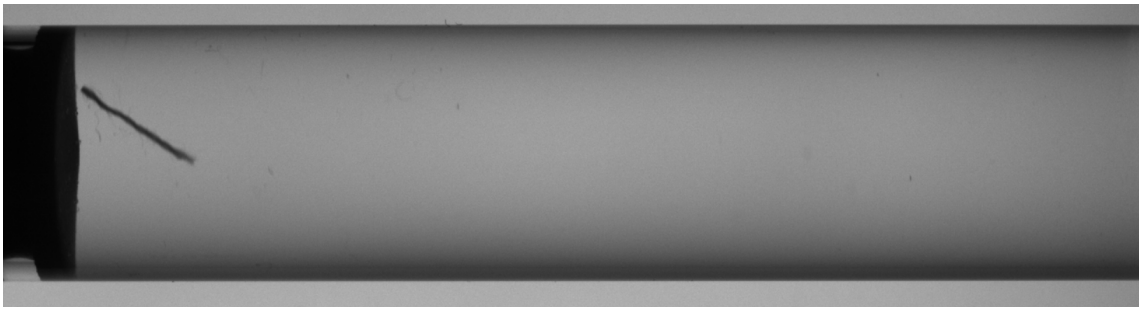


Figure 3.3: Image of cartridge with insulin recorded by a Basler (acA1600-20gc) camera at 20 fps. For visualization purposes the image is rotated 90°. The cartridge is rotating with a speed of 1186 rpm and the camera exposure time is set to 300 μs. A fiber particle can be observed floating in the liquid just above the rubber stopper at the bottom of the cartridge.

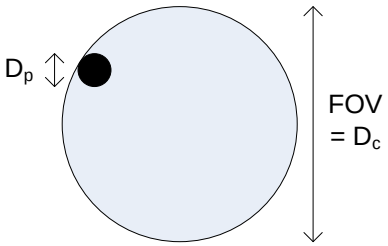


Figure 3.4: Cylindrical container with diameter D_c including a particle with diameter D_p.

The smallest particle to be detected has a diameter of D_p.The necessary camera resolution can be computed given a minimum size of the particles. As mentioned above, for simplicity particles are assumed to be circular. To achieve a feasible resolution we need a minimum of two pixels per particle. To calculate the minimum image resolution (R_i) we need to divide the FOV by the smallest diameter of the particle:

$$R_i = 2 \cdot \frac{FOV}{D_p}$$

assuming there is a perspective transformation from the world coordinates to the image plane that preserves similarity of object dimensions. For a cartridge with the dimensions of 75x10.9 mm (including 9 mm for the height of the cup) we need a camera resolution of 1500 pixels for a particle with a minimum size of 100 μm. In our test setup the Basler camera (acA1600-20gm/gc) has a resolution of 1624x1234 pixels. In our recordings the camera is rotated 90° such that the cartridges are oriented with the height parallel with the horizontal (1624 pixels) orientation of the image. Thus, the smallest detectable particle is approximately 92 μm.

3.2.1.1 Exposure Time

When the glass container is rotating the exposure time must be adjusted to reduce blurring caused by fast moving particles. The distance we allow the particles to move depends on the smallest particle we would like to detect. The speed of a particle traveling on the cylinder surface can be found by

$$v = \pi \cdot D_c \cdot f_{cyl} \tag{3.1}$$

where (f_{cyl}) is the rotation frequency of the cylinder in Hz and D_c is the outer diameter of the container. We can compute the desired exposure time (E_t) from equation 3.1 and the minimum

accepted movement of a particle, (Δs), during image acquisition. The required exposure time can be found to be:

$$\Delta s = v \cdot E_t = \pi \cdot D_c \cdot f_{cyl} \cdot E_t \tag{3.2}$$
$$\Leftrightarrow E_t = \frac{d_m}{f_{cyl} \cdot D_c \cdot \pi}$$

For a cylinder rotating 1200 rpm with a minimum particle size of 100 μm and a maximum allowed movement corresponding to twice its own size ($\Delta s = 200\mu m$), the exposure time can be calculated to:

$$E_t = \frac{200\mu m}{{}^{1200}/_{60s} \cdot 10.9mm \cdot \pi} = 292\mu s$$

3.2.1.2 Sampling

The method used for estimating three-dimensional positions of particles requires a sequence of images for each container while it is rotating. A number of frames are needed, equally distributed around each container. In order for the method to work properly a steady state for all particles is needed, such that they are rotating with the same frequency as the container. For more information about particles floating inside the liquid of the container see appendix B.2. However, in order to achieve such a steady state, the container must be rotated with a high frequency, e.g. 1000 rpm. For a camera to obtain e.g. 10 frames at this rotation speed, a frame rate of 167 fps is required. High speed cameras are available on the market that are able to record at high speeds with a high resolution. For instance the CMOS camera pco.dimax HC [19] provides a resolution of 2000x2000 pixels at a speed of 2277 fps. In our research lab, however, only Basler cameras with lower frame rates are available.

Therefore, instead an intentional undersampling (or bandpass sampling) is performed. In order to get n equally distributed images around a container, the container rotation frequency f_{cyl} can be expressed mathematically as:

$$f_{cyl} = f_s \cdot m \pm f_s \cdot \frac{1}{n} = f_s \cdot (m \pm \frac{1}{n}) \tag{3.3}$$

where f_s is the camera frame rate, n is the number of frames and m is a non-negative integer. The formula states that the container is rotating with a frequency that is a multiple, m, of the camera frame rate plus/minus a fraction of this rate. The result is n equally distributed images around the container. If f_s is 20 Hz, m is set to 1, n is set to 100 frames and we want a frequency less than the multiple (using minus), we get a rotation frequency of:

$$f_{cyl} = 20\,\mathrm{Hz}(1 - \frac{1}{100}) = 19.8\,\mathrm{Hz} = 1188\,\mathrm{rpm}$$

The undersampling theorem states that

$$\frac{2f_H}{k} \leq f_s \leq \frac{2f_L}{k-1} \tag{3.4}$$

where k is an integer satisfying $1 \leq k \leq \lfloor \frac{f_H}{f_H - f_L} \rfloor$ and f_H and f_L are the upper and lower bandedge frequencies [20, p. 56]. The above mentioned steady state assumption states that all movement in the images between frames is happening with a single frequency equal to the container/cylinder rotation frequency, $f_{cyl} = 1188\,\mathrm{rpm}$. Small variations and uncertainties in f_{cyl}

result in a narrow frequency band around the nominal rotation frequency. Rewriting equation 3.4 and using $k = 2$ gives:

$$f_H \leq \frac{k \cdot f_s}{2} = \frac{2 \cdot 20\,\mathrm{Hz}}{2} = 20\,\mathrm{Hz} = 1200\,\mathrm{rpm}$$
$$f_L \geq \frac{(k-1) \cdot f_s}{2} = \frac{(2-1) \cdot 20\,\mathrm{Hz}}{2} = 10\,\mathrm{Hz} = 600\,\mathrm{rpm}$$

This states that all motion with frequencies between 10 and 20 Hz can be reconstructed perfectly, whereas frequencies outside this range introduce aliasing.

3.2.1.3 Acquisition

When acquiring images using the described method with a rotation frequency of the container almost synchronized with the camera frame rate, the precision of the acquisition time is subject to strict demands. A small variation in the frame rate of 20 Hz can result in a distortion of the particles in the images, resulting in wrong estimations of their three-dimensional positions.

Normally this problem could be dealt with by recording the time stamps of the images at the end of exposure time. However, the setup available at our lab only allows time stamps at the end of data storage and with a precision of 1 ms. Using a variation of equation 3.2, this corresponds to a particle movement of:

$$\Delta s = v \cdot t_{cam} = \pi \cdot D_c \cdot f_{cyl} \cdot t_{cam} = \pi \cdot 10.9mm \cdot \frac{1200}{60s} \cdot 1ms = 649\mu m$$

which is approximately six times the size of the smallest detectable particle. The precision of the time stamps is therefore not sufficient to compensate for potential inaccuracies in the camera frame rate. Instead, a dedicated computer for recording images with high frame rates on a solid state drive is used, so that a constant frame rate of 20 Hz can be assumed. In a commercial product, however, a solution using more accurate time stamps would be preferred.

3.2.2 Lens

In this section we will present the theory and calculation for finding the optimal lens parameters like focal length, exposure time and k-stop number for the aperture of the camera system.

3.2.2.1 Focal Length

For a thin lens in air, the focal length is the distance from the optical center of the lens to the sensor plane of the camera as illustrated in figure 3.5.

The focal lengths are usually specified in millimeters. The model assumes that the working distance L to the object is at least ten times the focal length. The focal length and field of view of a lens are inversely proportional. In general the focal length is fixed, but it is also common that the working distance is fixed and the focus point can be adjusted. Lenses with a short focal length (less than 12 mm) produce images with a significant amount of radial distortion. We would like to avoid rectification of the images to compensate for such a distortion by expensive camera calibration. From a simple model using the similar triangles of the camera geometry we have:

$$\frac{S_s}{f} = \frac{FOV}{L}$$

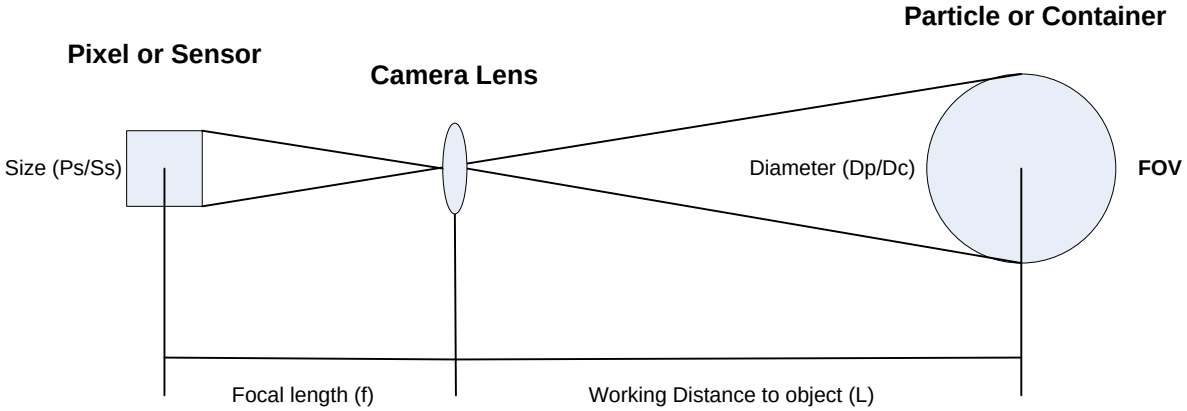


Figure 3.5: Optical model for a particle recorded in pixels by a camera system.

where S_s is the camera sensor size, f is the focal length, FOV is the field of view and L is the working distance. The camera magnification factor β can be expressed as the relation between the camera sensor size and the field of view aligned horizontally or vertically with the glass container:

$$\beta = \frac{S_s}{FOV} \tag{3.5}$$

A more accurate equation [21] should be used for calculating the needed focal length based on the magnification factor:

$$f = L\frac{\beta}{\beta + 1} \tag{3.6}$$

In our test setup we use an optical lens with an adjustable focal length, and the working distance can also be varied. The Basler camera has a sensor size of 7.16x5.44 mm^2 and a pixel size of 4.4x4.4 μm^2. Recordings are performed with a working distance of 180 mm. We can calculate the focal length given a preferred working distance and FOV of 45 mm, only viewing the glass surface of the container. Two types of lenses are used in the experiments with adjustable focal lengths of 1.4-16 mm and 1.4-25 mm.

In our case with the cartridge oriented horizontally we have a magnification factor of $\beta = {}^{7.16}/_{45} = 0.1591$. Here a focal length of $f = \frac{180mm \cdot 0.1591mm}{1.1591mm} = 24.7mm$ is needed. The working distance is approximately 7.3 times the focal length and thus close for the thin lens model not to hold. However, we assume that it is sufficient.

3.2.2.2 Depth of Field

The depth of field (DOF) has an impact on focus for particles in all positions inside and outside the glass container. We would like to have particles in focus on the front of the container and still keep an optimal focus on the back side seen from the camera position. The aperture has an impact on the depth of field. A minimized aperture increases the depth of field, and by increasing the aperture the depth of field is reduced. A small aperture requires more light or a longer exposure time. An f-stop number, k, which forms an adjustable size on the lens, controls the opening of the aperture and can be computed by [16]:

$$k = -\frac{(\sqrt{DOF^2 + L^2} - L)f^2}{2 \cdot DOF(f - L)P_s} \tag{3.7}$$

where P_s is the pixel size. In our case we have a desired depth of field equal to the container diameter of 10.9 mm. We have an f-stop number of (all units in mm):

$$k = -\frac{(\sqrt{10.9^2 + 180^2} - 180)24.7^2}{2 \cdot 10.9(24.7 - 180)0.0044} = 13.5$$

3.2.3 Illumination

The quality of lighting is a critical aspect for creating robust and timely vision inspection. The way an object is illuminated has an impact on the quality of the image and how easy it will be to handle by image processing like enhancement, noise reduction and segmentation. An optimal illumination will eliminate any costly preprocessing of the image allowing it to be segmented without any pre-filtering. Here we have experienced that a combination of illumination and camera settings is important for a high quality image.

The basis for the choice of lighting is determined from the basic characteristics related to light direction and whether it is possible to highlight desired features and reduce present noise factors. Light direction forms the foundation for interaction between the illumination and object to be inspected given the reflection, transmission and absorption properties. The goal with illumination is to maximize the contrast for the features of interest and minimize the contrast elsewhere. Considerations for the robustness related to production environment and ambient light must be included in a final setup. In our case we have a glass container with a shiny surface. Here we would like to reduce the reflection of light from the glass surface.

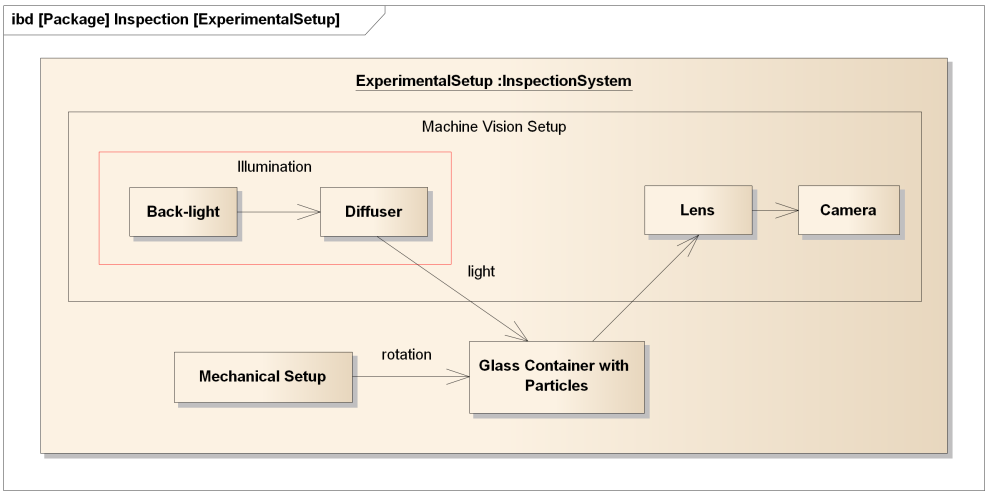


Figure 3.6: Internal block diagram for machine vision setup with illumination of glass container.

Back-lighting is chosen with a diffuse illumination (figure 3.6) since the primary objective is to inspect edges and contours of the container and the particles. The diffuse light ensures a uniform illumination of the container and thus a white uniform background of the image without any texture. The back-light ensures a light field where the contour of the container will be visible and where no light will be reflected. The influence of production and ambient light will be reduced with a high intensity of the back-light. The contour of particles will then be seen as dark spots as illustrated in figure 3.3. A white LED-lamp and a diffuser is used in our test setup as background lighting. LED-lamps are the standard method used for illumination in machine vision [16, 22]. It is homogeneous and has a long life time with resistance towards process disturbance. For inspecting glass bottles care must always be taken to shield a glass object from ambient light

during inspection. Failure to do so will result in poor image quality with highlights occurring in a host of unexpected places [22, p. 1203]. Therefore all our experiments are performed in a shielded room without any ambient light to cause false reflections.

To improve visibility of the glass surface and edges of the container a diffuse dome light could be used as illustrated in figure 3.7. It is a very effective lighting for curves and specular surfaces. In combination with the back-light this approach would properly improve the images of glass surfaces for inspection.

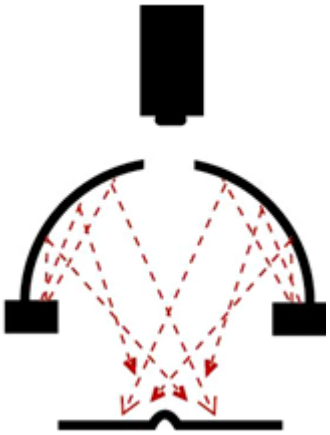


Figure 3.7: Dome diffuse light for front illumination of an object.

3.2.4 Setup Summary

A number of different camera, container and particle parameters can be selected in order to find the setup values for focal length, f-stop number, minimum particle detection size and exposure time. Table 3.1 shows the parameters of a number of selected cameras that are available from suppliers, used by InnoScan and present in our research facilities. The table is created based on inspection of cartridges with the dimensions of 66x10.9 mm, a working distance of 180 mm. A limited field of view of 45 mm is used in order to achieve a maximum camera resolution by only viewing the glass surface skipping the view of the tip and container bottom. The minimum particle size is calculated for a horizontal orientation og the cartridge within the image.

Camera Type Basler	**Resolution (H x V pixels)**	**Frame Rate** [Hz]	**Pixel Size** [μm^2]	**Sensor Size** [mm^2]	**Particle diameter** [μm]	**focal length** [mm]	**f-stop number** k
acA645-100gc	658x492	100	9.9x9.9	6.52x4.89	137	22.8	5.0
acA1600-20gc	1624x1234	20	4.4x4.4	7.16x5.44	55	24.7	13.5
acA2000-50gm	2048x1088	50	5.5x5.5	11.26x5.98	44	36.0	24.8
acA1000-120km	1024x1025	120	5.5x5.5	5.63x5.63	88	20.0	6.9

Table 3.1: Basler Area Scan (acA) Camera types

From the table we see that in theory it should be possible to detect a particle in the image with the cameras acA1600, acA2000 and acA1000. The smallest particle of 100 μm will then be found in the image plane covering at least 2x2 pixels. The exposure time should be set to 300 μs for a rotation speed of 1200 rpm to ensure a sharp image of the particle. Illumination with a diffuse background light should be used to create a white uniform background in the image. Particles will be viewed as black blobs in the images.

3.3 Glass Container with Particles

The requirements for the glass containers used in our experiments will be specified in this section. We are limited to evaluate our methods on cartridges and syringes that fit the mechanical supporting device in our research facilities. Cartridges of the dimensions 66x10.9 mm with a glass thickness of 0.85 mm are used for our initial experiments. In our final experiments this is accompanied by syringes of the dimensions 56x8.5 mm with a glass thickness of 0.80 mm. The minimum particle diameter to be detected is 100 μm. Particles could have different sizes and shapes. Inside the cartridge we have used liquids with viscosities between 0.91 and 1200 $mPa \cdot s$ (millipoise) corresponding to the viscosity of water and glycerol at $25°C$. The densities of water and glycerol are respectively 1000 and 1261 kg/m^3. In some of our experiments pieces of aluminum (2700 kg/m^3) are used as particles with a density higher than glycerol ensuring particles to be detected at the bottom of the container.

The particle velocity relative to the rotating container depends on parameters like viscosity, density, gravity and rotation speed of the container. We assume that particles are not moving relative to the rotating container according to theory and observations in appendix B section B.2.

If an object is transparent and background lighting is used as described in chapter 3.2.3, refraction will occur [21]. Due to difference in the refraction index of air, glass and liquid the container will behave like a lens. Particles will be observed as distorted and magnified. In appendix B section B.1 a model is described that corresponds with our observation. We are not able to see all particles on the back side for a cartridge filled with liquid. Particles are hidden due to refraction and their 2D position and size will not be correct since they are magnified due to the lens effect. For this reason we have decided not to use particles from images when they are observed moving on the back side of the rotating container.

3.4 Experimental Data Set

Overall, three different data sets are used for the experiments. An initial data set consisting of 6 different cartridges are used for developing the algorithms and choosing between different alternatives. This data set consists of the containers labeled A-E and is described thoroughly in appendix D table D.1. Another data set consisting of 40 containers (F-J) is divided into two separate sets. One is used as a training set with 20 containers (table D.2) and one is used as a test set with the remaining 20 containers (table D.3). The division of the 40 containers is conducted based on their nature, such that half of the containers labeled F are put into the training set and the other half is put into the test set etc. In this way the training and test sets are independent but still represent the same types of challenging containers.

Generally the data set is constructed to represent a wide range of challenges to the system. Some containers have particles tied to the inner and outer surfaces of the glass, others are empty or have particles floating inside the liquid. Different liquids and different materials and sizes for the particles are used. A short description of the different containers is listed below. For a complete description of individual containers, see appendix D.

- **A-E** : 6 cartridges containing glycerol/insulin with dots and particles on the inside and outside of the glasses
- **F** : 12 cartridges containing insulin with particles inside liquid from a numbered test set provided by InnoScan

- **G** : 10 cartridges containing insulin marked with black or white dots on the outer glass surfaces
- **H** : 11 syringes containing gel and particles provided by InnoScan (numbered according to specification)
- **I** : 4 syringes containing gel marked with black dots on the outer surfaces (numbered according to specification)
- **J** : 3 cartridges containing glycerol marked with dots and particles

3.4.1 Recordings

For all containers in the training set and test set a rotation speed of 1186 rpm is used to capture in total 87 frames. As described in chapter 3.2.1.2 this corresponds to 87 equally distributed images around each container due to undersampling. For all images the exposure time of the camera is set to 300 μs. With a frame rate of 20 Hz, recording 87 images takes $\frac{87\,frames}{20\,frames/s} = 4.35s$.

In order to approach the desire of handling 11 containers every second, this time must be lowered significantly. Therefore the algorithm is also trained and tested based on fewer numbers of frames than 87. In practice this should be done by varying the rotation speed. However, this would also result in significantly more recordings, which is time consuming both during the experiments and the following manual labeling. Therefore, instead the image sequence is downsampled with an integer factor such that the algorithm can be tested on respectively 87, 44, 29, 22, 18, 15, 13 and 11 frames.

As mentioned in chapter 3.2.1.2, the underlying assumption about the system is that the particles are in a steady state, such that they are rotating with the same frequency as the container at all rotation speeds. This means that downsampling the 87 frames corresponds to recording images at different rotation speeds according to equation 3.3. That is, in addition to undersampling we also downsample the 87 frames instead of performing the actual recordings at different rotation speeds.

3.4.2 Manual Labeling

In order to allow supervised learning and subsequent testing of the algorithm, a manually labeled data set is needed. Therefore, all observed particles in all acquired images have been pointed out manually and given a label indicating whether they are on the outside or inside of the corresponding glass containers.

These manual labels serve as the ground truth, specifying which particles should be detected and what classes they belong to (in/out). By searching in a given radius around each hand-labeled particle in every image, particles detected by the algorithm can be linked to observed particles. This can be done both on image-level and on image-sequence-level. On the image-level a detected particle is matched if it is sufficiently close to a labeled particle. On image-sequence-level a detected particle corresponds to a tracked sequence of image-level particles. A detected particle is matched on image-sequence-level if this sequence is sufficiently close to the corresponding sequence of labeled particles. For both image- and image-sequence-level a radius of 50 px is used in order to take into account imprecise hand labels and potential distortions caused by large particles.

In this way, statistics can be found both on image-level and image-sequence-level. Subsequently an evaluation can be made, whether detected particles are classified correctly as being

either on the outside or inside of the glass containers. However, in order to construct data suitable for evaluation, some decisions and conventions are needed. These decisions both regard particles that are present in the container but not detected by the algorithm (missing detections) and particles that are detected by the algorithm but not actually present in the container (false detections). A convention is used, utilizing the overall purpose of the system - being able to reject containers with particles inside while accepting containers with particles outside. This implies:

1. **Missing detections**. A container with a particle not detected by the system corresponds to a container with a particle detected on the outside, both being approved during inspection. Therefore, a particle not detected by the system but pointed out in manual labeling is treated as a particle detected on the outside when evaluating the algorithm. The implications of this decision are that particles that are actually on the outside will not influence the accuracy of the system, whereas particles that are actually on the inside will introduce false acceptances of containers. In an overall perspective, a false acceptance is the most critical, since this can ultimately lead to personal injury caused by impure medicine.

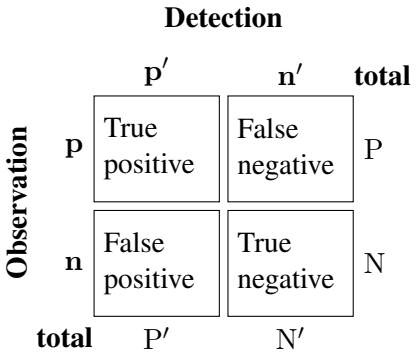
2. **False detections**. A non-existing particle detected by the system is a false detection and should ideally not influence the inspection result. However, if the system classifies the particle as being inside the container, a false rejection of the container will occur, given that no other particles are inside. Therefore, in order to include this event, the non-existing particle can be treated as an actual particle located on the outside of the container, even though this has not been labeled manually. The implications of this decision are that detection of particles on the outside will not influence the accuracy of the system, whereas detection of particles on the inside will introduce false rejections of containers.

3. **Multiple detections of the same particle**. When a particle is detected more than once on image-sequence-level by the algorithm, a decision needs to be made for how to handle this situation. Theoretically, the multiple detected particles can be estimated with different radii resulting in detected particles both on the inside and outside of the container. Therefore, multiple detections are handled by treating the actual particle as several particles with the same observation (in/out). The implications of this decision are that disagreements between the multiple detected particles will result in both correct and false particle classifications, whereas compliance will result in either multiple correct or multiple false particle classifications.

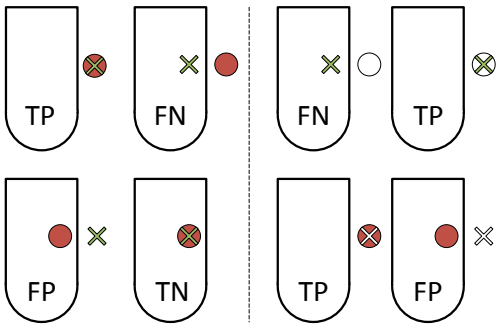
By applying the above conventions, a unique correspondence is achieved between observed and detected particles. This correspondence makes it possible to construct a confusion matrix, visualizing the overall performance of the system [23]. The confusion matrix lists the number of occurrences for containers being classified as accepted/rejected related to both observations and detections. The following conventions are used:

- **True positive (TP)**: a container is marked as accepted both by observation and detection.
- **False positive (FP)**: a container is marked as accepted by detection but is rejected by observation.
- **True negative (TN)**: a container is marked as rejected both by observation and detection.
- **False negative (FN)**: a container is marked as rejected by detection but is accepted by observation.

In this convention a positive is regarded as an acceptance based on actual/observed particle positions, where all particles are located on the outside of the container surface. On the other hand, a negative is regarded as a rejection, where just one particle located on the inside of a container is sufficient for rejection. Figure 3.8a illustrates the applied confusion matrix conventions with symbolic notation. Figure 3.8b illustrates the different observation/detection scenarios. To the right in the figure special cases are illustrated as described above.



(a) Confusion matrix relating observations (actual classes) to detections (predicted classes).



(b) Visualization of confusion matrix conventions. Red circles indicate observations, green crosses indicate detections and white circles/crosses indicate missing observations/detections.

Figure 3.8: Confusion matrix conventions. (a) illustrates the confusion matrix and (b) illustrates different observation/detection cases.

3.5 Artificial Data Set

In order to evaluate different aspects of the system separately while achieving a large and representative data foundation, a particle simulator is designed. The purpose of the simulator is to simulate the physical environment including the rotation device, the glass container, the camera and potential particles tied to the container. By including all these parameters in a simulator, a scenario can be constructed, where uncertainties and inaccuracies for a number of realistic parameters can be varied:

- **Camera position.** A displacement along the z-axis corresponding to the actual working distance of 180 mm is used.
- **Frame rate for the camera.** As described in chapter 3.2.1.3 a problem regarding the actual frame rate of the camera during image acquisition and inaccurate time stamps has been experienced. The overall method for estimating the position of particles relies heavily on exact values of the frame rate and rotation frequency. Therefore a possibility for changing the frame rate and adding noise to the time stamps is included in the simulator.
- **Focal length for the camera.** Along with the camera position the focal length determines the size of the container and the particles in the image, and is therefore important in order to simulate realistic particle coordinates for the positioning algorithm.

- **Image resolution.** The resolution of the image determines the size of the particles in pixels. A resolution of 1624 x 1234 is used corresponding to the Basler (20 Hz) camera.
- **Visible angle range.** Due to different optical phenomena like refraction of light rays in the glass and liquid only a part of the glass container and the particles inside is visible in the captured images. The simulator therefore includes an option for "hiding" a portion of the particles in the image sequence so only particles moving in an angle range corresponding to the front side of the cylinder are visible.
- **Perspective view.** The simulator uses a pinhole camera model in order to include the perspective camera projection. This enables us to examine imprecision introduced by the varying distance between a particle and the camera when the container is rotating. It also allows the possibility of designing and testing a correction of the perspective coordinate distortions.
- **Noise in blob segmentation.** Due to noise in the image and inaccuracies in the segmentation and tracking algorithms the detected blob positions will be overlaid with noise. By assuming a Gaussian distribution this noise can be added to the particle coordinates.
- **Rotation frequency of the glass container.** As described in chapter 3.1 an offset on the actual rotation frequency of the container, as opposed to the desired frequency, has been observed. This offset influences the estimated radii of the particles. An option of adjusting this offset and adding an uncertainty to its value is therefore implemented.
- **Dimensions of the glass container.** Since different kinds of containers such as ampoules, vials, cartridges and syringes can be used in the actual inspection system, an option of specifying the exact dimensions of the glass cylinder has been implemented. Especially the glass thickness is an important parameter that directly determines what precision is needed in the estimation of the particle radii.

The simulator can generally be divided into three steps. Algorithm 1 explains in pseudocode the simulation of image coordinates for particles. The simulation of the blob area is done alongside by modeling it as a constant with an overlaid harmonic oscillation.

1. **Construction of 3D-objects and -points in world coordinate system.**
   This step includes calculating the coordinates of the glass container and the particles rotating with it. The world coordinate system has the Origo located in the center of the container, the z-axis pointing towards the position of the camera and the y-axis pointing along the vertical direction of the container. A whole sequence of particle positions is created by utilizing information about the camera frame rate and the container rotation frequency. A rewritten version of the formula 3.3 can express the number of samples equally distributed around the container for one entire revolution:

$$n = \mp \frac{1}{\frac{f_{cyl}}{f_s} - k} \tag{3.8}$$

   The $\mp$-sign relates to formula 3.3, since it depends on whether the container rotation frequency is larger or smaller than a multiple of the camera frame rate. n three-dimensional particle positions can be constructed, distributed equally around the container in a radius specified as an input to the simulator. A portion of these corresponding to the chosen angular range visible by the camera can be passed on to the viewing transformation together

with the coordinates of the container. A possibility of adding individual time stamps has also been implemented to simulate the time stamp precision of 1 ms for the camera setup as described in chapter 3.2.1.3.

2. **Viewing Transformations**
   The purpose of the viewing transformations is to transform the 3D world coordinates to 2D camera coordinates. In the literature this transformation is described in a number of ways. However, generally the terms of extrinsic and intrinsic camera matrices are used to describe respectively the world to camera coordinate transformation and the image plane transformation from metric to pixel units [24, 25, 26]. In addition comes a projection from the camera coordinates in 3D to the image plane in 2D. This projection can either be orthographic or perspective depending on the camera lens. Using an orthographic projection will produce an image similar to a telecentric lens, whereas a perspective projection assumes a pinhole camera approximating a rectilinear lens [22, p. 1204]. In our setup a rectilinear lens is used and the simulator therefore includes a perspective projection. The combined transformation from 3D world coordinates to 2D image coordinates in pixels can be expressed as:

$$\mathbf{x}_{image} = \mathbf{K} \cdot \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T} \cdot \mathbf{X}_{world} \tag{3.9}$$

   In this equation $\mathbf{X}_{world}$ is a vector containing 3D coordinates of e.g. a particle and $\mathbf{x}_{image}$ is a vector containing 2D image coordinates in pixels - both vectors being expressed in homogeneous coordinates. $\mathbf{T}$ is a translation matrix describing the translation of the camera relative to the world coordinate system. $\mathbf{R}$ is the combined rotation matrix and can be broken down into the product of three rotation matrices, $\mathbf{R} = \mathbf{R_y} \cdot \mathbf{R_z} \cdot \mathbf{R_x}$, corresponding to the convention of the coordinate system.$\mathbf{P}$ is a $3x4$ perspective projection matrix that effectively removes the 4th coordinate, thereby introducing the z-coordinate as the homogeneous coordinate in the 2D image coordinate system. $\mathbf{K}$ is the intrinsic camera matrix. A number of different conventions are used for describing this in the literature. However, a simple version includes parameters known from the actual test setup [25]:

$$\mathbf{K} = \begin{bmatrix} f \cdot s_x & 0 & n_x \\ 0 & f \cdot s_y & n_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

   where f is the focal length of the camera, s_x and s_y are the scale factors relating pixels to distances and n_x and n_y represent the principal point (center of the image) in pixels.

3. **Manipulation of 2D image coordinates**
   When the particles and container coordinates have been projected into 2D image coordinates, noise can be added to these to simulate inaccuracies in the segmentation and tracking algorithms. For this, Gaussian distributed noise with a given variance is added to the (x, y)-coordinates.

Figure 3.9 shows an example of using the simulator with five particles using a total of $n = 20$ frames. The dimensions of the container are adjusted to fit the whole image for visualization purposes. The two particles on top are placed in a radius corresponding to the inner surface, whereas the three lower particles are placed in a radius corresponding to the outer surface of the container. For all five particles varying visible angle ranges are applied, and for the particle in the bottom a considerable amount of noise ($\sigma^2 = 30$) is added to the image coordinates.

input : particles radii
output: particles image coordinates

construct extrinsic matrices ($\mathbf{T}$, $\mathbf{R}$);
$\mathbf{K} = 3 \times 3$ intrinsic camera matrix ; // eq. 3.10
$n =$ number of points around the container ; // eq. 3.8
for $p \in particles$ **do**
    $\mathbf{X}_{world,p} = 3 \times n$-matrix with particle points in 3D world coordinates;
    add homogeneous coordinates to $\mathbf{X}_{world,p}$;
    $\mathbf{x}_{image,p} =$ projected 3D metric to 2D pixel coordinates ; // eq. 3.9
    remove homogeneous coordinates from $\mathbf{x}_{image,p}$;
end
add noise to $\mathbf{x}_{image,p}$;

Algorithm 1: Simulator algorithm for generation of particle image coordinates.

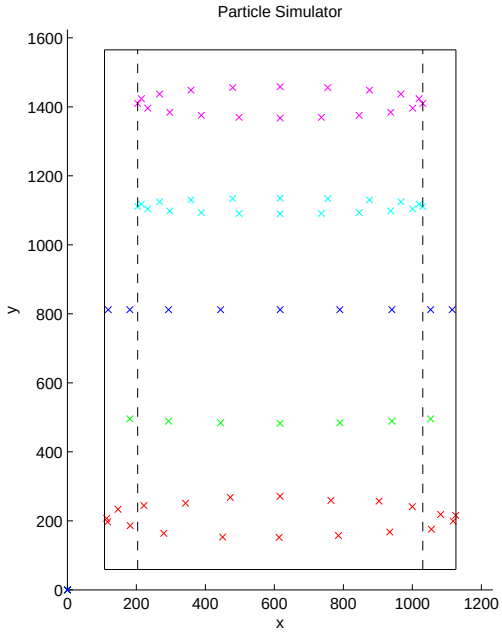


Figure 3.9: Example with 5 particles generated by the particle simulator.

3.6 Summary

In this chapter we have presented an experimental setup consisting of a mechanical setup and a machine vision setup. The purpose of the mechanical setup is to rotate medical glass containers with liquid with a specified rotation frequency. The machine vision setup includes illumination of the containers as well as a camera setup acquiring images with fixed intervals being able to inspect particles with sizes down to 55 μm.

From the experimental setup sequences of images have been acquired for a total of 40 containers. These represent the data set and are split into a training set and test set used for respectively training the proposed algorithms and testing them in a final evaluation. A particle simulator has been developed allowing the acquisition of an artificial data set. This is used to evaluate different aspects of the system separately while achieving a large and representative data foundation.

4

First Algorithm

In this chapter an overall method is presented for positioning particles three-dimensionally in glass containers. The method assumes precise measurements of both the container rotation frequency and the camera frame rate. By synchronizing these two frequencies, such that during a sample period for the camera, a particle moves only a small horizontal distance relative to the previous frame, a sequence of images can be acquired showing circular motion paths of particles tied to the glass container. By estimating the circular motion paths, a classifier can decide if a particle is inside or outside the glass container based on the measured radius of the circle.

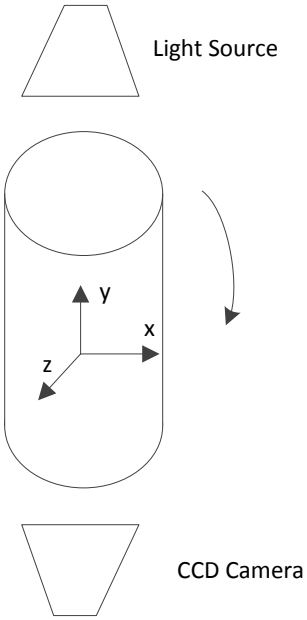


Figure 4.1: Method for capturing an image sequence of the rotating container. The defined three-dimensional coordinate system is drawn on top of the cylinder.

The situation is depicted in figure 4.1 where the glass container is rotating while a CCD camera is capturing the video sequence. A light source is illuminating potential particles tied to the container from the back seen from the camera. A coordinate system is defined such that the x-axis points along the transverse axis of the cylinder, the y-axis points along the longitudinal axis and the z-axis points towards the position of the camera. In this way the particles in the

captured images will have x- and y-coordinates. The main problem is to estimate the z-coordinate by estimating the radius of the circular particle paths.

As illustrated in the SysML [27] activity diagram 4.2 the algorithm takes as input a sequence of images of the rotating container together with parameters concerning the environmental setup such as the camera frame rate and the rotation frequency of the container. By using image processing techniques like edge detection and Hough transformation the container is measured in order to find its diameter and center coordinates. Simultaneously a segmentation algorithm seeks to find moving particles (blobs) in the image sequence and extract features like position and area for each particle blob. By matching blobs in consecutive images based on their properties or pixel intensity distributions, the found blobs can be matched from frame to frame and passed to a tracking algorithm. Here a path of movement is extracted for each particle based on the frame-wise blob matching throughout the image sequence. Utilizing environmental parameters like the camera frame rate and the container rotation frequency, a circle can be fitted to each sequence of particle coordinates. By comparing the estimated radii of the circular paths with the estimated container diameter a classification can be made to decide if a particle is inside or outside the glass container.

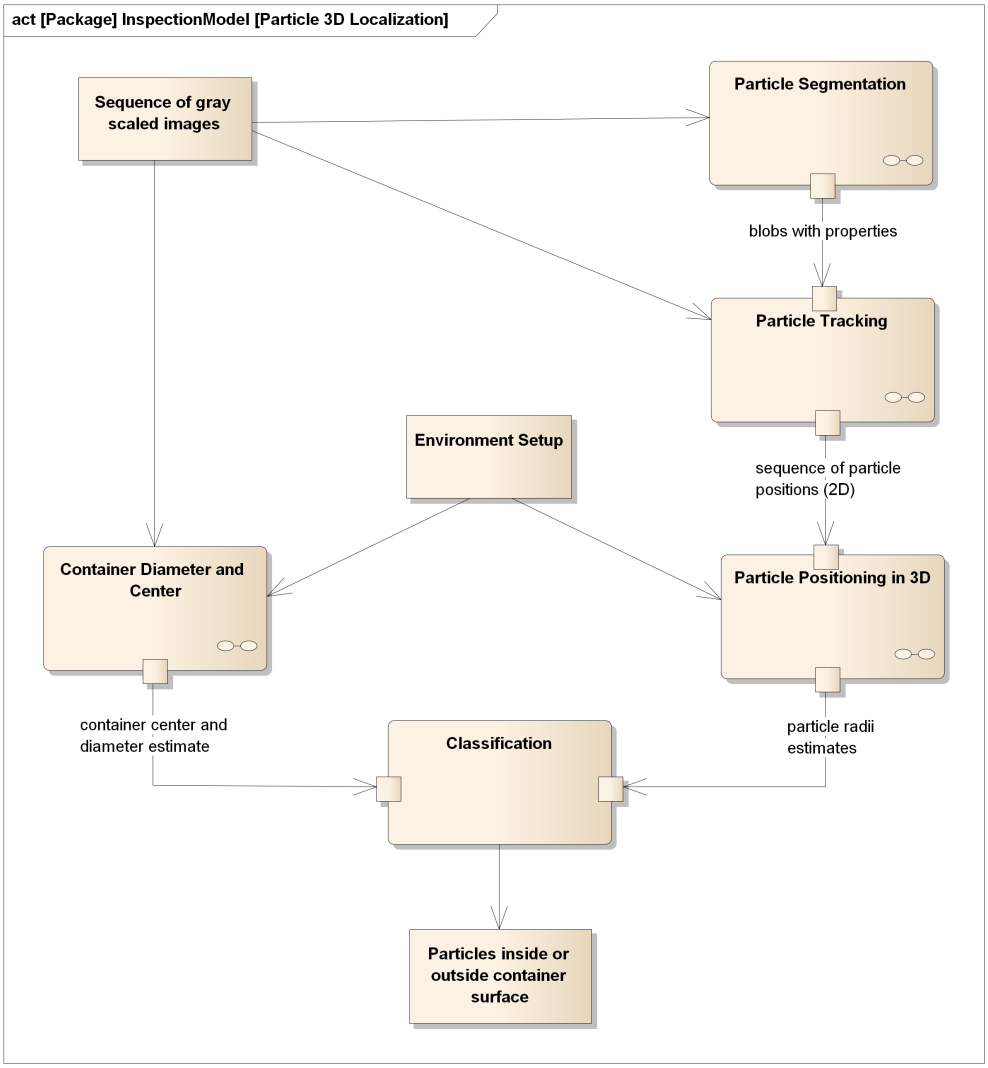


Figure 4.2: SysML activity diagram for the first algorithm.

In the following sections the different stages of the algorithm are described in more detail. Each section includes a small evaluation discussing each algorithm with alternative solutions, parameters and results. The general results of the overall algorithm conducted on a test set including several containers are presented and discussed in chapter 7.1.

4.1 Container Diameter and Center

In order to decide whether particles are located on the inside or the outside of a glass container we need to measure the center and diameter of the glass container. Inspection will be performed on similar containers and we can assume that the diameter varies less than a millimeter between each measurement. Our method concerns finding straight lines in the image so that we can estimate the diameter by finding the distance between parallel lines. Here we use the longest line found by a Hough transformation [28] and compute the shortest distance of parallel lines to this longest line. The distance that is closest to a manually measured average diameter will be used as an exact diameter for the currently inspected container. The method ensures a robust algorithm that also takes into account incorrect measurements and at the same time provides an individual center and diameter measurement.

The algorithm that we have developed and evaluated in finding the center and diameter is visualized in a SysML activity diagram [29] as illustrated in figure 4.3. In the following each part of the algorithm will be explained in detail.

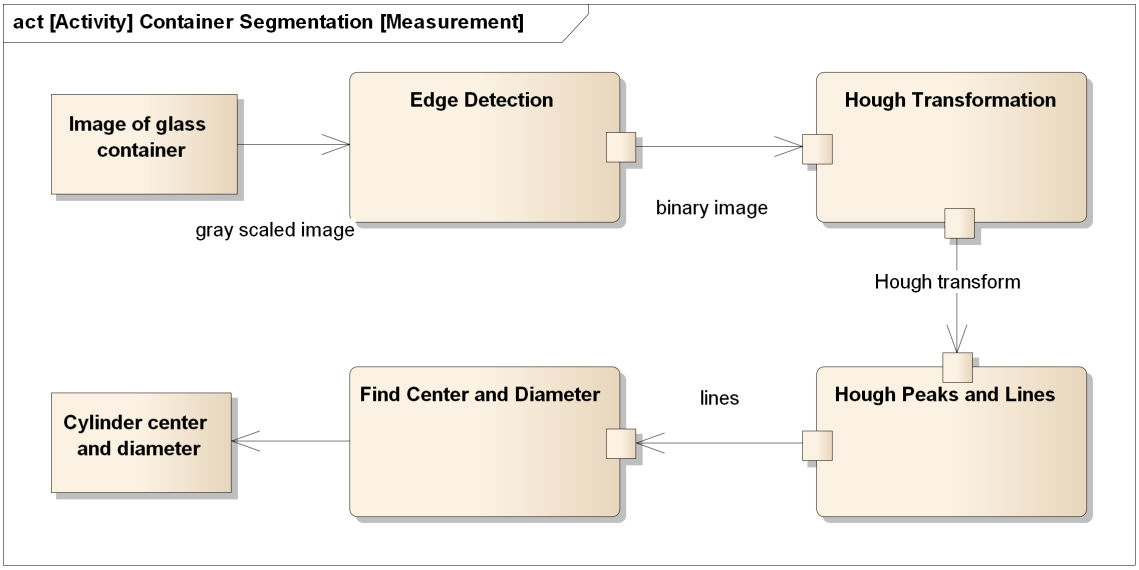


Figure 4.3: SysML activity diagram for diameter and center measurement.

4.1.1 Edge Detection

Edges characterize boundaries of the glass container and are therefore of fundamental importance in order to find lines. Edge detection is based on abrupt changes in pixel intensities and defines the boundary between regions. In [30] three fundamental steps are involved in edge detection.

- Image smoothing for noise reduction by low-pass filtering
- Extraction of all edge points that are candidates for becoming an edge point
- Edge localization by thresholding in order to find members of a set of points comprising an edge

First image smoothing is performed by using a Gaussian low-pass filter [3]. The Gaussian filter performs well and no ringing will occur as opposed to other low-pass filters like for instance the Butterworth filter. We like to avoid ringing causing artifacts that could produce false edges. In [17] edge detection using template matching or differential gradient methods is described. In either case the aim is to find where the intensity gradient magnitude is sufficiently large to be taken as a reliable indicator of the edge of a boundary. In our algorithm only differential gradient methods using either the first derivative or second derivative have been investigated. For first derivative methods filters like Sobel, Prewitt and Roberts are good, and a simple convolution mask using the Roberts operator performs acceptable as illustrated in figure 4.4. Here, longer edges with less noise are found unlike for Sobel and Prewitt edges. The Laplacian operator is a second derivative operator and is only sensitive towards changes in the intensity gradient. Applying the Laplacian operator followed by a global adaptive thresholding using Otsu's method [3, p. 742-747] performs well in finding the whole contour of the glass container. Less promising results are found using more advanced methods like Laplacian of Gaussian and, the most widely used, Canny edge detection. Here more details are found in the image than are needed for a simple diameter measurement.

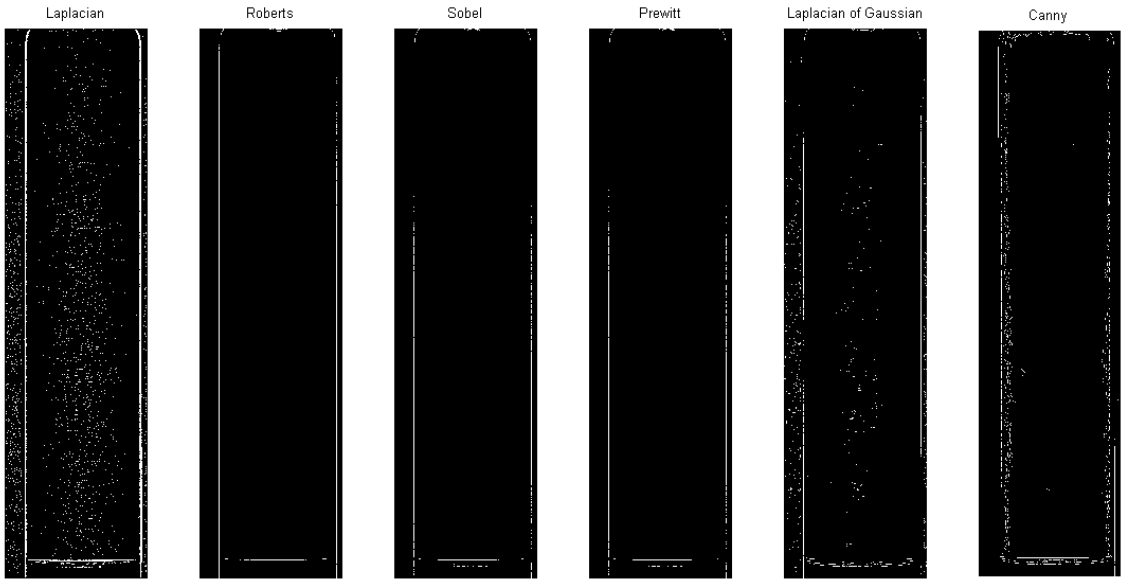


Figure 4.4: Comparing different edge detection methods for a background image of the cartridge.

Our experiments will be based on either using the Laplacian, Sobel or Roberts operator in selecting the best method for diameter measurement.

4.1.2 Hough Transform, Peaks and Lines

The Hough transform (HT) is the most common way of detecting straight edges [17]. The basic concept involved in locating lines by the HT is point-line duality [3, p. 733]. A point P can be defined either as a pair of coordinates or in terms of the set of lines passing through it. A line is parametrized using the slope and y-intercept by the equation $y = ax + b$. Infinitely many lines pass through (x, y), but all satisfy the equation for varying values of a and b. Duda and Hart [28] have proposed the so-called normal parametrization given by equation

$$xcos(\theta) + ysin(\theta) = \rho$$

This parametrization specifies a straight line by the angle of its normal θ and its algebraic distance ρ from the origin. These two values, taken in conjunction, define a polar coordinate. This corresponds to a sinusoidal curve in the polar (θ, ρ) plane, which is unique for each point. Every point from the original image will then appear as a sinusoidal curve in the HT and lines will appear as intersections of these curves representing a number of points on the same line. When many points in the original image are lying on the same line a high intensity of crossing sine curves will be observed in the HT image as illustrated in figure 4.5.

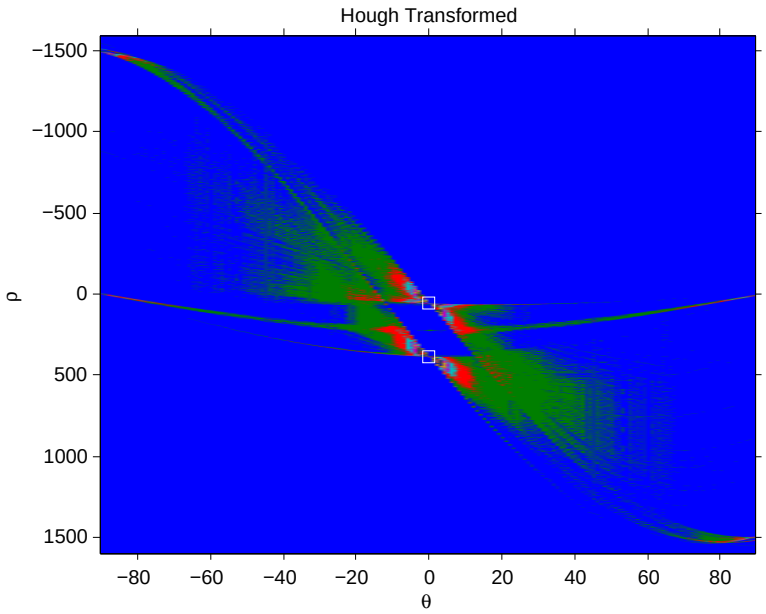


Figure 4.5: Polar coordinates of the Hough transform for the edge thresholded image.

The result of the linear HT on the edge thresholded image is a two-dimensional matrix with one dimension being the quantized angle θ and the other dimension the distance ρ. After the transformation the maximum peaks in the HT matrix are found. Each peak represents an intersection of many curves in the polar plane for which a number of edge points form a straight line. Each peak represents a line in the edge image that is finally found as "Hough Lines" illustrated in 4.6.

RANSAC [31] is an alternative model-based search scheme that could be used instead of HT. For line detection it is effective to make a sequence of hypotheses about the target objects using a selected number of points found by edge detection. RANSAC is more compact requiring less information storage than the HT. RANSAC is iterative using only a few points in each iteration, whereas HT stores the whole parameter space in memory. Since we have not investigated RANSAC for line detection we have only evaluated our method using the Hough transform.

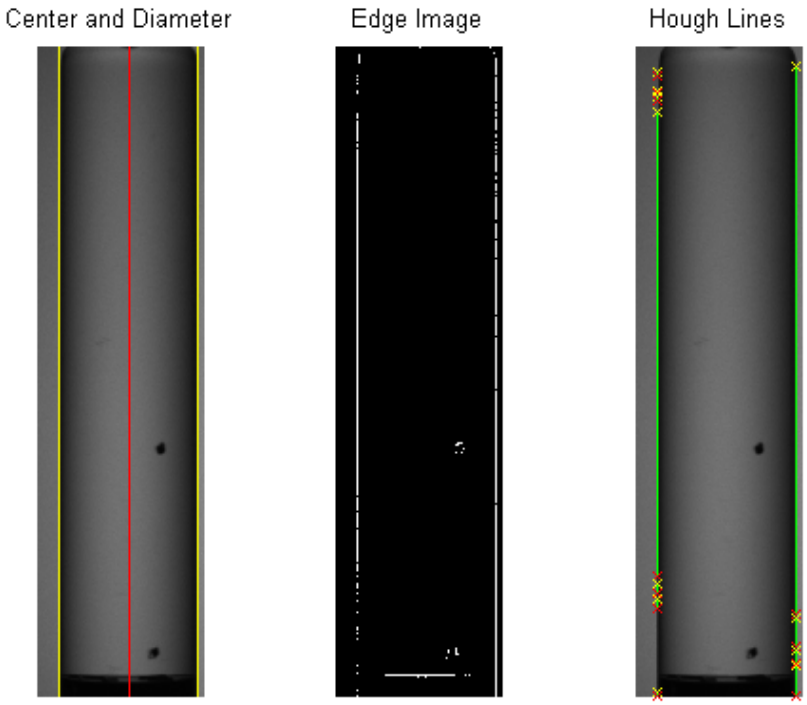


Figure 4.6: Diameter and center of cartridge found using Roberts edge operator and Hough transformation to calculate distance between parallel lines.

4.1.3 Finding Diameter and Center

Given a number of straight lines found by the Hough transform we need to compute the diameter and center of the cylindrical glass container. Our approach is first to find the longest Hough line. Given the geometry of the cylinder there is a high probability that the longest line will correspond to one of the borders of the cylinder.

Secondly we compute the cross product of all lines in relation to the longest line. If all lines are modeled as vectors in a two-dimensional euclidean space the euclidean norm of a cross product between two vectors will be zero indicating parallel lines. For all lines parallel to the longest line we compute the minimum distance as described by Dan Sunday [32]. From the minimum distances the diameter closest to a manually measured pixel diameter is chosen. The manually measured reference diameter ensures that defective measured distances will be discarded. The method will then find an individual diameter measurement within a tolerance of an average diameter. The center is computed using the x-coordinate of the longest line and the measured radius/diameter.

The final result is illustrated in 4.6 finding the diameter and center of a cylindrical glass container.

4.1.4 Evaluation

We have evaluated the described algorithm for finding the diameter and center using a background image of recordings from 6 different glass containers. The background image is found by median filtering 10 images from the video sequence of the rotating glass container.

The algorithm is sensitive towards the selected edge detection method as seen from the results in table 4.1, comparing the Roberts, Sobel and Laplacian operators for different containers from the data set in appendix D table D.1. A manually measured average diameter of 328 pixels is calculated.

Edge Operator	AR Insulin	CS Insulin	A1 Insulin	CS1 Insulin	D1 Empty	E1 Glycerol
Sobel	335	671	330	364	338	330
Laplacian	322	324	322	324	280	325
Roberts	328	328	330	330	328	330

Table 4.1: Diameter measurement in pixels on cartridges using different edge operators.

Using the Sobel operator we only manage to measure a correct diameter in two cases (A1 and E1). It fails in the other tests since only short Hough lines are found and therefore the shortest distances between the lines tend to be longer. The Laplacian operator in general measures a smaller value of the diameter, up to 5-8 pixels. In the case of cartridge D1 (an empty container) the Laplacian operator finds edges for both the inner and outer border of the glass surface. The inner diameter is in this case found with a diameter of only 280 pixels. The Roberts edge operator measures a diameter very close to the manually measured average value of 328 pixels in all test cases.

Conclusion: The Roberts edge operator in combination with the Hough transform is chosen for diameter and center measurement, since it provides the best result in all test cases.

4.2 Particle Segmentation

In order to track particles tied to the glass container we first need to detect particles in each image. This is done by image segmentation where we find particle blobs and extract properties from these. An overview of the algorithm is illustrated in figure 4.7. First the image is segmented as a binary image containing blobs for every particle found in the image. The boundary of the container edges are removed to ignore light fluctuation and disturbances caused by the rotating container. A narrow area corresponding to the thickness of the glass surface is selected. Particle blobs are extracted using the connected components algorithm, and region properties of the blobs such as center and area are found.

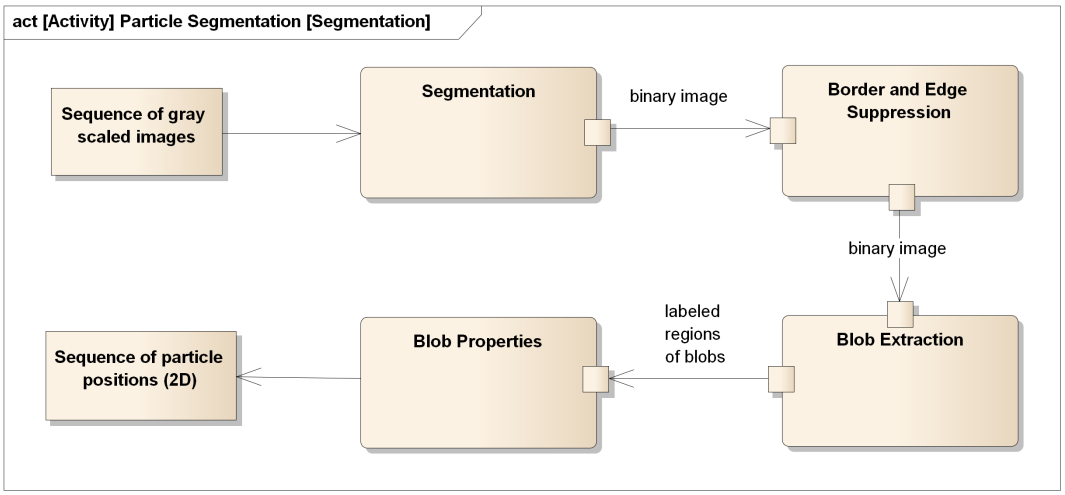


Figure 4.7: Decomposition of algorithms for particle segmentation.

Figure 4.8 illustrates one frame of the video sequence, where some of the intermediate steps in the segmentation process using background subtraction are shown. First a background image is found and the center and outer border of the container are marked with horizontal lines. The segmented image is found as the difference between the cropped and the background image. Finally blobs are identified using connected components illustrated in the last container with green bounding boxes. The design of each algorithm will be explained in detail in the following chapters followed by an evaluation of the described method.

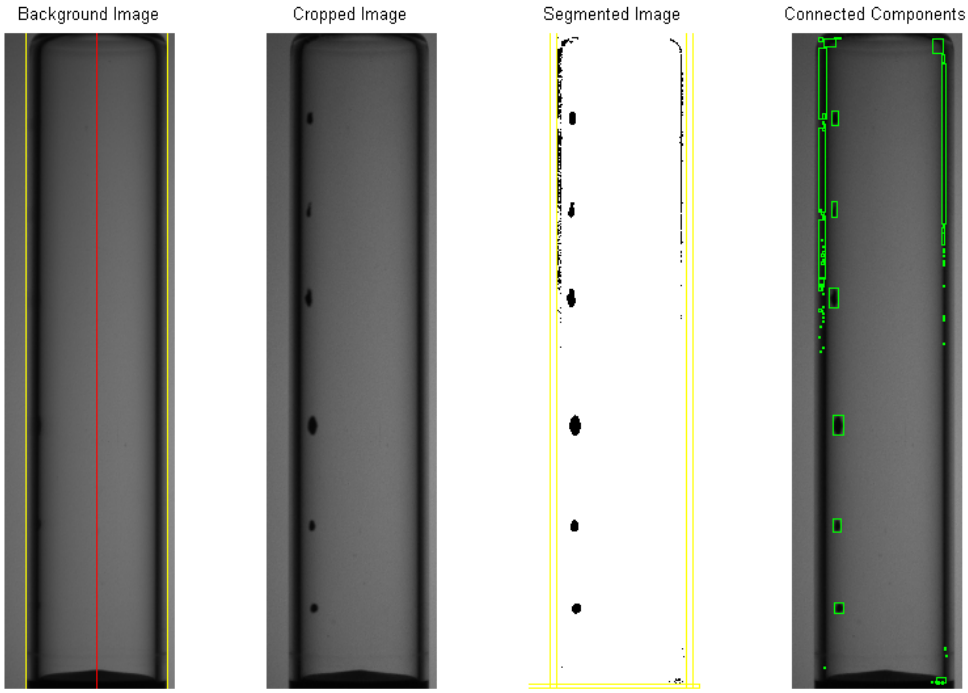


Figure 4.8: Background subtraction for a cartridge with particles.

4.2.1 Segmentation

Segmentation subdivides an image into its constituent regions or objects. In our case the goal is to find regions of interest for where particles could be located. The glass container is illuminated with a diffuse background light, and with a transparent glass container we will get a high intensity level in all pixels. Only the borders of the container and particles that create a dark shadow in the image will be visible as dark areas (blobs) on a white background. We will look for discontinuity of the pixel level intensity values in the image by using thresholding, edge detection and morphological operations.

We have developed two different methods for particle segmentation with the purpose of localizing and identifying particles in each image. The first method, Global Segmentation, uses an adaptive method to find a global threshold value to segment the image in black and white. The second method, Background Subtraction, uses different threshold values for every pixel. Here a background image is created and subtracted from each image in the video sequence. Both methods will be described in the following and finally evaluated.

4.2.1.1 Global Segmentation

It is possible to find a single global threshold value when the intensity distributions of image pixels for objects and the background are sufficiently distinct. In these cases we have a high contrast between dark particles and the white background. By forming a histogram of all pixel intensities the challenge is to find a threshold value that separates particles from the pixel intensities of the background. It is more likely to select a "good" global threshold value for the whole image if the histogram peaks are tall, narrow, symmetric and separated by deep valleys.

To improve the shape of histograms a method of using edges improving global thresholding is described in [3, p. 749]. The method considers only those pixels that lie on or near the edges between objects and the background. An immediate and obvious improvement is that the histograms used for finding the global threshold value would be less dependent on the relative sizes of objects and the background. Without using this method the histogram will be dominated by a large peak caused by the high concentration of the illuminated background. If only the pixels on or near the edges between particles and the background were used, the resulting histogram would have peaks of nearly the same height.

We have adapted the method of using edges in finding a global threshold value on our problem. Global segmentation is realized by the algorithm summarized below, where we have added a Gaussian blurring filter before finding edges.

1) Compute a blurred Gaussian filtered image $f_b(x, y)$

2) Compute an edge image $f_e(x, y)$ based on $f_b(x, y)$ using the Laplacian operator

3) Specify a value for thresholding the edge image to produce a binary image $g_{eT}(x, y)$

4) Use the thresholded edge image as a mask to select pixels from $f_b(x, y)$ corresponding to "strong" edge pixels by computing a histogram, using only the pixels in $f_b(x, y)$ that correspond to the locations of the 1-valued pixels in $g_{eT}(x, y)$

5) Use the Otsu's method on the histogram to find a global threshold value

6) Finally the image $f_b(x, y)$ is segmented using the global threshold value multiplied by a configuration factor producing the segmented image $g_{sT}(x, y)$

The result of the algorithm above is a segmented binary image where particles are visible. However, we have found that very small particles will not be detected by this method. The final segmentation is therefore improved by combining the thresholded edge image $g_{eT}(x, y)$ with the globally segmented image $g_{sT}(x, y)$. A union between the two logical binary images is computed

$$g_T(x, y) = g_{eT}(x, y) \cup \neg g_{sT}(x, y)$$

The method is illustrated in figure 4.9 for a cartridge with a particle outside and inside the glass surface. This method is limited to only find particles inside the border of the glass container where we have a high contrast in the image. Here the background illumination is an important factor in order to improve the contrast and reduce the width of the black borders in g_{sT}.

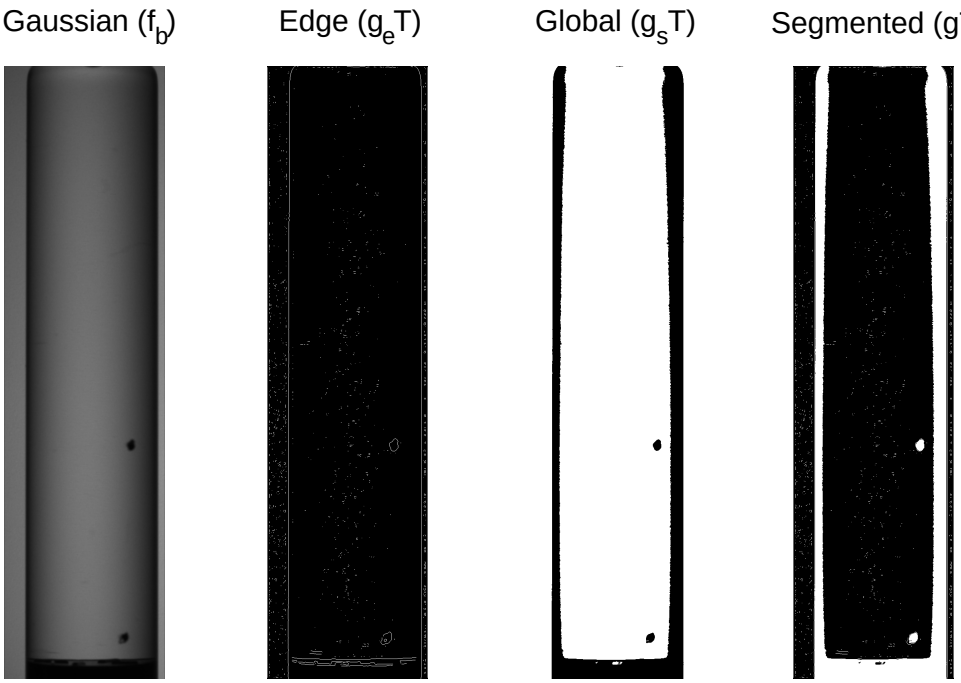


Figure 4.9: Particle segmentation combining the edge image with the globally thresholded image.

4.2.1.2 Background Subtraction

To improve the segmentation as described in the previous section we have created a background image of the illuminated cartridges without particles. Based on a number of images the background can be found by using a median filter for every pixel location. The filter computes the median value for every pixel by extracting pixels at the same spatial location from a number of images in the video sequence. Here we have selected 9 images with equal intervals from the sequence. In the case where we have a sequence of 18 images of the rotating container every second image is used.

The method is simple but efficient since we are now practically using an individual segmentation threshold for every pixel in the image. The background segmented image is created as a binary image where every pixel intensity is set to 1 if the intensity is less than a certain percentage of the background pixel intensity. In this case darker areas than a white background will be found as particle blobs.

A variation of this method is also explored where both darker and brighter pixels within a certain percentage of the background pixel intensities are used. Using this method results in an improved segmentation at the bottom area of the container, where we have the rubber stopper. In this case particles are observed as brighter blobs on a dark background. The disadvantage, however, is that we get more irregular blobs and false particle detections due to variation of the illumination. Therefore only darker pixels are used in our background subtraction method, but an improvement of the method is still relevant for proper detection of particles in the area of the rubber stopper.

The segmented image (using only darker pixels than the background) is illustrated in figure 4.10. In this case segmentation does not detect the boundary of the cylinder but only particles. However, this is not the case if the container is not rotating straight around its own center as in figure 4.8.

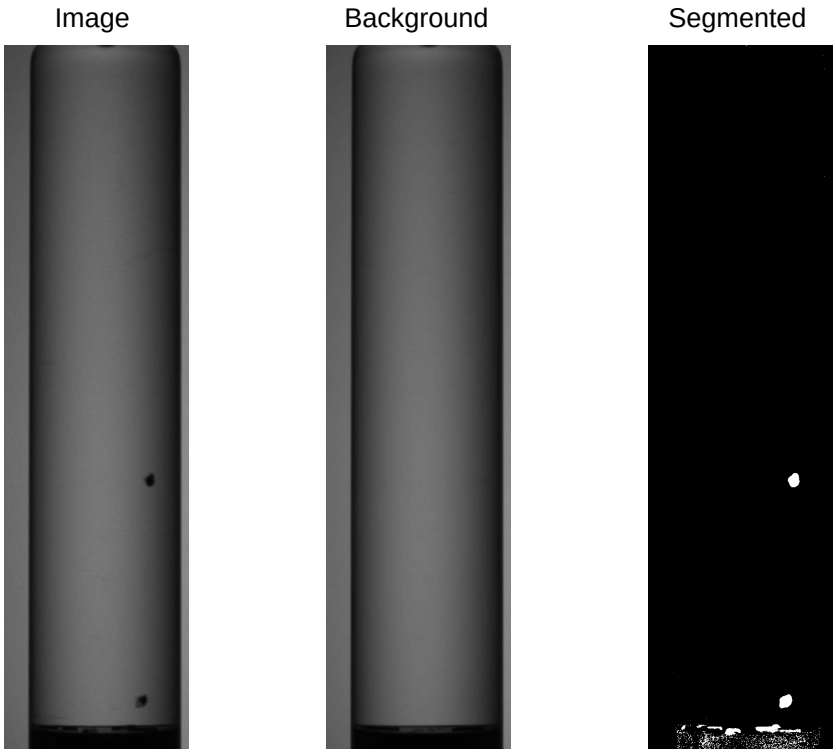


Figure 4.10: Particle segmentation using the background image for thresholding.

4.2.2 Border and Edge Removal

We would like to remove the border of the rotating container to avoid false detections of particles, which has been noticed being a problem as illustrated in the previously described segmentation algorithms. We have already found the diameter and center of the cylinder and know the thickness of the glass surface. Based on this information we have implemented an algorithm that removes the boundary and bottom of the glass cylinder. A rectangle is specified to ignore blobs at the border of the container corresponding to the thickness of the glass surface in pixels. The algorithm will extend the width of the rectangle until it finds the white illuminated background area. This feature is especially needed for global segmentation to ensure that particles do not merge with the borders. The dynamic border and edge removal are illustrated in figure 4.11

4.2.3 Blob Extraction and Properties

After segmentation and edge removal morphological operations [3, p. 627] are performed as a preprocessing in filtering the image. The algorithm can be configured to perform dilation, erosion, opening or closing using a 3x3 cross structuring element SE. A dilation expands particle blobs and erosion shrinks them. Opening is an erosion followed by a dilation. It generally smooths the contour of the particles, breaks narrow convex holes and eliminates thin protrusions. Closing is a dilation followed by an erosion and tends to smooth sections of contours like opening. Opposite opening, in general it fuses narrow breaks, eliminates small holes and fills gaps in the contour.

After morphological filtering, extraction of connected components is computed using the algorithm as described in [3, p. 645-647]. Let image A be a set containing one or more connected components formed in the image X_0 whose elements are zeros (background). We start by initializing image X_0 with one white pixel found in A. The iterative procedure below is used to find each connected element.

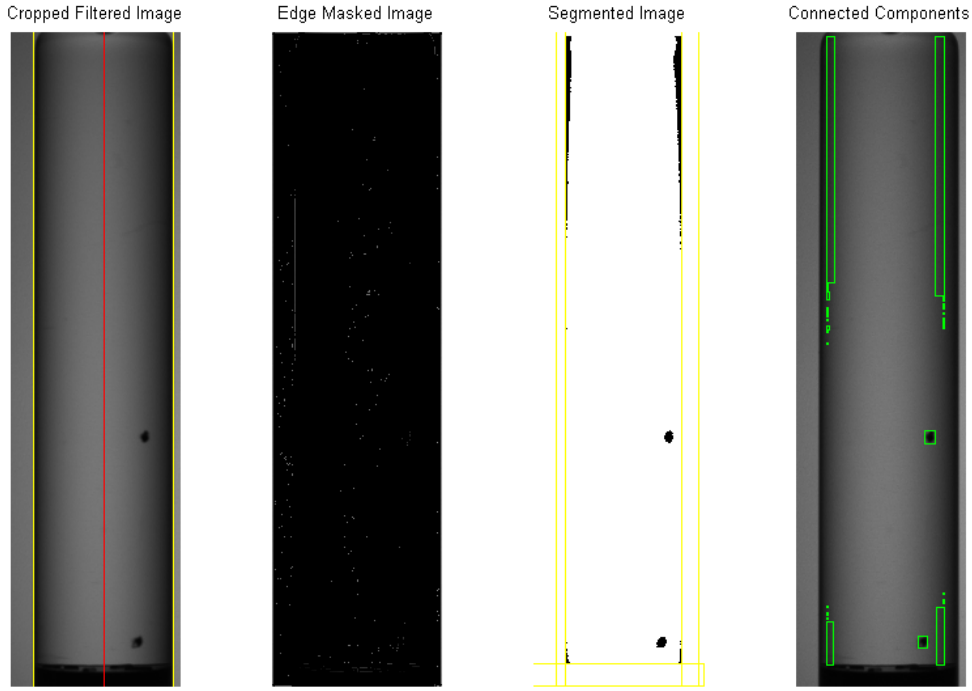


Figure 4.11: Global segmentation and matching for a cartridge where the boundary of the edges is dynamically removed (marked with yellow lines).

$$X_k = (X_{k-1} \oplus SE) \cap A$$

The procedure terminates when $X_k = X_{k-1}$, with X_k containing all the connected components of the input image A. Every time the procedure terminates the connected component (particle blob) is labeled with a unique number. Here we can choose between a 4 or 8-connected neighborhood SE in labeling the binary image. With an 8-connected neighborhood SE we will get less particles since corners of particles close to each other will be labeled as the same particle. After labeling using connected components, features are extracted for each particle blob.

- **Area**: the number of pixels in the labeled region is used to specify the area of the particle blob
- **Center**: the center of mass of each region is found to specify the center of the particle
- **BoundingBox**: the bounding box is found as the smallest rectangle containing the region

The result can be seen in figure 4.11 where particles are marked with a green bounding box.

4.2.4 Evaluation

The first evaluation concerns comparing the global and background segmentation methods. First we will evaluate the two methods on an empty (no liquid) test cartridge (D1) containing 3 marks on the outside and 3 marks on the inside of the glass surface. Image samples of the cartridge are found on the attached DVD described in appendix E.

After segmentation we are only looking for particles with an area between 28 - 4000 pixels found within a number of pixels from the borders of the container. This is to minimize illumination

disturbances caused by the rotating container. A high intensity of the back-light improves contrast and particle segmentation in the border area of the container which is improved for our final test.

All particles detected in all frames are plotted as illustrated in figure 4.12. Here we can see how the two methods are both able to detect the 6 marks on the cartridge. We also see false detections at the bottom and top of the cartridge. Especially for the global segmentation method we have false detections close to the left border of the container. For both methods we are missing matches at the border of the container. As expected we also see tracks formed as ellipses caused by a camera projection from a circular 3D motion to a 2D motion path in the image plane.

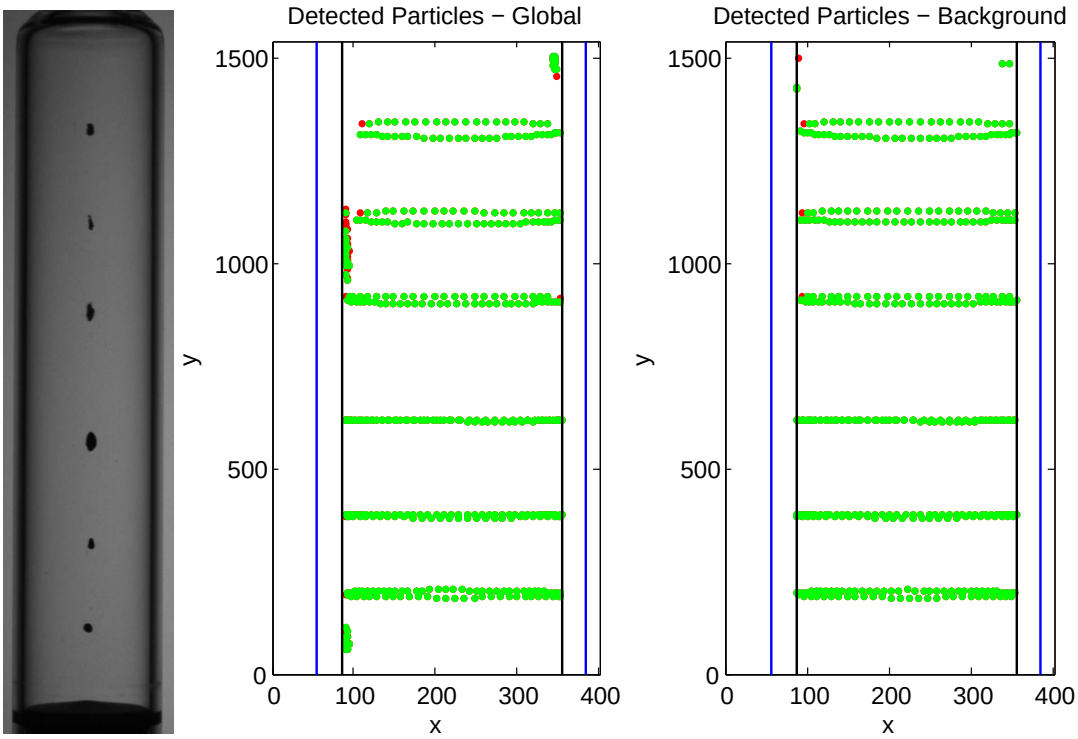


Figure 4.12: Detected particles found in all frames of a rotating cartridge with 6 particles, comparing the global and background segmentation methods. The outer diameter of the container is marked with blue lines and the window for detection is marked with black lines. Red markers indicate that no match in the previous frame was found for the particle blob.

In figure 4.13 we see an important difference between the two segmentation methods when we look at the particle area as a function of the x-position. Here another test cartridge with glycerol (A1) containing two markers of the same area on the inside and outside of the glass surface is tested. As expected the background subtraction method shows that the particle area is largest at the center of the container. When there is liquid in the container we also see the magnification effect of particles on the backside. They have an area between 1400 - 1800 for the background subtraction method. This method has also detected a very small particle at the bottom of the container only visible due to the magnification effect. Finally we observe that particles on the backside occasionally fail to match. This is due to a too small search radius and is only related to the matching by correlation algorithm described in the next chapter.

Our results are summarized in table 4.2 comparing the two methods. Here the number of detected particles is listed for a number of recorded test samples. The test samples A1, CS1, D1 and E1 are described more detailed in appendix D table D.1.These tests are performed by filtering particles within 60 pixels from the borders of the cartridge and from the bottom to limit the number of false detections.

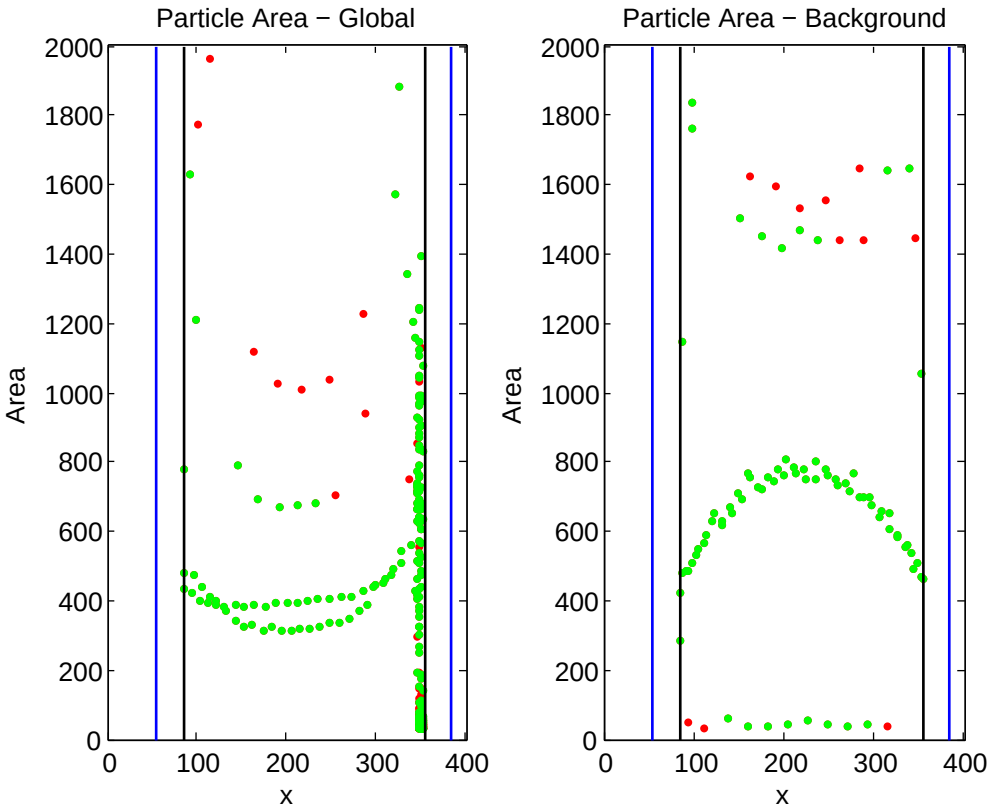


Figure 4.13: Detected particles found in all frames of a rotating cartridge with 2 particles of the same size. The particle area is plotted as a function of the horizontal (x) position on the glass container. Red markers indicate that no match in the previous frame was found for the particle blob.

Test cartridge	**Global Detections**	**Background Detections**
A1	52	60
CS1	249	3053
D1	264	262
E1	198	239

Table 4.2: Particle detection using global or background segmentation.

When we compare the two methods the background method is able to detect more particles than the global method. The background method is able to segment particles closer to the border and bottom of the container compared to the global method. In test case CS1 dust has been adhered to the surface and only the background subtraction method is able to detect a lot of particles as we should expect.

Conclusion: To conclude, we will use the background subtraction method and pre-filter particles in the border area, corresponding to the size of the glass surface in pixels in order to reduce false detections caused by fluctuation of the illumination and vibrations in the rotating container. Background subtraction has the highest number of detections and is able to segment particles close to the bottom and borders of the container and with a correct area. It is assumed that the large amount of false detections can be handled by the tracking algorithm, since this only accepts particles that conform with a motion assumption.

4.3 Particle Tracking

Particle tracking concerns matching found blobs between consecutive images in order to determine how many particles are present and to extract a sequence of two-dimensional positions for each. An overview of the algorithm is illustrated in figure 4.14. When a blob is detected in a particular frame we will search for matching blobs in the previous frame of the video sequence. The result is a list of particle blobs with properties for each frame and a list of matches with the previous frame. Blob matching is performed by using centers and a fixed window size searching for matching particle blobs in the previous frame of the video sequence. Finally particles are linked throughout the entire image sequence. A path of particle movement is extracted for each blob representing a portion of a circular path around the front side of the container.

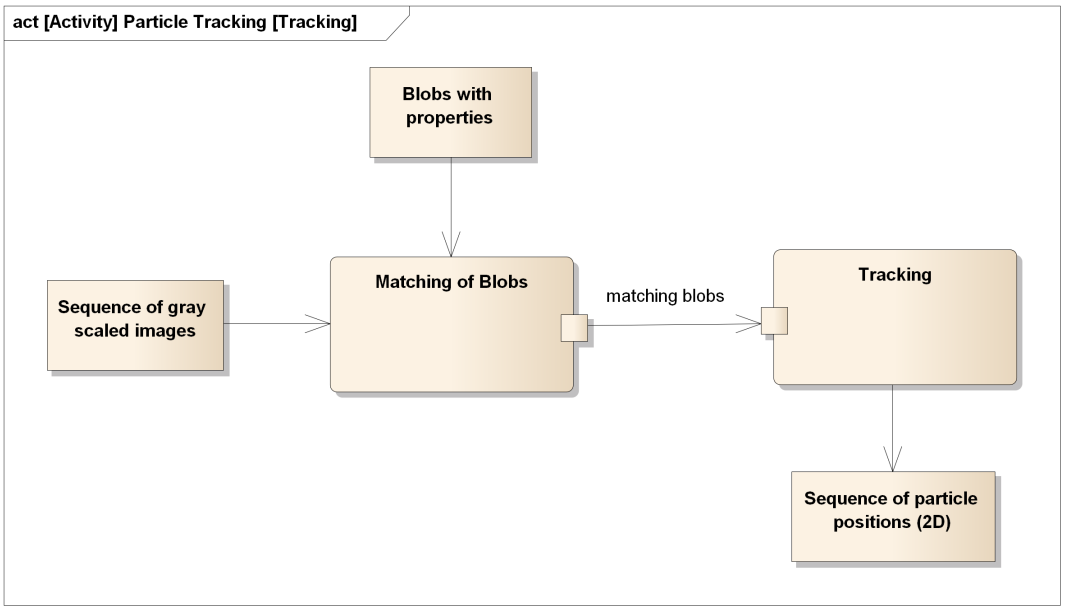


Figure 4.14: Decomposition of algorithms for particle tracking.

The design of each algorithm will be explained in detail in the following chapters followed by an evaluation of the described method for particle tracking.

4.3.1 Matching of Blobs

Matching of blobs follows blob extraction in that we wish to match particle blobs between frames of the video sequence. For every detected blob in a particular frame we will search for matches in the previous frame of the sequence. A special case happens at frame 1 where we search for matches in frame N, such that the tracking becomes circular. In figure 4.15 one frame of the video sequence is shown where matching is based on the previous frame for two different glass containers.

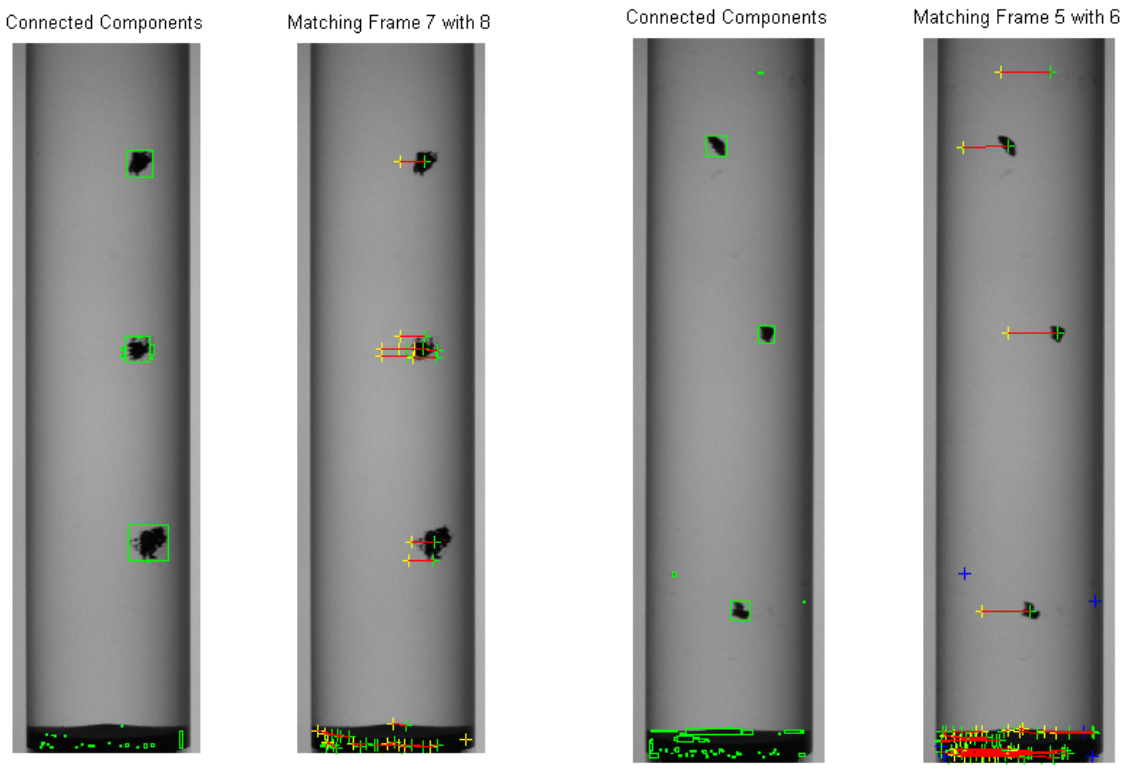


Figure 4.15: Matching of blobs between two frames for two different cartridges with particles. Red lines between a green and a yellow marker illustrates a blob match between a current and a previous frame.

The purpose is to provide accurate information to the tracking algorithm about particle matches using information about the intensity of pixels in a square area of the particle center. The algorithm will search for matches in a specified maximum search radius to the center of the particle. Peter Kovesi [33] has implemented two different matching algorithms where only one of them is applied on our problem. Alternatively our own method, **matchbyarea** that finds the best matching particles by matching the particle area could be used. The two methods are both examples of best-first searches that only choose a single best match from a given heuristic evaluation metric [34]. A more advanced approach would be to use track-splitting where multiple hypotheses are generated in order to include also the second best match and postpone the decision about the best path to a later stage [35].

matchbycorrelation: Matching by correlation generates putative matches between previously detected feature points in two images by looking for points that are maximally correlated with each other within windows surrounding each point. This is a simple-minded normalized correlation measure. Only points that correlate most strongly with each other in both

directions are selected. The algorithm builds a correlation matrix that holds the correlation strength of every point relative to every other point. A correlation matrix is computed as the squared sum of pixel intensities within the window size.

matchbyarea: Matching using particle area compares all particles found in the previous frame within a specified rectangular window. The search window is limited to fit a rectangular area in the previous frame where we would expect to find a matching particle based on a maximum particle displacement. This maximum displacement depends on the image sampling rate and the rotation frequency and the radius of the glass container. The best particle match is found as the best matching particle area within the specified rectangular window. The method is very fast since we only compare particle centers and areas instead of pixel intensities.

In the first evaluation of our method, both match by correlation and match by area have been evaluated. Both methods are efficient and able to match particles from frame to frame. In figure 4.15 the matching is illustrated for the match by area method.

A number of related methods for feature matching exist and are used especially in the field of dense stereo correspondence [24, p. 545-587]. Here similarity measures are used to compare pixel values in order to determine how likely they are to be in correspondence. Various algorithms exist and the most common pixel-based matching algorithms include *sum of squared intensity differences* (SSD), *absolute intensity difference* (SAD) and *normalized cross-correlation* (NCC), the latter behaving similarly to SSD and the method implemented by Peter Kovesi.

The census transform [36] is another alternative that converts each pixel inside a moving window into a bit vector representing which neighbors are above or below the central pixel. This method is quite robust against large-scale, non-stationary exposure and illumination changes. It would be very relevant and efficient [37] to implement on a graphics hardware (GPU) since real-time execution is an issue in our case.

4.3.2 Linking Matches

As described above the most simple form of tracking performs a best-first search on particle blobs between consecutive images. These frame-wise matches can be linked to form a path of blob positions throughout the whole image sequence. This path represents a particle movement corresponding to a portion of a circular path around the container. Subsequently the coordinates can be fitted to a circle, which is described in the next section.

As shown in section 4.2 the particle segmentation finds all sorts of blobs, only some of which being actual particles. The main objective of the simple tracking algorithm is to filter these blobs so only those representing realistic particles are included in the subsequent processing. In this, the following parameters are used to sort out blobs and sequences of blobs:

- **Area**: only blobs with a larger area than 15 px are included. This area roughly corresponds to a circular particle with a radius of 70 μm. By measuring the width of the container as 10.9 mm in metric values and 330 px in the image the area can be calculated as:
  $A = \pi \cdot r^2 = \pi \cdot (70\mu m \cdot \frac{330px}{10.9mm})^2 = 14.1px.$

- **Sequence length**: only blobs that are matched in more than 15 consecutive images are included.[1] This requirement ensures that wrong matches are discarded.

[1] When fewer than 87 frames are available, this requirement is reduced accordingly.

- **Travel distance**: only particles that throughout their sequence travel a distance more than 60 pixels are included. This ensures that non-moving particles are discarded. However, it also means that only particles moving in a circular path with a minimum diameter of $60px \cdot \frac{10.9mm}{330px} = 2mm$ are included.
- **Travel direction**: only particles traveling on the front side of the glass container are desired. Particles traveling on the backside are exposed to refractions in both the glass and the liquid when viewed by the camera, and the particle position and area will therefore be distorted. By requiring a travel direction corresponding to particles moving on the front side of the glass container, problems regarding refraction are minimized.

Figure 4.16 shows tracking sequences from different test units (A1 and D1) plotted on top of an image. The first example shows a sequence with 33 frames distributed around the container, resulting in approximately 12 visible points. The last two examples show two sequences with only 14 frames, resulting in only 4-5 visible points.

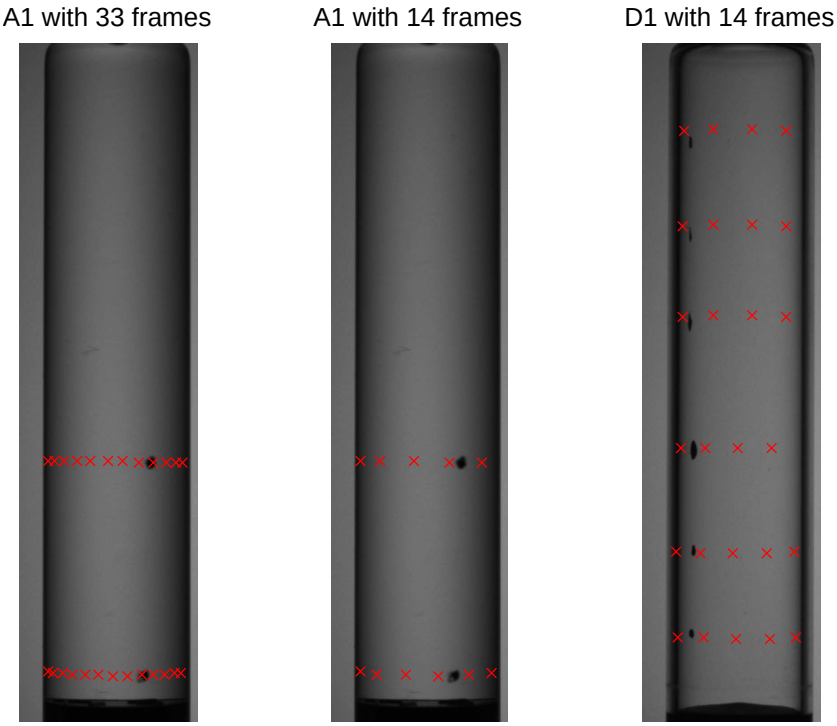


Figure 4.16: Sequences of tracked blobs on the front side of the glass container.

4.3.3 Evaluation

As the most simple form of tracking the described method in this chapter works as a proof of concept. However, the method is extremely sensitive towards uncertainties and variations in the segmentation and matching algorithms. If just one match between blobs in consecutive images is missed, the tracking is lost, and the particle will thus not be detected by the overall system. Likewise if not all blobs in a sequence have areas larger than the minimum limit, the particle will not be detected. A significant improvement to the overall system will therefore be to extend the tracking algorithm with prediction capabilities and tolerances towards missing matches, for instance by using track-splitting. In evaluation of the matching algorithm and deciding on whether to use

matching by correlation or by area the method is analyzed and described in detail in appendix C section C.1.

Conclusion: We conclude that match by area is the best algorithm achieving the highest tracking rate and lowest number of false particle detections on the glass surface. Match by correlation fails compared to match by area for large particles at the bottom of the container with increasing distance between centers in two successive frames.

4.4 Particle Positioning in 3D

As stated in the problem definition the goal of this thesis is to investigate and propose an algorithm that can detect and position particles three-dimensionally in a glass container. From the images the x- and y-coordinates of the particles can be found, and by utilizing the knowledge about the physical movement of the particles either inside or outside the container, the z-coordinates can be estimated. However, an exact estimation of the z-coordinate over time is not the ultimate goal. Instead a radius estimation of the circular path that a particle is moving in is the desired parameter. Thus, the problem of estimating particle positions in 3D comes down to fitting a circle to the measurements of 2D particle position using an assumption of a constant angular velocity during image acquisition.

The relation between the camera frame rate, the container rotation frequency and the number of frames for one observed revolution of the container in the sequence of images was stated in equation 3.3 and is repeated here:

$$f_{cyl} = f_s \cdot m \pm f_s \cdot \frac{1}{n} \tag{4.1}$$

where f_{cyl} is the container rotation frequency, f_s is the camera frame rate, n is the number of frames for a revolution and m is a non-negative integer. The formula states that the container is rotating with a frequency that is a multiple of the camera frame rate plus/minus a fraction of it resulting in n frames for a revolution recorded by the camera.

An estimation of the motion of individual particles can be found by assuming a steady state of the rotating liquid in the glass container during image acquisition. This assumption implies that all particles move with a constant angular velocity in a circular path with a given radius either inside or outside the container. By detecting the same particle in multiple consecutive images, the displacement in pixels can be used to estimate a circular path. Normally this problem will have infinitely many solutions, but if we use the assumption about a constant angular velocity and relate consecutive points, a unique solution can be found. As input a sequence of geometric center coordinates of the tracked particles are used.

When the circular path of the particle movement is in front of the camera, only the x-coordinates of the particles will vary due to the rotation. Thus, instead of estimating a circular path from both the image coordinates, the x-coordinates can be compared to the expected motion under a central force expressed in polar coordinates. The expected motion in the x-axis can be expressed as:

$$\hat{x}_k = r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) + x_c \tag{4.2}$$

This expression states that the k'th coordinate has moved an angle of $\omega \cdot T_s \cdot (k-1)$ relative to the first coordinate and that the first measurement started at an angle of θ_0 relative to the x-axis. Here

$T_s \cdot (k-1)$ is the time stamp of the k'th measurement. By comparing these expected coordinates with the measured x-coordinates an error function can be constructed that we wish to minimize. By writing a sum of squared differences between the measured x-coordinates and the expected coordinates from equation 4.2 we have:

$$f(r, \theta_0) = \sum_{k=1}^{K} (x_k - r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) - x_c)^2 \tag{4.3}$$

The error depends on the measured points and the chosen values for the radius, r, and the initial angle, θ_0. By minimizing the error function subject to r and θ_0 the best fit representing the measured points can be found. The goal is to find:

$$\begin{bmatrix} r & \theta_0 \end{bmatrix}^\intercal = arg\,min\, f(r, \theta_0) \tag{4.4}$$

Different methods can be used to find this minimum. A simple and efficient method is gradient descent with a fixed step size [38]. Here a vector, $\mathbf{a} = \begin{bmatrix} r & \theta_0 \end{bmatrix}^\intercal$, consisting of the two unknown parameters, r and θ_0, is updated iteratively to find the minimum of the error function as:

$$\mathbf{a_{k+1}} = \mathbf{a_k} - \alpha \nabla f(\mathbf{a_k}) \tag{4.5}$$

By continuously taking a small step in the negative direction of the gradient the parameters are gradually modified from a starting guess. The gradient of f is given by

$$\nabla f(\mathbf{a_k}) = \begin{bmatrix} \frac{\partial f}{\partial r} \\ \frac{\partial f}{\partial \theta_0} \end{bmatrix} \tag{4.6}$$

$$\frac{\partial f}{\partial r} = \sum_{k=1}^{K} -2 \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) \cdot (x_k - r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) - x_c)$$

$$\frac{\partial f}{\partial \theta_0} = \sum_{k=1}^{K} 2 \cdot r \cdot sin(\omega \cdot T_s \cdot (k-1) + \theta_0) \cdot (x_k - r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) - x_c)$$

Figure 4.17 shows how two parameters, r and θ_0, evolve for each iteration of the gradient descent algorithm. The partice simulator described in chapter 3.5 is used to simulate 5 particles having different radii, starting angles and y-positions. As input to the algorithm the exact simulated positions shown in figure 3.9 are used. For all particles the colors of the plots match with each other in the figures. As described above a fixed step size is used to find the best fit. In this case individual values for the two parameters are used: $\alpha_1 = 8 \times 10^{-3}$ for r and $\alpha_2 = 1 \times 10^{-6}$ for θ_0. As a starting guess for r, $x_{max} - x_c$ is used.

In figure 4.18 five circles reconstructed from their estimated radii are plotted together with their corresponding measured points, where the z-coordinates are reconstructed with the formula:

$$\hat{z}_k = r \cdot sin(\omega \cdot T_s \cdot (k-1) + \theta_0)$$

Together with the estimated circles the actual outer and inner surfaces of the container are drawn with black dashed lines. The particles represented by the magenta, cyan and red circles are all estimated correctly within an unnoticeable margin. They all have measured points spread all around the container, whereas the particles represented by the blue and the green circles only have visible points of the front side of the container (see figure 3.9). Although a considerable amount of noise is added to the red particles, these are still estimated much more precisely than the blue and the green.

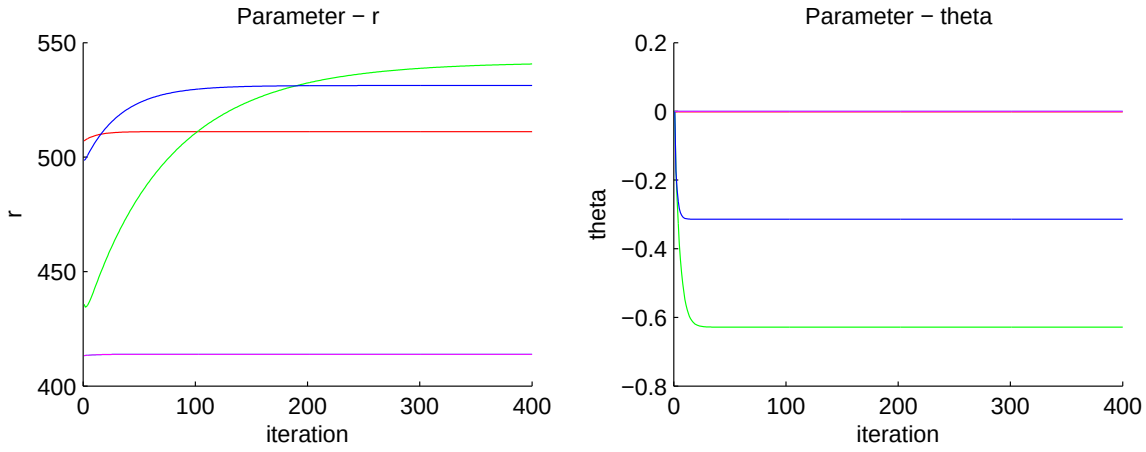


Figure 4.17: Estimation of parameters for circle fitting.

This problem is due to the perspective projection that introduces a distortion of the x-coordinates depending on their varying distances to the camera around the container. A more comprehensive explanation and an algorithm for correcting the coordinates is discussed in appendix C section C.2. However, in practice the perspective correction is not used because it requires exact parameters of the physical environment in order to perform reliably.

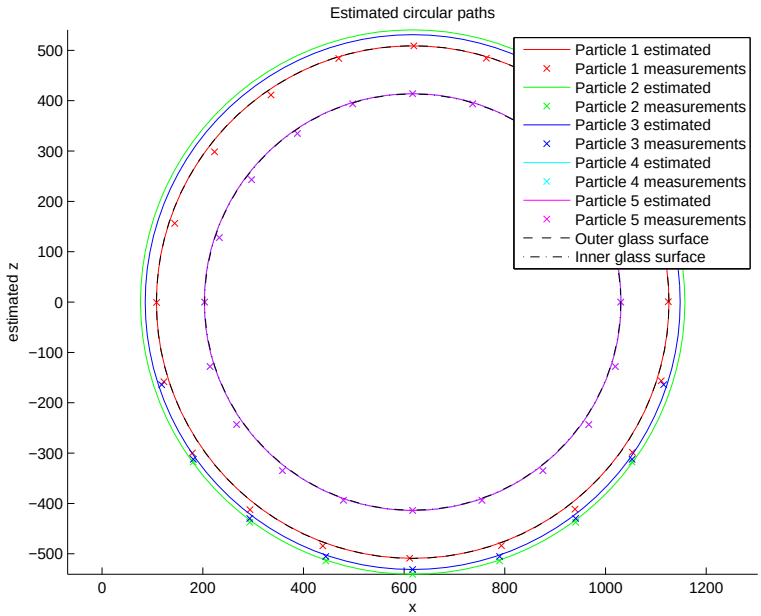


Figure 4.18: Estimated circles from measurements. Together with the circles the measurements are plotted, where the z-coordinates have been reconstructed from the estimated radii.

Choosing the gradient descent method for finding the best fit of a circle to the measurements is justifiable, since the error function has only one minimum when $r \in [0, \infty[$ and $\theta \in [0; 2\pi]$. From equation 4.3 it is clear that the error-function is periodic with 2π and that it is an odd function, because it has another minimum at $f(-r, \theta_0 \pm \pi)$. Figure 4.19 illustrates this with a contour plot of the error-function for the red particle in figure 4.18.

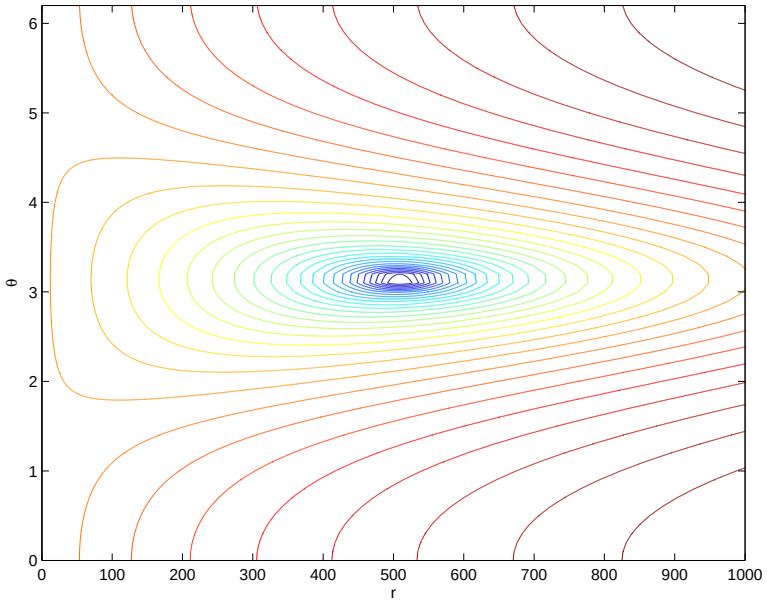


Figure 4.19: Contour plot of the error-function for the positioning algorithm. The error-function has a single minimum when $r \in [0, \infty[$ and $\theta \in [0; 2\pi]$.

4.4.1 Evaluation

For the actual system the most interesting parameter to determine is the number of measurements needed for estimating the radius of a particle. The number of measurements relates directly to the number of revolutions of the container and thereby also the acquisition time per unit. Today InnoScan can handle a maximum of 11 units/s corresponding to approximately 90 ms per unit. With a camera frame rate of 20 fps the acquisition time for e.g. 10 points (frames) around the container is $\frac{10\,frames}{20\,frames/s} = 0.5s$.

Figure 4.20 shows a simulation of two particles - one located on the inner surface of the glass container (green) and one located on the outer surface (red). The mean estimated radii are shown related to how many measurements/frames were used. Only a portion of the measurements corresponding to an angle range of 133° are made visible to the positioning algorithm, so that it only uses particles on the front side of the container. Gaussian noise with a standard deviation of 1.09 is added to the measurements - a value corresponding to the precision of 100 hand-segmentations of the same particle. The black dashed lines indicate one standard deviation above and below the mean estimated radii.

As can be seen from the figure the positioning algorithm only needs 9 points around the container in order to estimate the particle radii such that no confusions between inner and outer particles happen. This corresponds to an acquisition time of $\frac{9\,frames}{20\,frames/s} = 0.45s$. However, this estimate most likely is too optimistic. Uncertainties in the estimation of the container center, x_c, vibrations in the physical setup and uncertainties in the camera frame rate or the container rotation frequency can all contribute to larger errors. Therefore a reasonable estimate might be to double the theoretical limit and use at least 18 frames for each rotating container.

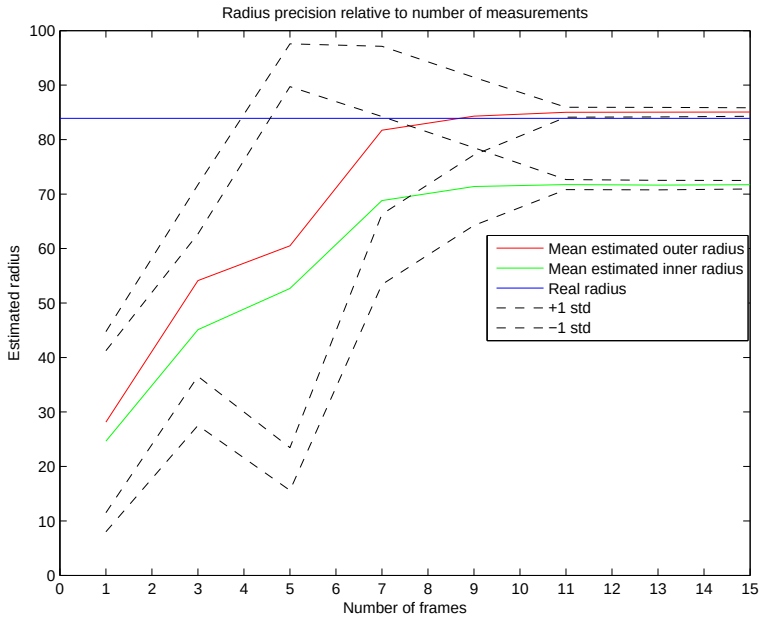


Figure 4.20: Precision of the positioning algorithm based on number of frames captured.

4.5 Classification

The ultimate decision whether a particle should be marked as being on the outside or inside of the glass container is handled by a classifier. The classifier works by taking as inputs the estimated radii for the found particles, and a decision is made by comparing these with a given threshold value. Overall, two configurations of the classifier can be used:

1. A global threshold value used for all particles from all containers. This approach is simple and the threshold value can be trained by using a training set with either supervised or unsupervised learning. However, the method will only work for similar containers that have the exact same diameter and glass thickness.

2. A local value depending on the measured diameter of the container from section 4.2. This approach utilizes information about the container from the images and makes the decision based on this. A global offset to the local decision boundary may be trained by using a training set with supervised learning. The method is robust towards variations in the container diameter and glass thickness and may even be used for completely different types of containers.

Since the second method appears to be the most robust, it has been chosen for the final system design and verification. The decision boundary for a container is described by the equation:

$$r^* = \frac{d_c}{2} + \mathit{offset} \tag{4.7}$$

where r^* is the decision boundary, d_c is the measured container diameter and *offset* is a global offset that can be trained with supervised learning. Since half the measured container diameter corresponds to the outer surface of a container, the most promising decision boundary will probably be slightly smaller, resulting in a negative *offset*. In order to find the best value of this *offset*, a

ROC[2] curve can be generated. A ROC curve is a two-dimensional graph plotting the true positive rate (TPR) as a function of the false positive rate (FPR) [23]. By varying the decision boundary offset, different pairs of (FPR, TPR) can be found. The FPR and TPR can be calculated directly from a confusion matrix as:

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{4.8}$$

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{4.9}$$

In chapter 3.4.2 these categories are described on a container-level - regarding each container as accepted/rejected based on all of its particles. However, when training the classification, the evaluation must be performed on a particle-level. This corresponds to have a container for each particle, so that the decision to accept/reject is based on only one particle. Missing and false detections are caused by the detection part of the algorithm (segmentation and tracking), which is ignored when training and evaluating the classification algorithm. Therefore, only particles that are both observed and detected are included in the calculation of the confusion matrix entries (TP, FP, TN, FN).

In figure 4.21 a ROC curve is generated by varying the *offset* to the decision boundary. The entire training set described in appendix D (table D.2) is used. The blue line denotes the ROC curve, the black box denotes an offset of 0 corresponding to a decision boundary equal to half the measured diameter of the containers, and the red box denotes the optimal offset when no more false positives are allowed than at 0 offset.

From the ROC curve it is clear that using an offset of 0 is not the best solution for this data set. Actually, an increase in the TPR can be achieved without also increasing the FPR. At an offset of -8.9 px (red box) a TPR of 0.91 and a FPR of 0.0 is achieved. This means that no false positives occur at this offset.

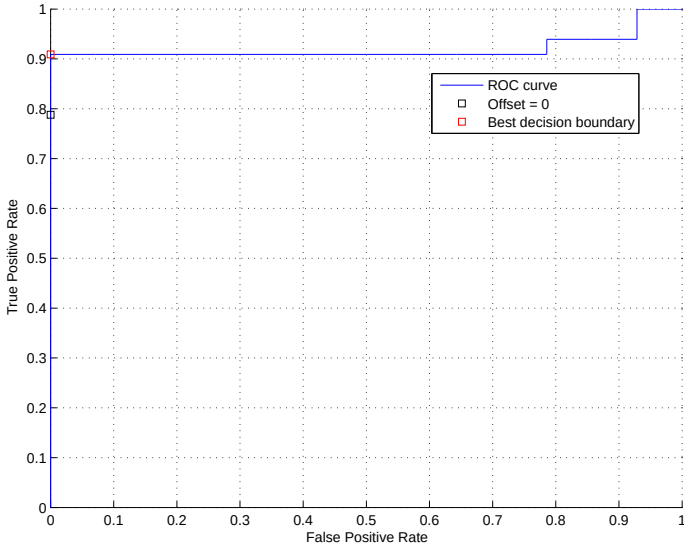


Figure 4.21: ROC curve generated by varying an offset to the decision boundary. The blue line denotes the ROC curve and the black cross denotes an offset of 0 corresponding to a decision boundary equal to the measured radius of the containers.

[2]Receiver operating characteristic

However, a biased estimate is most likely found when estimating the performance of the algorithm on the same data set that was used to train the decision boundary. Therefore, cross validation can be used to assess how the performance will be on an independent data set. A common type of cross validation is K-fold cross validation, where the training set is randomly partitioned into K independent subsets [39, p. 483]. One of the subsets is used as a test set while the remaining subsets are combined as a training set. The classifier is trained K times, each time replacing the test set with another subset. The performance is estimated by averaging over the K found errors, whereas the optimal decision boundary can be found as the decision boundary of the classifier with the smallest error.

Figure 4.22 shows a confusion matrix generated by performing 4-fold cross validation on the training set. Since the classifier is trained on a particle-level, the data set consists of a total of 36 particles that are both observed and detected (and belonging to the 20 containers in the training set).

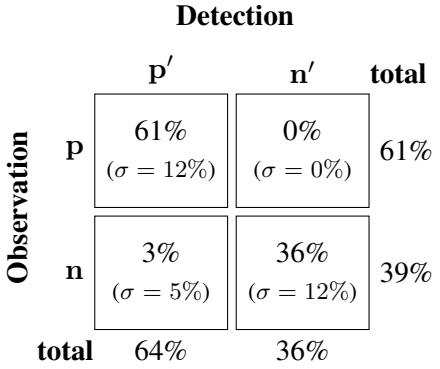


Figure 4.22: Confusion matrix for estimated classification performance on particle-level. The numbers are calculated as mean values over the $K = 4$ iterations. Numbers in brackets specify the standard deviations.

The optimal decision boundary offset is found by choosing the offset from the classifier with the best performance. This is evaluated by calculating the accuracy given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.10}$$

The classifier with the best performance has an accuracy of 1 when using an offset of -8.9 px. This is chosen as the global offset parameter used in the system.

Conclusion: A classifier using a local value depending on the measured diameter of the container is used as a decision boundary. A global offset of -8.9 px has been trained and is added to this value in order to maximize the accuracy of the classification.

4.6 Summary

The first algorithm functions as a prototype of the overall system, containing separate algorithms for segmentation, tracking, positioning and classification. It is evaluated and tested on an actual test set in chapter 7, where both system results as well as the individual algorithm results are presented. From this evaluation it is clear that the main drawback of the first algorithm concerns the timing performance. In the problem definition it is stated that the system should be able to inspect 11 containers per second corresponding to approximately 90 ms of computation time for each container. However, in a purely sequential and non-optimized version of the first algorithm as presented above, the execution time is much higher. Therefore, in the following chapter a real-time design of the first algorithm is presented.

Following this, two advanced tracking algorithms are proposed in order to investigate and solve problems regarding tracking of the particle trajectories. These problems concern missing blob detections due to occlusion and false detections due to noise in the background subtraction.

5

Real-time Design

The goal is to design a real-time system executing an optimized version of the first algorithm achieving a capacity of up to 11 containers per second.

A high speed Basler Camera (avA1000-120km) providing a frame rate of 120 fps and a resolution of 1024x1024 pixels could be used. Would it be possible to realize a real-time system being able to process an optimized version of the algorithm given the limitation of the acquisition rate with 11 – 87 frames for inspection of the glass container? Theoretically the maximum capacity would then be $\frac{120fps}{11frames/container} = 10.9$ containers per second for 11 images and a camera acquisition rate of 120 fps. With 22 or 29 images a higher accuracy is achieved (see results for the first algorithm in chapter 7.1), although in this case the system would only be able to inspect respectively 5.5 and 4.1 containers per second.

The Graphics Processing Unit (GPU) is a powerful and programmable general purpose processor. The GPU has evolved into a manycore processor that supports a highly parallel and multithreaded computational platform. The technology is driven by market demands for real-time and high-definition 3D graphics, especially by the computer gaming industry. Recent models like the GeForce GTX TITAN provides 4500 GFLOPS and 290 GB/s [40]. GPUs are able to solve problems outside their original application domain due to their highly parallel architecture. They are mainly suitable for implementation of data-parallel algorithms. Here the same algorithm is executed on many data elements in parallel, where it is important that the ratio of arithmetic operations is higher than the number of memory accesses.

We would like to design a real-time realization of our proposed algorithm using data-parallel computing, based on a combination of CPU and GPU computation. It should be created as a fast, flexible and extendible design. The algorithm will be modeled and verified using the Parallel Computing Toolbox from MATLAB [41] in designing an optimized version for parts of the first algorithm. It would be possible to increase the performance even more by porting the algorithm to a C++/CUDA [40, 42] implementation reusing the CUDA kernels from the MATLAB model.

The following chapter gives an introduction to how the system could be optimized for a real-time realization. The concept of data-parallel computation will be explained followed by an optimized realization of the background extraction as well as the positioning algorithm.

The final achieved performance results will be presented later in chapter 7 with reference to the first algorithm and an initial optimized algorithm.

5.1 Real-time Optimization

A real-time computer system depends not only on the logical results of the computations, but also on the physical instant at which the result is produced [43]. In our case real-time refers to a soft deadline indicating the time at which computation must be completed after image acquisition. This deadline depends on the speed of the carousel transporting the glass containers. It is not catastrophic if this deadline is missed as long as we are able to inspect every passing glass container. That is, the throughput must be constant, but the latency might vary due to buffered images and postponed processing.

Task parallelism distributes execution processes across different parallel computing cores, whereas data parallelism distributes the data instead. Task parallelism is today realized by the operating system in combination with homogenous multicores. In our experiments a Windows processing platform with a CPU of 6x2 cores is used. Data parallelism in terms of single program multiple data (SPMD) [41] is realized on Graphical Processing Units (GPUs). SPMD means that identical code runs on multiple threads where each thread can have different, unique data for that code. This is the basic programming paradigm for the GPU architecture. What part of the algorithm would be possible to speedup? Amdahl's law [44] specifies the maximum speedup S attainable by parallelizing a serial program is:

$$S = \frac{1}{(1-P) + \frac{P}{N}} \tag{5.1}$$

where P is the proportion of code that can parallelized and N is the number of processors over which the parallel portion of code runs. The maximum speedup attainable by parallelism is the reciprocal of the proportion of code that is not parallelizable. Equation 5.1 says that if 10% of a given code base is not parallel, even on an infinite number of processors it cannot attain more than a tenfold speedup. Therefore the first serialized version of the algorithm must be analyzed for optimization and parallelization.

If a camera is chosen with a frame rate of 120 fps we would only have $\frac{11 frames/container}{120 fps} = 92ms$ of processing time for 11 frames. In this case a speedup of 106 times is required of the algorithm according to table 7.1. Here we have a total serial execution time of 9.8 sec for 11 frames. According to Amdahl more than 99% must be parallelized running on an infinite number of processors. This would be very hard to achieve and therefore alternative methods must be considered.

Using pipeline processing as known from digital logic design or multiprocessor pipelining, a multiple number of processing cores could increase the throughput of the system. In our case the algorithm can inspect a number of glass containers simultaneously. By this approach a longer latency for each inspection would be allowed, since the required average throughput will still be achieved using parallel concurrent inspections running on several cores.

In this section it is discussed how to optimize and parallelize the first sequential algorithm design. The topics to be considered are listed below.

1. **Profiling** and identification of hotspots in the algorithm
2. **Partitioning** parts of the algorithm relevant for data parallel computation
3. Finding **alternative efficient methods** at the **algorithm level** especially for the positioning algorithm
4. **Designing an optimized data parallel** algorithm for combined tracking and positioning

Profiling results of the first algorithm are described and presented in chapter 7.1.2.4. Here the algorithms for extracting the background image and positioning are the most critical parts. The timing performance of calculating the background image will be optimized as a data-parallel GPU implementation. Improving the timing performance of positioning will be achieved by finding a more efficient optimization algorithm for circular radius estimation. These optimizations and improvements will be described in the following sections after an introduction to data parallel computation.

5.2 Data-parallel Computation

In data-parallel computing it is important that the algorithm is designed for parallel computation to achieve an optimal performance. *"Data parallelism refers to the program property whereby many arithmetic operations can be safely performed on the data structures in a simultaneous manner"* [45]. Here we will use the NVIDIA CUDA programming model that is an extension to the C language enabling to write code for the GPU as a data-parallel computing device.

The execution starts on the host (CPU) that copies data to the GPU and invokes a kernel function processing the data in parallel. When a kernel function is invoked, or launched, the execution is moved to the device (GPU). Here a large number of threads are generated to take advantage of multithreading provided by hardware. All the threads that are generated by a kernel during an invocation are collectively called a grid working in parallel on the data structure. Data structures could be vectors or matrices in 2D or 3D. The GPU executes similar threads processing the data in parallel. The kernel functions (or simply kernels) typically generate threads to exploit data parallelism operating on different parts of the data. Here it is important that threads operate as independently as possible. Especially the output result should be stored at different global memory locations. For a matrix multiplication as example, the entire matrix multiplication computation can be implemented as a kernel where each thread is used to compute one element of output matrix $P = M \cdot N$. In this example, the number of threads used by the kernel is a function of the matrix dimension ($n \times n$). This method eliminates sequential loops enumerating all elements of the matrix and reduces the time complexity from a $O(n^2)$ computational problem to $O(1)$ seen from the executing thread's point of view.

5.3 Optimized Extraction of Background Image

The background image is found by using a temporal median filter for every pixel location in a number of frames. The filter computes the median value for every pixel by extracting pixels at the same spatial location from a number of images in the video sequence. Here we have selected 9 images with equal time intervals from the sequence. The median filter is implemented as a CUDA kernel using the Parallel Computing Toolbox from MATLAB to invoke the kernel. The kernel function loads pixels from all 9 images in the z-dimension to local memory. This is to reduce slow global memory access in the following sorting function. It then performs a bubble sort and selects the median value. A large number of threads will in parallel generate a result for every pixel that forms the median filtered background image. A complete code list for the background extraction kernel and MATLAB script to launch the CUDA kernel is found in appendix F section F.2.

Executing this implementation we achieved a background extraction computation time of only 27 ms compared to 12 s with an image size of 415x1580 pixels. We have gained a speedup of more than **400 times** with the CUDA acceleration of the MATLAB model.

5.4 Optimized Positioning Algorithm

Before designing a data-parallel algorithm for tracking and positioning we will optimize the positioning step at the algorithm level. One of the most time consuming parts of the first algorithm is positioning where especially the circular radius estimation using gradient descent is critical. Four alternative optimization algorithms are selected with the purpose of evaluating performance based on 12 particles. The average time for radius estimation of one particle applying different optimization algorithms is summarized below.

Gradient descent (0.81 sec) as used in the first algorithm described in chapter 4.4

Levenberg-Marquardt (0.0064 sec) (Newton) is described in the following chapter 5.4.1

Conjugate gradient (0.074 sec) method is a mixture of steepest descent and Newton's method. Typically it performs better than the steepest descent but not as well as Newton's method. Here we have used the MATLAB Poblano toolbox[1].

Particle swarm (12.9 sec) performs a randomized global search updating a population set of candidate solutions called a swarm.

Genetic algorithm (10.3 sec) is a global search heuristic that mimics the process of natural evolution. It is inspired by phenomena such as inheritance, mutation, selection and crossover.

Particle Swarm and the Genetic algorithm are global optimization algorithms and would be able to solve problems where local minima exist. Although in our case we only have one minimum they are still evaluated since they would be interesting for a data-parallel realization. However, they both perform slower and do not achieve the same accuracy as the other methods.

The Levenberg-Marquardt algorithm achieves a speedup of more than 125 times within the same accuracy as the gradient descent method. Conjugate gradient performs approximately 11 times better. The Levenberg-Marquardt algorithm will be described in detail in the following chapter and will later be used in a combined optimized tracking and positioning algorithm in chapter 6.1.

5.4.1 Levenberg-Marquardt

In chapter 4.4 we described how the steepest descent method is used to estimate the radius of the circular trajectory that a particle is moving in. Steepest descent is not always the most efficient approach because only small steps towards the minimum are used in each search iteration. In Newton's method the first and second derivatives are used in order to gain faster convergence. Newton's method is obtained by a quadratic approximation using the Taylor series expansion of f about the current estimated point $\mathbf{x}^{(m)}$, for the m'th iteration. Applying the First-Order Necessary Condition [38, p. 83] and using the notation $\mathbf{g}^{(m)} = \nabla f(\mathbf{x}^{(m)})$ for the gradient we have an estimate of the minimum at the next iteration:

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - F(\mathbf{x}^{(m)})^{-1}\mathbf{g}^{(m)}$$

where F is the Hessian matrix of f at $\mathbf{x}$. The m'th iteration of Newton's method can be written in two steps as

[1] `https://software.sandia.gov/trac/poblano`

1. Solve $F(\mathbf{x}^{(m)})\mathbf{d}^{(m)} = -\mathbf{g}^{(m)}$ for $\mathbf{d}^{(m)}$
2. Set $\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \mathbf{d}^{(m)}$

Step 1 requires the solution of an $n \times n$ system of linear equations. In [38, p. 162-165] Newton's method is applied for a nonlinear data-fitting problem, also called a *nonlinear least-squares problem*. The problem concerns:

$$minimize \sum_{k=1}^{K} (e_k(\mathbf{x}))^2 \tag{5.2}$$

where $e_k : \mathbb{R}^n \rightarrow \mathbb{R}, k = 1, ..., K$, are error or cost functions. Defining $\mathbf{e} = \Big[e_1, ..., e_K\Big]^T$, we write the objective function as a quadratic function $f(\mathbf{x}) = \mathbf{e}(\mathbf{x})^T\mathbf{e}(\mathbf{x})$. Applying Newton's methods we need to compute the gradient and Hessian of f as done in [38] where Newton's method applied to the nonlinear least-squares problem gives:

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - (J(\mathbf{x})^T J(\mathbf{x}) + S(\mathbf{x}))^{-1} J(\mathbf{x})^T \mathbf{e}(\mathbf{x}) \tag{5.3}$$

where J is the Jacobian matrix of $\mathbf{e}$:

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_1}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial e_1}{\partial x_n}(\mathbf{x}) \\ \vdots & & \vdots \\ \frac{\partial e_K}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial e_K}{\partial x_n}(\mathbf{x}) \end{bmatrix} \tag{5.4}$$

and the matrix $S(\mathbf{x})$ involves the second derivatives of the function $\mathbf{e}$. The (p, j)'th component of $S(\mathbf{x})$ is found by:

$$S_{pj}(\mathbf{x}) = \sum_{k=1}^{K} e_k(\mathbf{x}) \frac{\partial^2 e_k}{\partial x_p \partial x_j}(\mathbf{x})$$

In some applications the matrix $S(\mathbf{x})$ can be ignored because its components are negligibly small. In this case Newton's algorithm reduces to what is commonly called the *Gauss-Newton method*:

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - (J(\mathbf{x})^T J(\mathbf{x}))^{-1} J(\mathbf{x})^T \mathbf{e}(\mathbf{x}) \tag{5.5}$$

A potential problem is that the matrix $J(\mathbf{x})^T J(\mathbf{x})$ may not be positive definite and thus may not be invertible. This problem can be overcome using a Levenberg-Marquardt modification:

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - (J(\mathbf{x})^T J(\mathbf{x}) + \mu_m I)^{-1} J(\mathbf{x})^T \mathbf{e}(\mathbf{x}) \tag{5.6}$$

where I is the identity matrix and μ is a small value ensuring a positive definite (and thus invertible) matrix. In the literature equation 5.6 is referred to as the *Levenberg-Marquardt algorithm*. The term $\mu_m I$ could be viewed as an approximation to $S(\mathbf{x})$ in Newton's algorithm, equation 5.3.

Our goal is to apply Newton's method to our problem of estimating the circular path of a particle's movement. In chapter 4.4 the expected motion of a particle in the x-axis was expressed as:

$$\hat{x}_k = r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) + x_c \tag{5.7}$$

where r is the radius, T_s is the sample time (time between two frames), k is the time step, θ_0 is the starting angle for a particle and x_c is the center of the container. With appropriate choices of

the parameters r and θ_0 we can formulate a data-fitting problem and by applying the *Levenberg-Marquardt algorithm* we obtain a nonlinear least-squares problem with the error function:

$$f(\mathbf{x}) = \sum_{k=1}^{K}(e_k(\mathbf{x}))^2 = \sum_{k=1}^{K}(x_k - r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) - x_c)^2$$

where $\mathbf{x} = [r, \theta_0]^T$ is a vector consisting of the optimization variables. The Jacobian of the error function can be calculated using equation 5.4:

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_1}{\partial r}(\mathbf{x}) & \frac{\partial e_1}{\partial \theta_0}(\mathbf{x}) \\ \vdots & \vdots \\ \frac{\partial e_K}{\partial r}(\mathbf{x}) & \frac{\partial e_K}{\partial \theta_0}(\mathbf{x}) \end{bmatrix}$$

The terms on the two columns can be calculated as:

$$\frac{\partial e_k}{\partial r}(\mathbf{x}) = -cos(\omega \cdot T_s \cdot (k-1) + \theta_0) \qquad \frac{\partial e_k}{\partial \theta_0}(\mathbf{x}) = r \cdot sin(\omega \cdot T_s \cdot (k-1) + \theta_0).$$

We then have a final optimized version of the algorithm by inserting the Jacobian matrix for our problem into equation 5.6 and computing a value for μ whenever the eigenvalues are negative or very small for the matrix multiplication $J(\mathbf{x})^T J(\mathbf{x})$. This algorithm is used to achieve a speedup of more than 100 times within the same accuracy as the gradient descent method finding a local minimum within 4-10 iterations. Here the steepest descent method with a fixed step size can use more than 1000 iterations.

5.5 Summary

To meet the real-time performance requirements of up to 11 containers per second we have proposed a high speed Basler Camera (avA1000-120km) providing a frame rate of 120 fps. With this camera a capacity between 4-11 containers per second would be the theoretical maximum with the proposed algorithm. In this case a container rotation speed close to 7000 rpm is required, instead of the 1186 rpm used in our experiments.

With such a high framerate the algorithm needs to be optimized in order to meet the desired capacity for the whole system. Based on profiling of the first algorithm we have identified the most critical parts of the algorithm which concern extraction of the background image, tracking/-matching and positioning. Here we have suggested a data-parallel design for relevant parts of the algorithm targeting the GPU platform. For the background extraction a speedup of more than 400 times is achieved with a CUDA acceleration of the MATLAB model. Optimization of the positioning algorithm is done by using the Levenberg-Marquardt algorithm achieving a speedup of more than 100 times. The above mentioned optimizations do not have any impact on the accuracy of the algorithm.

A new combined tracking and positioning algorithm will be designed for data-parallel computation on the GPU platform presented in following chapter. Here the goal is to improve both performance and accuracy.

6

Advanced Tracking

In the first tracking algorithm presented in chapter 4.3 different problems and limitations are observed when evaluating the results. These concern:

Noise in the image introduced by changes in illumination or improper blob segmentation. This is especially a problem in the boundary area of the containers. Here the first tracking algorithm often detects invalid tracks when only a limited number of frames is available.

Missing blobs in a particle track. This is especially a problem for small particles and at the border of the container. The blob of a particle is changing as it passes the front side of the container having its maximum at the center of the rotating container. The first algorithm cannot handle missing blobs and will loose tracking.

Self occlusion in the case where we have particles at the same horizontal position of the rotating container. When passing each other on the front and back side particles will occlude each other.

Background occlusion occurs for particles at the bottom area of the container when they are occluded by the robber stopper.

Complex particle shapes for particles of irregular form. The segmentation algorithm will create more blobs from the same particle. This will cause a problem in change of area and center position, and the tracking might get lost.

Real-time processing requirements are high in order to process 11 containers/second. The implementation should be independent of the number of particles and noise measurements present.

The above mentioned problems relate directly to the general problems of multi-target tracking algorithms. Here the problems of noise, occlusion (missing detections), computational complexity and track maintenance are common issues. In the following we present two algorithms trying to solve some of the problems of the first tracking algorithm. First, however, a general and more profound introduction to the problem of multi-target tracking is given.

The aim of a tracking algorithm is to generate the trajectory of an object over time locating its position in every frame of a recorded sequence of images. In this regard the term target is often used for the tracked object, whereas a track refers to the sequence of measurements (the trajectory)

constituting the target. Every tracking method requires an object detection mechanism either in every frame or when the object first appears. The most common approach for object detection is to use information in a single frame as done in the first tracking algorithm. Here background subtraction is used to find any significant changes in an image region from the background model. The output is a number of regions with properties containing the centroid position and area representing the detected objects. In [46] a taxonomy is defined dividing tracking methods into three different categories:

Point Tracking. Objects detected in consecutive frames are represented by points, and the correspondence of points between frames is based on previous object states which can include object features like position and motion.

Kernel Tracking. A kernel refers to a template for the object shape and appearance, where motion is usually tracked in the form of a parametric transformation such as translation, rotation or affine transformation.

Silhouette Tracking. Here tracking is performed by estimating the object region in each frame.

Since point representation is suitable for tracking objects that occupy a small region such as particles, our problem relates to this category. In our case the object is a particle represented by a feature vector including a two-dimensional point location and an area.

Point correspondence in itself can be a difficult problem in the presence of noise and occlusion (missing detections). However when the problem concerns multi-target tracking it is further complicated by the need to associate the right measurements with the right targets (particles). Also, as in our case, targets may both initiate and terminate at any time, thereby introducing a time-varying number of targets.

The point tracking problem can generally be divided into two broad categories defined as *deterministic* and *statistical* methods [46].

Deterministic methods. Deterministic methods for point tracking assign costs of associating measurements to objects. The cost function can be defined by a combination of different constraints, typically based on a known motion model. The point correspondence problem can thus be formulated as an optimization problem. A solution, which consists of one-to-one correspondences chosen from the set of all possible point associations between two frames can be obtained by assignment methods like the Munkres [47] assignment algorithm or greedy search methods.

Statistical methods. Object measurements obtained from the recorded and segmented frames unavoidably contain noise. Statistical methods for point correspondence model the problem probabilistically by taking into account the statistics of the measurement noise and by utilizing a state-space model describing the physics of the system. Each object has a state possibly including position, velocity and acceleration. The state of each object is maintained by a Kalman filter, handling the prediction of future states and the management of noisy measurements. The data association problem is solved by associating measurements to objects based on computed probabilities. Some models include an option for initiating and termination objects continuously, others have a fixed number of tracked objects. In this way the point correspondence problem is solved before applying the Kalman filter to the objects.

In the following chapters two multi-target point tracking algorithms are presented. The **Circular Trajectory Tracker** (CTT) belongs to the family of deterministic methods using a bottom-up

design approach. It is an improved extension of the first algorithm designed for a data-parallel realization to meet real-time requirements. The second algorithm, **Multiple Hypothesis Tracking** (MHT), is widely accepted as one of the preferred methods for complex multi-target tracking problems [48]. It is in the family of probabilistic methods and can be considered as a top-down design approach to our problem. As opposed to CTT it processes frames recursively such that its guess about target states evolves over time. Following the description of the two algorithms an evaluation comparing the first tracking algorithm to CTT and MHT is performed using the particle simulator described in chapter 3.5.

6.1 Circular Trajectory Tracker

In the first algorithm particle blobs are found for each frame and then tracked between two successive frames by finding correspondences of blobs. These frame-wise correspondences are finally linked to form a trajectory of particle movements throughout the whole image sequence. Every track represents a trajectory for a particle movement corresponding to a portion of a circular path around the container. In the first algorithm a simple approach is used to find correspondences using the blob area and a limited search window around the found particle. No consideration of performance is taken into account in the realization of this algorithm and it has a number of limitations as mentioned in the introduction of chapter 6.

A new optimized combined tracking and positioning algorithm is presented where some of the principles from the first algorithm will be reconsidered with the goal of improving performance and accuracy. We would like to develop an algorithm suitable for data-parallel computation in finding radius estimates incorporated into the tracking algorithm of particles as a motion model. Here the basic idea is to apply the optimized Levenberg-Marquardt algorithm described in chapter 5.4.1 on a combination of possible particle tracks found in a sequence of frames. The expected particle trajectory is computed on these possible tracks.

In the following chapter related work will be introduced which is used as inspiration in designing the circular trajectory tracker. In the problem statement the tracking problem is defined followed by two chapters that explain how the CTT algorithm increases noise immunity and handles missing detections. A design is presented that will be efficient for a data-parallel realization on a GPU with the goal of meeting the real-time requirements of the designed system. Finally a brief evaluation of the CTT algorithm is presented that demonstrates its noise immunity and handling of missing detections.

6.1.1 Related work

In [49] a restrained optimal assignment decision tracker (ROAD) is presented that concerns the case where objects are tracked with the positional information as the sole feature for identification. This type of tracking problem is called the *motion correspondence* problem, that is, finding corresponding measurements through an image sequence based solely on the measured point position. ROAD is in the category of deterministic methods developed to establish motion correspondence in the field of multi-target tracking. It expresses predictions about the position of a moving point based on historical track information either using the nearest-neighbor between 2 frames or using a smooth motion model looking at points in 3 successive frames. It is also able to model spurious and missing measurements where interpolation is used to estimate missing points. Finally it computes a global motion model of the overall motion for all tracks to average out individual track motion errors over time. Our presented CTT uses the knowledge of the circular trajectory replac-

ing the need for a global motion model and interpolation in finding missing points. In addition to the positional information it uses the measured particle blob area.

In [50] four alternative heuristics are used to calculate the cost for solving the motion correspondence problem:

- Using 2 frames: Nearest Neighbor (NN) - a point in frame k is matched with the nearest point in frame $k-1$
- Using 3 frames: Minimum Acceleration (3MA) - the position of the point in frame $k-1$ is used along with the current position to estimate a velocity
- Using 4 frames: Minimum Change in Acceleration (4MA) - here the velocity is estimated as in (3MA) together with the smallest change in acceleration using frames $k+1$ and $k+2$
- Using 4 frames: Best Estimate (BE) - extension to the (4MA) method using a cost function for estimating the position in two frames ahead ($k+2$) based on velocity and acceleration

Some of these cost functions could be applied instead of our described method when we do not know the particle trajectory in advance. CTT uses 3 frames to find motion correspondences given the known particle circular trajectory. It defines a cost function similar to 3MA applied on the change in particle size instead of position.

In [51] a framework based on graph theory is used for finding point correspondences in monocular image sequences over multiple frames. Here the problem of finding multiple frame correspondences is formulated as a graph theoretic problem. Multiple frame correspondence relates to finding the best unique path for each point using a number of frames. They use a window of frames in order to handle occlusions for a short duration. In [52] a real-time semi-dense point tracking algorithm is presented to track a high number of points in a video sequence in real-time. They use a GPU handling 10.000 points per frame at 55 fps with a resolution of 640x480 pixels. These papers have been an inspiration for how the CTT algorithm is designed for a fast data-parallel realization.

6.1.2 Problem statement

To formulate a problem by combining tracking and positioning we use the definitions for point tracking algorithms as described in [49], [50] and [51]. However, in the following text it is reformulated such that blob measurements are described by a feature vector composed of a two-dimensional center point and blob area.

Let a sequence of k frames correspond to k time instances $1 \leq k \leq K$, and let $\mathbf{x}_k = \{\mathbf{x}_k(1), \mathbf{x}_k(2), ..., \mathbf{x}_k(M_k)\}$ be the set of M_k points measured at time k. By measurements we mean particle blobs detected by the segmentation algorithm as described in chapter 4.2. Each measured point in set $\mathbf{x}_k$ is a feature vector representing 2D particle blob coordinates and area. A feature vector for point $\mathbf{x}_k(i)$ would look like: $\langle r_k(i), c_k(i), s_k(i)\rangle$ where $r_k(i)$ and $c_k(i)$ represent the row and column position of a blob center and $s_k(i)$ is the blob area. The number of measured points M_k at time k can be either smaller (occlusion or missing detections) or larger (spurious/false detections) than the number of real particles moving in the 3D world.

Multi-target tracking is the problem of finding a set of L tracks that represent the motion of L points through the 2D space from time 1 to K where each track represents a real particle. A track T of length n is an ordered n-tuple of corresponding point measurements
$\langle \mathbf{x}_1(m_1), \mathbf{x}_2(m_2), \ldots, \mathbf{x}_k(m_k)\rangle$, with $1 \leq m_k \leq M_k$.

Let the measurement $\mathbf{x}_k(i)$ denote the i'th position in the k'th frame. A point tracking algorithm then tries to find a point $\mathbf{x}_{k+1}(j)$ for each measurement in $\mathbf{x}_k$ such that $\mathbf{x}_{k+1}(j)$ is the

position of the particle in frame $k+1$ that was at position $\mathbf{x}_k(i)$ in frame k. In order to determine which of all the points in $\mathbf{x}_{k+1}$ to choose, a tracking algorithm defines a cost ϕ_{ij}^k for each pair of $\mathbf{x}_k(i)$ and $\mathbf{x}_{k+1}(j)$. The optimal solution to the tracking problem would make links between points in succeeding frames such that

$$\Phi = \sum_k \sum_i \sum_j \phi_{ij}^k$$

is minimized. This is in general known to be an NP-hard multidimensional assignment problem [49]. This algorithm has an order of $O(n^3)$ time complexity. In our case for 100 frames with an average of 50 points in every frame we have a time complexity of 250.000 iterations to compute just the cost function. A brute-force approach would not be realistic even with a GPU realization. The most common approximation is to restrict the number of frames over which Φ is optimized, referred to as a greedy matching approximation [51].

Most tracking algorithms restrict the $\mathbf{x}_{k+1}$ investigated measured points of possible matches for each $\mathbf{x}_k(i)$ by imposing a limit on the distance a particle can travel from one frame to the next. A conflict occurs when $\phi_{ij}^k = \phi_{nj}^k$ for $i \neq n$, so that two particles in frame k match equally well with the same particle in frame $k+1$. In this context, a tracking algorithm is specified by two parameters: the heuristic used to calculate the cost using a 3-point correspondence between 3 successive frames and a method used to break conflicts. Essential symbols used to describe the circular trajectory tracker are summarized in table 6.1.

k	Time step / frame number
$\mathbf{x}_k$	Set of measured blobs for frame k
M_k	The number of measurements in set $\mathbf{x}_k$
$\mathbf{x}_k(i)$	The i'th measurement in $\mathbf{x}_k$
$\hat{\mathbf{x}}_k(i)$	The i'th estimated point position (x, y) at time k
$r_k(i)$	Measured row-position (y) for blob i at time k
$c_k(i)$	Measured column-position (x) for blob i at time k
$s_k(i)$	Measured area of blob i at time k
$T_k(i)$	Track with end point in frame k at measured blob i
ϕ_{ij}^k	Cost function for 2-point correspondence between 2 frames at time k
ϕ_{hij}^k	Cost function for 3-point correspondence between 3 frames at time k
P_x	A matrix that contains all current particle blob features $c_k(m_k)$ for $1 \leq m_k \leq M_k$
P_y	A matrix that contains all current particle blob features $r_k(m_k)$ for $1 \leq m_k \leq M_k$
P_s	A matrix that contains all current particle blob features $s_k(m_k)$ for $1 \leq m_k \leq M_k$
M_b	A matrix that contains 3-point correspondences in backward direction
M_f	A matrix that contains 3-point correspondences in forward direction
M	A matrix that contains correspondences in backward direction where conflicts are solved
L	A matrix that contains the length of tracks
R	A matrix that contains the radius estimate of tracks
O	A matrix that contains the θ_0 estimate of tracks
Y	A matrix that contains the average y-position of tracks
A	A matrix that contains the average blob areas of tracks

Table 6.1: Symbol list for the CTT algorithm of commonly used notations. All matrices contain the *i*'th parameter (row) at time *k* (column).

The challenges and limitations concerning accuracy and performance for the first algorithm can be reduced to three major areas that we would like to include in an improved algorithm.

1. The algorithm should *increase immunity of noisy blobs* in segmented frames
2. The algorithm should improve particle tracking by *handling missing blobs and occlusion*
3. The algorithm should be *designed for data-parallel computing* to meet the real-time processing requirements

6.1.3 Increase Immunity of Noise Blobs

We will first present the principles for how immunity of noise blobs can be increased. Here a method is presented where more information is used to reduce the search window, and the proposed circular particle trajectory is taken into consideration. The aim is to reduce false correspondences if noise blobs are introduced by illumination or segmentation. In the first algorithm this would be possible since only the blob area and not the position is taken into account.

In the first algorithm the tracking algorithm uses a sequence length to filter valid particle tracks. It filters tracks on their lengths ensuring that blobs matched in less than a number of consecutive frames are ignored. The length is set to accept only valid particle tracks and is adjusted according to the number of frames. In the first algorithm a requirement of at least 3 consecutive measurements is used when only 15 frames are available. For 11 and 13 frames at least 2 consecutive measurements are required. If we use 3 frames instead of 2 we are able to improve the first algorithm in finding correspondences between 3 successive frames. To increase accuracy we will search for a correspondence of a particle blob $\mathbf{x}_k(i)$ in the previous frame $\mathbf{x}_{k-1}$ and in the next frame $\mathbf{x}_{k+1}$.

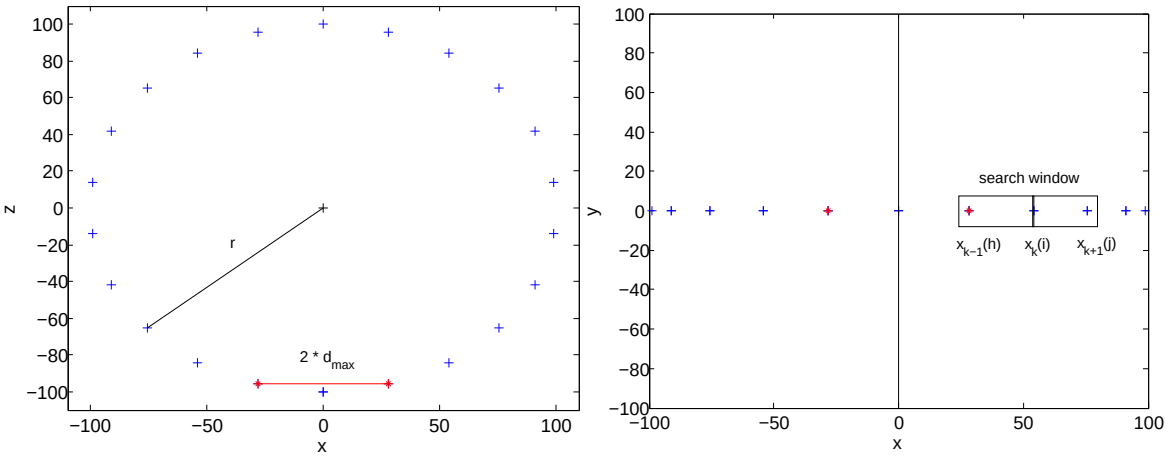


Figure 6.1: Top and front view of the container with one particle moving in a radius (r) with 22 measurements distributed equally around the container. The black line in the left figure indicates the center line of the rotating container. A search window indicates the area to constrain search for 3-point correspondence.

Figure 6.1 at the left illustrates the top view of a container with a particle located in radius r from the center of the rotating container. Its position is marked with a blue cross in every frame with 22 measurements distributed equally around the container. From the front view of the container we observe an acceleration of the particle position when it is located at the left side of the center line and a deceleration at the right side of the center line. This observation could be used

in limiting the search for correspondences between 3 successive frames. For a measured particle blob $\mathbf{x}_k(i)$ in frame k we can write for the particle column position:

$$c_k(i) - c_{k-1}(h) < c_{k+1}(j) - c_k(i) \qquad when \qquad c_k(i) < c_{cl} \tag{6.1}$$

$$c_k(i) - c_{k-1}(h) > c_{k+1}(j) - c_k(i) \qquad when \qquad c_k(i) > c_{cl} \tag{6.2}$$

where c_{cl} is the column position of the center line of the container. In appendix G section G.1 a proof of the above equations 6.1 and 6.2 is given. Equation 6.1 says that for points found at the left side of the center line an acceleration between 3 successive frames should be used. Equation 6.2 says that for points found at the right side of the center line a deceleration should be used. In practice, however, for correspondences of points lying within a window from the center line we do not limit our search due to inaccurate measurements. Here a window of $2 \cdot d_{max}$ is used where d_{max} is the maximum particle movement between two frames. d_{max} can be found using the container rotation frequency f_{cyl} and camera sampling frequency f_s together with the measured radius r of the container. We can find d_{max} given the circular trajectory by:

$$d_{max} = 2 \cdot a \cdot r \cdot sin(\frac{\omega}{2}), \qquad \omega = 2\pi \frac{f_{cyl}}{f_s}$$

where $a > 1$ is a constant to be adjusted assuming that particles will be located close to the surface of the container. In our experiments this value is set to $a = 1.3$. Instead of using a fixed search window size, as done in the first algorithm, we could also limit this window taking the position into account. In figure 6.1 a search window is illustrated to constrain the search for 3-point correspondences. The search window can be squeezed depending on the blob position. Given the circular trajectory (see figure 6.2) a minimum search window in the x-direction at frame k is found by:

$$d_k = a \parallel c_k(i) - \hat{c}_{k+1}(i) \parallel = a \parallel c_k(i) - r \cdot sin(\theta + \omega) \parallel$$

where $\hat{c}_{k+1}(i)$ is the estimated position in next frame and θ can be found using trigonometry of the blob position relative to the center line as:

$$\theta = cos^{-1}(\frac{c_k(i) - c_{cl}}{r})$$

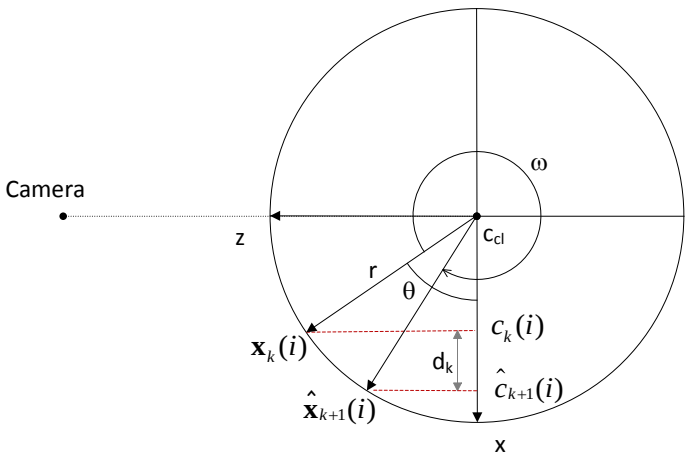


Figure 6.2: Top view of the rotating container with trigonometry for computing the minimum search window given the circular particle trajectory.

To summarize, we will search for corresponding points in 3 successive frames using a search window depending on the position relative to the center line of the rotating container and the knowledge about acceleration and deceleration in speed. The best correspondence is found by defining a cost function for the blob area using a second derivative approximation:

$$\phi_{hij}^{k} = s_{k+1}(j) - 2 \cdot s_k(i) + s_{k-1}(h) \tag{6.3}$$

Here we minimize the change in area over 3 frames. According to the observation in figure 4.13 the area changes as a sine function. Therefore the above approximation will be appropriate for minimizing the change in blob area.

6.1.4 Missing Blobs and Occlusion

Based on correspondences between 3 successive frames a number of tracks will be found by linking point correspondences to form a trajectory of the particle movement. In the following, this part of the algorithm will be called the *3-point correspondences and linking*. A number of possible tracks will be found using the linking approach as described for the first tracking algorithm. The result will be a number of particle tracks where the estimated radius and θ_0 are found using the Levenberg-Marquardt algorithm (see equation 5.6). An estimate of the y-position for a given track is found as the mean of the measured y-positions. If a blob measurement for a real track is missing in one frame the result could be two similar detected tracks as illustrated in figure 6.3.

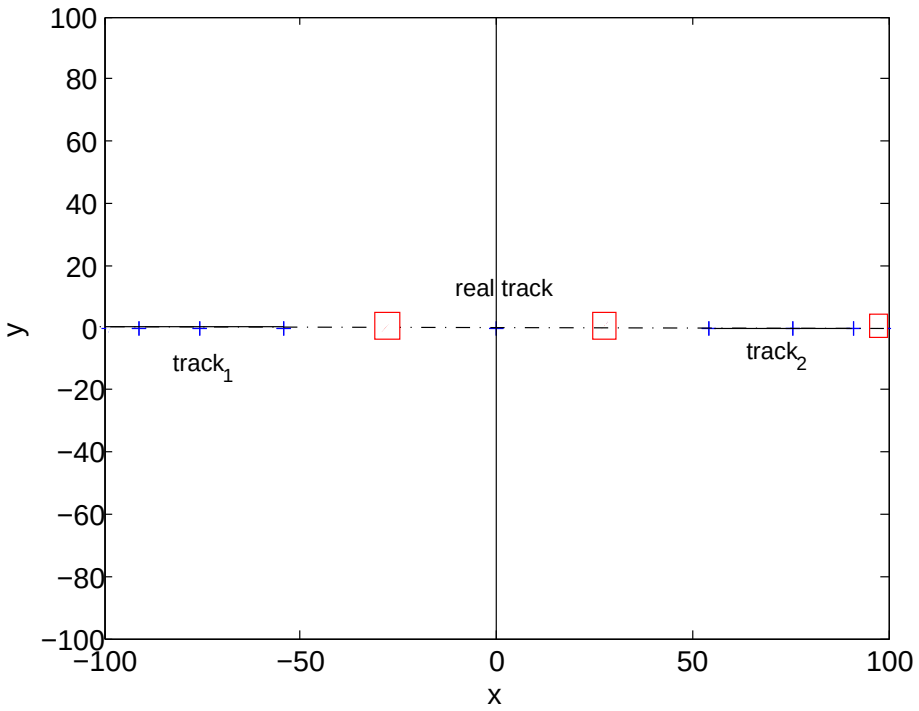


Figure 6.3: A real particle track is found as two tracks since 3 measured points are missing (marked with red squares).

To handle this situation an extension of the first algorithm is proposed. The idea is to compute the expected particle trajectory based on the initial possible tracks found by the 3-point correspondences and linking algorithm. For every track found, the expected particle trajectory for points on

the front side are estimated by using the input estimate of radius r, θ_0 and mean row position y.

$$\hat{\mathbf{x}}_k(i) = \begin{bmatrix} r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) + x_c \\ y \end{bmatrix}$$

where $\hat{\mathbf{x}}_k(i)$ is the estimated point position (c, r) in the current frame k. This is an approximation assuming that the perspective camera projection only has a minimum impact on the y-position.The blob area $s_k(i)$ is not used and therefore omitted from the estimated feature vector $\hat{\mathbf{x}}_k(i)$.

The algorithm will then search for corresponding points in all frames and compute a cost function that defines how well measured points are fitting the expected particle trajectory. The outcome is a number of tracks where some of them contain the same measured points. These tracks are finally filtered selecting the ones with a minimum cost. For instance, in figure 6.3 two tracks will be found containing all measured points together with estimated missing points. However, only the one with the minimum cost will be selected. By this method we will be able to cope with missing particle blobs and occlusion as long as at least 3 corresponding points are found in a particle track. The immunity for noisy blob measurements is also expected to increase, since only measurements fitting with the estimated circular trajectory are chosen as correspondences.

In the following chapters the algorithm is described in detail with the purpose of designing a data-parallel tracking algorithm.

6.1.5 Data Parallel Design

The goal is to create a design that meets the real-time requirements as described in chapter 5. In the work of designing a new improved tracking and positioning algorithm a bottom-up approach is used. Having the data-parallel computation model in mind we will define a model for our tracking problem that is suitable for a data-parallel implementation. A number of kernels will be designed with the purpose of parallel execution on a GPU. Increasing the number of particle blobs in each frame should not increase the execution time since more threads would automatically be instantiated on the GPU. In theory we would have one thread processing one particle blob or one particle track.

The GPU executes similar threads processing the data in parallel. For data-parallel computation it is important to design data-structures that are suitable for a kernel execution. Here we will form three matrices P_x, P_y and P_s all of dimension $M_{max} \times K$, where M_{max} is the maximum number of particle blobs in any frame and K is the number of frames. Each matrix represents its own blob feature from $\mathbf{x}_k(i)$ as defined in the problem statement. In the following nomenclature of a measurement $\mathbf{x}_k(i)$, i relates to the matrix row position and k to the column position.

Since the recorded frames contain samples distributed equally around the glass container our search for tracks starts with 3-point correspondences between successive frames $k-1$, k and $k+1$. The search is made circular such that the last frame K is linked to the first frame 1. The algorithm computes a result in a number of matrices with dimensions $M_{max} \times K$ as listed below:

- L contains the lengths of the tracks
- R contains the radii estimates of the tracks
- O contains the θ_0 estimates of the tracks
- Y contains the average column (y) positions of the tracks
- A contains the average blob areas of the tracks

The matrix L will have a non-zero entry for each track positioned at the k'th column indicating the time of the last point correspondence. A non-zero entry in the L matrix indicates a valid track $T_k(i)$ ending in frame k using the i'th measurement. The other matrices contain the computed properties for the tracks at the same location. The length will be positive if a track is found on the front side of the container and negative when moving in the opposite direction on the back side.

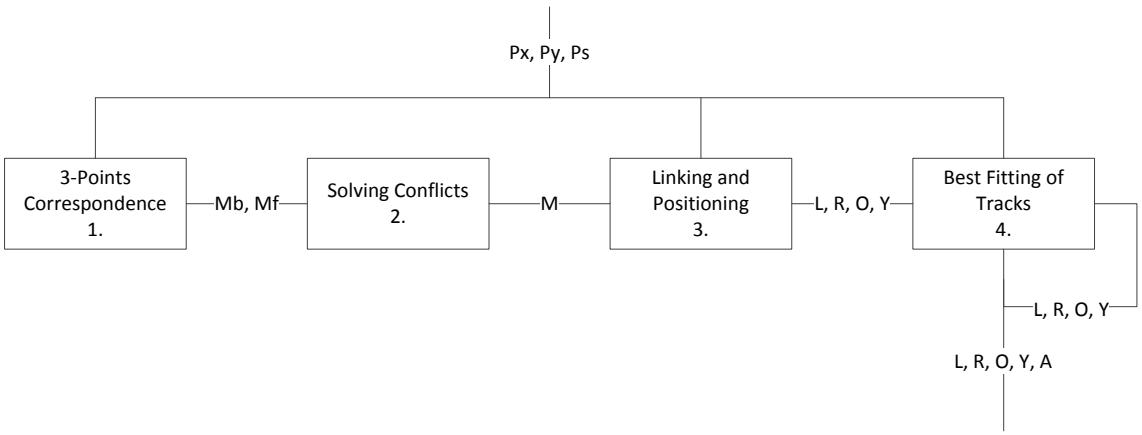


Figure 6.4: Data-parallel algorithm composed of kernels performing an optimized circular trajectory tracking.

The algorithm is composed of a number of kernels as illustrated in figure 6.4 that when executed in sequence produce the resulting L, R, O, Y and A matrices. As input it takes the blob features specified in the matrices P_x, P_y and P_s.

The first, second and third kernels are designed for data-parallel computing. These kernels include the improvements concerning immunity of noise and are expected to have a higher performance even without an implementation in CUDA. The goal for the fourth kernel is to increase accuracy by improving the particle detection rate handling missing blob detections and occlusion. A summary of each kernel is listed in the following.

The first kernel finds 3-point correspondences and produces the matrices M_b and M_f with the index of corresponding feature points in one frame backward $(k-1)$ and one frame forward $(k+1)$. It uses the knowledge about the circular trajectory and a cost function ϕ^k_{hij} using the blob area as described in chapter 6.1.3.

The second kernel solves conflicts by using the matrices M_b and M_f as input. It search for correspondences between frame k and $k-1$ using the indices in M_b and M_f. The output matrix M contains indices for backward correspondence for where tracks can be found. Conflicts are solved ensuring that a point that has a match in forward direction also has a corresponding match in backward direction.

The third kernel finds tracks $T_k(i)$ based on measured feature points in P_x, P_y and P_s and correspondences found in M. It uses a similar linking approach as described for the first tracking algorithm followed by an estimation of the particle position. The result is a number of initial tracks specified in the matrices L, R, O and Y.

The fourth kernel refines the estimated tracks found by the first three kernels. This is done by fitting all measured points to an expected track based on a first estimate of the circular particle trajectory. The method is able to handle missing blob detections and occlusion as

described in chapter 6.1.4. It filters tracks belonging to the same particle trajectory and selects the best fit based on two cost functions.

The design of each kernel will be described in the following chapters illustrated with a simple example.

6.1.5.1 First Kernel – 3-Point Correspondence

The first kernel searches for correspondences one frame backwards and forward within a rectangular window defined by a limitation of the particle movement in the x- and y-direction as described in chapter 6.1.3. It finds correspondences within the window using the particle blob area by minimizing a cost function as described in equation 6.3. The purpose is to find corresponding points in the matrices P_x, P_y and P_s such that the cost function ϕ^k_{hij} for each 3-point correspondence of $\mathbf{x}_{k-1}(h)$, $\mathbf{x}_k(i)$ and $\mathbf{x}_{k+1}(j)$ is minimized.

The result is a matrix M_b that contains a row index for a corresponding point in the previous frame $(k-1)$ and a matrix M_f that contains a row index for a corresponding point in next frame $(k+1)$.

In the example below we have a sequence of $K = 11$ frames where only the first 6 frames are shown. Each frame has 700×1624 pixels with a maximum length of $M_{max} = 6$ measurements for a frame. P_x, P_y and P_s are the matrices containing the blob input center points and area. The 3rd and 6th frame have 6 measurements, the others only have 5.

$$P_x = \begin{bmatrix} 755 & 690 & 538 & 644 & 603 & 456 \\ 617 & 458 & 749 & 436 & 643 & 716 \\ 481 & 576 & 502 & 504 & 476 & 460 \\ 617 & 722 & 791 & 568 & 804 & 564 \\ 432 & 804 & 684 & 770 & 742 & 708 \\ 0 & 0 & 443 & 0 & 0 & 797 \end{bmatrix} \quad P_y = \begin{bmatrix} 1390 & 264 & 812 & 812 & 618 & 239 \\ 448 & 812 & 75 & 434 & 1358 & 1178 \\ 1004 & 1006 & 261 & 619 & 428 & 1366 \\ 1201 & 1199 & 1193 & 82 & 1000 & 424 \\ 812 & 1380 & 1006 & 1004 & 812 & 619 \\ 0 & 0 & 622 & 0 & 0 & 812 \end{bmatrix}$$

The matrix M_b contains the row indices in frame $k-1$ for the best 3-point correspondences in backward direction, and the matrix M_f contains the indices in frame $k+1$ for the best correspondences in forward direction. When no matches are found within the rectangular search window a zero is inserted.

In frame $k = 3$ we have found a backward (M_b) correspondence for blob $i = 1$ with blob $h = 2$ in frame $k = 2$. The feature vectors for the corresponding blobs found in M_b are $\mathbf{x}_3(1) = [812, 538, 15]$ and $\mathbf{x}_2(2) = [812, 458, 15]$. They are located at the same row position ($y = 812$), and the column (x) position indicates that the particle $\mathbf{x}_3(1)$ is moving from left to right (front side) since we have increasing x-coordinates. Using forward correspondence, in frame $k = 3$ we find in M_f a correspondence for blob $i = 1$ with blob $j = 1$ in frame $k = 4$. The feature vector for this is $\mathbf{x}_4(1) = [812, 644, 15]$. The x-position is still increasing indicating that we have a point on the front side.

$$P_s = \begin{bmatrix} 15 & 16 & 15 & 15 & 15 & 15 \\ 16 & 15 & 16 & 15 & 16 & 16 \\ 15 & 15 & 16 & 15 & 15 & 16 \\ 15 & 15 & 15 & 16 & 15 & 15 \\ 16 & 15 & 15 & 15 & 15 & 15 \\ 0 & 0 & 15 & 0 & 0 & 15 \end{bmatrix}$$

$$M_b = \begin{bmatrix} 0 & 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 6 & 2 & 0 \\ 1 & 4 & 0 & 0 & 0 & 3 \\ 0 & 0 & 3 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad M_f = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 1 & 4 & 0 \\ 4 & 4 & 0 & 0 & 0 & 5 \\ 0 & 0 & 5 & 0 & 6 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

6.1.5.2 Second Kernel – Solving Conflicts

Based on the input results of the 3-point correspondences given in the forward and backward matrices, conflicts are solved. We will only search for tracks where we have a correspondence in both backward and forward direction as illustrated in figure 6.5.

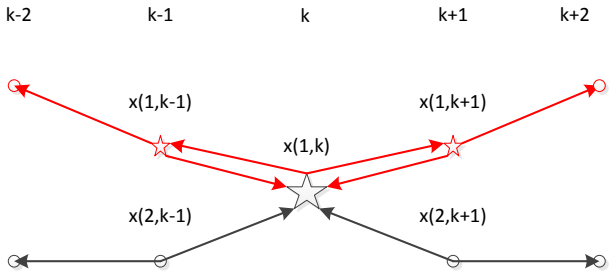


Figure 6.5: A point in frame *k* with 3-points correspondence forward and backward. The track will only be accepted if a correspondence in both directions is found (illustrated with red).

The result is a matrix M that contains backward corresponding indices for where a match is found in both directions. In case a forward or backward correspondence is found but no other point correspondences exist in the opposite direction, the point is also accepted. The result is a matrix M with backward correspondences where conflicts have been solved.

In the example below no conflicts were found and the M matrix is similar to the M_b matrix, except for situations marked with red where a forward match is found without a corresponding backward match. For $M(1,1) = 4$ this is due to a corresponding forward match in frame 11 which is omitted from the example.

$$M_f = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 1 & 4 & 0 \\ 4 & 4 & 0 & 0 & 0 & 5 \\ 0 & 0 & 5 & 0 & 6 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 4 & 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 6 & 2 & 0 \\ 1 & 4 & 4 & 0 & 0 & 3 \\ 0 & 0 & 3 & 5 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

6.1.5.3 Third Kernel – Linking and Positioning

The third kernel searches for possible tracks based on input from the M matrix. A track end point is found in M if there is a correspondence with a point $\mathbf{x}_k(i)$ in the previous frame $\mathbf{x}_{k-1}(h)$ and no point correspondence exists in the next frame $\mathbf{x}_{k+1}(j)$. When an end point is found a track is created by linking backwards in the M matrix. By linking we continue to add points to the track as long as a non-zero entry is found in M. In figure 6.6 it is illustrated how the matrix M is linked based on the example in the previous chapter. Tracks are created for the blue, yellow, red and green lines indicated in the matrix L with the lengths of the tracks located at their end point

$T_k(i)$. Because the yellow track continues in forward direction, we do not have an entry in the matrix L. One track composed by the matrix entries (2,4), (3,5) and (4,6) is not visualized in M but continues in forward direction like the yellow track.

$$M = \begin{bmatrix} 4 & 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 6 & 2 & 0 \\ 1 & 4 & 4 & 0 & 0 & 3 \\ 0 & 0 & 3 & 5 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Figure 6.6: Linked tracks found in the matrix *M*. A blue, yellow, red and green track is found on the front side of the container.

The lengths of the tracks are inserted into the L matrix with the number of links.

$$L = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

A track is created based on the same conditions as for the first algorithm described in chapter 4.3.2. Here heuristics are used to filter blobs and tracks so only those representing realistic particles are found. Limitations in **area**, **sequence length** and **travel distance** are used. In addition to the **travel distance** a short track can still be accepted if it is close to the center of the rotating container.

A special situation is handled in the case where an end position for a track is not found. This would be the case for a particle where the first kernel finds correspondences between all successive frames. To handle this situation a search for circular tracks is performed for points in the last frame K. If a circular track is found, only the trajectory of the track in the travel direction of the front side of the container is used.

A particle's position is estimated along the x-axis as a harmonic motion with a known angular velocity ω given the camera frame rate and the rotation frequency of the container. An estimate of the radius r and starting angle θ_0 is finally computed using equation 6.4 and the Levenberg-Marquardt algorithm as described in 5.4.1:

$$\begin{bmatrix} \hat{r} & \hat{\theta_0} \end{bmatrix} = arg \min_{r,\theta_0} \sum_{k=1}^{K} (x_k - r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) - x_c)^2 \tag{6.4}$$

where x_c is the center and r is the radius of the rotating container. Here the error is minimized using the assigned track measurement x_k in frame k relative to the estimated x-position, where $T_s \cdot (k-1)$ is the time stamp of the k'th measurement in a track of length K. The first measurement started at an angle θ_0 relative to the x-axis. The estimates $\hat{r}$ and $\hat{\theta_0}$ are inserted into the matrices R and O at the end locations of the tracks $T_k(i)$. A mean value of the y-position and area of a track are inserted into the matrices Y and A at the same location. Only tracks on the front side are inserted. The matrices below illustrate the resulting R and Y.

$$R = \begin{bmatrix} 193 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 191 & 0 & 0 & 0 \\ 0 & 0 & 0 & 189 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 187 \end{bmatrix} \quad Y = \begin{bmatrix} 1392 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1197 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1004 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 812 \end{bmatrix}$$

6.1.5.4 Fourth Kernel – Best Fitting of Tracks

The method is inspired from the RANSAC [31] algorithm but instead of using a random subset for an initial model estimate, a model for each track is initially estimated using the track parameters found by the first three kernels. Inspiration has also been taken from Kalman filtering where unknown model variables are estimated based on a recursive two-step process in first predicting the states using a physical model and secondly in the update step comparing the prediction to measurements. Kalman filtering produces estimates of the current state variables along with their uncertainties using a Gaussian distribution. In the following presented method a best fitting approach is used instead of predicting uncertainties, but the basic idea of a physical model where the unknown model variables are updated in each iteration is kept.

The main idea with this kernel is to improve the initial result produced by the first three kernels. If we adjust the heuristic parameters to a minimum by accepting tracks with a short **sequence length** and **travel distance** we will probably increase the particle detection rate but also the number of false detections. If we could somehow improve the result by filtering the number of false detections without decreasing the particle detection rate, a better result could be achieved.

The idea is to use the estimated model parameters, radius (r) and angle (θ_0) for every track found by the third kernel. We will then compute an estimate for the circular trajectory for points on the front side of the container and search for the best fit of measurements.

If a point measurement is missing in frame k the estimated point will be used. A number of cost functions are defined to find the tracks with highest probability of representing a real particle detection.

An initial track $T_k(i)$ is formed by the first kernels producing the matrices R, O, Y and L as input to the best fitting kernel described in the following. For every track found in L the expected particle trajectory for points on the front side are estimated using the input estimate of radius r, θ_0 and mean row position y.

$$\hat{\mathbf{x}}_k(i) = \begin{bmatrix} r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) + x_c \\ y \end{bmatrix} \tag{6.5}$$

where $\hat{\mathbf{x}}_k(i)$ is the estimated position of a point in the circular track at time k. The blob area $s_k(i)$ is not used. However, future extension of the algorithm could include a mean estimate of the measured blob area. An estimated track consists of all points moving from left to right, such that:

$$\hat{T}_k(m_k) = \langle \hat{\mathbf{x}}_1(m_1), \hat{\mathbf{x}}_2(m_2), \ldots, \hat{\mathbf{x}}_k(m_k) \rangle \tag{6.6}$$

The estimated position $\hat{\mathbf{x}}_k(i)$ is used in a cost function using the nearest neighbor of point correspondence between measured and estimated position

$$\phi_{ik} = \| \begin{bmatrix} r_k(i) & c_k(i) \end{bmatrix}^T - \hat{\mathbf{x}}_k(i) \| \tag{6.7}$$

where ϕ_{ik} is the euclidean distance between the measured and estimated point position given by the circular particle trajectory from equation 6.5. The cost is only computed for points lying within

a defined window of the estimated position of $\hat{\mathbf{x}}_k(i)$. This window is set to a maximum radius of $R_f = 12$ pixels from the estimated position. In a future extension of the algorithm the window could be adjusted to cover the maximum variance of the y-position and the uncertainty of the blob center measurement.

The algorithm includes all frames and estimates a position of missing blob detections. If no corresponding points are found within the window, the estimated point $\hat{\mathbf{x}}_k(i)$ is used in forming the track. New estimates of the radius $\hat{r}$ and angle $\hat{\theta_0}$ are finally computed using equation 6.4 and the Levenberg-Marquardt algorithm for the new track $T_k(i)$.

Two cost functions are defined and used to filter valid tracks. The first cost function computes a value between 0 and 1 indicating how accurate the measured points fit the track

$$\phi_{Tf} = \frac{\sum_k \phi_{m_k k}}{M_m R_f} \tag{6.8}$$

Here M_m is the number of measured points found fitting the track and R_f is the radius of the defined search window. If the computed fitting cost ϕ_{Tf} is below a predefined threshold the track will be accepted as valid. The second cost function computes a value between 0 and 1 indicating how many measured points were found relative to the expected number of points on the front side of the container:

$$\phi_{Tl} = 1 - \frac{M_m}{M_e} \tag{6.9}$$

where M_e is the number of estimated points on the front side. If similar tracks are found including at least two of the same points, the track with the minimum cost is selected.

A simplified pseudocode of the fourth kernel is summarized in algorithm 2 below. Each thread that the kernel initiates will first compute the entry of the matrix L corresponding to the specific thread. If a track end point is found it starts to construct the expected track $\hat{T}_k(i)$ for particle blob positions on the front side. Then it searches for measured points that fit the expected track, and finally the track cost is computed. If the cost is below a configured threshold the track is accepted.

In summary the method uses information of the initial tracks found by the first, second and third kernels. It estimates track points $\langle \hat{\mathbf{x}}_1(m_1), \hat{\mathbf{x}}_2(m_2), \ldots, \hat{\mathbf{x}}_k(m_k) \rangle$ based on initial radius and angle estimates. The algorithm uses the estimated track to search for correspondences between estimated and measured points in frames belonging to the front side of the circular particle trajectory. It then iterates a specified number of times between finding point correspondences and re-estimating the circular trajectory. Finally it uses the two cost functions to filter tracks ensuring that only real particles are detected by eliminating duplicate tracks. This is done by first evaluating the track length cost ϕ_{Tl} and secondly the track fitting cost ϕ_{Tf} and selecting the best track if a measured point belongs to different detected tracks. In this way the algorithm will be able to handle missing blob detections and occlusions by using an estimate of the missing blob detections.

6.1.6 Initial Evaluation

This chapter contains an initial evaluation of the optimized Circular Trajectory Tracker (CTT). The rotating container is simulated using the particle simulator described in chapter 3.5. Here the simulator generates 5 particles of the same size on the outside of the container with different y-positions. It computes 22 frames in which the particle blobs are simulated. The purpose is to illustrate how CTT is able to handle noise and missing particle blobs.

input : measured feature points in matrices P_y, P_x, P_s and matrices L, R, O and Y
output: matrices L, R, O, Y and A containing new estimates for each track $T_k(i)$

compute thread position at i, k
if $T_k(i)$ *in* $L > 0$ **then**
  construct expected track on front side $\hat{T}_k(i)$ from equation 6.5 and 6.6
  forall the $k \in K$ **do**
    compute cost ϕ_{ik} using equation 6.7 for all points in k
    to find best point fitting within a specified window
    if *minimum cost of* ϕ_{ik} *is found* **then**
      add point $\mathbf{x}_k(i)$ to track $T_k(i)$
    else
      add estimated point $\hat{\mathbf{x}}_k(i)$ to track $T_k(i)$
    end
    $k = k - 1$
    if k = *0* **then**
      k = K
    end
  end
  compute track fitting cost ϕ_{Tf} according to equation 6.8
  compute track length cost ϕ_{Tl} according to equation 6.9
  if *track cost* < *threshold* **then**
    re-estimate $\hat{r}$ and $\hat{\theta}_0$ for final track $T_k(i)$ using equation 6.4
    update L with length of track $T_k(i)$
    update R with estimated radius $\hat{r}$
    update O with estimated angle $\hat{\theta}_0$
    update Y with mean of measured blob y-position found in P_y
    update A with mean of measured blob area found in P_s
  end
end

Algorithm 2: Kernel for re-estimating tracks using best cost fitting of the circular particle trajectory.

The most important parameters for the CTT and the simulator are adjusted as listed below:

- $K = 22$ – number of frames in the recordings of the rotating container
- **area** ≥ 15 – minimum size of particle blobs
- **sequence length** > 2 – minimum number of successive particle blobs
- $a = 1.3$ – factor used to reduce the search window (see chapter 6.1.3)
- $R_f = 12$ – radius in number of pixels to search for best fitting points
- $\phi_{Tf} < 0.80$ – threshold for cost fitting (equation 6.8)
- $\phi_{Tl} < 0.40$ – threshold for cost length (equation 6.9)

From figure 6.7 we observe that 2 false particle detections are found on the inside of the glass container. This is for a simulation where we have a mean of 100 added random noise blobs in every frame.

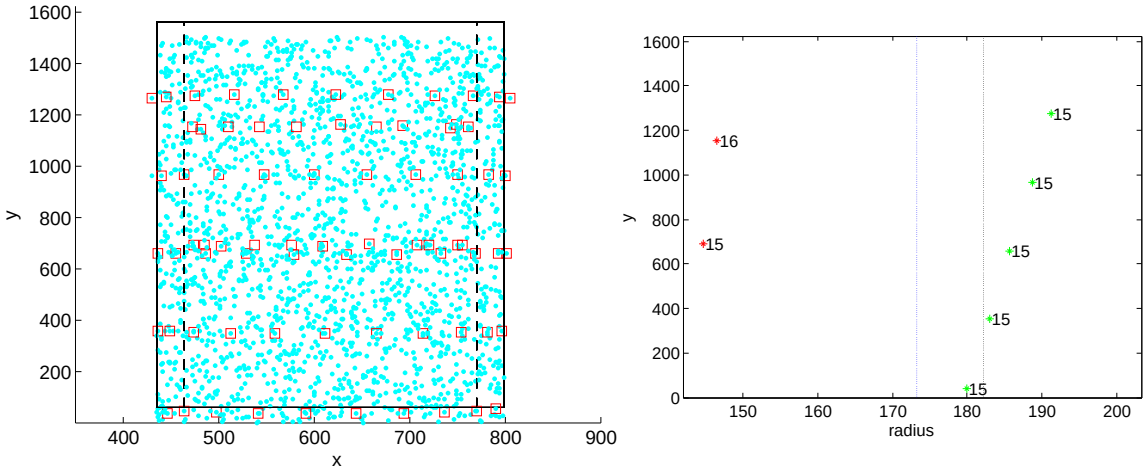


Figure 6.7: 5 particles generated by the simulator with a mean of 100 added noise points in each frame. The left figure illustrates with blue dots the noise and particle points for all frames, and a red square indicates a found particle track. The right figure shows that five particles were found on the outside and that 2 false detections were found on the inside of the container surface. The black line (left) indicates the measured radius of the container and the blue line (right) is the decision boundary.

From figure 6.8 we observe that 4 out of 5 particles were detected correctly. Particle P1 has too many missing points, since at least 3 corresponding points in a successive frame sequence is required by CTT. We also observe that the radius for particle P4 is now estimated to be on the inside of the surface of the simulated glass container. Since the decision boundary (see chapter 4.5) is inside the glass surface it is still detected correctly as being on the outside.

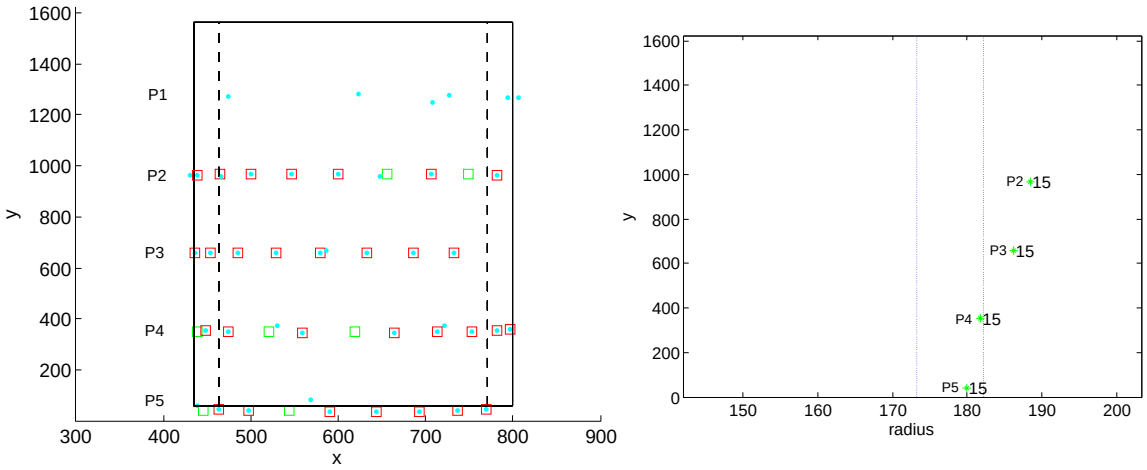


Figure 6.8: 5 particles generated by the simulator where some points are missing in each frame. The left figure illustrates with blue dots the particle points for all frames, and a red square indicates a found particle track where missing points are estimated by CTT with a green square. The right figure shows that only four particles were found on the outside.

Figure 6.9 illustrates how the execution time for tracking and positioning depend on the number of particles and noise measurements added to each frame. Both algorithms are based on the

optimized positioning algorithm as described in chapter 5.4. Kernels are implemented in MATLAB using 'for' and 'parfor' loops as a proof of concept. A 'parfor' loop in MATLAB spawns more threads on the CPU. Each thread runs in parallel on different cores to simulate a GPU realization of the kernels. A final optimized CUDA implementation of the CTT kernels are left for future work. The CTT algorithm performs faster than the first algorithm as a function of added noise. The critical part of the the CTT algorithm concerns the fourth kernel. Here most of the time is used on filtering similar tracks. However, no effort has been made to optimize this part of the algorithm running in MATLAB.

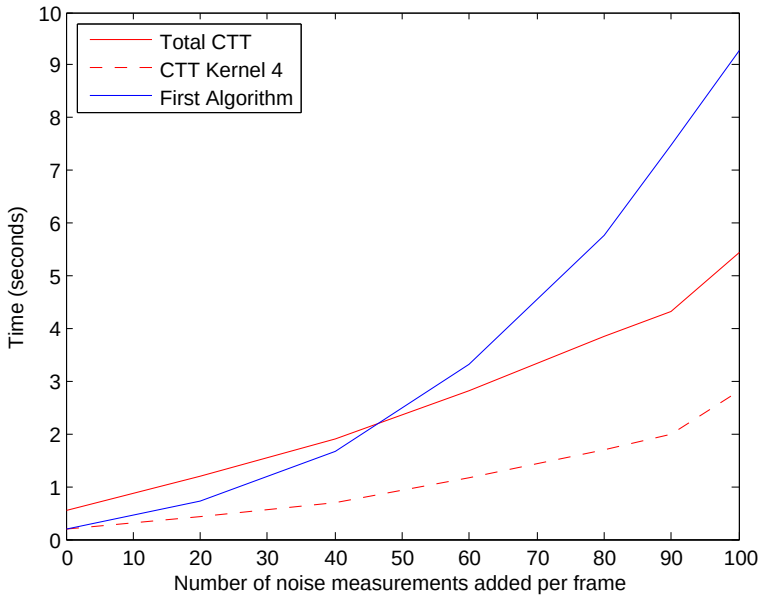


Figure 6.9: Execution time for tracking and positioning for CTT and the first algorithm measured for tracking 20 particles and adding noise. Half of the CTT execution time is used by the fourth kernel.

To conclude, CTT is able to handle missing blob detections and has proven to be tolerant towards added noise detections using the simulator. The performance looks promising even though a fully parallel (GPU) implementation is not yet realized. Here we would expect the algorithm to be less affected by the number of particles and noise. A detailed comparative evaluation of CTT against the first algorithm concerning accuracy and performance will be discussed later in chapter 6.3 using the simulator and finally in chapter 7 on a real data set of container recordings.

6.2 Multiple Hypothesis Tracking

Multiple Hypothesis Tracking (MHT) is a statistical/probabilistic method for multi-target tracking. It was originally presented by Donald B. Reid in 1979 [53] and is today considered one of the preferred methods for multi-target tracking [48].

MHT handles both noise, occlusion and track initiation/termination. It does this by constructing hypotheses that represent all combinations of targets and data associations. The decisions of specific data associations are postponed until several frames have been processed, such that the found targets and their tracks are the most likely given the history of measurements. The probabilities are calculated based on model specific parameters concerning the probability of initiating and terminating tracks as well as statistics concerning noise (false detections) and occlusion (missing detections).

Although MHT is one of the most powerful algorithms for solving these complex problems, it is definitely not the only one in the literature. In the following chapter an introduction and related work concerning probabilistic multi-target tracking algorithms is presented. A number of these may be suited for solving our problem. However, in practice the decision of using MHT was taken based on the documentation and computational performance of available open source software.

6.2.1 Introduction

Probabilistic multi-target tracking can generally be divided into two problems, namely the problem of data association and the problem of object state estimation [46]. As mentioned in the introduction data association (or point correspondence) refers to the problem of associating the right measurements with the right targets. Object state estimation refers to the problem of estimating a track for a target – that is a sequence of states for the target through time.

Object state estimation is a general problem of tracking, occurring in both multi- and single-target tracking problems. When the problem of data association has been solved, each found target can be isolated as a single-target tracking problem. In this problem, state estimation can be made probabilistically by utilizing a state space representation and applying Bayesian recursive estimation to compute the posterior distribution of the states given all measurements [46]. This recursion consists of a prediction step utilizing the state space model and update step. The update step utilizes the acquired measurements as well as statistics concerning noise figures of the process and the measurements. If the included parameters of the state space model are linear and normally distributed, the recursive Bayesian estimation simplifies to the Kalman filter [54]. In practice this is often used to estimate the tracks of the individual found targets in multi-target tracking. In problems where the state space model is nonlinear, a typical approach is to use the extended or unscented Kalman filter.

The problem of data association, on the other hand, is considerably more difficult and also interacts with the object state estimation of found tracks, since predicted states might help in determining which measurements should be associated to which targets. The different algorithms vary in complexity depending on which problems they try to solve. A general definition of scans can be used to categorize the algorithms. A *single-scan algorithm* makes a final decision of data association based on the current measurements of a frame. On the other hand, a *multi-scan algorithm* postpones the final decision of data association until more frames have been processed. This is also known as deferred decision logic.

An example of a single-scan (although not probabilistic) algorithm is Gaussian Nearest Neighbor (GNN) in which a track always associates the closest measurement in the given frame. A constraint involves that the measurement must be within a validation region of the track and that a measurement can only be associated with one track. Here a fixed number of targets is known and propagated through time, meaning that track initiation and termination are not supported. Another example is the Joint Probabilistic Data Association filter (JPDA) [55] in which soft assignments are made between measurements and tracks. Again the number of tracks is constant and known beforehand, but instead of assigning a unique measurement to each track, a weighted combination of all measurements is used. The weights are computed as probabilities of associating a given measurement to a given track based on the posterior probability that the measurement originated from the track [46]. By using a combination of all measurements in updating each track, a simple Kalman filter can be used to maintain the tracks [46, 55].

Examples of multi-scan algorithms are: the track splitting filter [56], 0-1 integer programming [57], Multiple Hypothesis Tracking (MHT) [53], Rao-Blackwellized Particle Filter (RBPF) [58], Markov Chain Monte Carlo Data Association (MCMCDA) [59] and the Probability Hypothesis Density filter (PHD) [60, 61]. These algorithms all have the power of using past measurements to form the most likely data associations. However, they vary in their approaches and their abilities to handle noise and occlusion and to initiate and terminate tracks.

The track splitting filter creates a tree structure for maintaining the different data association possibilities back in time. Instead of choosing the closest measurement to a track in the current frame as GNN, it postpones the decision a number of frames and chooses the data association that minimizes the total sum of distances [35]. The number of tracks is constant and subsequent track pruning must be made in order to ensure that no tracks share the same measurements.

0-1 integer programming takes all measurements in all frames and seeks to form a number of tracks. This is done as an optimization problem maximizing the posterior probability distribution while constraining the tracks to be disjoint [35].

MHT generates hypotheses for all possible data associations. A possibility for a measurement to be a false detection or a new track is incorporated such that the algorithm also handles initiation and termination of tracks.

RBPF divides the problem into the subproblems of data association and estimation of the number of targets. A particle filter is used to sample from the set of possible data associations and target initiation and termination processes. Rao-Blackwellization is used to evaluate some of the filtering equations analytically as opposed to pure sampling methods.

MCMCDA uses Markov Chain Monte Carlo methods to sample directly from the set of possible data associations. In this way it avoids the need to enumerate all data associations as is typically done in MHT, while still allowing track initation and termination.

PHD uses the concepts of random finite sets to obtain a problem formulation independent of the number of targets present. Using the concepts of recursive Bayesian filtering the finite sets of targets are predicted and updated using either a particle filter or Gaussian mixtures for representing the probability density functions [62].

Algorithm	Scan	Track-initiation	Track-termination	Occlusion[1]	False alarm
Global Nearest Neighbor (GNN)	Single			(✓)	(✓)
Joint Probabilistic Data Association filter (JPDA)	Single			✓	✓
Track-Splitting filter	Multiple				(✓)
0-1 integer programming	N	✓	(✓)	✓	✓
Multiple Hypothesis Tracking (MHT)	Multiple	✓	✓	✓	✓
Rao-Blackwellized particle filter (RBPF)	Multiple	✓	✓	✓	✓
Markov Chain Monte Carlo Data Association (MCMCDA)	Multiple	✓	✓	✓	✓
Probability Hypothesis Density filter (PHD)	Multiple	✓	✓	✓	✓

Table 6.2: Comparison of multi-target tracking algorithms.

Table 6.2 shows a comparison of the above mentioned multi-target tracking algorithms. Seen from the table, only MHT, RBPF, MCMCDA and PHD support full track initiation and termination. Since this is a vital need in our application such that we can track an unknown number of targets (possibly zero), these are the only candidates for further investigation. Because these algorithms are quite complex and therefore difficult and comprehensive to implement, the choice is based primarily on the documentation and computational performance of available open source software. Implementations were found of all four algorithms. However, only the implementation of MHT was both well documented, seemed to run efficiently and gave promising results at the same time.

Therefore, in the following chapters MHT is applied on our problem, even though alternative (and newer) approaches might prove to perform better.

6.2.2 Theory

Multiple Hypothesis Tracking (MHT) is a multi-target tracking method originally proposed by Reid in 1979 [53]. Although the algorithm has a long history, its application in real life problems is expanding due to the continuous increases in computer processing power [63]. Since the original contribution by Read, different approaches of reducing the computational load have been proposed. Some of these deal with efficient clustering of hypotheses so that hypotheses that do not share measurements can be treated independently. Other approaches deal with ways of reducing the number of hypotheses generated and propagated, thus achieving a suboptimal algorithm in which the hypothesis generation is performed non-exhaustively [64].

As mentioned, the main challenge of a multi-target tracking algorithm is to handle the data association. Measurements in a given frame must either be associated with known existing targets (prior targets), regarded as false alarms or create new targets. In order to take into account all these possibilities, multiple hypotheses can be generated representing all the different combinations. A hypothesis is a collection of measurements from different frames representing a track. In this way hypotheses are generated and propagated through the image sequence, and for each frame and each

[1] Also called missing measurements

measurement within a frame, more hypotheses are generated. Graphically this can be thought of as an expanding hypothesis tree, because each hypothesis splits into several new hypotheses for each frame. This is much like the track-splitting filter mentioned above [56]. Naturally, this procedure is impractical because of the exponential increase in hypotheses.

Therefore, hypothesis reduction is essential in order to perform MHT in practice. Hypothesis reduction is also known as pruning because the hypothesis tree is practically pruned in the process. It is a method for deleting unlikely hypotheses and merging similar hypotheses along the way, such that the number of hypotheses does not expand infinitely.

In figure 6.10 a conceptual overview of the MHT algorithm is shown. In this overview implementation specific optimizations concerning hypothesis clustering are abstracted away. The individual steps of the algorithm are:

1. **Measurements**. New measurement set for the current frame.
2. **Gating**. Only measurements that are close to prior targets can be associated with these. Different metrics can be used for calculating how well a measurement fits with a target.
3. **Hypothesis Generation**. A measurement can either belong to a prior target, create a new target or be a false alarm. Different hypotheses are generated to account for the different possibilities.
4. **Hypothesis Evaluation**. The generated hypotheses are given probabilities.
5. **Hypothesis Reduction**. Only the most likely hypotheses are kept alive. This step also includes track pruning.
6. **Track State Prediction**. For each hypothesis the predicted position at the next time step is calculated using a state space representation and a Kalman filter.

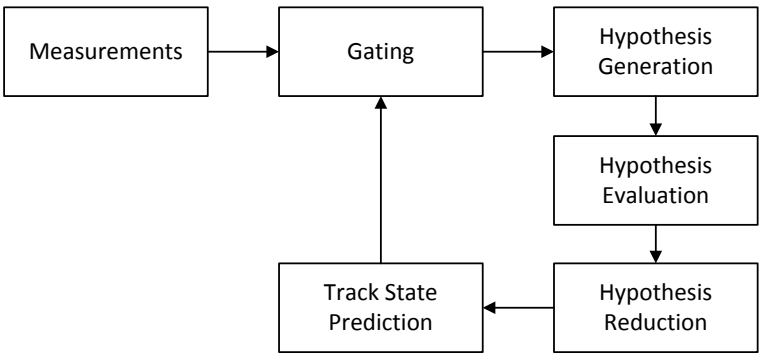


Figure 6.10: Algorithm overview for Multiple Hypothesis Tracking.

In the following description each of the six steps are described in depth. Throughout the description the symbols in table 6.3 are used to describe measurements, state vectors, tracks, data associations and hypotheses.

k	Time step / frame number
$\mathbf{x}_k$	Measurement set for frame k
M_k	The number of measurements in measurement set $\mathbf{x}_k$.
$\mathbf{x}_k(i)$	The i'th measurement in the measurement set $\mathbf{x}_k$
$\mathbf{X}^k$	All measurements up to and including time k
$\hat{\mathbf{z}}_k(j)$	State vector estimate for target j at time k
$\hat{\mathbf{Z}}^k(j)$	All state vector estimates for target j up to and including time k
Ω^k	The set of all hypotheses at time k
I_k	The number of hypotheses at time k
Ω_i^k	The i'th hypothesis at time k
$p(i)$	The index of the parent hypothesis at time $k-1$ from which the i'th hypothesis at time k originated
P_i^k	The probability of the i'th hypothesis at time k
ψ_{ki}	The measurement-to-track associations for the i'th hypothesis at time k
$\psi_{ki}(m)$	The measurement-to-track association for the m'th measurement at time k according to the i'th hypothesis
H	Observation model in the state space representation
R	Measurement covariance matrix in the state space representation
$P_k(j)$	Error covariance matrix of the Kalman filter for target j at time k
$B_k(j)$	Innovation covariance matrix of the Kalman filter for target j at time k
V	Observation area of the sensor/camera as the total number of pixels
$f_{\mathcal{N}(\mu,\Sigma)}$	Multivariate normal distribution with mean μ and covariance matrix Σ
$f_{\mathcal{P}(\lambda)}$	Poisson distribution with expected value λ
N_{DT}	Number of associations with prior targets (previously detected targets)
N_{FT}	Number of associations with false alarms (false targets)
N_{NT}	Number of associations with new targets
N_{PT}	Number of prior targets
N_{px}	Number of pixels in the image
P_D	Probability of detecting a prior target in the current frame
β_{FT}	Density of false alarms
β_{NT}	Density of new targets

Table 6.3: Symbol list.

6.2.2.1 Measurements

A measurement set at time k is denoted by $\mathbf{x}_k$ and consists of M_k measurements, such that $\mathbf{x}_k = \{\mathbf{x}_k(1), \ldots, \mathbf{x}_k(M_k)\}$. Some of these belong to prior targets, some belong to new targets and some are false alarms. For each frame a new measurement set is available, and by predicting the new states of the prior targets in the algorithm, new data associations can be made based on the gating procedure.

6.2.2.2 Gating

The purpose of the gating procedure is to limit the number of generated hypotheses. A gate (validation region) is created around each prior target j such that only measurements that are positioned within this gate can be associated with the target. The gate size is an application specific setting depending on the known physics and motion model of the system. It can be modeled as an n-dimensional ellipse described by a maximum Mahalanobis distance from the predicted track

state. At time k the predicted track state $\hat{\mathbf{z}}_k(j)$ of target j, the error covariance matrix $P_k(j)$ and the measurement covariance matrix R are assumed to be known from the previous predicted track state. The criteria for the m'th measurement $\mathbf{x}_k(m)$ being inside a constant Mahalanobis distance, η, is given by the equation [53]:

$$(\mathbf{x}_k(m) - H\hat{\mathbf{z}}_k(j))^T B_k(j)^{-1} (\mathbf{x}_k(m) - H\hat{\mathbf{z}}_k(j)) \leq \eta^2 \tag{6.10}$$

Here H is the observation model mapping the state space to the measurement space. $B_k(j)$ is the covariance matrix given by the innovation covariance update formula of the Kalman filter. This incorporates both the covariances matrix $P_k(j)$ of the predicted state and the covariance matrix R of the measurements by the equation:

$$B_k(j) = HP_k(j)H^T + R \tag{6.11}$$

In case of a nonlinear state space model, the extended Kalman filter and its update equations can be used. In equation 6.10 and 6.11 it is assumed that all measurements in the current frame have the same covariance.

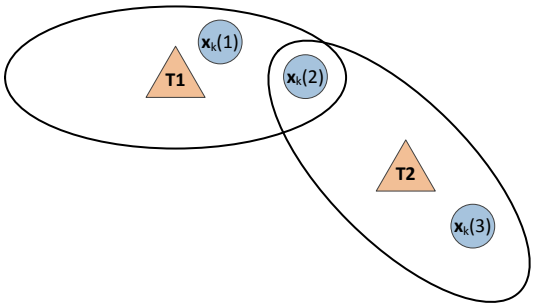


Figure 6.11: Simple example of multiple hypothesis gating. Two prior targets, $T1$ and $T2$, and a measurement set at time k consisting of three measurements are present. The ellipses denote the gates or validation regions of the prior targets. Only measurements within these gates can be associated with the prior targets.

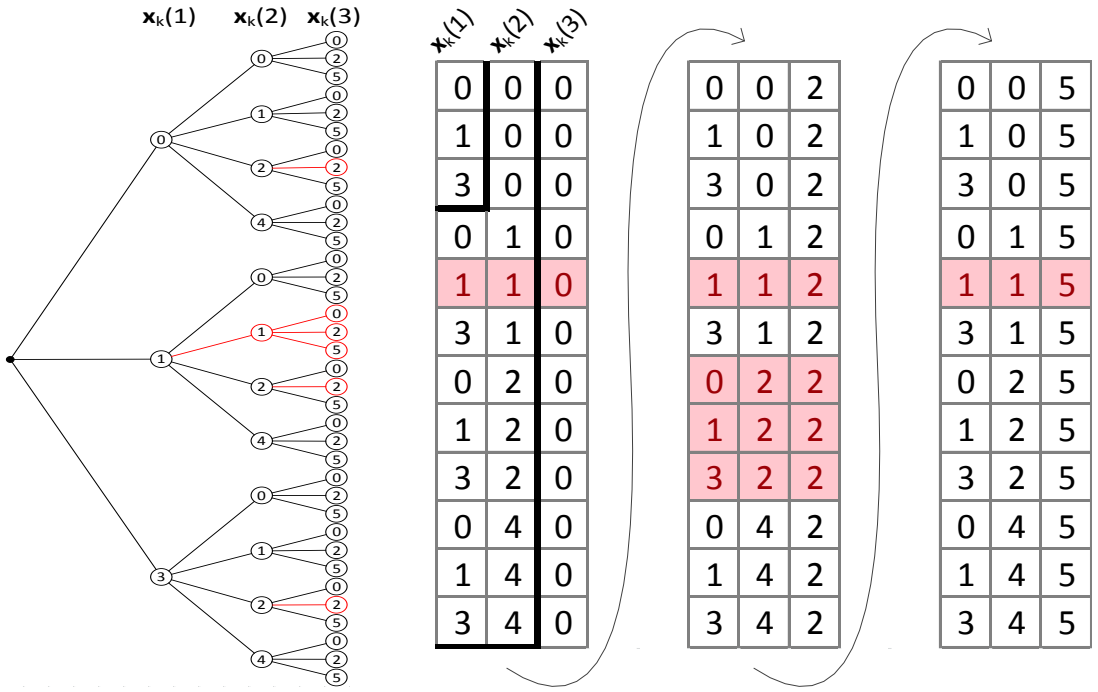
In figure 6.11 a simple example including two prior targets, $T1$ and $T2$, and a measurement set at time k consisting of three measurements is depicted. For each target a gate is drawn indicating which measurements are association candidates for the target. For $T1$ measurement 1 and 2 are candidates, and for $T2$ measurement 2 and 3 are candidates.

6.2.2.3 Hypothesis Generation

For the example in figure 6.11 30 valid hypotheses can be generated. These can be mapped as both a hypothesis tree and a hypothesis matrix as illustrated in figure 6.12.

Figure 6.12a illustrates all hypotheses as a tree by iterating over the three measurements (the three columns). All numbers denote which targets the measurements are associated with. The depth of the tree increases as more measurements are processed. The first level of the tree handles the three combinations for the first measurement, $\mathbf{x}_k(1)$. This can either be a false alarm (0), belong to the first target (1) or be a new target (3). Since the gate of target $T2$ does not include $\mathbf{x}_k(1)$, this association is not valid. The association to a new target is designated by a number one above the number of currently known targets. For the first measurement a new target is denoted (3) since only two targets are known.

In the second level of the tree, the second measurement, $\mathbf{x}_k(2)$, is processed. Since this is within the gates of both targets, four hypotheses can be generated for each of the three previous



(a) Hypothesis tree. (b) Hypothesis matrix.

Figure 6.12: Hypotheses for the simple example. Red crossings and lines denote invalid combinations, since a target cannot be associated with more than one measurement in a frame. (a) illustrates the hypotheses as a tree and (b) illustrates them as a matrix.

hypotheses. The second measurement can either be a false alarm (0), belong to the first (1) or second (2) target or be a new target (4).

In this way 36 hypotheses can be generated. However, not all hypotheses are valid, since a target cannot be associated with more than one measurement per frame. Therefore, 6 of the hypotheses are marked as invalid (red), and only 30 are generated.

Figure 6.12b illustrates how the hypotheses can be represented in a computer as a matrix. In this representation the rows denote hypotheses and the columns denote measurements. Solid black lines are drawn on top of the matrix to illustrate how the matrix is expanded for each processed measurement.

Mathematically, we express the set of all hypotheses at time k as $\Omega^k = \{\Omega_1^k, \ldots, \Omega_{I_k}^k\}$, where Ω_i^k denotes the i'th hypothesis of the set, and I_k is the total number of hypotheses at time k. We use superscript notation for k because it is the combined set of all hypotheses up to and including the k'th time step. These hypotheses associate all measurements up to and including the k'th time step, $\mathbf{X}^k = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$.

6.2.2.4 Hypothesis Evaluation

All generated hypotheses are evaluated with a probability based on the model parameters and their associated measurements. That is, we seek to estimate the probability of a hypothesis i based on all measurements up to and including time k:

$$P_i^k = P(\Omega_i^k|\mathbf{X}^k) \tag{6.12}$$

Because hypotheses originate from prior hypotheses (illustrated in the hypothesis tree), a recursive relationship can be written based on a parent hypothesis, $\Omega_{p(i)}^{k-1}$, and the measurement-to-track associations for the current frame, ψ_{ki}. $p(i)$ denotes the index of the parent hypothesis – that is, the hypothesis in frame $k-1$ from which the current hypothesis i originated. ψ_{ki} denotes the data association for the i'th hypothesis in frame k. In figure 6.12 ψ_{ki} corresponds to a branch of the hypothesis tree or a row of the hypothesis matrix, relating each of the measurements in frame k to either prior targets, new targets of false alarms. $\psi_{ki}(m)$ denotes the measurement-to-track association for the m'th measurement at time step k according to the i'th hypothesis. The recursive relationship states that a new hypothesis is the union of a prior hypothesis and a measurement-to-track association [65]:

$$\Omega_i^k = \{\Omega_{p(i)}^{k-1}, \psi_{ki}\} \tag{6.13}$$

Applying this relationship between prior and current hypotheses, a recursive formula for calculating the probabilities of hypotheses can be stated. Here it is also utilized that the accumulated measurement set up to time k can be expressed by the union of the previous and new sets, $\mathbf{X}^k = \{\mathbf{X}^{k-1}, \mathbf{x}_k\}$. Using Bayes rule, where first the definition of conditional probability and second the chain rule for conditional probabilities is applied, we get:

$$\begin{aligned}
P_i^k &= P(\Omega_i^k|\mathbf{X}^k) \\
&= P(\Omega_{p(i)}^{k-1}, \psi_{ki}|\mathbf{X}^{k-1}, \mathbf{x}_k) \\
&= \frac{P(\Omega_{p(i)}^{k-1}, \psi_{ki}, \mathbf{X}^{k-1}, \mathbf{x}_k)}{P(\mathbf{X}^{k-1}, \mathbf{x}_k)} \\
&= \frac{P(\mathbf{x}_k|\Omega_{p(i)}^{k-1}, \psi_{ki}, \mathbf{X}^{k-1}) P(\psi_{ki}|\Omega_{p(i)}^{k-1}, \mathbf{X}^{k-1}) P(\Omega_{p(i)}^{k-1}|\mathbf{X}^{k-1}) P(\mathbf{X}^{k-1})}{P(\mathbf{X}^{k-1}, \mathbf{x}_k)} \\
&= \frac{1}{c} \underbrace{P(\mathbf{x}_k|\Omega_{p(i)}^{k-1}, \psi_{ki}, \mathbf{X}^{k-1})}_{\text{measurement likelihood (1)}} \underbrace{P(\psi_{ki}|\Omega_{p(i)}^{k-1}, \mathbf{X}^{k-1})}_{\text{association likelihood (2)}} \underbrace{P(\Omega_{p(i)}^{k-1}|\mathbf{X}^{k-1})}_{\text{prior (3)}}
\end{aligned} \tag{6.14}$$

where the normalizing constant $c = \frac{P(\mathbf{X}^{k-1}, \mathbf{x}_k)}{P(\mathbf{X}^{k-1})} = P(\mathbf{x}_k|\mathbf{X}^{k-1})$ is the same for all hypotheses and thus can be omitted.

In order to derive an expression for this probability, the terms denoted (1), (2) and (3) are treated individually. They comprise:

(1) **Measurement likelihood**. The probability of drawing the current measurement set $\mathbf{x}_k$ given the considered hypothesis.

(2) **Association likelihood**. The probability of the measurement-to-track associations ψ_{ki} given the prior hypothesis and the model parameters.

(3) **Prior**. The probability of the prior hypothesis $\Omega_{p(i)}^{k-1}$. This is known from the previous iteration.

(1) Measurement likelihood The measurement likelihood term expresses the probability of the measurement set $\mathbf{x}_k$ given the considered hypothesis. Assuming that the M_k measurements are independent, the joint probability can be written as the product of the individual probabilities as:

$$\begin{aligned} P(\mathbf{x}_k|\Omega_{p(i)}^{k-1}, \psi_{ki}, \mathbf{X}^{k-1}) &= \prod_{m=1}^{M_k} P(\mathbf{x}_k(m)|\Omega_{p(i)}^{k-1}, \psi_{ki}, \mathbf{X}^{k-1}) \\ &= \prod_{m=1}^{M_k} f\left(\mathbf{x}_k\left(m\right)\right) \end{aligned} \tag{6.15}$$

where

$$f\left(\mathbf{x}_k\left(m\right)\right) = \begin{cases} {}^1\!/_V & \text{if } \mathbf{x}_k(m) \text{ is associated to a false alarm or new target} \\ f_{\mathcal{N}(H\hat{\mathbf{z}}_k(j), B_k(j))}\left(\mathbf{x}_k(m)\right) & \text{if } \mathbf{x}_k(m) \text{ is associated to a prior target } j \end{cases} \tag{6.16}$$

In this expression it is assumed that both false alarms and new targets have a uniform distribution within an area V. The measurements that are associated with false alarms or new targets through ψ_{ki} are thus given the probability ${}^1\!/_V$. Measurements that are associated with prior targets are given a probability depending on their Mahalanobis distances to the predicted track state. This is modeled by the function $f_{\mathcal{N}(H\hat{\mathbf{z}}_k(j), B_k(j))}$ evaluating a multivariate normal distribution $\sim \mathcal{N}(H\hat{\mathbf{z}}_k(j), B_k(j))$ at $\mathbf{x}_k(m)$. Here, H and $B_k(j)$ are the observation model and the covariance matrix as described in chapter 6.2.2.2.

(2) Association likelihood The association likelihood term expresses the probability of the measurement-to-track associations ψ_{ki} for the considered hypothesis. As for the measurement likelihood it is conditioned on the prior hypothesis and the model parameters. From the prior hypothesis $\Omega_{p(i)}^{k-1}$ the number of prior targets can be extracted as N_{PT}. And from the association hypothesis ψ_{ki} three numbers can be extracted:

- N_{DT} (Detected Targets) - number of associations to prior targets
- N_{FT} (False Targets) - number of associations to false alarms
- N_{NT} (New Targets) - number of associations to new targets

The association likelihood is composed of three independent events whose probabilities can be multiplied together:

$$\begin{aligned} P(\psi_{ki}|\Omega_{p(i)}^{k-1}, \mathbf{X}^{k-1}) =& P(N_{DT}, N_{FT}, N_{NT}|\Omega_{p(i)}^{k-1}) \cdot P(\text{configuration}|N_{DT}, N_{FT}, N_{NT}) \cdot \\ & P(\text{assignment}|\text{configuration}) \end{aligned} \tag{6.17}$$

In this expression the first term denotes the probability of drawing the numbers N_{DT}, N_{FT} and N_{NT} based on the model parameters and the prior hypothesis. The second term, configuration, denotes the probability of assigning, without order, the N_{DT}, N_{FT} and N_{NT} targets to the specific measurements in the current measurement set. The last term, assignment, denotes the probability of assigning, with order, the N_{DT} prior targets to N_{DT} specific current measurements.

First the probability of drawing the numbers N_{DT}, N_{FT} and N_{NT} can be calculated based on the prior hypothesis $\Omega_{p(i)}^{k-1}$ and the model parameters. The probabilities of drawing N_{DT}, N_{FT} and N_{NT} are handled separately. A binomial distribution[2] is used to calculate the probability of detecting N_{DT} of the N_{PT} prior targets. Here P_D is a model parameter describing the general probability of detecting a prior target in the current frame. The probability is given by:

$$P(N_{DT}|\Omega_{p(i)}^{k-1}) = \binom{N_{PT}}{N_{DT}} P_D^{N_{DT}}(1-P_D)^{N_{PT}-N_{DT}} \tag{6.18}$$

The probabilities of detecting N_{FT} and N_{NT} can be modeled in the same way as binomial distributions. However, the number of trials in the distribution model is no longer N_{PT} since false alarms and new targets do not originate from prior targets. Instead the number of trials is equal to the total number of pixels in the image N_{px}. If this is the same as the sensor area $N_{px} = V$, the probability P_F of an occurrence of a false alarm in a given pixel is equal to the density of false alarms $P_F = \beta_{FT}$.

Because the total number of pixels is much larger than N_{PT}, an approximation of the binomial distribution can be utilized to reduce the computational complexity of the algorithm. In the limit $N_{px} \to \infty$ the binomial distribution is equal to the Poisson distribution[3] [66], such that:

$$P(N_{FT}|\Omega_{p(i)}^{k-1}) = \binom{N_{px}}{N_{FT}} P_F^{N_{FT}}(1-P_F)^{N_{px}-N_{FT}} \approx \frac{(\beta_{FT}V)^{N_{FT}}}{N_{FT}!} e^{-\beta_{FT}V} \tag{6.19}$$

The exact same approximation is made for the probability of new targets. The combined probability of drawing the numbers N_{DT}, N_{FT} and N_{NT} is thus:

$$P(N_{DT}, N_{FT}, N_{NT}|\Omega_{p(i)}^{k-1}) = f_{\mathcal{B}(N_{PT},P_D)}(N_{DT}) f_{\mathcal{P}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{P}(\beta_{NT}V)}(N_{NT}) \tag{6.20}$$

where $f_{\mathcal{B}(N_{PT},P_D)}(N_{DT})$ denotes a binomial distribution with N_{PT} trials and a probability of P_D evaluated at N_{DT}, and $f_{\mathcal{P}(\beta_{FT}V)}(N_{FT})$ denotes a Poisson distribution with an expected value $\lambda = \beta_{FT}V$ evaluated at N_{FT}.

Next, the probability of the configuration is calculated. A configuration is an assignment of specific measurements to count as either N_{DT}, N_{FT} or N_{NT}. The probability of the configuration is based on how many possible combinations there are of drawing N_{DT}, N_{FT} and N_{NT} from the measurement set. The binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is used to count the number of combinations that k items can be drawn from n items when the order does not matter. This is used to count the total number of combinations of drawing N_{DT} from the M_k measurements, N_{FT} from the remaining $M_k - N_{DT}$ and N_{NT} from $M_k - N_{DT} - N_{FT}$:

$$\binom{M_k}{N_{DT}}\binom{M_k - N_{DT}}{N_{FT}}\binom{M_k - N_{DT} - N_{FT}}{N_{NT}} = \binom{M_k}{N_{DT}}\binom{M_k - N_{DT}}{N_{FT}}\underbrace{\binom{N_{NT}}{N_{NT}}}_{=1}$$

[2]The binomial distribution $\binom{n}{k}p^k(1-p)^{n-k}$ gives the probability of drawing k successes from n independent trials when the order does not matter.

[3]The Poisson distribution $\frac{\lambda^k}{k!}e^{-\lambda}$ gives the probability of drawing k successes in a fixed interval of time/space with a known density of occurrence λ.

The probability of the configuration is then given by one divided by this number, because we seek the probability of one specific configuration:

$$P(\text{configuration}|N_{DT}, N_{FT}, N_{NT}) = \frac{1}{\binom{M_k}{N_{DT}}\binom{M_k - N_{DT}}{N_{FT}}} \tag{6.21}$$

Finally, the probability of the assignment of N_{DT} to specific measurements is calculated based on how many possible assignment combinations there are. The assignments of N_{FT} and N_{NT} are not included, because false targets are all associated with a 0 (chapter 6.2.2.3) and the order of assigning new targets is indifferent. Therefore the probability of assigning N_{FT} and N_{NT} is 1 in both cases. However, for the detection of prior targets we can use partial permutation[4] to count the number of possible assignments of the N_{DT} detected prior targets to the N_{PT} prior targets (when order does matter). As before, the probability of the assignment is then given by one divided by this number, because we seek the probability of one specific assigment:

$$P(\text{assignment}|\text{configuration}) = \frac{1}{\frac{N_{PT}!}{(N_{PT}-N_{DT})!}} = \frac{(N_{PT} - N_{DT})!}{N_{PT}!} \tag{6.22}$$

All in all, substituting equation 6.20, 6.21 and 6.22 into 6.17 gives the complete association likelihood:

$$\begin{aligned} P(\psi_{ki}|\Omega_{p(i)}^{k-1}, \mathbf{X}^{k-1}) =& f_{\mathcal{B}(N_{PT},P_D)}(N_{DT}) f_{\mathcal{P}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{P}(\beta_{NT}V)}(N_{NT}) \cdot \\ & \frac{1}{\binom{M_k}{N_{DT}}\binom{M_k - N_{DT}}{N_{FT}}} \frac{(N_{PT} - N_{DT})!}{N_{PT}!} \end{aligned} \tag{6.23}$$

(3) Prior The prior hypothesis term $P(\Omega_{p(i)}^{k-1}|\mathbf{X}^{k-1}) = P_{p(i)}^{k-1}$ expresses the probability of the prior hypothesis $\Omega_{p(i)}^{k-1}$. This is assumed to be known from the previous iteration, and it is therefore included directly in the expression of the current hypothesis probability P_i^k.

Combining (1), (2) and (3) By substituting equation 6.16 and 6.23 into 6.14 and simplifying, we get [53, 65]:

$$P_i^k = \frac{1}{c'} P_D^{N_{DT}} (1 - P_D)^{N_{PT} - N_{DT}} \beta_{FT}^{N_{FT}} \beta_{NT}^{N_{NT}} \left(\prod_{m=1}^{N_{DT}} f_{\mathcal{N}(H\mathbf{z}_k(\psi_{ki}(m)), B_k(\psi_{ki}(m)))} (\mathbf{x}_k(m)) \right) P_{p(i)}^{k-1} \tag{6.24}$$

where for brevity and ease of notation the M_k current measurements are reordered such that the first N_{DT} measurements correspond to detections of previous targets. Equation 6.24 is the essential part of the MHT algorithm. It describes the probability of a given hypothesis recursively based on prior hypotheses. Often the logarithm of equation 6.24 is used due to different computational advantages [48]. These comprise numerical precision with small probabilities, performing additions rather than multiplications and performing multiplications rather than exponentiations [65].

[4] In combinatorics the number of partial permutations $P_{n,k} = \frac{n!}{(n-k)!}$ is defined as the number of ways that k items from a sequence of n unique items can be arranged in differently ordered sequences.

6.2.2.5 Hypothesis Reduction

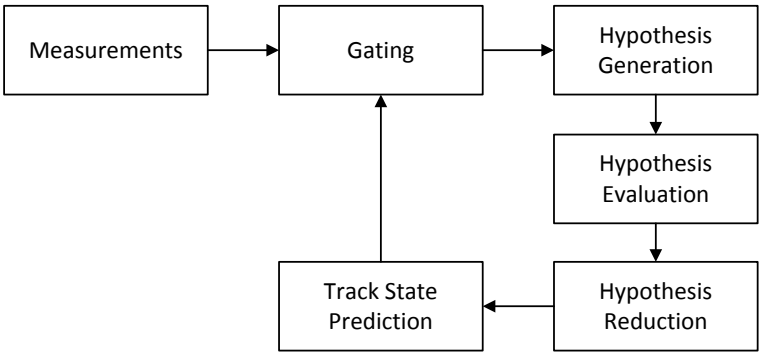


Figure 6.13: Algorithm overview for Multiple Hypothesis Tracking.

After the evaluation of the hypotheses, the hypothesis reduction step (figure 6.13) seeks to reduce the number of hypotheses. As described in chapter 6.2.2.3 a hypothesis is created for every possible association combination of measurements and targets in each frame. However, this number rises exponentially as the number of frames and measurements increases. It has been shown that for a given hypothesis the maximum number of new hypotheses generated in a single frame can be calculated from the equation [67]:

$$N = \sum_{N_{DT}=0}^{min(M_k, N_{PT})} \binom{M_k}{N_{DT}} \binom{N_{PT}}{N_{DT}} N_{DT}! \, 2^{M_k - N_{DT}} \tag{6.25}$$

where N_{PT} is the number of prior targets and M_k is the number of measurements in the current measurement set. In this calculation it is assumed that all measurements are within the gates (validation regions) of all prior targets. A simple example with one single frame with $M_k = 20$ measurements and $N_{PT} = 0$ known prior targets generates a total of 1.048.576 hypotheses. If, instead, we have $M_k = 50$ measurements as is often the case in our problem, a total of $1.13 \cdot 10^{15}$ hypotheses are created for just the first frame.

Therefore, it is critical to reduce the number of hypotheses in order to have a computationally feasible algorithm. This is also called hypothesis pruning. Different approaches are used to reduce the number of hypotheses based on their likelihood. However, since they all discard hypotheses that later on potentially could prove to be the most likely hypotheses, we are only guaranteed a sub-optimal solution. Some approaches to hypothesis pruning are:

- The most simple approach to hypothesis reduction is to delete hypotheses that have a probability (calculated by equation 6.24) below a given threshold. However, this strategy tends to delete hypotheses containing new targets, because their probabilities are low until the targets are associated with more measurements [68]. Therefore, heuristics can be used to allow low-probability hypotheses to survive when they contain new targets.

- Another approach is to use n-scan-back pruning [69], in which a final decision regarding data associations for a given frame k is made at time $k+n$. This is equivalent to the general notion of an n-scan algorithm that looks back n frames in order to choose the most likely data associations. When $n = 0$ we have a zero-scan algorithm. Here MHT approaches more simple multi-target tracking algorithms such as the probabilistic data association filter [53].

- Another strategy is to perform hypothesis merging [53]. The idea is that similar hypotheses can be merged and represented by a single hypothesis. The similarity criterion can depend

on the number of contained targets and the distances between them. A distance between two targets j_1 and j_2 could for instance be calculated based on the distance between their track state predictions $\hat{\mathbf{Z}}^k(j_1)$ and $\hat{\mathbf{Z}}^k(j_2)$ and the distance between the eigenvalues of their covariance matrices. If the distance is smaller than a given threshold, the hypotheses are merged, and the combined probability is calculated as the sum of the individual hypothesis probabilities.

In practice a combination of hypothesis merging, n-scan-back pruning and using the most probable hypotheses with an upper limit is often chosen [35].

Track Management An important part of the MHT algorithm concerns track management. In the above chapters the notion of prior targets is used to denote the number of targets that a parent hypothesis contained. However, since each measurement within a frame can originate from a new target, this number naturally rises exponentially. Therefore, instead of just calculating the number of prior targets, we distinguish between tentative and confirmed tracks:

- A tentative track is a track which has not yet been confirmed. All new targets within a hypothesis start as tentative tracks, and most tracks are deleted before being confirmed.
- In the conceptual MHT, a confirmed track is a track which survives the hypothesis reduction and has a probability of existence equal to one [53]. In figure 6.12b hypothesis reduction corresponds to deleting rows in the hypothesis matrix. A track j is confirmed when (after pruning) all entries in a column m of the hypothesis matrix are equal. This means that the probability of associating the m'th measurement to the j'th track is 1, and thus the track is confirmed.

  In other more practical MHT variants, another definition of confirmed tracks is used. Here a track is confirmed when it has been associated with measurements in a number of frames. An M-of-N criterion is often used, in which a measurement must be associated with the track in M out of N consecutive frames to get confirmed [70]. This approach is used to generate hypotheses for only some data associations in equation 6.15 and 6.16 by favoring associations with confirmed tracks over tentative tracks [71].

The differentiation between tentative and confirmed tracks is also important for visualizing the output of the tracker. A typical MHT algorithm outputs the confirmed tracks of the hypothesis with the highest probability. If all tracks of this hypothesis were visualized, all noise measurements would also appear as tracks.

In addition to the M-of-N criterion, track termination (deleting a track) is often performed when a track (tentative or confirmed) is not detected within the last K frames [70]. This heuristic is another strategy for reducing the number of hypotheses.

6.2.2.6 Track State Prediction

The last step in the algorithm before closing the iteration loop and processing a new measurement set at time $k+1$ concerns the track state prediction. As described in chapter 6.2.2.2 the gating procedure seeks for measurements within a validation region of the prior targets generated in previous frames. Each track can be handled as a single-target tracking problem, and in this way the state of the track can be updated and predicted with a Kalman filter. The purpose of the track state prediction is to provide a prediction of the state of a track at time $k+1$ when it is known at time k. In the prediction step of the Kalman filter the previous estimate of the state vector $\hat{\mathbf{z}}_k(j)$

for track j along with a potential control input $\mathbf{u}_k$ is used to predict an a priori estimate of the state vector at the next time step. Simultaneously an estimate of the new error covariance P_{k+1} is predicted:

$$\hat{\mathbf{z}}_{k+1}^{-}(j) = F\hat{\mathbf{z}}_k(j) + B\mathbf{u}_k \tag{6.26}$$

$$P_{k+1}^{-} = FP_kF^T + Q \tag{6.27}$$

$\hat{\mathbf{z}}_k(j)$ is the state estimate for track j at time k, F is the state transition matrix, B is the control-input matrix coupling the potential input vector $\mathbf{u}_k$ to the state variable, P_k is the error covariance matrix[5] and Q is the process covariance matrix.

As described in chapter 6.2.2.3 each hypothesis contains several tracks. However, some tracks are shared by multiple hypotheses, and therefore the track state estimate $\hat{\mathbf{z}}_k(j)$ to propagate has to be chosen with consideration. Different approaches can be used [68]:

- **Highest probability**. The most simple approach is to use the track state from the hypothesis with the highest probability. However, this has the undesirable effect of influencing the gating- and evaluation-procedure of the next iteration such that the probability of other likely hypotheses is probably reduced.
- **JPDA**. The principles behind the Joint Probabilistic Data Association filter can be utilized to combine all state estimates of tracks based on the probabilities of the corresponding hypotheses. If j denotes the given track number and i denotes a hypothesis containing track j, the state estimate can be computed as a weighted sum of all track hypotheses:

$$\hat{\mathbf{z}}_k(j) = \sum_{i=1}^{I_{kj}} P_i^k \hat{\mathbf{z}}_{ki}(j) \tag{6.28}$$

  where $\hat{\mathbf{z}}_{ki}(j)$ denotes the state estimate of track j at time k in hypothesis i, and I_{kj} denotes the number of hypotheses containing track j.

The update formula of the Kalman filter is only used if a measurement in the current frame is associated with the track. Otherwise another prediction is carried out without updating the model with a measurement. The measurement update step (also referred to as the correction step) of the Kalman filter is given by:

$$K_k = P_k^{-}H^T(HP_k^{-}H^T + R)^{-1} \tag{6.29}$$

$$\hat{\mathbf{z}}_k(j) = \hat{\mathbf{z}}_k^{-}(j) + K_k(\mathbf{x}_k(j) - H\hat{\mathbf{z}}_k^{-}(j)) \tag{6.30}$$

$$P_k = (I - K_kH)P_k^{-} \tag{6.31}$$

Here K_k is the Kalman gain that decides how much the new measurement is weighted compared to the previous estimate of the state. By iterating between the prediction and update steps the Kalman filter is continuously updated every time a new measurement is available. Since the combination of the previous estimate and a new measurement is described as a Gaussian, the filter does not need to store all previous measurements but is completely described by its current state.

In the above equations the matrices F, B, H, R and Q are all assumed to be constant. However, these can depend on the time k and can therefore change with each time step or measurement.

[5]The error covariance is defined as the covariance of the error $\mathbf{e}_k = \mathbf{z}_k(j) - \hat{\mathbf{z}}_k(j)$, where $\mathbf{z}_k(j)$ is the actual state of the track and $\hat{\mathbf{z}}_k(j)$ is the estimated state.

For instance R often changes with time because the uncertainty of a measurement can vary depending on the environment.

6.2.3 Extensions

The above description of MHT is the conceptual MHT described by Reid in 1979 [53]. Because of the exponential rise in the number of hypotheses as well as the demanding calculations of hypothesis probabilities before pruning, this algorithm is in practice considered to be computationally infeasible [48]. However, extensions, modifications and alternative implementations have been proposed that enable real-time execution of the algorithm. Reid himself proposed a method of decomposing the problem into a number of smaller problems using clustering of the hypotheses [53]. This method is explained in appendix H section H.1. However, in order to perform MHT in real-time, sub-optimal approaches are required, in which the solution is not guaranteed to be optimal. These approaches include hypothesis reduction, track deletion and track merging.

One of the significant improvements in the computational feasibility of MHT is to use Murty's algorithm [72] to generate hypotheses more efficiently [64]. In this approach only the m-best (most probable) hypotheses are formed without enumerating all possible combinations.

Also, another way of addressing the problem called Track-Oriented MHT (TOMHT) does not maintain hypotheses from frame to frame but instead stores track sets. This method is described more thoroughly in appendix H section H.2.

6.2.4 Application

In the following section the application of MHT in our problem domain is treated. First a state space representation describing the physics of the moving particles is derived and discussed. Following is a description of the specific MHT implementation and its advantages and disadvantages applied to our problem. Subsequently track filtering as performed in the first tracking algorithm is described with the purpose of removing unrealistic tracks found by MHT. Finally the estimation of the MHT model parameters for a simulated environment as well as for the training set of actual containers is treated.

6.2.4.1 State Space Model

The physical system consisting of particles moving in liquid corresponding to the rotating glass containers can be described mathematically with a state space representation. The purpose of this representation is to provide a compact description of the motion model relating states of particles to the measurements from the segmentation algorithm. For every track that is maintained by MHT a Kalman filter is used to estimate the state of the object at the next time step. As described in chapter 6.2.2.2 this estimate is used to sort out all measurement-to-track associations that do not match the estimated state with a gate. In chapter 6.2.2.4 it is used to favor measurement-to-track associations close to the estimate by assigning higher probabilities to their hypotheses.

A simple state space representation of the system models the particles in a similar way to the one utilized in chapter 4.4. That is, a particle's position along the x-axis can be modeled as a harmonic motion with a known angular velocity. The initial angle, θ_0, and the radius, r, of this motion are unknown but can be estimated from a number of measurements of the image's x-coordinates. As stated in equation 4.2 the expected motion in the x-axis can be expressed as:

$$\hat{x}_k = r \cdot cos(\omega \cdot T_s \cdot (k-1) + \theta_0) + x_c$$

This expression states that the k'th coordinate has moved an angle of $\omega \cdot T_s \cdot (k-1)$ relative to the first coordinate and that the first measurement started at an angle of θ_0 relative to the x-axis. Here $T_s \cdot (k-1)$ is the time stamp of the k'th measurement. This serves as the foundation for constructing the state space representation.

We assume an explicit discrete time-invariant nonlinear system. This means that the system can be described by the following equations:

$$\mathbf{z}_k = f\left(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}\right) \tag{6.32}$$

$$\mathbf{x}_k = h\left(\mathbf{z}_k, \mathbf{v}_k\right) \tag{6.33}$$

In these $\mathbf{z}_k$ is the state vector at time k consisting of the unknown variables of the system. f is the state transition function coupling the state variables to the next time step. $\mathbf{u}_k$ is an optional control input vector to the system (e.g. motion input). $\mathbf{w}_k$ is a process noise vector with a zero-mean Gaussian distribution and covariance matrix Q. $\mathbf{z}_k$ is the measurement vector consisting of measurement data from the segmentation algorithm, and $\mathbf{v}_k$ is a measurement noise vector independent of $\mathbf{w}_k$ but also with a zero-mean Gaussian distribution and covariance matrix R. h is the observation function coupling the state variables to the measurements.

In the most simple model, the state transition function can be described as a linear operation when using a polar coordinate system for describing the position along the x-axis for a particle. The states are the radius r, the angle θ and the y-coordinate. A control input handling the known constant angular velocity is applied, and by assuming purely additive noise the state transition function can be written as:

$$\begin{aligned}
\mathbf{z}_k = \begin{bmatrix} r_k \\ \theta_k \\ y_k \end{bmatrix} &= f\left(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}\right) \\
&= A\mathbf{z}_{k-1} + B\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{k-1} \\ \theta_{k-1} \\ y_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \omega \cdot T_s + \mathbf{w}_{k-1}
\end{aligned} \tag{6.34}$$

where A is the state transition matrix coupling the state variables to the next time step and B is the control matrix coupling the control input to the states.

The measurement vector consists of x- and y-coordinates for each frame. By applying a non-linear measurement function, the x-coordinate can be related to the polar coordinate description, whereas the y-coordinate relates directly to the third state. Assuming that we have purely additive measurement noise, the measurement model can be written as:

$$\begin{aligned}
\mathbf{x}_k = \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} &= h\left(\mathbf{z}_k, \mathbf{v}_k\right) \\
&= \begin{bmatrix} r_k \cdot cos\left(\theta_k\right) \\ y_k \end{bmatrix} + \mathbf{v}_k
\end{aligned} \tag{6.35}$$

The covariance matrices, Q and R, are used by for instance a Kalman filter to handle the uncertainties of the process and the measurements. For the process noise we can model Q as a diagonal

matrix with entries corresponding to the uncertainties of r, θ and the y-coordinate. The noise in the radius, σ_r^2, depends on the assumption about stationary particle movements described in appendix B section B.2. However, vibrations in the mechanical rotation setup will also influence the radius.

For the measurement noise, we can model the covariance matrix R as a diagonal matrix with entries corresponding to the variance of the measured x- and y-coordinates in the images. These are assumed to be equal, such that $\sigma_{\hat{x}}^2 = \sigma_{\hat{y}}^2$.

However, specific values for the variances have not been estimated since the utilized implementation of MHT does not include a Kalman filter for handling the state space model.

Appendix H section H.3 includes a description of an extended state space model. Here the area of the particle blobs as well as an estimate of the angle θ are included in the measurement vector.

Considerations The state space representation described above seeks to describe the physics of the system as well as the relation to the available measurements as good as possible. However, the best model might not necessarily be the one that best describes the physics. Two different goals can be stated for the state space representation:

- Since the model includes the radius of a tracked particle, an estimation of the state using this representations could possibly solve the particle positioning problem. In this way particle tracking and positioning could be combined, and the need for subsequent circle fitting as in chapter 4.4 would disappear. In MHT the Kalman filter is used to estimate the states by processing the associated track measurements recursively. Since our model is nonlinear, the normal Kalman filter can not be used. Instead the extended Kalman filter that basically uses a first order Taylor expansion about the estimate of the current mean and covariance can be used to approximate the state. However, the extended Kalman filter is not an optimal estimator and therefore it might not be sufficient for estimating the radius with enough accuracy.

- In MHT the main reason for using a Kalman filter for each track is to help perform a better data association. In this regard the Kalman filter needs to perform better at all times compared to if we did not use any prediction filter at all. If this is not the case, the data association is compromised and we might experience worse results of the tracking algorithm. Figure 6.14 illustrates this problem if, for instance, the angle θ is estimated such that a particle is assumed to be on the back side of the container instead of the front side. The measured x-coordinate is the same along the dashed line, so the model will fit exactly as good as with $-\theta$ and the same radius. However, this estimation results in a search to the left in the next frame (red dot), whereas the other guess would result in a search to the right. This ambiguity might be solved by adding (nonlinear) constraints to the model as well as using for instance a particle filter instead of a Kalman filter to account for a non-Gaussian probability density function. Also, the assumption about purely additive noise in the measurement model should be revised. However, this has not been investigated.

6.2.4.2 Software

As mentioned in chapter 6.2.1 the implementation of multiple hypothesis tracking (MHT) is all but trivial. It requires a lot of programming effort and results in a large amount of code [55]. Therefore we have not prioritized an implementation of the algorithm ourselves. Instead we have found three

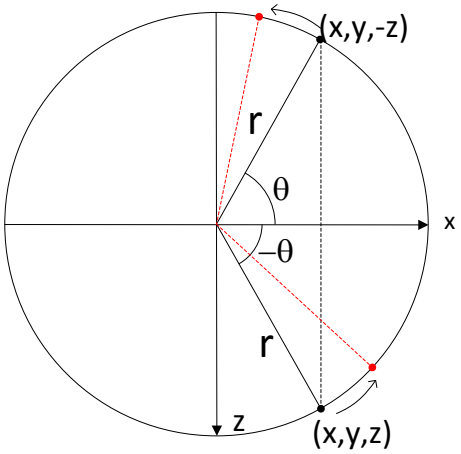


Figure 6.14: Problem in state space representation due to ambiguity in polar coordinate transformation.

different open source implementations. The first is written in pure MATLAB code[6], the second in pure C-code[7] and the third in Java but with a MATLAB wrapper[8]. The Java implementation was chosen because it is fairly well documented [63, 73, 74] and performs well in MATLAB with a tolerable execution time.

The chosen implementation is actually an API for a generic Multiple Hypothesis Library (MHL) that is designed to fit a wide range of problems. Therefore it does not address application specific optimizations and variations such as probability modeling, motion models and sensor models [73]. Also, the interface to MATLAB through the wrapper is rather simplified such that information regarding hypotheses and measurement-to-track associations is not accessible. The operations of the library are implemented close to the original formulation of the conceptual MHT as described in the theory above. Therefore, the implementation is also Hypothesis-Oriented as proposed by Reid. The implementation uses Murty's algorithm [72] to generate the 6 best hypotheses and n-scan-back pruning [69] to make a final decision regarding data associations 6 frames back ($n = 6$).

The consequences of using the MHL implementation that have been discovered are:

1. **Measurements**. The implementation is only able to take two-dimensional positional measurements as input, resulting in tracking without the measured area of a blob. Therefore the extended state space representation described in chapter H.3 is not applicable, and the algorithm will generally have a disadvantage in tracking particles with greatly varying blob areas compared to the first tracking algorithm. In practice the measurements are of the form: $\mathbf{x}_k(i) = \begin{bmatrix} x_k(i) & y_k(i) \end{bmatrix}^T$, where x and y are the two-dimensional coordinates.

2. **Motion model**. Specifically the lack of a motion model is a major issue in our case, because the known physics of the particle movements can not be modeled within the algorithm. The state space representations described in chapter 6.2.4.1 could potentially provide crucial information in the gate computation in equation 6.10. If the prediction of a track state at time $k + 1$ is better than the current measurement at time k, the gate would more likely include the correct track measurement. At the same time it will give the correct measurement-to-

[6] http://en.pudn.com/downloads174/sourcecode/math/detail806954_en.html
[7] http://mediafutures.cs.ucl.ac.uk/people/IngemarCox/downloads/
[8] http://www.multiplehypothesis.com/downloads.html

track association a higher probability according to equation 6.15 and 6.16. In practice, the lack of a motion model corresponds to the state space representation:

$$\begin{aligned} \mathbf{z}_k &= A\mathbf{z}_{k-1} + \mathbf{w}_{k-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + \mathbf{w}_{k-1} \\ \mathbf{x}_k &= H\mathbf{z}_k + \mathbf{v}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \mathbf{v}_k \end{aligned} \tag{6.36}$$

where $\mathbf{v}_k$ is always zero because we trust all associated measurements 100%. This correspond to a null-matrix for the measurement covariance matrix R.

3. **Gating**. In MHT the gate (or validation region) is a multidimensional ellipse according to the dimension of the measurement vector. As described in equation 6.10 and 6.11 the radii of this ellipse depend on the error covariance matrix $P_k(j)$ of track j and the measurement covariance matrix R. However, when R is the null matrix, the radii will always be equal resulting in a circular gate. Therefore, it is not possible to ensure that a particle is allowed to move more along the x-axis than the y-axis from time k to $k+1$.

4. **Tentative/confirmed tracks**. The implementation only outputs the current states of the confirmed tracks. However, confirmed tracks often originate from tentative tracks in earlier frames as described in chapter 6.2.2.5. Therefore, if a track is not immediately confirmed, we miss the associated measurements while the track is only tentative. Figure 6.15 shows a simulation of this phenomenon where the number of lost measurements is related to the density of false alarms, β_{FT}. Only when the density of false alarms is equal to zero, no measurements are lost. At a mean of 100 added noise measurements per frame, an average of 4 measurements of a real track are lost before the track is confirmed. Since we do not have access to previous measurement-to-track associations, these measurements from tentative tracks can not be acquired. An improvement to the MATLAB wrapper would therefore be to include a method of acquiring all measurements associated to a track through time.

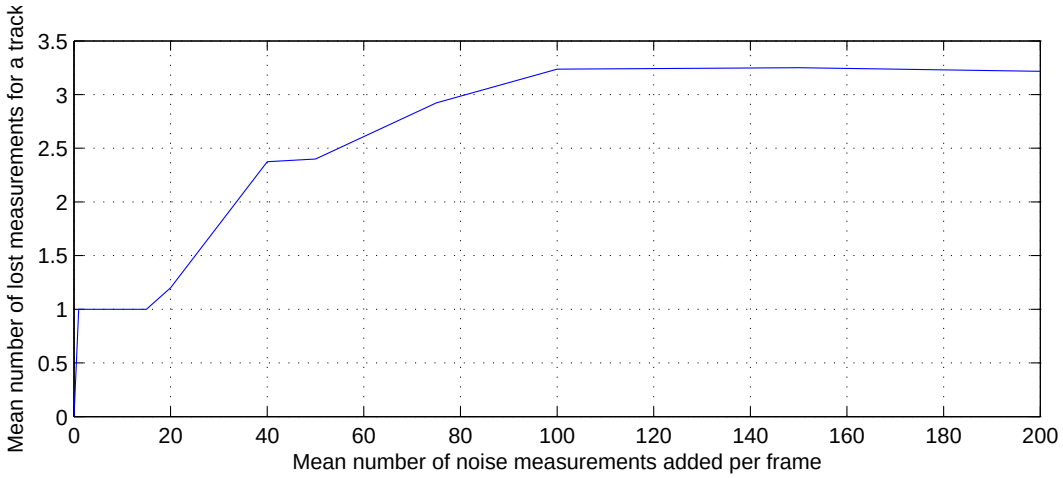


Figure 6.15: Simulation of lost measurements caused by non-confirmed tracks. The mean number of lost measurements for a track is plotted relative to the mean number of added noise measurements per frame.

5. **Best hypothesis**. The implementation outputs the track states of the currently best hypothesis recursively as would normally be the case for a radar tracker, where an operator monitors the currently known targets. In our application we are not interested in the intermediate best hypotheses but instead the overall best hypothesis at time K when all frames have been processed. However, if this was possible with the implementation, we would still have to store tracks when they were deleted, because a hypothesis does not contain information about previously terminated tracks.

6.2.4.3 Track Filtering

Since MHT might generate spurious tracks in a noisy environment, we still have to perform subsequent filtering of the found tracks as was done in the first tracking algorithm. The exact same filters are applied as in chapter 4.3.2. These are:

- **Area**: only blobs with a larger area than 15 px are included.
- **Sequence length**: only blobs that are matched in more than 15 consecutive images are included.[9]
- **Travel distance**: only particles that throughout their sequence travel a distance more than 60 pixels are included.
- **Travel direction**: only particles traveling on the front side of the glass container are included.

In addition to these a filter is added requiring the travel distance along the y-axis to be maximum 50 px. This value is found by evaluating the maximum travel distance of the y-coordinates in the manually labeled training set. The entire training set described in appendix D (table D.2) is used, in which the maximum travel distance was found to be 43 px. Therefore, with a margin, a maximum allowed travel distance along the y-axis is chosen to be 50 px.

6.2.4.4 Initial Evaluation

Figure 6.16 shows an example of using the implementation of MHT in a simulated environment using the particle simulator described in chapter 3.5. The simulator generates 5 particles equally distributed along the y-axis of the container. A mean of 10 noise measurements are added per frame, and a probability of missing detections of $1 - P_D = 0.1$ is used. In the figure all measurements (particles and noise) are marked with blue dots. Red boxes denote measurements that are associated to tracks found by MHT. Black circles denote missing measurements, such that a particle measurement (blue dot) is not present at these locations. As seen from the figure MHT (combined with subsequent track filtering) has found all 5 tracks. In the track located at the middle of the container a total of 4 missing detections occur. This happens with both one and two consecutive missing measurements in a row. At the right side of the same track MHT has associated one of the noise measurements to the track instead of assigning it to a false alarm. This happens partially because the implementation does not contain a physical model of the particle movements and partially because the gate is circular and not elliptical as would be more appropriate for the specific application.

In chapter 4.3.1 the matches between blobs was done from frame to frame, and frame N was matched with frame 1, such that the tracking was circular. In MHT this refinement of the

[9] This number changes when fewer than 87 frames are available

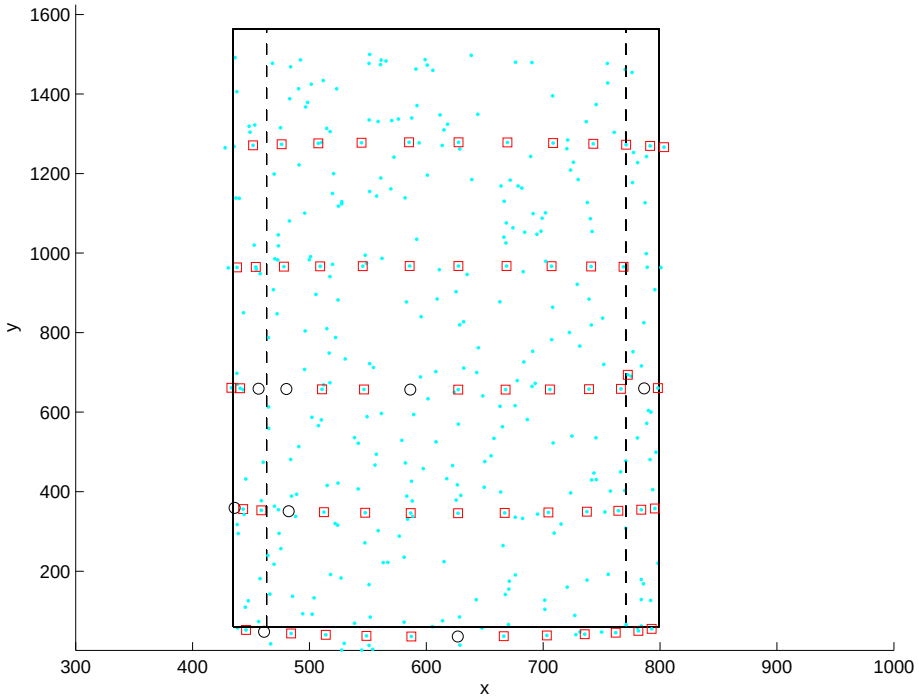


Figure 6.16: Multiple Hypothesis Tracking applied at a simulated problem with 5 generated particles. A mean of 10 noise measurements are added per frame, and a probability of missing detections of $1 - P_D = 0.1$ is used.

algorithm is not easily implemented, since the final decision regarding data associations is done 6 frames back using n-scan-back pruning as described above. Therefore, instead the sequence of all measurements $\mathbf{X}^K$ is concatenated with the first half of the measurements $\mathbf{X}^{K/2}$ such that we have a total of $\frac{3K}{2}$ frames with the last $\frac{K}{2}$ being a copy of the first. In this way we are guaranteed to track a particle for an entire passage over the front side of the glass container regardless of the starting angle θ_0. Subsequent track filtering ensures that no overlapping tracks occur due to this fix.

Parameter Estimation In contrast to the first tracking algorithm presented in chapter 4.3 MHT is a probabilistic algorithm containing environmental parameters related to the problem domain. These parameters were described thoroughly in chapter 6.2.2.4 and are summarized below:

- P_D - Probability of detection. The probability that a previously detected track (prior track) is also detected in the current frame.
- β_{FT} - Density of false alarms. How many false alarms (false detections) on average are present per pixel per frame.
- β_{NT} - Density of new targets. How many new targets on average are present per pixel per frame.

When performing MHT it is assumed that we have estimated these parameters from the problem domain and the environment. However, different methods of estimating them must be applied when using simulations and the actual data set.

In the simulations presented in the next section, we know the exact parameters for these in a simulated environment, and the MHT algorithm can be tested against these using a particle simulator. The formulas used for calculating the parameters are:

$$P_D = 1 - P_{\text{Undetected}} \tag{6.37}$$

$$\beta_{FT} = \frac{\lambda}{V} \tag{6.38}$$

$$\beta_{NT} = 2\frac{N_p}{3N/2}\frac{1}{V} \tag{6.39}$$

Here, P_D is almost directly given as a parameter in the simulation by the probability of undetected particles $P_{\text{Undetected}}$. The calculation of β_{FT} and β_{NT} as densities is done by dividing with the container area V. β_{FT} can be calculated by the mean number of added noise measurements per frame λ as $\beta_{FT} = \lambda/V$. The calculation of β_{NT} is based on the idea that a total of N_p particles are present in the training set. Because we concatenate the measurements with the first half such that the total number of frames is $\frac{3N}{2}$ as described above, all particles are guaranteed to appear two times on the front side of the container. Therefore β_{NT} is calculated as 2 times the total number of particles divided by the number of frames and the container area.

On the actual dataset we can train the parameters based on the training set described in appendix D (table D.2). This is done by calculating the number of true detections (TP), false detections (FP) and missing detections (FN) of blobs and the total number of particles N_p present in the training set. The three algorithm parameters can then be calculated as:

$$P_D = \frac{TP}{TP + FN} \tag{6.40}$$

$$\beta_{FT} = \frac{FP}{N}\frac{1}{V} \tag{6.41}$$

$$\beta_{NT} = \frac{3}{2}\frac{N_p}{N}\frac{1}{V} \tag{6.42}$$

The calculation of β_{FT} is done by taking the total number of false detections and divide by the number of frames and the container area. The calculation of β_{NT} is based on the idea that a total of N_p particles are present in the training set. Half of these will have a starting angle on the back side of the container ($\theta_0 \in [0; \pi]$) appearing only one time on the front, and half of these will have a starting angle on the front side of the container ($\theta_0 \in]\pi; 2\pi[$) appearing two times on the front. The average appearance on the front is thus $\frac{3}{2}$. Therefore β_{NT} is calculated as $\frac{3}{2}$ times the total number of particles divided by the number of frames and the container area.

The actual numbers found by evaluating the training set and the hand labels are:

- $P_D = 0.88$. The number is quite low which is caused by missing detections primarily located at the edges of the container.
- $\beta_{FT} = {}^{45}/_{V} = 7.59 \cdot 10^{-5}$. In average 45 noise measurements are present per frame.
- $\beta_{NT} = {}^{0.88}/_{V} = 1.49 \cdot 10^{-6}$. In average 0.88 new targets are present per frame.

6.3 Evaluation

As mentioned in the introduction the two proposed tracking algorithms differ by being respectively deterministic and probabilistic in their nature. The Circular Trajectory Tracker (CTT) defines cost functions of data-associations and seeks to minimize these iteratively. Multiple Hypothesis Tracking (MHT), on the other hand, defines and evaluates data-association hypotheses with probabilities based on combinatorics and model parameters. Another fundamental difference between the two is that CTT is an offline algorithm processing all frames and measurements at the same time, whereas MHT is an online algorithm improving its output recursively based on incoming measurements. Since our problem is cyclic in its measurements the most meaningful approach would probably be an offline algorithm. In the specific implementation MHT is a 6-scan algorithm, whereas CTT can be categorized as a 3-scan algorithm although its method of handling occlusion might be seen as a K-scan approach (K being the total number of frames). Commonalities for the two algorithms include their gating procedures as well as the track filtering procedure removing unrealistic particle detections. Also, both algorithms contain parameters that must be trained prior to execution. For MHT these parameters are probabilistic and easily trained based on a training set. For CTT, however, the interconnection between the different parameters remains unknown, and the training must therefore be performed manually.

In the following, both the first tracking algorithm and the two advanced tracking algorithms are tested and evaluated based on simulated data. As described in the introduction to this chapter some of the problems observed in the first tracking algorithm concern added noise and missing detections. These aspects can be simulated using the particle simulator described in chapter 3.5. By using a simulator we can test the tracking algorithms in a more controlled environment with a large amount of generated data in order to gain a statistical foundation. Throughout the simulations we refer to the first tracking algorithm by FTA, the Circular Trajectory Tracker by CTT and Multiple Hypothesis Tracking by MHT. In all, three different scenarios are considered in order to test performance and evaluate the three algorithms with respect to each other:

1. **Added noise**. The algorithms are tested with an increasing amount of uniformly distributed noise measurements on the simulated glass container. This corresponds to increasing the density of false alarms β_{FT} in MHT.
2. **Missing detections**. The algorithms are tested with an increasing probability of missing detections corresponding to lowering the probability of detection P_D in MHT.
3. **Number of measurements**. The algorithms are tested on their ability to track particles with a decreasing number of available frames/measurements. Added noise and missing detections are applied corresponding to the noise level and probability of missing detection in the training set described in appendix D (table D.2).

Measurement noise is not simulated for more reasons. Often measurement noise is relevant if the sensor technology comes with large uncertainties like for instance in radar. However, in our case the measurement noise caused by inaccuracies in the camera technology is far exceeded by other phenomena like refraction in the fluid and projective transformations of particle shapes. Instead of zero-mean normal-distributed noise, these phenomena include offsets and nonlinearities that are not covered in this thesis. In addition, adding noise to the measurements without at the same time adding false detections will not introduce data-association errors, and thus the simulation can not be performed easily by varying just a single parameter.

In all simulations the numbers of true positives (TP), false positives (FP) and false negatives (FN) are counted. A true positive occurs if a sequence of measurements (a minimum of 15) of a found track is sufficiently close to the corresponding sequence of generated real particles. This is the same criteria as described in chapter 3.4.2. However, here a radius of only 5 px is used as the "closeness criteria" because the accuracy of the simulator is in general much better than the process of hand labeling particles. Likewise, a false positive occurs if a found track can not be associated with a real particle, and a false negative occurs if a real particle is not detected.

In all simulations the radii of the generated particles are equal to the radius of the container. This is done to ensure that only relevant parameters are varied and because the subsequent positioning of the particles is not the focus of the evaluation. In all but the last simulation the number of frames used is 87 corresponding to the experimental setup described in chapter 3.4.1. In the last simulation this number is varied by downsampling the number of frames with an integer factor.

Added noise The three algorithms are tested and compared under the influence of added noise (false alarms/detections). The noise is uniformly distributed in the container area, and the number of noise measurements per frame is drawn from a Poisson distribution as described in chapter 6.2.2.4. The noise measurements also include an area which is uniformly distributed over the range of the real particle blob areas. Figure 6.17 shows a simulation of the impact of added noise for the three tracking algorithms. A total of 20 particles are generated per run, and a mean of 50 runs is calculated for each plotted value-pair. The results are shown as the particle detection rate (also known as recall) and the precision of the particle detection as a function of the mean number of noise measurements added per frame. Precision and recall are both measures of particle detection performance. Precision is the fraction of detected particles that represent real particles, whereas recall is the fraction of real particles that are detected [23]. That is:

$$Precision = \frac{TP}{TP + FP} \tag{6.43}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.44}$$

As seen from the figure FTA generally has the lowest precision, meaning that it finds a lot of false tracks that do not represent real particles. Also, when more than 100 noise measurements are present per frame, it tends to miss some of the real tracks resulting in false negatives. MHT generally performs better with a higher recall and a considerably higher precision. However, CTT unquestionably outperforms both MHT and FTA since it has a precision and recall of both 1 throughout the entire simulation.

Missing detections The three algorithms are tested and compared under the influence of missing detections/measurements. A probability of detection P_D for a particle in a frame is varied to introduce missing measurements. Figure 6.18 shows a simulation of the impact of missing measurements for the three tracking algorithms. A total of 20 particles are generated per run, and a mean of 50 runs is calculated for each plotted value-pair. The results are shown as a function of the probability of missing measurements, $1 - P_D$. When $1 - P_D = 0.1$ it means that in average every 10'th measurement of a particle is missing. Because of the probabilistic simulation this means that some particles do not experience missing measurements at all, whereas other particles might experience several missing measurements, potentially in conjunction.

From the figure we see that the recall of FTA falls drastically even for low probabilities of undetected measurements. This is due to the fact that FTA requires a consecutive sequence of matches and therefore does not tolerate missing detections. Instead, several tracks are created

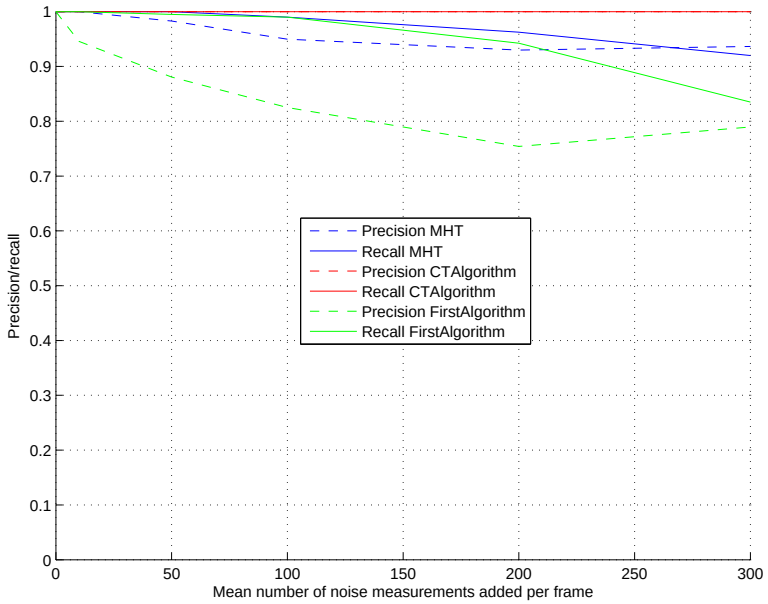


Figure 6.17: Simulation of precision and recall as a function of added noise. MHT is plotted with blue, CTT is plotted with red and FTA is plotted with green. Solid lines denote recall and dashed lines denote precision. The noise is uniformly distributed in the container area and the x-axis denotes the mean number of noise measurements added per frame. A mean value based on 50 repetitions is shown.

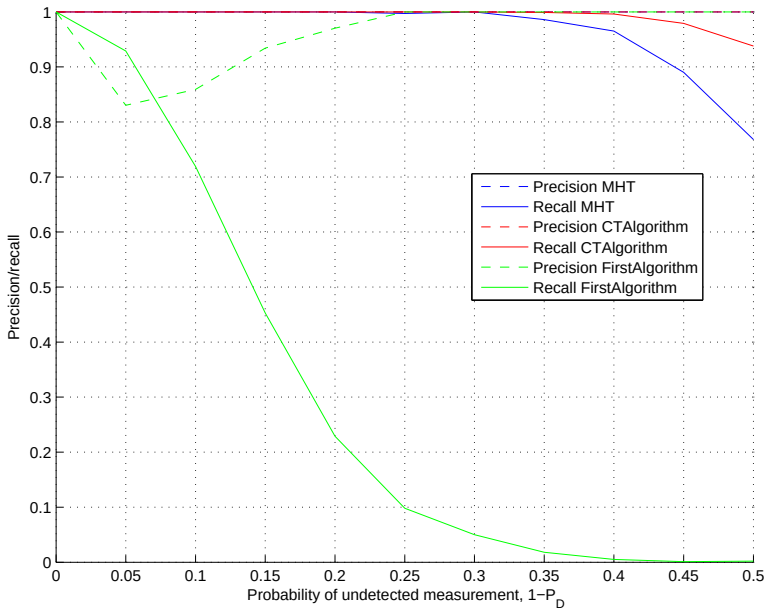


Figure 6.18: Simulation with 20 targets and missing data points. The targets are distributed equally vertically along the container with random initial angles, θ_0. The x-axis denotes the number of missing measurements per frame, and the y-axis denotes the number of targets tracked by the algorithm. The missing measurements are randomly drawn from the set of target measurements for each frame.

(one for each consecutive sequence of measurements), and only those consisting of at least 15 matches count as true positives. Therefore, the precision is also lowered because the small tracks are considered false positives. MHT and CTT both track particles perfectly up to a probability of undetected measurements at $1 - P_D = 0.25$. However, at around $1 - P_D = 0.3$ the recall of MHT drops off indicating that it starts loosing the tracks or does not contain enough matches. The same happens for CTT at around $1 - P_D = 0.4$.

Number of measurements The three algorithms are tested and compared with a varying number of frames (and thus measurements). By downsampling the original 87 frames by integer factors, the algorithms are evaluated when respectively 87, 44, 29, 22, 18, 15, 13 and 11 frames are available. In order to simulate a realistic environment, the noise characteristics and missing measurements observed in the training set described in appendix D (table D.2) are used. As described in chapter 6.2.4.4 this corresponds to a mean of 45 noise measurements added per frame and a probability of undetected particles of $1 - P_D = 0.12$. Figure 6.19 shows a simulation of the performance of the three tracking algorithms as a function of the number of frames. When the number of frames available is lower than 87, so is the requirement of a minimum of 15 detections in a track sequence before a track is marked as a true positive. The exact requirement used to evaluate found tracks by their number of matches is $N_{\text{matches}} = \left\lceil \frac{15 \cdot N_{\text{frames}}}{87} \right\rceil$.

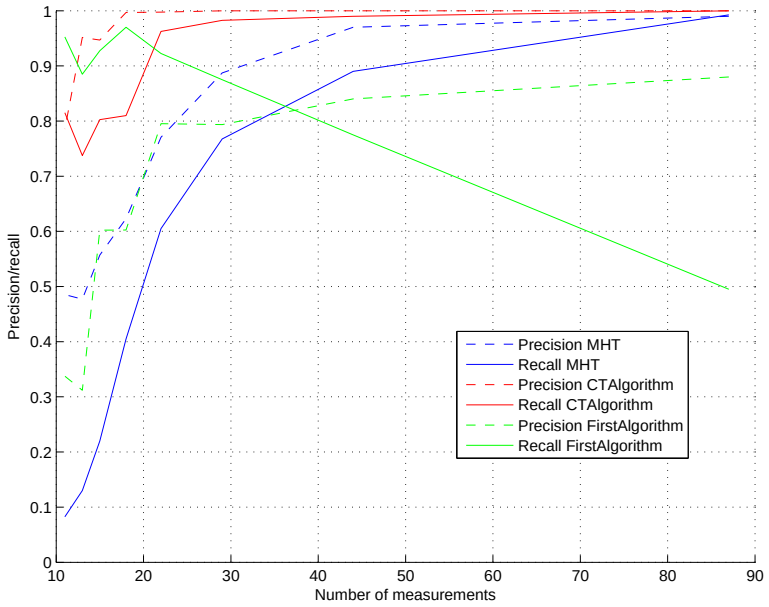


Figure 6.19: Simulation of precision and recall as a function of the number of measurements available for each track. MHT is plotted with blue, CTT is plotted with red and FTA is plotted with green. Solid lines denote recall and dashed lines denote precision. The maximum number of frames available is set to 87 as in the actual dataset. By downsampling the original 87 frames by integer factors, the algorithms are evaluated when fewer frames are available.

From the figure we see that FTA has poor performance when the number of frames is large. As in the simulation of missing detections this is due to the fact that FTA requires a consecutive sequence of matches and therefore does not tolerate missing detections. Because the number of required matches falls with the number of frames, FTA is more likely to track a consecutive sequence of matches, and therefore the recall gets better. The precision on the other hand drops when fever frames are available since the search area is larger, and thus noise measurements are more likely to be associated with the track. At 87 frames MHT experiences almost the same

performance as CTT, detecting all particles. However, both the precision and recall of MHT drop significantly below 44 frames. This is partly due to the circular gate and the lack of a motion model resulting in wrong data associations and partly due to the lost measurements before a track is confirmed as described in chapter 6.2.4.2. CTT generally experiences the best performance down to 22 frames. Below this, the performance drops off due to the parameter settings of the algorithm and the requirement of a minimum of 3 consecutive matches as described in chapter 6.1.5.

6.3.1 Summary

The three algorithms, FTA (First Tracking Algorithm), CTT (Circular Trajectory Tracker) and MHT (Multiple Hypothesis Tracking), have been evaluated above based on their performances in a simulated environment. Table 6.4 summarizes the results of these simulations and adds complexity and execution time to the comparison of the algorithms.

Algorithm	Complexity	Added noise	Missing detections	Number of frames	Execution time
FTA	🟢	🔴	🔴	🟢	🔴
CTT	🟡	🟢	🟢	🟢	🟡
MHT	🔴	🟡	🟢	🔴	🟡

Table 6.4: Simulator-based performance comparison of the three proposed tracking algorithms – First Tracking Algorithm (FTA), Circular Trajectory Tracker (CTT) and Multiple Hypothesis Tracking (MHT). Green denotes good performance, yellow denotes medium performance and red denotes poor or no performance.

FTA is the least complex algorithm. It performs well at only a few number of available frames, however it is not resistant towards added noise and missing detections. The execution time is high, but could easily be lowered significantly using the same matching principles as in CTT.

CTT is considerably more complex. However, it also handles both added noise and missing detections and does this at only a few number of frames available. It is faster than FTA but still slow compared to the real-time requirements of the inspection system. It is expected to be faster for a data-parallel implementation of the designed CUDA kernels.

MHT is the most complex of the algorithms. It handles missing detections almost as well as CTT and handles added noise to some extent. However, it requires more available frames, because it looses measurements before confirming a track. The execution time is relatively high with just a few measurements per frame, but because it has a maximum number of hypotheses, the increase when excessive noise is added is much lower than both FTA and CTT. Since most of the MHT code is implemented in JAVA it is already to some extent optimized for speed.

The performance on the real dataset is presented in chapter 7. Here the overall results of the system as well as individual algorithm results of the three tracking algorithms are presented.

7

Results and Discussion

In this thesis three different algorithms have been proposed. The **first algorithm** covers the whole system including segmentation, tracking, positioning and classification of particles and containers. The **second algorithm**, the Circular Trajectory Tracker, builds upon the first algorithm and replaces the tracking- and positioning-steps of this in order to increase accuracy and timing performance. The **third algorithm**, Multiple Hypothesis Tracking, replaces only the tracking-step of the first algorithm as an advanced general method for multi-target tracking.

In the following these three algorithms are evaluated based on the test set of 20 containers described in appendix D table D.3.

For verification and comparing the results of the three different algorithms a machine vision setup is made as described in chapter 3. The list below summarizes the conditions for achieving the results as described in the following chapters.

- **Camera and lens:** A Basler camera (acA1600-20gc) with a 25 mm focal length lens, an f-stop number of approximately 8 and an exposure time of 300 μs is used. The camera is horizontally oriented achieving an optimal resolution. A working distance of 180 mm is used from camera to glass container.

- **Rotation speed:** A steady rotation speed of the container (1186 rpm) relative to the camera frame rate (20 fps) results in 87 frames distributed equally around each glass container. It is assumed that the vibrations in the mechanical rotation device are minimal such that the container rotates straight about its own center axis. The recorded sequence of frames is processed by skipping a number of frames in steps of 0-7 images resulting in a reduced needed number of frames for processing. By this method we are able to compare the accuracy of the algorithms when we have between 11-87 images in a sequence.

- **Particle:** It is assumed that particles are not moving relative to the rotating container at the steady rotation speed according to theory and observations in appendix B.2. Background illumination ensures that particles are visible as dark blobs on a white background due to light absorptions.

- **Container:** Only cylindrical and transparent glass containers with clear liquids can be inspected. The algorithms require an average measured pixel diameter of the glass container, and the area of the image to be inspected is defined by a crop area.

7.1 First Algorithm

This section contains a presentation of the results concerning the first algorithm described in chapter 4 and summarized below.

- **Container Diameter and Center** measured by using Roberts edge detection, Hough Transform and finding a best diameter.
- **Segmentation** using a background image and looking for dark blobs relative to a white illuminated background.
- **Tracking** particle trajectories using a successive number of blob matches in a sequence of images (only when a particle is moving on the front side of the glass container). Matches are found by comparing particle areas and by using blob centers within a rectangular window between successive frames.
- **Positioning** particles by estimating a radius for the particle circular trajectory with a circle fitting of tracked blob centers.
- **Classification** using estimated particle radii and measured diameters of glass containers.

The recorded images are used in the presentation of the final results and findings followed by a discussion with focus on where and how to improve the algorithm.

7.1.1 System Results

Figure 7.1 shows the results of the evaluation of the test set as a confusion matrix linking the observed values (manually labeled) to the detected values by the algorithm. 9 containers are accepted correctly and 9 containers are rejected correctly. However, one container is rejected incorrectly (FN) and one container is accepted despite actual particles inside (FP). The false negative concerns the container G309 where noise in the bottom area causes a false particle detection which is estimated to be inside the container. The false positive concerns the container H9 that contains an extremely small particle overlooked by the segmentation algorithm due to its low contrast and small blob area. See appendix E for videos of a selected number of containers from the test set.

		Detection		
		p′	**n′**	**total**
Observation	**p**	45% (9)	5% (1)	10
	n	5% (1)	45% (9)	10
	total	10	10	

Figure 7.1: Confusion matrix for the test set. Numbers in brackets specify the actual container counts.

The above results were acquired with a camera frame rate corresponding to 87 frames per revolution.

With a frame rate of 20 fps this corresponds to an acquisition time of $\frac{87\,frames}{20\,frames/s} = 4.4s$, which is far from the desired objective (see chapter 1.2). In section 4.4.1 the theoretical limit for how many frames were needed in order to distinguish particles on the inside from particles on the outside of a container was estimated. Here a value of at least 18 frames per revolution was anticipated which corresponds to an acquisition time of $0.9s$ with the current camera setup. In figure 7.2 the accuracy of the system is plotted as a function of how many frames are used. The accuracy is described by the equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

As expected the best accuracy is obtained at 87 frames. The most significant drop happens at around 29 frames, below which the accuracy falls off drastically. Contrary to expectation, at 11 and 13 frames the accuracy increases again. This phenomenon is caused by a general rise in false particle detections due to a smaller "sequence length" in the tracking algorithm (section 4.3.2) when fewer frames are used. In the figure two lines are drawn indicating the accuracies if naive guesses on all particles being inside or outside are conducted. An accuracy of 55% is achieved if all particles are classified as being inside the containers.

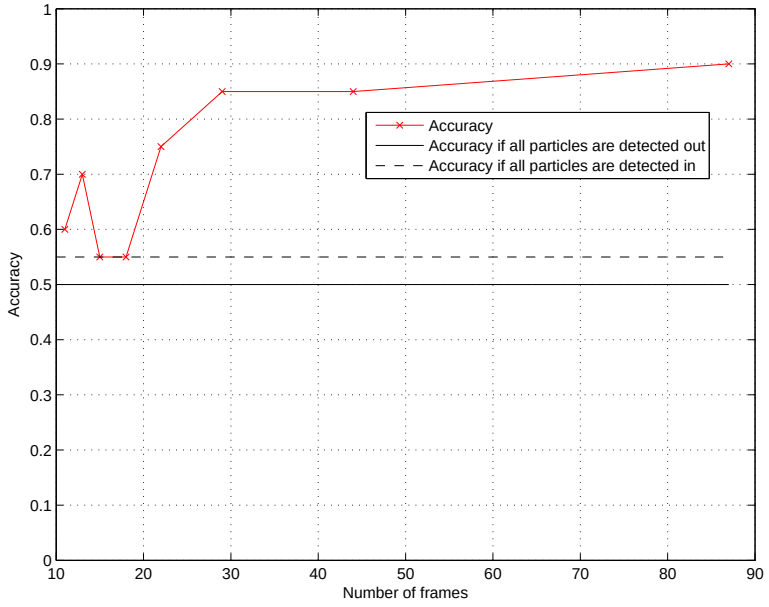


Figure 7.2: Accuracy plotted as a function of the number of frames used in the algorithm.

7.1.2 Algorithm Results

In the above description, the performance of the system is evaluated on a container-level – regarding each container as accepted/rejected based on all of its particles. In addition to this, it is possible to evaluate the performance on a particle-level. This corresponds to having a container for each particle, so that the decision to accept/reject is based on only one particle. By evaluating the algorithm on a particle-level, more detailed results concerning the individual parts of the algorithm can be investigated. The following three sections evaluate individual parts of the algorithm:

- **Blob Detection**. The segmentation algorithm is evaluated on image-level based on how many of the manually labeled particle-points are detected as blobs and how many false detections occur.

- **Particle Detection**. The tracking algorithm is evaluated on image-sequence-level based on how many particles are detected and how many false detections occur.
- **Particle Positioning and Classification**. The positioning and classification algorithms are evaluated in conjunction based on how many of the particles are classified correctly as being either inside or outside the containers.

Finally, execution times of the algorithm in MATLAB are evaluated with relation to the required performance of the system.

7.1.2.1 Blob Detection

Blob detection concerns the segmentation algorithm and its performance. The segmentation algorithm finds a number of blobs in each image that represent potential particles. If a blob matches a manual labeled particle on image-level it is a true positive. If no matches are found, it is a false positive. On the other hand, if a hand labeled particle is not found by a blob, a false negative occurs. As described in chapter 3.4.2 a blob is matched if it is sufficiently close to a manual labeled particle (within a radius of 50 px).

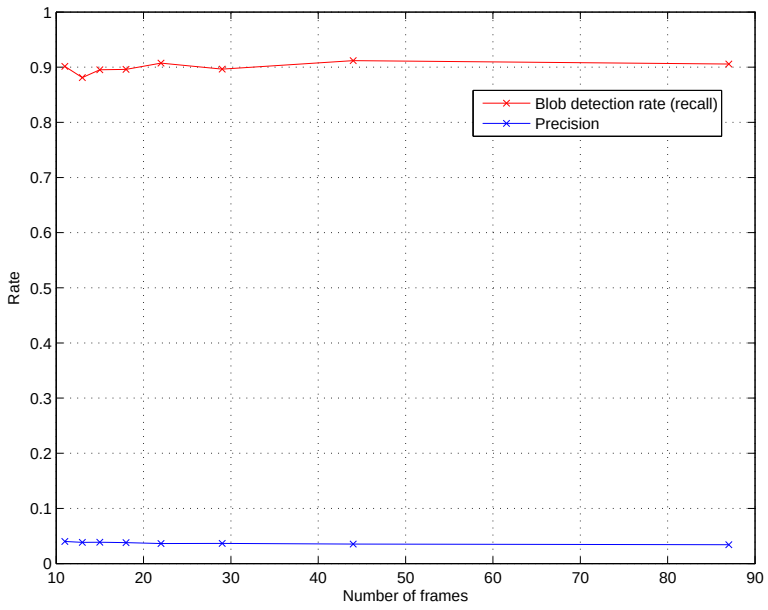


Figure 7.3: Blob detection rate (recall) and precision of the segmentation algorithm. The rates are plotted as a function of the number of frames used.

In figure 7.3 the blob detection rate (also known as recall) and the precision of the blob detection are illustrated as a function of the number of frames used. Precision and recall are both measures of the detection performance. Precision is the fraction of detected blobs that represent real particles in each image, whereas recall is the fraction of real particles that are detected in each image [23].

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.3}$$

As expected the performance of the blob detection is independent of the number of frames used, since blob detection is performed on image-level and not image-sequence-level. Around 90% of all particles are detected in each image. The 10% missing particles are caused by particles located at the bottom and the border of containers and particles that are either too small or too blurred to be segmented. Only around 4% of the detected blobs actually represent real particles. This is mainly due to a significant amount of false detections at the bottom and at the borders of the containers. An average of around 45 false blob detections occur at every frame. However, since these are not all matched from frame to frame in the tracking algorithm, they do not necessarily cause a problem.

To conclude, an improvement in the segmentation algorithm would primarily concern particles located at the bottom of containers. Here, the general assumption about light absorption rather than light reflection does not always hold, and thus not all particles are detected. The large number of false detections can be compensated for in the tracking algorithm, but adjusting a parameter like the minimum size could also improve the precision.

7.1.2.2 Particle Detection

A particle is detected by the algorithm if it is successfully tracked in a successive number of frames on the front side of the glass container. If a particles matches a manually labeled particle it is a true positive. If no matches are found, it is a false positive. On the other hand, if a manually labeled particle is not detected, a false negative occurs. As described in chapter 3.4.2 a particle is matched if it is sufficiently close to a manually labeled particle in a sequence of images. The length of this sequence is determined by the number of frames used. At 87 frames a sequence of 15 consecutive matches must occur, and at 11 frames 2 consecutive matches must occur, in order for a particle to be detected.

In figure 7.4 the particle detection rate (also known as recall) and the precision of the particle detection are illustrated as a function of the number of frames used. Here, precision is the fraction of detected particles that represent real particles, whereas recall is the fraction of real particles that are detected.

The particle detection rate is relatively constant at around 74% (34 particles detected out of 46 in total). This means that most particles are detected even though the number of frames is decreased from 87 to 11. As for the blob detection, missing particles are caused by particles located at the bottom of containers and particles that are either too small or too blurred to be segmented. The majority of the missing particle detections occur at the container J401 that consists of several small air bubbles inside the container. A few particles loose tracking due to occlusion of other particles and some are not tracked due to large changes in appearance between frames. One particle is located closely to the center of the container, such that its radius is very small. Due to a minimum travel distance required by the tracking algorithm in order to filter out noise, this is incorrectly filtered out and causes a missing particle.

The precision of the particle detection falls off drastically at around 29 frames. At this point a significant increase in false detections occurs, mainly because fewer blobs are needed in a consecutive sequence in order to detect a particle. This phenomenon introduces particles caused by incorrectly matched noisy blobs, especially at the bottom of the containers. At 87 frames around 11% (4 particles) of the detected particles are false detections. These are caused by small or blurred particles that are not observed during hand labeling, even though they are actually present in the containers. If these are detected by the algorithm they contribute as false detections.

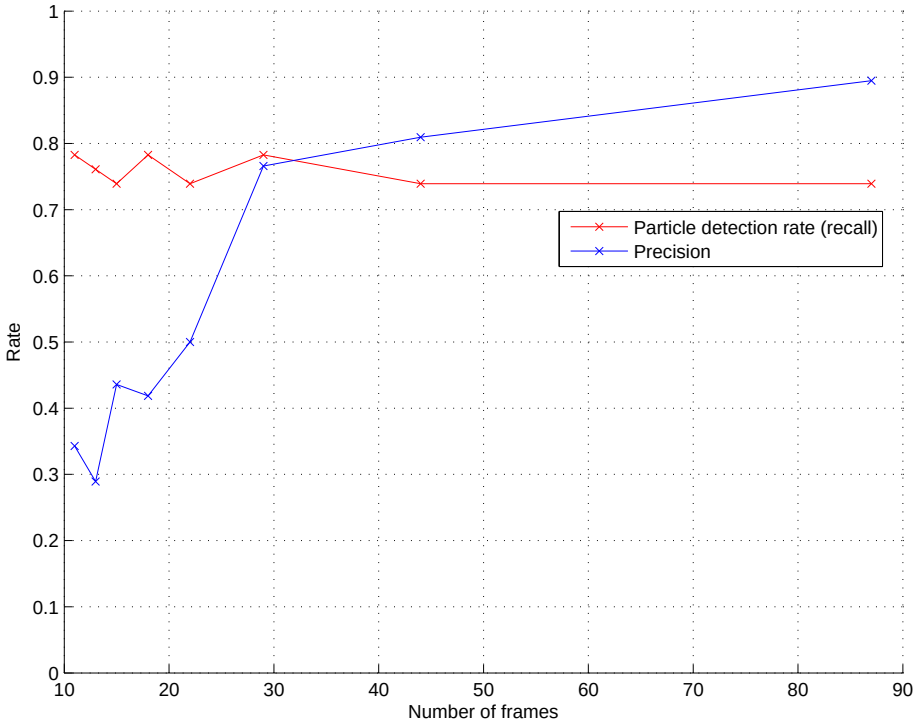


Figure 7.4: Particle detection rate (recall) and precision of the tracking algorithm. The rates are plotted as a function of the number of images used.

To conclude, a minimum number of 29 frames must be used in order to achieve an acceptable performance. An improvement in the tracking algorithm would primarily concern minimizing false detections when the number of frames is decreased. Also, an ability to detect particles even though they are not matched strictly in a consecutive sequence can improve the performance. A tracking algorithm that utilizes a motion model of each particle could contribute to a larger precision and simultaneously enable the detection of particles in spite of missing blob detections or matches.

7.1.2.3 Particle Positioning and Classification

The positioning algorithm is difficult to test in isolation since the hand labeling is only performed on classification basis. This means that true information is only available on whether a particle is located on the inside or outside of a container, and not on the actual values of the particle radii. However, by constructing a histogram of the estimated radii and coloring the counts based on their origin, a rough picture of the positioning performance can be obtained. Figure 7.5 illustrates a histogram of the estimated radii for 87 frames. These have been normalized based on the measured container diameters by the equation:

$$r_{norm} = \frac{r}{d_c/2 + \mathit{offset}} \tag{7.4}$$

where r is the estimated radius, d_c is the measured container diameter and *offset* is a global offset trained in chapter 4.5.

The green and red bars denote particles that are detected and observed manually being respectively on the outside and inside of the containers. The blue bars denote false detections of particles – that is, particles that are not observed manually but are detected by the tracking algorithm.

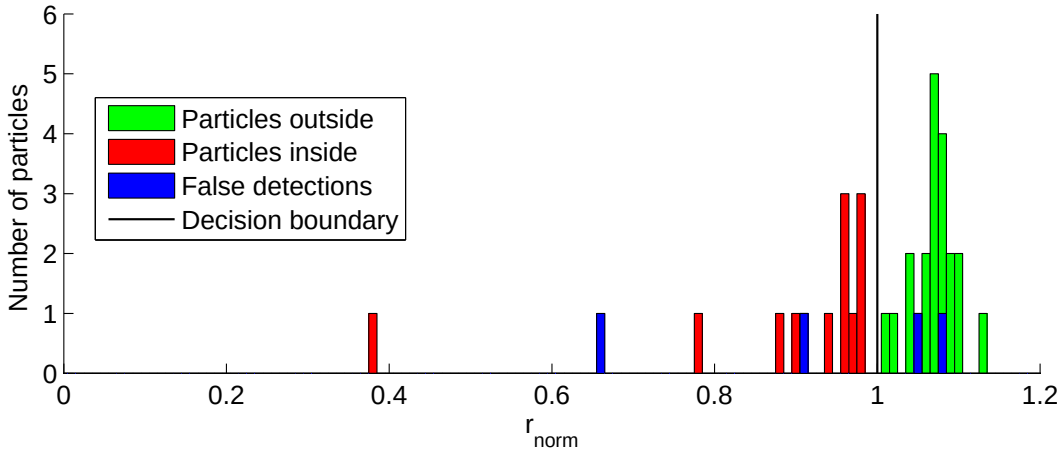


Figure 7.5: Histogram of estimated radii. The radii are normalized to the measured container diameters using equation 7.4.

A black line denotes the decision boundary in which the trained *offset* is included. From the figure it is clear that all observed particles are classified correctly. However, two particles (green bars) are positioned very close to the decision boundary. These concern two particles from the container G309 where large particles skew the blob centers when close to the container edges in the images. This phenomenon introduces distortions of the x-coordinates resulting in smaller estimated radii. Three 'outliers' having small radii are also noticed. One of these is a false detection belonging to the container G309 as mentioned above. Due to noise in the bottom of the container a false detection occurs during tracking and is positioned within the container. The other two are observed particles (red bars) where the smallest belongs to the container H10 and the other belongs to H7. Because these particles are positioned within a high viscosity gel, they are not pushed outwards in the liquid during rotation. Therefore, although being 'outliers' in the figure, they actually represent the true radii of the particles.

Using only the particles that are both detected and manually observed (green and red in figure 7.5) the confusion matrix in figure 7.6 is constructed based on the test set. **100% of the particles are classified correctly** based on their estimated radii. 21 particles are classified correctly as being outside the containers and 13 are classified correctly as being inside.

		Detection		
		p′	**n′**	**total**
Observation	**p**	62% (21)	0% (0)	21
	n	0% (0)	38% (13)	13
	total	21	13	

Figure 7.6: Confusion matrix for positioning and classification on particle-level. Numbers in brackets specify the actual particle counts.

In figure 7.7 the accuracy of the positioning and classification algorithms is plotted as a function of how many frames are used.

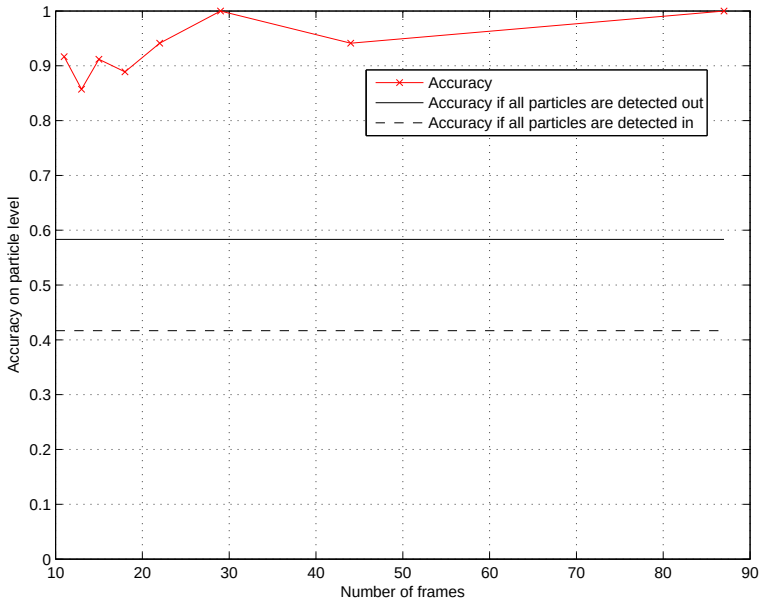


Figure 7.7: Accuracy of positioning and classification plotted as a function of the number of frames used in the algorithm.

At 87 frames the accuracy is 100% as stated above. At 44 frames this falls off a bit, because two large particles in the container G309 are misclassified as being inside, because their estimated radii are too small. This is caused by a general problem with large particles that are estimated with too small radii. At around 29 frames the accuracy falls off a bit. This is caused by the fact that fewer measurements of each particle are available, and thus the uncertainty increases. In the figure, two lines are drawn indicating the accuracies if naive guesses on all particles being inside or outside are conducted. An accuracy of 58% is achieved if all particles are classified as being outside the containers.

To conclude on the results for the positioning and classification algorithms, the performance of these is quite accurate. A problem concerns large particles that are generally estimated with a too small radius. As described above this problem is caused by distortions in the x-coordinates of the blobs when they are close to the container edges in the image. A solution to this problem could be to compensate for the x-coordinate distortions by utilizing the decreasing blob area at the edges. However, this would involve an expansion to the algorithm, which is not the focus of this first version of the system.

7.1.2.4 Timing

The total average processing time for one glass container is illustrated in figure 7.8. The execution time is measured based on the first version of the algorithm implemented in MATLAB running on Windows with an Intel i7 Core processor as described in appendix F section F.1. From the figure it is observed that the algorithm to extract the background image is the most time consuming part of the system. The background extraction process is independent of the number of frames since the same number of images are used every time the background image is computed. If the algorithm is executed sequentially the acquisition curve defines the limit being able to inspect a new container

with the given camera frame rate. Loading images requires approximately half of the acquisition time. The positioning time varies nonlinearly as a function of the number of frames and will be a limiting factor for a few number of frames. The increase in positioning time correlates with the increasing number of false particle detections as discussed in the previous chapter, since the execution time depends on the number of particles whose position must be estimated.

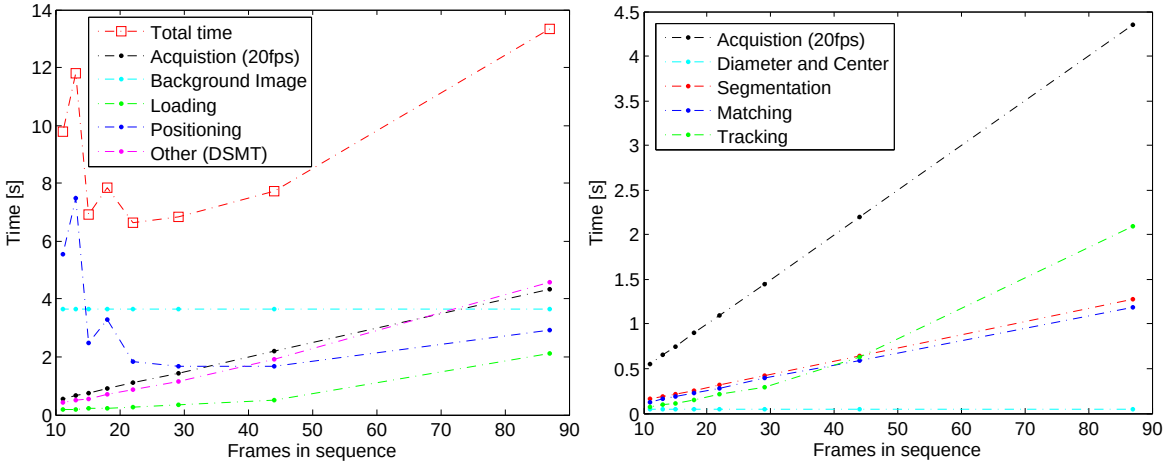


Figure 7.8: Total execution time for processing one glass container as a function of the number of frames. Acquisition indicates the recording time with a Basler camera of the type acA1600-20gc (20 fps). DSMT abbreviates Diameter, Segmentation, Matching and Tracking.

In figure 7.8 to the right is illustrated the execution time for cylinder diameter and center measurement, particle segmentation, matching and tracking. Only the diameter and center measurement is independent of the number of frames.

In achieving a real-time version of the system, the background extraction and the positioning methods need to be optimized. From table 7.1 it is clear that the MATLAB version of the system needs to be optimized in the area of 3-18 times being able to achieve the same performance as the camera acquisition rate with a serial realization of the algorithm.

Number of frames	**Acquisition time**	**DSMT time**	**Total time**
11	0.55	0.41	9.8
13	0.65	0.49	11.8
15	0.75	0.56	6.9
18	0.9	0.69	7.9
22	1.1	0.86	6.6
29	1.45	1.2	6.8
44	2.2	1.9	7.7
87	4.35	4.6	13.3

Table 7.1: Mean execution time per container in seconds for different parts of algorithm as a function of the number of frames.

A real-time design of the algorithm is described in chapter 5, and figure 7.9 illustrates the timing results after an implementation of the proposed optimization. Here a GPU implementation of the background extraction and the positioning algorithm using the Levenberg-Marquardt opti-

mization is realized. From the figure it would be possible to execute the optimized first algorithm running in parallel with image acquisition using a camera frame rate of 20 fps.

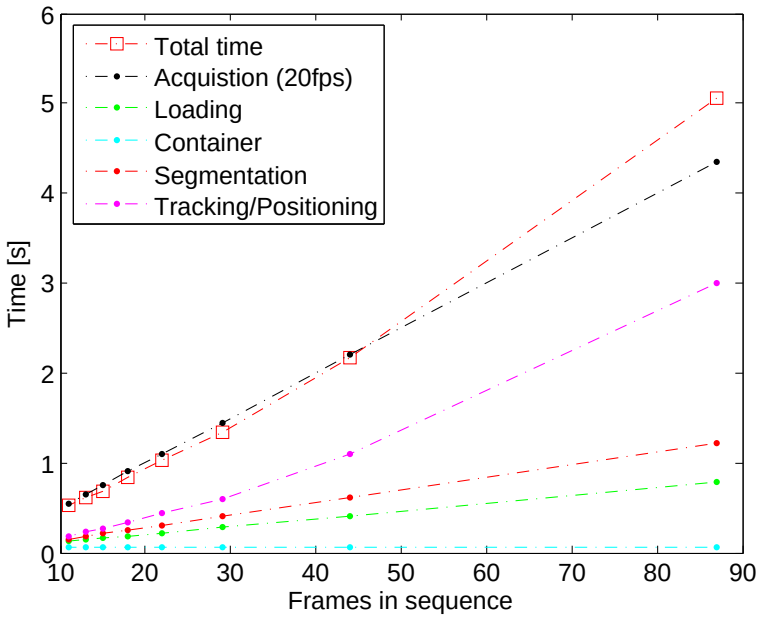


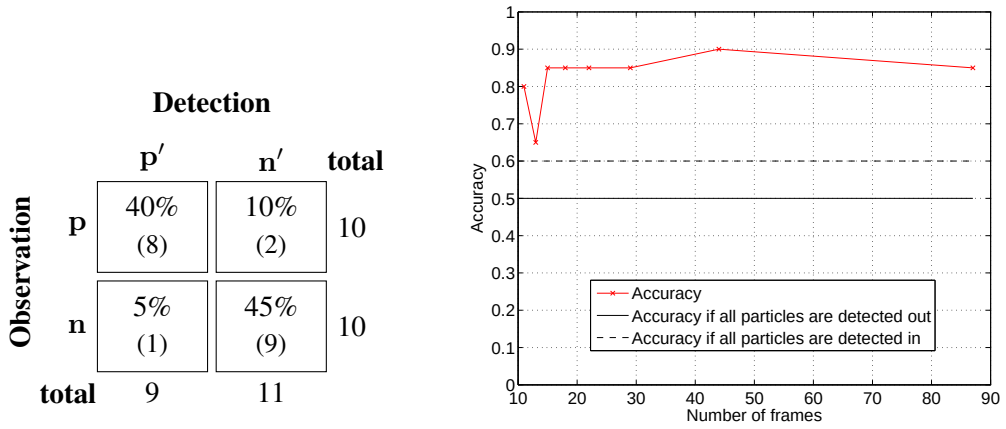
Figure 7.9: Total execution time after optimization of the background extraction and positioning algorithm. Time for tracking and positioning is combined in "Tracking/Positioning". "Container" includes background extraction and diameter and center measurement.

7.2 Second Algorithm – Circular Trajectory Tracker

The second algorithm uses the exact same principles as the first algorithm. However, instead of the steps 'Tracking' and 'Positioning' a new combined tracking and positioning algorithm called Circular Trajectory Tracker (CTT) is used. CTT has been developed by using a bottom-up approach based on the first algorithm and is described in chapter 6.1.

7.2.1 System Results

Figure 7.10a shows the results for the evaluation of the test set. 8 containers are accepted correctly and 9 containers are rejected correctly. However, two containers are rejected incorrectly (FN) and one container is accepted despite actual particles inside (FP). The false negatives concern the containers G309 and G310. The false positive concerns container H9 that contains an extremely small particle which is overlooked. Both G309 and H9 are incorrectly classified for the same reasons as described for the first algorithm.



(a) Confusion matrix for the test set. Numbers in brackets specify the actual container counts.

(b) Accuracy plotted as a function of the number of frames used in the algorithm.

Figure 7.10: Confusion matrix and accuracy for CTT.

In figure 7.10b the accuracy of the system is plotted as a function of how many frames are used. The accuracy falls off below 15 frames, which is better than the first algorithm. Here a drop happens already at 29 frames.

Contrary to expectation the best accuracy is obtained at 44 frames. Here the container G310 is classified correctly, whereas G309 and H9 are still misclassified as described above. A false particle blob is detected for container G310 in the bottom area of the container where we have a dark background. The first algorithm also detects a false particle at the same location, but this happens at 44 frames instead of 87 frames. The explanation relates to a different way of filtering valid tracks in the CTT algorithm.

7.2.2 Algorithm Results

In the above description, the performance of the system is evaluated on a container-level – regarding each container as accepted/rejected based on all of its particles. As for the first algorithm, the

evaluation can also be done on a particle-level. However, since CTT is only influencing the tracking and positioning, the evaluation of blob detections is not repeated. Therefore in the following, the algorithm including CTT is evaluated based on particle detection and particle positioning and classification.

7.2.2.1 Particle Detection

In figure 7.11 the particle detection rate (recall) and the precision of the particle detection are illustrated as a function of the number of frames used.

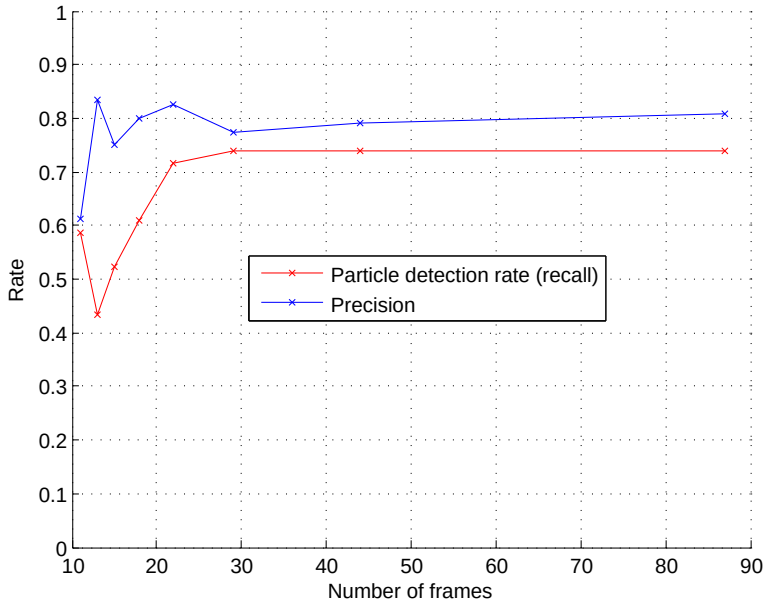


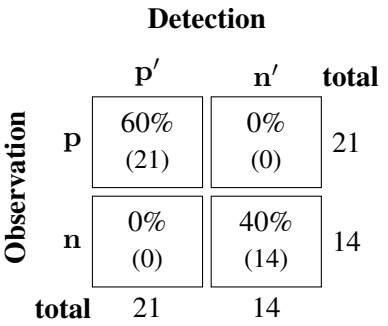
Figure 7.11: Particle detection rate (recall) and precision of CTT. The rates are plotted as a function of the number of frames used.

Between 22 and 87 frames the recall is relatively constant at around 74% as for the first tracking algorithm. Below this, the recall drops off because a minimum of 3 consecutive matches is required and because the threshold setting is set to at least 60% detected blobs required in order for a track to be valid. However, below 29 frames the precision is generally better for CTT than the first tracking algorithm, meaning that the number of false detections caused by noise measurements is lowered. The penalty is a drop in the particle detection rate. By adjusting the threshold setting it is possible to change this picture. A higher value would imply a higher recall but a decrease in precision.

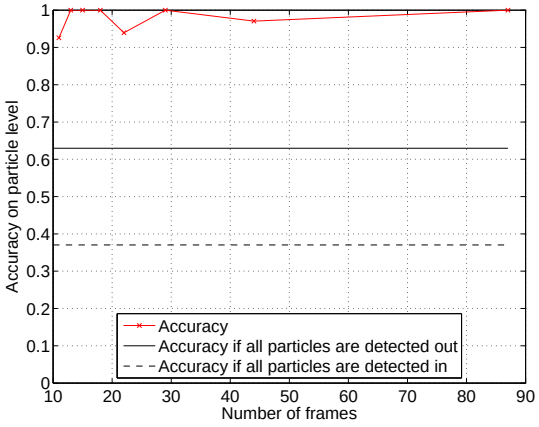
7.2.2.2 Particle Positioning and Classification

In figure 7.12a a confusion matrix for 87 frames on particle-level is constructed using CTT. As for the first algorithm, 100% of the particles are classified correctly based on their estimated radii. 21 particles are classified correctly as being outside the containers and 14 are classified correctly as being inside.

In figure 7.12b the accuracy of the positioning and classification algorithms is plotted as a function of how many frames are used. The graph is similar to the result of the first algorithm, but a higher accuracy is observed below 22 frames. The decision boundary offset is set to -10.2 based on the training set, that is, a smaller value compared to -8.9 for the first algorithm (see chapter 4.5). It indicates that we have a larger gap between radius estimates of particles inside and outside the



(a) Confusion matrix for positioning and classification on particle-level for 87 frames. Numbers in brackets specify the actual particle counts.



(b) Accuracy of positioning and classification plotted as a function of the number of frames used in the algorithm.

Figure 7.12: Confusion matrix and accuracy on particle-level for CTT.

containers, and it could explain a higher accuracy below 22 frames. The phenomenon is probably related to the improved CTT algorithm that now only selects the best fitting tracks.

7.2.2.3 Timing

Figure 7.13 illustrates the timing results where the processing time of the background extraction is optimized by a data-parallel realization in CUDA, and where the CTT algorithm is used for tracking and positioning. We observe the CTT algorithm being the critical part of the total execution time. From 44 frames to 87 frames a drop in time is observed due to specific settings of the CTT algorithm. In chapter 6.1.6 we found that kernel 4 was the most time consuming part of the algorithm.

For 87 frames tracks shorter than a sequence of 4 particle blobs are filtered away before kernel 4 is executed. Since this filtering is performed before kernel 4 a better timing performance is observed for 87 frames. Kernel 4 now uses less time in filtering tracks with duplicate particle blobs due to noisy short tracks. Requiring a sequence of at least 4 blobs for 22, 29 and 44 frames results in a higher performance and precision. Here the precision increases with an absolute value of 10% and the total execution time falls below the camera acquisition rate of 20 fps. For more details see appendix J section J.2. This improvement is not shown in the presented results of the CTT algorithm and further investigation of parameter optimization is left for future work.

With a camera frame rate of 20 fps we will be able to execute the algorithm in parallel with the circular trajectory tracker in MATLAB. Even with the redesigned data-parallel kernels, using 'for'/'parfor' loops in MATLAB, we have a timing performance similar to the camera acquisition rate of 20 fps.

The goal is to use a Basler camera with an acquisition rate of 120 fps. Here we see that the CTT algorithm as well as the segmentation are the limiting parts for a pipe-lined realization. If the algorithms were executed in parallel we still need to reduce the execution times for these algorithms. This is expected to be possible with a data-parallel realization in CUDA. Especially the background subtraction and subsequent morphological filtering operations should be possible to speed up efficiently. The performance of a GPU implementation of the CTT algorithm is expected to be independent of the number of particle blobs and noise measurements.

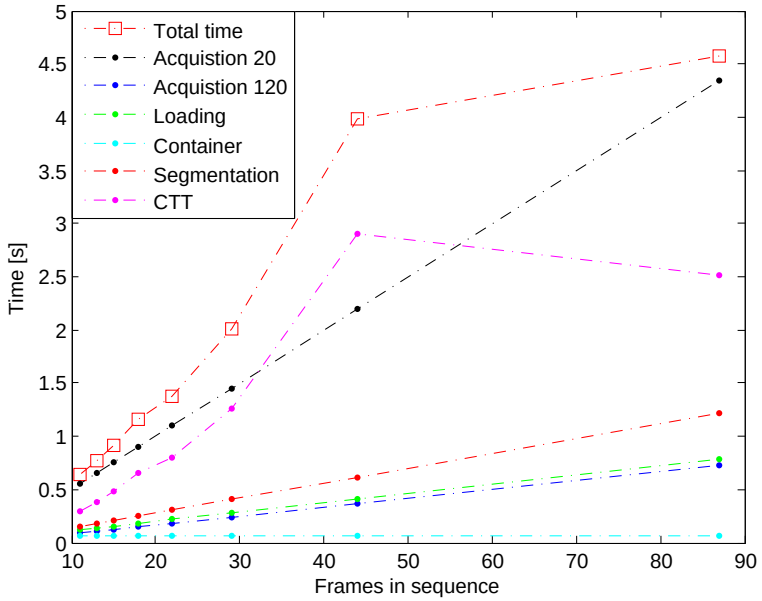


Figure 7.13: Average execution times of processing one glass container with optimized background extraction and the CTT algorithm. Acquisition indicates the recording time with a Basler camera of the type acA1600-20gc (20fps) or acA1000-120km (120fps). "Container" includes time for background extraction, diameter and center measurement.

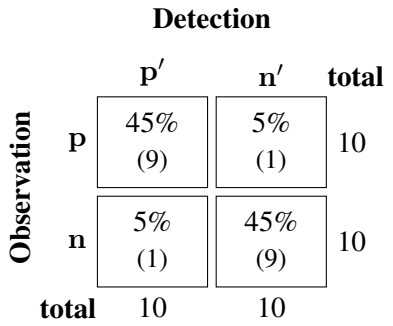
7.3 Third Algorithm – Multiple Hypothesis Tracking

As the second algorithm, the third algorithm also uses the exact same principles as the first algorithm. However, the 'Tracking' step is replaced with a probabilistic multi-target tracker called Multiple Hypothesis Tracking (MHT). In the literature MHT is one of the preferred methods for solving advanced multi-target tracking problems. In our problem it is applied as a top-down approach, and the theory behind as well as the adjustments of the method are described in chapter 6.2.

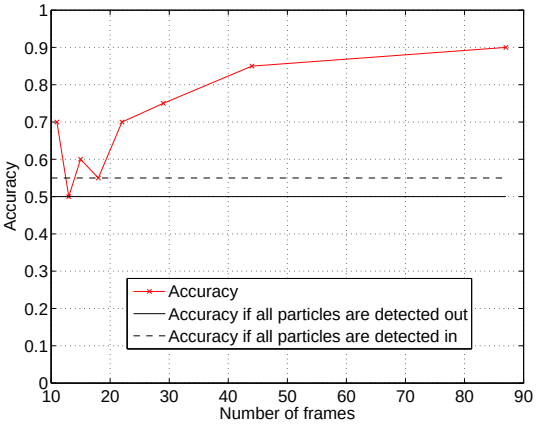
7.3.1 System Results

The confusion matrix in figure 7.14a shows the results of the evaluation of the test set. 9 containers are accepted correctly and 9 containers are rejected correctly. However, one container is rejected incorrectly (FN) and one container is accepted despite actual particles inside (FP). As for the first algorithms these concern G309 and H9. However, the specific MHT implementation is non-deterministic and thus can give different results from run to run. Because F68 contains a considerable amount of noise (false detections) near the particle, MHT does not always associate the right measurements with the track and therefore looses the track. This is caused by the gate (validation region) being circular and the fact that no state space model is used to incorporate the physics of the particle movements.

In figure 7.14b the accuracy of the system is plotted as a function of how many frames are used. As expected the best accuracy is obtained at 87 frames. However, the accuracy drops off already at 44 frames, whereas at the first tracking algorithm it drops off at 29 frames. This is partly due to the circular gate and the lack of a motion model resulting in wrong data associations and partly due to the lost measurements before a track is confirmed as described in chapter 6.2.4.2.



(a) Confusion matrix for the test set. Numbers in brackets specify the actual container counts.



(b) Accuracy plotted as a function of the number of frames used in the algorithm.

Figure 7.14: Confusion matrix and accuracy for MHT.

7.3.2 Algorithm Results

In the above description, the performance of the system is evaluated on a container-level – regarding each container as accepted/rejected based on all of its particles. As for the first algorithm, the evaluation can also be done on a particle-level. However, since MHT is only influencing the tracking and thus also the subsequent positioning, the evaluation of blob detections is not repeated. Therefore in the following, the algorithm including MHT is evaluated based on particle detection and particle positioning and classification. In contrary to the first and second algorithm, MHT is not evaluated on its timing performance since the primary interest lies in its accuracy and ability to handle noise and missing detections.

7.3.2.1 Particle Detection

In figure 7.15 the particle detection rate (recall) and the precision of the particle detection are illustrated as a function of the number of frames used.

The recall is relatively constant at around 74% as in the first tracking algorithm. This means that most particles are detected even though the number of frames is decreased from 87 to 11. However, the precision is generally much lower than the precision of the first tracking algorithm. Again this is caused by the circular gate and the lack of a motion model resulting in false particle detections from noise measurements. However, as for the first tracking algorithm some of the false detections are also caused by small or blurred particles that are not observed during hand labeling, even though they are actually present in the containers.

7.3.2.2 Particle Positioning and Classification

In figure 7.16a a confusion matrix for 87 frames on particle-level is constructed using MHT. As for the first algorithm, 100% of the particles are classified correctly based on their estimated radii. 21 particles are classified correctly as being outside the containers and 14 are classified correctly as being inside.

In figure 7.16b the accuracy of the positioning and classification algorithms is plotted as a function of how many frames are used. At 87 frames the accuracy is 100% corresponding to the results of the confusion matrix. Below this, the accuracy is somewhat lower because noise

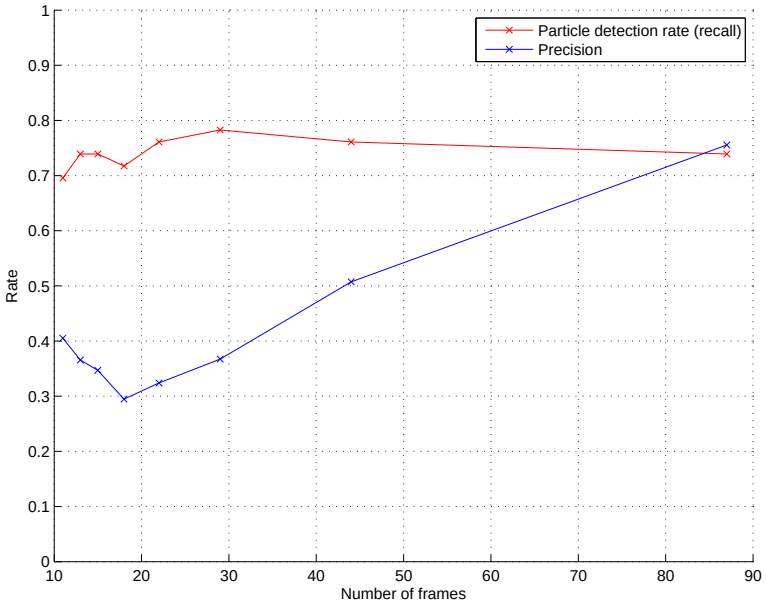
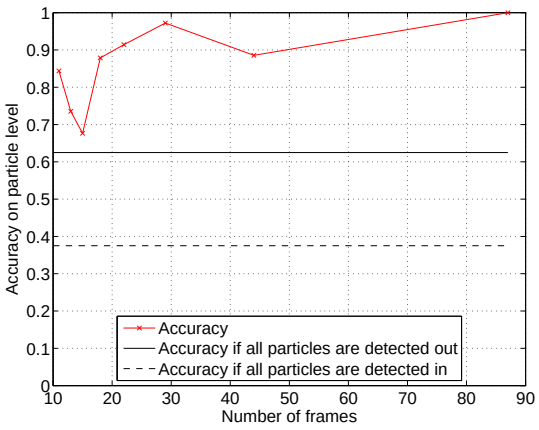


Figure 7.15: Particle detection rate (recall) and precision of MHT. The rates are plotted as a function of the number of frames used.

Observation \ Detection	p′	n′	total
p	62% (21)	0% (0)	21
n	0% (0)	38% (13)	13
total	21	13	

(a) Confusion matrix for positioning and classification on particle-level for 87 frames. Numbers in brackets specify the actual particle counts.



(b) Accuracy of positioning and classification plotted as a function of the number of frames used in the algorithm.

Figure 7.16: Confusion matrix and accuracy on particle-level for MHT.

measurements are more likely to be associated with the track when larger distances are present between particle measurements from frame to frame. An accuracy of 63% is achieved if all particles are classified as being outside the containers.

7.4 Comparison of Proposed Algorithms

The three algorithms, FA (First Algorithm), CTT (Circular Trajectory Tracker) and MHT (Multiple Hypothesis Tracking), have been evaluated above based on their performances on the test set. CTT and MHT are variations of the first algorithm where CTT replaces the tracking and positioning step, whereas MHT only replaces the tracking step. The particle segmentation step and the classification step are thus the same for the three algorithms.

Figure 7.17 summarizes the results by showing the accuracy of the three algorithms as a function of the number of frames. FA generally outperforms MHT. At 87 and 44 frames they perform equally well, but at 29 and 22 frames FA has a considerably higher accuracy. Below 87 frames CTT achieves better performance than both FA and MHT. The highest accuracy occurs at 44 frames because noise measurements in a single container are interpreted as a real particle at 87 frames.

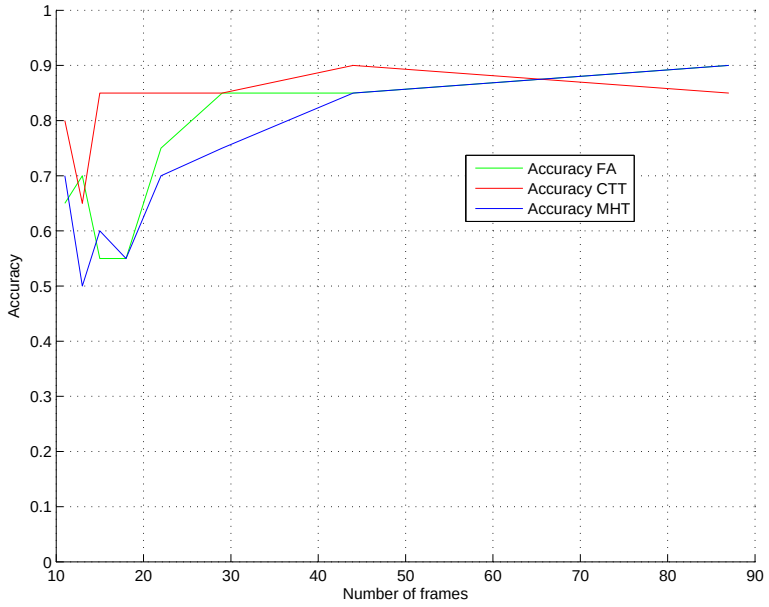
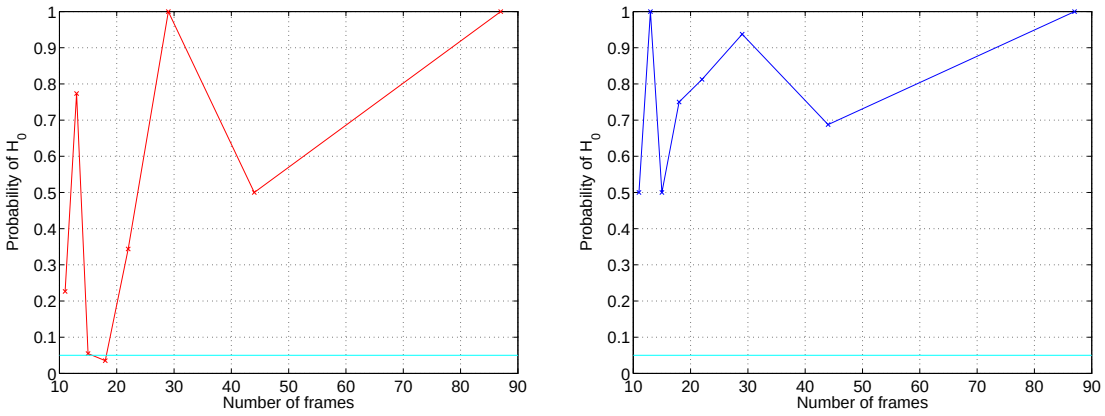


Figure 7.17: Accuracy plotted as a function of the number of frames used for all algorithms. FA denotes the first algorithm, CTT denotes the Circular Trajectory Tracker and MHT denotes Multiple Hypothesis Tracking.

Due to the small test set, a single tracking error in a container can contribute with a 5% decrease in the accuracy because the test set consists of only 20 containers. Therefore, statistically a variation in the accuracy between two algorithms might not necessarily mean that one is better than the other. In order to find the statistical foundation for determining this, hypothesis testing can be performed in which we define the null hypothesis H_0 to be that two algorithms are equal in their performances. By doing this we can calculate the probability of H_0 using McNemar's test to compare the paired results of the two algorithms. If this probability is smaller than 5%

we reject H_0 in favor of an alternative H_1 hypothesis stating that one algorithm performs better than the other. The theory and calculations behind the McNemar test are available in appendix I. Figure 7.18 illustrates the probabilities of the null hypotheses as a function of the number of frames. Figure 7.18a shows the comparison of FA and CTT, whereas figure 7.18b shows the comparison of FA and MHT. From the graphs there is no general statistical confidence to justify a difference between FA and respectively CTT and MHT. However, at 18 frames the alternative hypothesis H_1 (stating that CTT performs better than FA) is supported, since the probability of H_0 is less than the significance level of 0.05. Therefore, when using only 18 frames CTT is a preferred method to FA.



(a) McNemar test for FA and CTT.

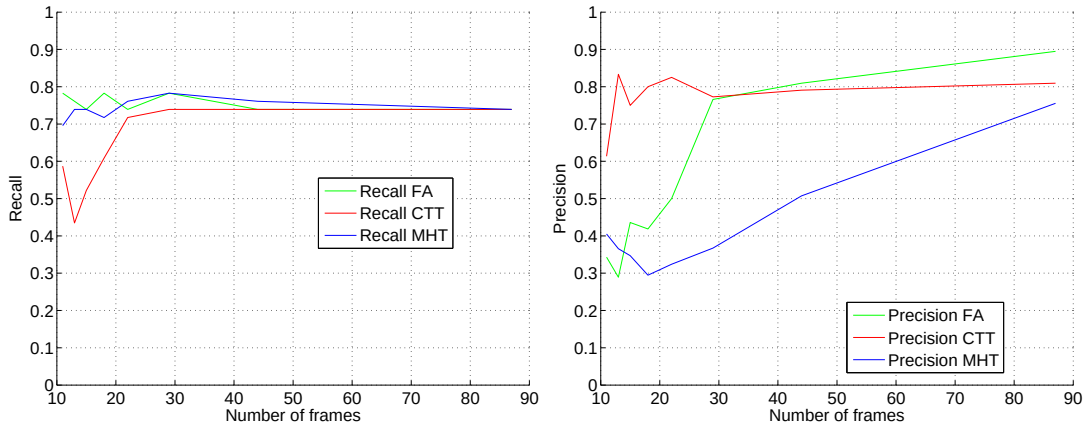
(b) McNemar test for FA and MHT.

Figure 7.18: McNemar test comparing the first algorithm to respectively CTT (a) and MHT (b). The probability of the null hypothesis is shown as a function of the number of frames. The green line indicates a significance level of 0.05.

Figure 7.19 shows the particle detection rate (recall) and the precision of the three algorithms as a function of the number of frames. The recall is generally the same until 22 frames, meaning that all algorithms find the same amount of real particles. Below this the recall of CTT falls off due to the settings of the algorithm and a minimum of 3 consecutive matches required in order to detect a particle.

The precision, however, differs significantly for the three algorithms. FA has a higher precision than both CTT and MHT when several frames are available. However, when only a few frames are available CTT experiences a better precision, rejecting more false particle detections. MHT has the lowest precision, meaning that it finds the most false particles. Since the data on particle-level is not paired, we can not perform the McNemar test as described above.

Based on the above description, deciding which algorithm performs the best is difficult from the limited amount of test data. However, since the complexity is rather low for FA while being considerably higher for both CTT and MHT, no clear motive for selecting an alternative algorithm to FA is present. Simulations show that in other circumstances when more noise and occlusions are present, CTT will generally outperform both FA and MHT. A more comprehensive test set might show the same, but based on the currently available data, a real-time implementation of the first algorithm is recommended for solving the problem of detecting and three-dimensionally positioning particles tied to containers.



(a) Recall. (b) Precision.

Figure 7.19: Particle detection rate (recall) and precision for all three algorithms. FA denotes the first algorithm, CTT denotes the Circular Trajectory Tracker and MHT denotes Multiple Hypothesis Tracking.

7.5 Summary

Based on the above evaluation of the proposed algorithms, the first algorithm is recommended for solving the problem of detecting and three-dimensionally positioning particles tied to containers. Multiple Hypothesis Tracking (MHT) and the Circular Trajectory Tracker (CTT) each possess advantages when applied on simulated data, but due to the limited amount of data present in the test set, no clear motive for selecting an alternative algorithm to the first algorithm is present.

The proposed algorithm has an accuracy of 90% on a test set of 20 containers when using the maximum number of acquired frames. However, at this accuracy the acquisition time[1] is 4.4 seconds, whereas the execution time is close to 5 seconds for each container according to figure 7.9.

With statistical significance CTT performs better than the first algorithm using only 18 frames. At 18 frames CTT has an accuracy of 85%. Here the acquisition time is lowered to 0.9 seconds due to fewer frames, whereas the execution time is lowered to 1.16 seconds by using real-time optimization.

In order to meet the real-time requirements of inspecting 11 containers per second, however, a camera with a higher acquisition rate as well as a data-parallel optimized version of the first algorithm must be applied.

[1]The time it takes to record 87 frames

8

Future Work

In this chapter we will discuss three different areas important for the future work and for the actual implementation of the presented inspection method. Two areas are described covering the mechanical and machine vision setup which is important for the accuracy and performance of the final system. The algorithm will also be discussed with focus of extended verification, improvements and implementation.

8.1 Mechanical Setup

The CVT machine consists of a mechanical carousel that transports glass containers while they are rotating and being inspected. A mirror could be mounted in the center of the carousel to ensure a symmetrical horizontal tracking of the moving glass containers as illustrated in figure 8.1. This would especially be important for a long horizontal travel distance while recording many frames of rotating glass containers.

Another important issue is to ensure an *accurate sample rate* of the container as described in chapter 3.2.1.2. A high accuracy on the actual rotation speed and camera frame rate is required in order to obtain a high accuracy in the positioning of the particles. In the experimental setup $f_{cyl} = 1186\,\mathrm{rpm}$ and $f_s = 20\,\mathrm{fps}$ is used. Small variations and uncertainties in f_{cyl} or f_s can result in false particle radius estimations. Here an offset error of the f_{cyl} was measured to $2\,\mathrm{rpm}$, which is too high if not compensated for. The accuracy of the container rotation frequency can be improved by implementing a feedback loop (PID) in the electronic controller. This is not the case in the current version of the control software, although the needed encoder information from the motor revolution is available to implement such a feedback loop. Alternatively, this information could be passed to the tracking and positioning algorithms to compensate for the offset mathematically.

It is important for the container to rotate straight about its own center axis. If there is a large slack while the container is rotating, the impact will be false radius estimations for particles at the top and the bottom of the containers. From our experiments with the current setup this seems not to be a problem but has to be investigated further. A higher disturbance on the illumination at the borders of the container is also observed related to the slack.

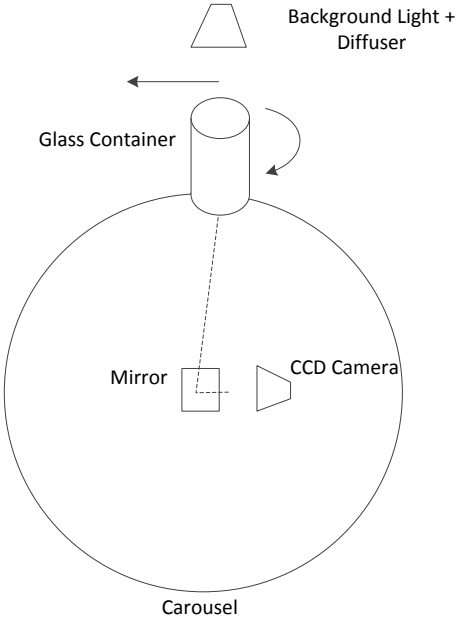


Figure 8.1: Camera and mirror placed in the center of the carousel to track the horizontal movement of the glass container.

8.2 Machine Vision Setup

Concerning the machine vision setup, illumination is very important and needs high attention. A higher intensity of the background light has to be investigated for improving the contrast between particles and background.

With the current experimental setup small particles of fiber material are difficult to detect. Here the method of using a dark field is proposed which can be produced by having collimated light vertical to the camera position. The hypothesis is that particles will then be observed as bright spots on a dark background, similar to the 'Camera with Laser Sheet' method suggested in appendix A section A.3.

Selection of a high speed camera needs to be investigated in order to meet the real-time requirements and keep a high resolution and contrast. As described in equation 3.3 if the camera sample rate f_s increases, the container rotation frequency f_{cyl} will also increase. From equation 3.2 this means that we need to decrease the exposure time in order not to introduce further blurring of the particle blobs. However, this will also require a higher intensity of the illumination. In this way, all the parameters are related.

8.3 Algorithm

In our final test results we found a large number of false particle detections in the bottom area of the segmented images. The reason is related to the robber stopper where particles are observed on a dark background as described in chapter 4.2.1.2. Here a solution would be to improve the segmentation algorithm.

After solving the above mentioned problem, verification on a larger data set is important. With the number of glass containers used in our experiments we only have a narrow statistical

foundation. An improved verification of the algorithm on a data set with more than 100 containers of the same type with different types of particles would be preferable.

The algorithm could also be improved in a number of areas as described in the following.

Segmentation. Here the general goal would be to improve the precision and detection rate of real particle blobs. The precision can be improved in the bottom area of the containers by also including blobs whose pixels are brighter than the background. Alternatively the crop area could be changed to include only the area above the robber stopper. The expected performance gain is illustrated in appendix J section J.1. Here noise blobs detected in the area of the robber stopper are ignored achieving a higher accuracy than presented in this thesis. A method to increase the detection rate could be to combine the background subtraction with an edge filter. Here the same principle as for the 'Global Segmentation' method described in chapter 4.2.1.1 could be used. A data-parallel implementation for parts of the segmentation algorithm should be realized to meet real-time performance requirements.

Tracking (CTT). Here an evaluation of the algorithm on a larger data set is recommended especially to find how parameters listed in chapter 6.1.6 should be adjusted to achieve an optimal detection rate, precision and accuracy (see appendix J section J.2). Another important step would be to implement a data-parallel version of the proposed four kernels to achieve the required real-time performance. Missing detections seem not to be the most important problem in our data-set, and thus the fourth kernel that finds the best fitting tracks could be omitted. Here we would expect a result comparable to the first tracking algorithm, except that it would be much faster according to figure 6.9.

An alternative design of the algorithm could be explored and verified on a more comprehensive data set. Here a cost function is suggested to minimize the change in particle movement and direction instead of using a cost function for change in particle area. This method has only gained minor improvements (see appendix J section J.2) with a much higher complexity than the presented CTT.

Tracking (MHT). The most significant improvement of the implementation of Multiple Hypothesis Tracking (MHT) would be to include a motion model handled by a Kalman filter. This could potentially catch missing particles as well as exclude a vast number of false detections. The state space model described in chapter 6.2.4.1 would result in an elliptical gate (validation region) when finding point correspondences, and in this way a particle moving along the y-axis would be excluded. Another improvement would be to handle the issue concerning tentative/confirmed tracks as described in chapter 6.2.4.2. The current implementation of MHT only outputs confirmed tracks, and thus some measurements associated with a tentative track are lost before the track is confirmed.

Positioning. The impact of refraction between liquid, glass and air has not been fully investigated. Particles inside the liquid of the glass containers are both subject to refraction and magnification having an impact on the estimated radii. A model should be created to estimate the error of the radius estimates due to these phenomena.

For large particles the blob center is skewed resulting in a smaller estimated radius. The results could be that a container is rejected despite its particles being on the outside of the glass. Either the system should reject a container with particles above a certain size or the algorithm needs to compensate for this phenomenon.

9

Conclusion

This thesis proposes a method of detecting and positioning particles inside or outside containers with liquid medicine based on a sequence of images. The method is applied on a practical problem experienced by the Danish industrial company InnoScan A/S. InnoScan develops high-speed inspection machines for rejecting medical containers containing foreign particles based on two-dimensional images. However, a distinction between particles residing inside or outside glass containers is not possible due to the lack of a third dimension, and thus some glass containers with harmless dirt on the outside are unnecessarily discarded.

Using our proposed algorithm, particles are detected, tracked and positioned based on a sequence of images acquired under a constant rotation speed of the cylindrical containers. A container is rejected if any particles are estimated as being inside, whereas particles estimated outside are tolerated. The proposed algorithm has an accuracy of 90% on a test set of 20 containers when acquiring 87 frames per container. At this accuracy the acquisition time is 4.4 seconds, whereas the average execution time per container is 5 seconds. Acquiring only 18 frames per container, an accuracy of 85% is achieved, whereas the acquisition time is lowered to 0.9 seconds and the execution time is lowered to 1.16 seconds.

From evaluating the first proposed algorithm, classification errors result from false and missing particle detections in the tracking algorithm. Therefore, two advanced tracking algorithms are investigated and applied to compensate for problems concerning false detections and occlusion.

The Circular Trajectory Tracker is an algorithm based on the first tracking algorithm. It is designed as a bottom-up approach for data-parallel computation on a GPU platform. It allows missing detections of particle blobs while filtering out noise detections that do not fit with a realistic circular particle trajectory. It outperforms the first algorithm evaluated on a simulated data set and shows significant improvement in relation to the first algorithm for 18 frames on the actual test set. It is expected to meet the real-time requirements for a future implementation in a parallel programming language like CUDA.

Multiple Hypothesis Tracking is a probabilistic method generally accepted as one of the most promising methods for complex multi-target tracking. An open-source implementation of this advanced algorithm has been tailored to fit our problem such that an unknown number of particles can be tracked in a sequence of images. The method handles both occlusion and noise but due to the lack of a Kalman filter in the specific implementation, a motion model filtering out unrealistic tracks is not included. Evaluated on a simulated data set the algorithm outperforms the first algorithm regarding noise and missing detections. On the actual test set however, it shows no

improvements in relation to the first algorithm.

In order to apply the proposed method in practice, future work must focus on real-time aspects of both the camera acquisition time and the algorithm execution time. A camera with a higher acquisition rate as well as an optimized data-parallel version of the first algorithm are needed. Additionally the mechanical setup must be improved since inaccuracies in the rotation speed leads directly to inaccuracies in the estimated particle positions. However, this thesis serves as a proof of concept illustrating that the proposed method is both feasible and accurate.

Bibliography

[1] N. Zeuch, *Understanding and Applying Machine Vision*. John Wiley & Sons, Inc., Second ed., 1988.

[2] InnoScan A/S, "http://www.innoscan.dk," 1988.

[3] R. E. W. Rafael C. Gonzalez, *Digital Image Processing*. Pearson, Third ed., 2008.

[4] C. Wöhler, *3D Computer Vision*. Springer, Second ed., 2013.

[5] M. T. Giovanna Sansoni and F. Docchio, "State-of-The-Art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation," *Sensors*, vol. 9, pp. 568–601, 2009.

[6] J.-A. B. Francois Blais, Marc Rioux, "Practical Considerations For A Design Of A High Precision 3-D Laser Scanner System," in *Optomechanical and Electro-Optical Design of Industrial Systems*, pp. 225–246, 1988.

[7] F. Chen, G. M. Brown, and M. Song, "Overview of three-dimensional shape measurement using optical methods," *Optical Engineering*, vol. 39, no. 1, pp. 10–22, 2000.

[8] M. T. Giovanna Sansoni and F. Docchio, "Fast 3D profilometer based upon the projection of a single fringe pattern and absolute calibration," *Meas. Sci. Thechnolo.*, vol. 17, pp. 1757–1766, 2006.

[9] B. Buntz, "The Benefits of Optical 3D Inspection," *European Medical Device Technology*, vol. 1, no. 5, 2010.

[10] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–195–I–202 vol.1, 2003.

[11] T. Nielsen, F. Bormann, S. Wolbeck, H. Spiecker, M. D. Burrows, and P. Andresen, "Time-of-flight analysis of light pulses with a temporal resolution of 100 ps," *Review of Scientific Instruments*, vol. 67, no. 5, pp. 1721–1724, 1996.

[12] C. W. Kia Hafezi, "A general framework for three-dimensional surface reconstruction by self-consistent fusion of shading and shadow features and its application to industrial quality inspection task," in *Optical Metrology in Production Engineering*, vol. 5457, 2004.

[13] B. Barrois and C. Wohler, "3D Pose Estimation Based on Multiple Monocular Cues," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, 2007.

[14] M. N. Shree K. Nayar, Masahiro Watanabe, "Real-Time Focus Range Sensor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 995–1001, 1996.

[15] M. Raffel, C. Willert, and J. Kompenhans, *Particle Image Velocimetry, A Practical Guide*. Springer, Second ed., 2007.

[16] C. M. Simon Bogh, Mads Hvilshoj and J. Stepping, "Machine Vision - teori og praksis," Tech. Rep. 3.123, Aalborg University, Industry and Production, May 2007.

[17] E. R. Davies, *Computer and Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, Elsevier, 2012.

[18] F. W. Bruce Bachelor, *Intelligent Machine Vision: Techniques, Implementations and Applications*. Springer London, 2001.

[19] PCO, "http://www.pco.de/categories/high-speed-cameras/pcodimax-hs/," 2013.

[20] E. Ifeachor and B. Jervis, *Digital Signal Processing: A Practical Approach*. Electronic systems engineering series, Prentice Hall, 2nd ed., 2002.

[21] A. Hornberg, *Handbook of Machine Vision*. Wiley-VCH, 2006.

[22] B. Batchelor, *Machine Vision Handbook*. Springer, 2012.

[23] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006.

[24] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 1st ed., 2011.

[25] J. P. David A. Forsyth, *Computer Vision: A Modern Approach*. Prentice Hall, Second ed., 2011.

[26] P. Shirley and S. Marschner, *Fundamentals of Computer Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 3rd ed., 2009.

[27] sysML, "SysML, http://www.sysml.org/," 2008.

[28] R. Duda and P.E.Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[29] R. S. Sandford Friedenthal, Alan Moore, *A Prictical Guide to SysML*. Friendenthal, Sanford: Morgan Kaufman OMG Press, First ed., 2008. ISBN 978-0-12-374379-4.

[30] M. D. More and G.K.Andurkar, "Edge detetection techniques: a comparative approach," *World Journal of Science and Technology*, vol. 2, no. 4, pp. 142–154, 2012.

[31] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with application to image analysis and automated catography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[32] Dan Sunday, "http://geomalgorithms.com/a07-_distance.html," 2012.

[33] Peter Kovesi, "MATLAB and Octave Functions for Computer Vision and Image Processing," 2013.

[34] Z. Zhang, "Token tracking in a cluttered scene.," *Image Vision Comput.*, vol. 12, no. 2, pp. 110–120, 1994.

[35] I. J. Cox, "A Review of Statistical Data Association Techniques for Motion Correspondence," *International Journal of Computer Vision*, vol. 10, pp. 53–66, 1993.

[36] R. Zabih and J. Woodfill, "Non-parametric Local Transform for Computer Visual Correspondence," in *Proceedings of European Conference on Computer Vision, Stockholm*, pp. 151–158, Springer, 1994.

[37] X. Mei, X. Sun, M. Zhou, shaohui Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 467–474, 2011.

[38] E. Chong and S. Zak, *An Introduction to Optimization*. Wiley Series in Discrete Mathematics and Optimization, Wiley, 2011.

[39] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[40] NVIDIA, "CUDA C Programming Guide," Tech. Rep. PG-02829-001-v5.5, NVIDIA Corporation, July 2013.

[41] M. Inc., "Parallel Computing Toolbox User's Guide R2013b," tech. rep., The MathWorks Inc., 3 Apple Hill Drive Natrick, MA 01760-2098, 2013.

[42] NVIDIA, "CUDA C Best Practices Guide," Tech. Rep. DG-05603-001-v5.5, NVIDIA Corporation, July 2013.

[43] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer, 2nd ed., 2011.

[44] M. R. M. Mark D. Hill, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[45] W.-m. W. H. David B. Kirk, *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, First ed., 2010.

[46] O. J. Alper Yilmaz and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, 2006.

[47] J. Munkres, "Algorithms for the assignment and transportation problems," *SIAM*, vol. 5, no. 1, pp. 32–38, 1957.

[48] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 19, no. 1, pp. 5–18, 2004.

[49] C. Veenman, M. Reinders, and E. Backer, "Establishing motion correspondence using extended temporal scope," *Artificial Intelligence*, vol. 145, no. 1–2, pp. 227 – 243, 2003.

[50] K. Shafique and M. Shah, "A quantitative study of three-dimensional Lagrangian particle tracking algorithms," *Experiments in Fluids*, vol. 40, no. 2, pp. 301–303, 2006.

[51] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 1, pp. 51–65, 2005.

[52] A. M. Matthieu Garrigues, *Real Time Semi-dense Point Tracking*. Springer Berling Heidelberg, 2012.

[53] D. B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control*, vol. 24, pp. 843–854, 1979.

[54] J. Vermaak, S. Godsill, and P. Perez, "Monte Carlo filtering for multi target tracking and data association," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 41, no. 1, pp. 309–332, 2005.

[55] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *Control Systems, IEEE*, vol. 29, no. 6, pp. 82–100, 2009.

[56] P. Smith and G. Buechler, "A branching algorithm for discriminating and tracking multiple objects," *Automatic Control, IEEE Transactions on*, vol. 20, no. 1, pp. 101–104, 1975.

[57] C. Morefield, "Application of 0-1 integer programming to multitarget tracking problems," *Automatic Control, IEEE Transactions on*, vol. 22, no. 3, pp. 302–312, 1977.

[58] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1, pp. 2 – 15, 2007.

[59] S. Oh, S. Russell, and S. Sastry, "Markov Chain Monte Carlo Data Association for Multi-Target Tracking," *Automatic Control, IEEE Transactions on*, vol. 54, no. 3, pp. 481–497, 2009.

[60] R. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 39, no. 4, pp. 1152–1178, 2003.

[61] B.-N. Vo and W.-K. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4091–4104, 2006.

[62] D. E. Clark, *Multiple Target Tracking with The Probability Hypothesis Density Filter*. PhD thesis, HERIOT-WATT UNIVERSITY, 2006.

[63] D. M. Antunes, D. M. de Matos, and J. A. Gaspar, "A Library for Implementing the Multiple Hypothesis Tracking Algorithm," *CoRR*, vol. abs/1106.2263, 2011.

[64] I. Cox and S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 2, pp. 138–150, 1996.

[65] D. Geier, "Development and Evaluation of a Real-time Capable Multiple Hypothesis Tracker," Master's thesis, Technische Universität Berlin, 2012.

[66] R. Scheaffer, M. Mulekar, and J. McClave, *Probability and Statistics for Engineers*. Brooks/Cole, Cengage Learning, 2010.

[67] R. Danchick and G. E. Newnam, "Reformulating Reid's MHT method with generalised Murty K-best ranked linear assignment algorithm," *Radar, Sonar and Navigation, IEE Proceedings* -, vol. 153, no. 1, pp. 13–22, 2006.

[68] A. Amditis *et al.*, "Multiple Hypothesis Tracking Implementation," in *Laser Scanner Technology*, pp. 1–10, InTech, 2012.

[69] T. Kurien, "Issues in the design of practical multitarget tracking algorithms. Multitarget-Multisensor Tracking: Advanced Applications 1:43–83," 1990.

[70] S. Coraluppi and C. Carthel, "Modified Scoring in Multiple-Hypothesis Tracking," *Journal of Advances in Information Fusion*, vol. 7, no. 2, 2012.

[71] S. Coraluppi and C. Carthel, "Addressing the greedy-target problem in multiple-hypothesis tracking," in *Aerospace Conference, 2011 IEEE*, pp. 1–10, 2011.

[72] K. G. Murty, "An Algorithm for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, vol. 16, no. 3, pp. 682–687, 1968.

[73] D. M. Antunes, "Multi-sensor based Localization and Tracking for Intelligent Environments," Master's thesis, Instituto Superior Técnico, 2011.

[74] D. M. Antunes, "MHL API Javadoc." `http://www.multiplehypothesis.com/javadoc-mhl/`, 2011. Accessed: 2013-11-25.

[75] C. H. . M. Stephens, "A Combined Corner and Edge Detector," in *Proceedings of th 4th Alvey Vision Conference*, pp. 147–151, BMVA, 1988.

[76] Z. Z. Charles Loop, "Computing Rectifying Homographies for Stereo Vision," Tech. Rep. MSR-TR-99-21, Microsoft Research, research.microsoft.com, May 1999.

[77] R. Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp. 323–344, 1987.

[78] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," 2010.

[79] T. Yoshizawa, *Handbook of Optical Metrology - Principles and Applications*. CSC Press, 2009.

[80] R. Popp, K. Pattipati, and Y. Bar-Shalom, "m-best S-D assignment algorithm with application to multitarget tracking," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 37, no. 1, pp. 22–39, 2001.

[81] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 4 ed., 2007.

Appendices

Appendix A - Pre-project

This appendix contains a description of the methods and experiments performed in a pre-project with the purpose of selecting a method for the thesis.

We have created a number of innovative ideas that we have evaluated in a pre-study project with the purpose of deciding on methods relevant for investigation and basis for the thesis. We have made a test setup with a mechanical supporting device, laser and CCD cameras. The mechanical supporting device is attached to a motor with encoders and electronic control. This electronic control is connected to a CAN bus from where a plan can be downloaded that tells how to rotate the glass container. Figure A.1 shows a stereo setup with a glass container (cartridge) for inspection. The cartridge is mounted in the mechanical rotation device supported by a cup and a rotation tracking mark. Due to limited research facilities only the CCD Basler cameras are used with a resolution of 1624x1234 pixels (20 Hz) and 658x492 pixels (100 Hz). The cameras are connected to a Windows PC with a high speed Ethernet connection with frame grabbers and software for camera adjustment and image and video recording.

Initial experiments are described and discussed concerning the four methods listed below. **Video Sequence of Rotation** is the method we have selected as the most promising for the thesis.

1) **Line Scan Cameras**

2) **Stereo Vision**

3) **Camera with Laser Sheet**

4) **Video Sequence of Rotation**

A.1 Line Scan Cameras

This method is based on the camera system that InnoScan are currently using in their particle inspection stations. The idea is to perform recordings of the glass container surface from different angles. The mirror controlled by the inspection station should be moved scanning the glass container surface at scan lines L1 and L2 as illustrated in figure A.2. The method uses the existing inspection stations by unfolding the glass container surface from two different angles while the glass container is rotating. The method can either be realized with two stations, where the mirrors

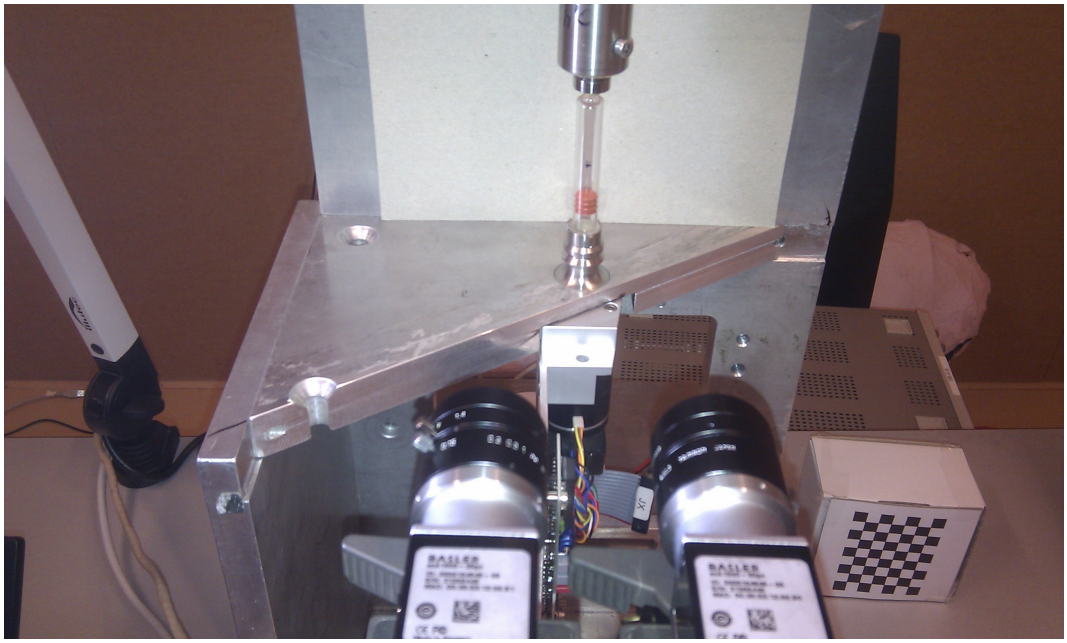


Figure A.1: Test setup with mechanical supporting device, electronic control, glass container (cartridge) and two Basler CCD cameras.

are viewing the glass container from different angles, or at the same station, where the mirror angle is changed after scanning one revolution of the glass surface. We assume that using the same station would achieve the best result since in this case we do not need to take the different camera calibration parameters into account and since the movement of the particles will be minimal.

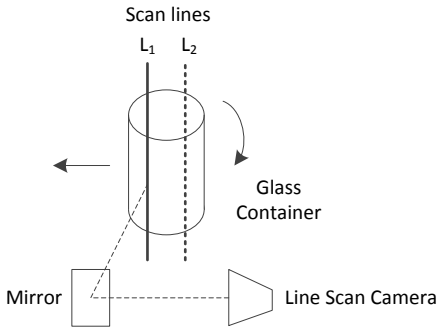


Figure A.2: Method based on mirror and line scan camera.

Having two images viewed from different angles it should be possible to use principles from stereo vision [24]. Here we would use sparse stereo correspondence to find matching particles in the two images. By knowing the geometry, given the angle of the mirror and the camera's intrinsic and extrinsic parameters, the hypothesis is to compute the distance of similar particles found in the two images and thereby determine the three-dimensional particle coordinates. In figure A.3 we have used images from two different stations where the same particles can be seen different places in the image. In this case the two images are recorded where the particle is moving in the liquid of the glass container between the two particle stations. Note that in this case the mirror angle is the same for the two particle stations due to a special recording procedure. We have used a Harris corner detection [75] and a matching algorithm to find corresponding points in the two images.

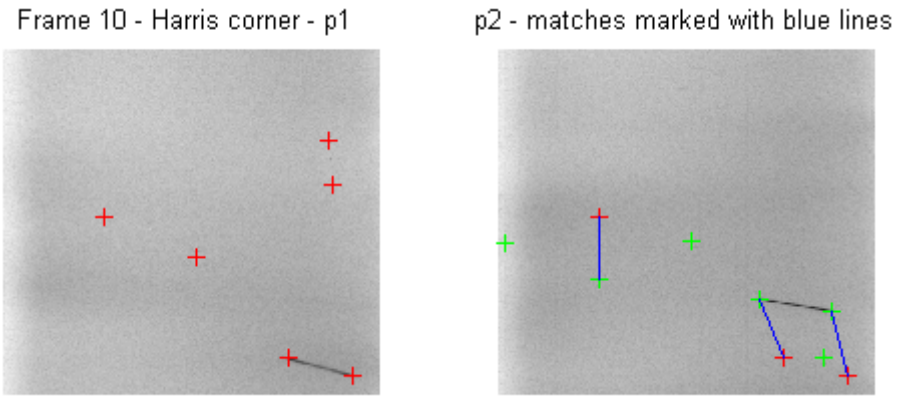


Figure A.3: Particle matching from two particle stations (p1, p2).

The method seems promising but we have chosen not to continue with this idea especially because we are unsure if the accuracy will be good enough. It would also be difficult to realize with the current test setup in our research laboratory. However, InnoScan has a similar test setup with a line scan camera and adjustable mirror that could be used to investigate this method further.

A.2 Stereo Vision

The pinhole model [25, 4] can be used for reconstruction of the three-dimensional structure of a scene from several images. In stereo vision a left and right image are used to find the depth information. The basic idea is to use geometry to find the transformation from the scene point in the 3D world coordinate system to the image plane. Triangulation is used based on the 3D points projection to the left and right image planes. This technique requires the camera pose found by calibration and searches for corresponding points in the two images. Assuming that we have parallel optical axes (the left and right images have been rectified [76]) ensures that the image scan lines are the epipolar lines, and we thus know the camera parameters. Along the scan lines in each image a search for corresponding points is made and the disparity between them is computed. In order to reconstruct the 3D scene from disparity values the fixed distance between the cameras called baseline (B) and focal length (f) of the cameras has to be known. The stereo vision coordinate system lies between both cameras on the baseline as in figure A.4. The x-axis is parallel with the image plane. In computer vision image origins usually lie at the top left corner of an image. Here, the image coordinate systems have their origins translated to the very middle of each image, the closets point to the focal point.

We can compute the depth (Z) of similar points along the left and right image scan lines given that they are epipolar lines (after rectification).

$$Z = f\frac{B}{x_r - x_l} \tag{A.1}$$

Intrinsic and extrinsic camera parameters can be found by camera calibration as described by Tsai [77]. The extrinsic parameters consist of a rotation matrix and translation vector describing how to map a point from the image plane to the 3D world coordinate system. The intrinsic parameters consist of focal length, pixel size, image center point (principal point) and radial distortion. The parameters can be formed in the intrinsic matrix and describe how to map from the image plane to the pixel coordinates in the image.

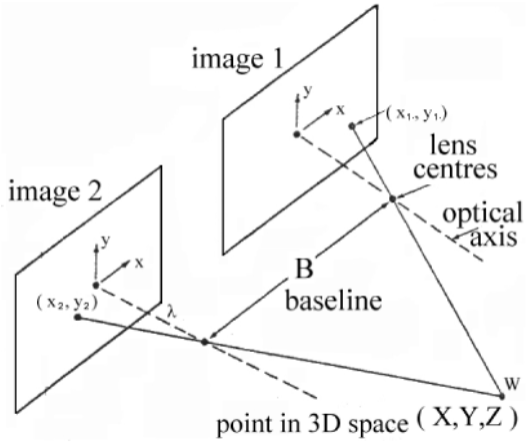


Figure A.4: Stereo image pinhole model.

Based on a set of stereo images a correspondence disparity map can be computed. In dense stereo vision algorithms matches are found for all points in the stereo images, whereas in sparse matching algorithms matches are found only for a selected number of feature points. From a disparity map and the camera matrix parameters the 3D coordinates of the pixels can be found. In sparse stereo vision the algorithm is optimized by only using a selected number of matching points as input to the 3D reconstruction.

A.2.1 Calibration and Experiments

In this method we will use two CCD cameras as illustrated in figure A.5. The cameras are calibrated to find the intrinsic and extrinsic camera matrices. Here we have used the camera calibration toolbox made by Bouguet [78].

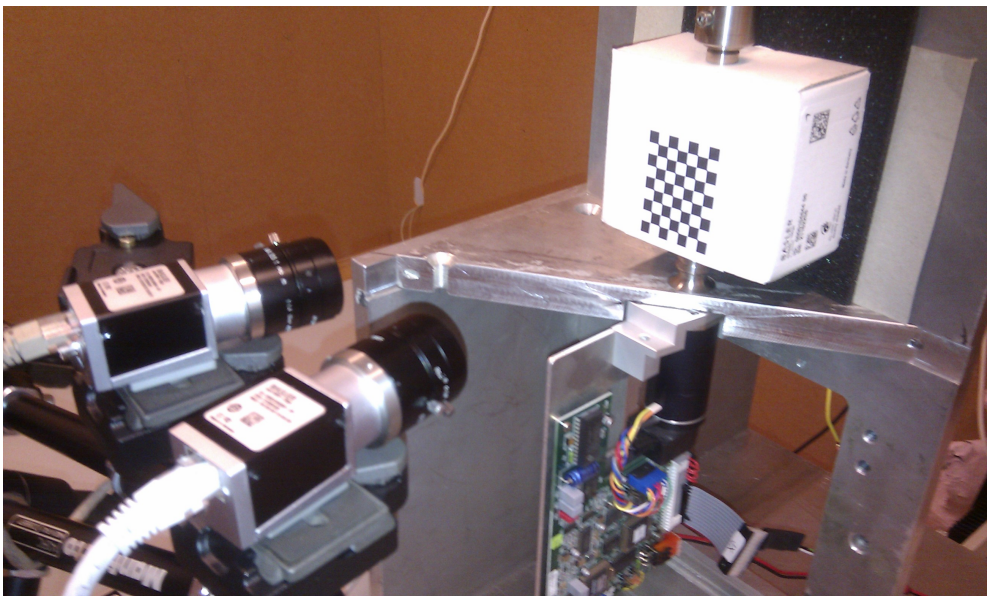


Figure A.5: Stereo Vision setup based on Basler CCD cameras.

A planar checkerboard is created with black and white squares with square dimensions of 6x6 mm. A number of 12 stereo calibration images were created by recording the checkerboard

in different positions and angles relative to the stereo cameras. The calibration toolbox is able to compute calibration data by extraction of corners from the checkerboard. In figure A.6 the reconstructed scene of the stereo setup based on images of the checkerboard in different positions is shown.

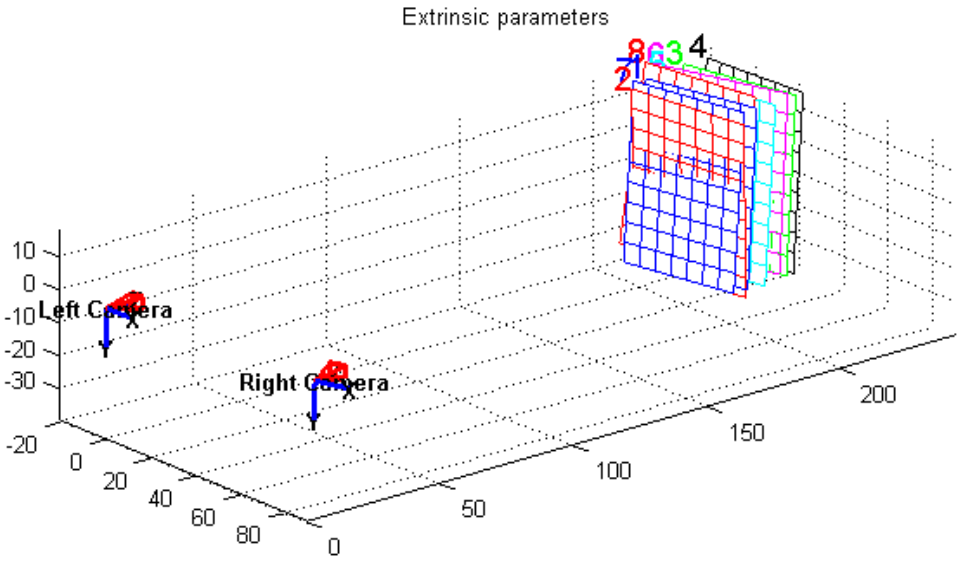


Figure A.6: Reconstruction of camera stereo scene based on extrinsic calibration matrix.

The toolbox computes a number of intrinsic parameters for the left and right camera covering: focal length, principal point and radial distortion with uncertainties. The extrinsic parameters are estimated given as a rotation vector (om) and translation vector characterizing the relative location of the right camera with respect to the left camera. Normally we will compute the left and right camera reference frames relative to each other through a rigid motion transformation. Points (X) in one image can be transformed by a 3D rigid transformation given a rotation matrix (R) and the translation vector (T):

$$X_r = R \cdot X_l + T \tag{A.2}$$

where R is a 3x3 rotation matrix corresponding to the rotation vector om. The relation between om and R is given by rodrigues rotation formula. Many wide-angle lenses have noticeable radial distortion [24] which the toolbox computes. The radial distortion model says that coordinates in the observed images are displaced towards the image center by an amount proportional to their radial distances. The radial distortion model consists of low-order polynomials:

$$\hat{x}_c = x_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4)$$
$$\hat{y}_c = y_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4)$$

where $r_c^2 = x_c^2 + y_c^2$ and κ_1 and κ_2 are called the radial distortion parameters. The following parameters are listed as a result of the stereo calibration:

Intrinsic parameters of the left camera (pixels):

Focal Length: $f_l = [3750.1, 3733.6] \pm [71.7, 71.4]$
Principal point: $c_l = [1035.5, 588.2] \pm [55.9, 34.3]$
Distortion: $\kappa_l = [0.31864, -6.36726, 0.00179, 0.01851, 0.00000]$

Intrinsic parameters of the right camera (pixels):

Focal Length: $f_r = [3716.1, 3700.0] \pm [72.2, 71.7]$
Principal point: $c_r = [642.3, 632.3] \pm [57.5, 39.5]$
Distortion: $\kappa_r = [0.19291, -3.11863, 0.00491, -0.01884, 0.00000]$

Extrinsic parameters (position of the right camera wrt the left camera):

Rotation vector: $om = [-0.00785, 0.40085, -0.00639] \pm [0.01212, 0.01998, 0.00277]$
Translation vector: $T = [-79.30026, -1.54489, 15.29275] \pm [0.68751, 0.26682, 2.64612]$

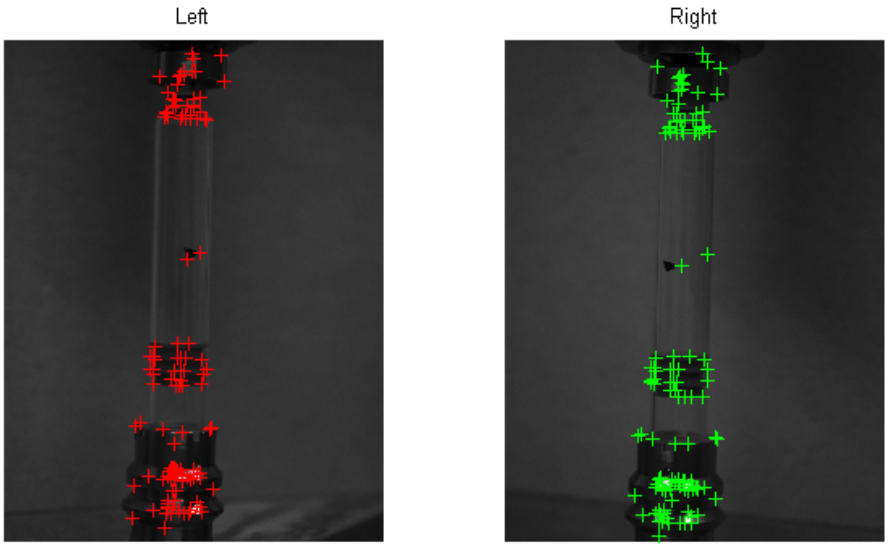


Figure A.7: Left and right stereo images of cartridge using Harris edge detection and sparse correspondence.

After calibration of the stereo camera we have recorded a number of stereo images and used Harris edge detector [75] to find matching points as input to a sparse feature based stereo 3D reconstruction. Figure A.7 shows a rectified stereo image pair of a cartridge where a number of selected points have been used to reconstruct the 3D image as illustrated in figure A.8. The points form a cloud of 3D points visualizing the 3D form of the cartridge for inspection. The object for inspection has been illuminated with a front light and a black background.

In dense stereo correspondence a number of methods exist for matching pixels found in the epipolar lines of the rectified stereo images. We have made experiments with a number of these methods in the category of block matching algorithms. They should be computational efficient and thus favorable for use in real-time vision systems. Here we have found the best result using the Zero Mean Normalized Cross Correlation (ZNCC) or Census transform [36]. A non-optimized 'C' implementation of the Census transform takes more than five minutes to compute for a stereo image pair of 1624x1234 pixels. A faster computation can be achieved by downscaling the images before correspondence matching. There is still a long way to the timing requirement of maximum 90 ms of the CVT machine. The resulting disparity map is illustrated in figure A.9 for the stereo image of the cartridge.

Our experiments with different stereo 3D algorithms seems promising for generating a 3D image of an object like the cartridge in our example. For finding particles on the surface or inside the cartridge this method is less promising since the distances are down to 100 μm. More work needs to be done finding a better solution for small particles inside or outside the glass container using principles of stereo vision.

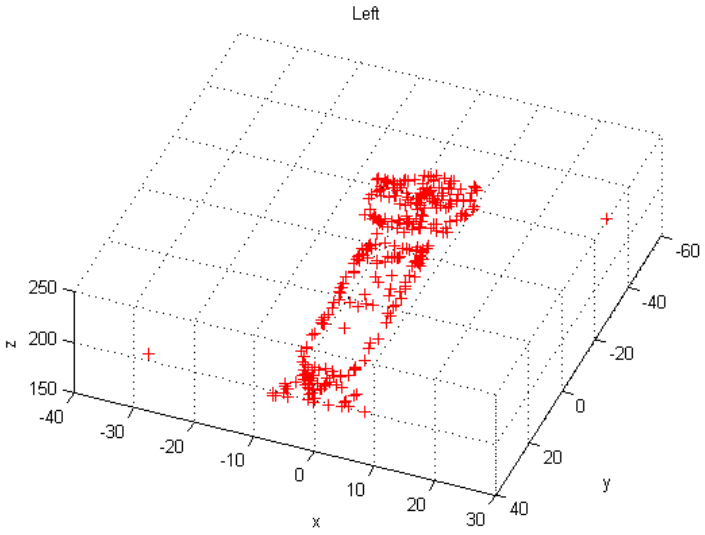


Figure A.8: 3D reconstruction of a cartridge based on sparse correspondence and the calibration matrix.

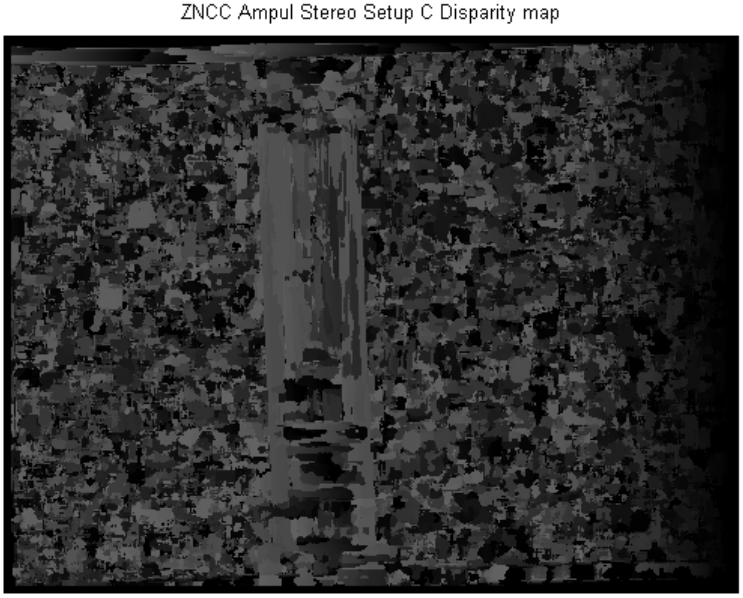


Figure A.9: Disparity map of cartridge based on dense correspondence using ZNCC block matching.

A.3 Camera with Laser Sheet

In this method we would use a full-frame CCD based digital camera and a laser sheet of light. The laser sheet is projected from an angle of 90 degrees relative to the camera position. While the glass container is rotating the camera records particles that are passing a cross section of the circular glass container with liquid. The laser illuminates the container and liquid with a narrow sheet of light. The setup is made in a dark room ensuring that only particles passing the laser sheet are reflecting the light. By adjusting the camera exposure time particles passing the laser sheet will be visible. The setup is illustrated in figure A.10.

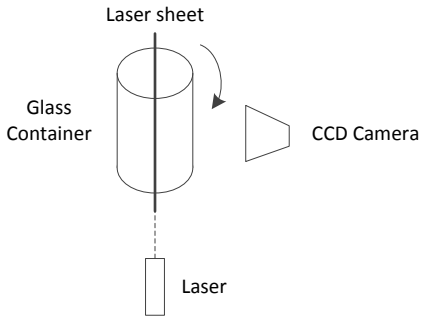


Figure A.10: Method based on laser sheet with CCD camera.

Experiments have been performed using the test setup with the Basler 20 and 100 Hz cameras. Figure A.11 illustrates how an empty cartridge glass container with a piece of adhesive tape on the surface is illuminated by the laser sheet. The Basler 20 Hz CCD camera is set to an exposure time of 2 sec with a resolution of 658x492 pixels. A number of experiments have been performed with cartridges containing insulin (actrapid) and a number of different types of particles.

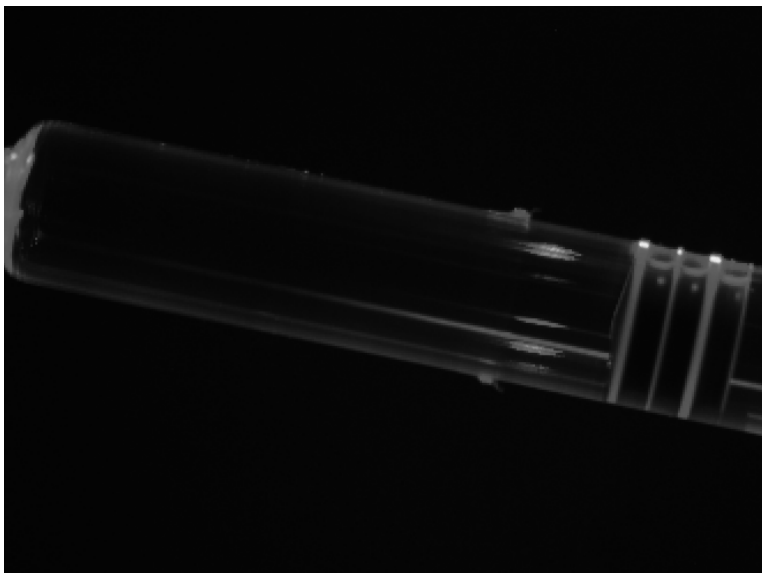


Figure A.11: Cartridge image rotating 1200 rpm with an exposure time of 2 sec.

This method has a number of drawbacks. A completely dark room will be difficult to create in the industrial setup of the machinery that InnoScan is manufacturing. Experiments have shown that cartridges with liquids are creating a diffusion of the laser light so that particles inside the

cartridge are difficult to see in the image. However, it seems possible to find particles on the outside of the cartridge. An optimal exposure time depends on the rotation speed. The best result is achieved when it is longer than the time of two revolutions. Image processing like thresholding and line search of the cartridge contour would be the method to estimate particles on the surface.

A.4 Video Sequence of Rotation

In this method a sequence of images are recorded where the camera frame rate is synchronized with the rotation of the glass container. In figure A.12 the setup is illustrated for the position of the camera, mirror and illumination of the glass container. In our test setup the mirror and the horizontally moving glass container is omitted. The method concerns recording a sequence of images where each image is recorded for every one revolution of the glass cylinder.

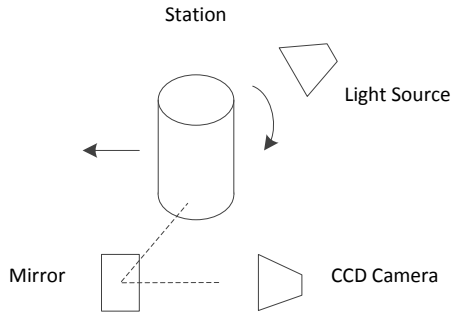


Figure A.12: Method based on CCD camera with image sequence of rotation.

Every time the cylinder has rotated one revolution a new frame is sampled after a short delay. Adding this short delay will ensure that the next frame is taken at a new position of the glass cylinder surface. This method makes it possible to record a video sequence covering the whole surface of the glass cylinder. We have derived a formula to compute the cylinder rotation frequency (f_{cyl}) by knowing the camera sample rate (f_s) and the number of frames (n) to record for one cylinder revolution

$$f_{cyl} = f_s(1 - \frac{1}{n}) \tag{A.3}$$

If a video sequence of 40 frames should be recorded for one revolution and the camera frame rate is $f_s = 20\,fps$ a rotation frequency of $f_{cyl} = 19.5\,Hz = 1170\,rpm$ is needed. In this case the total recording time will be $t = \frac{n}{f_s} = \frac{40}{20} = 2$ sec. We will need 2 seconds to record 40 frames partly overlapping but covering the whole cylinder surface.

The hypothesis is that by processing the sequence of video frames it should be possible to track the particle position. We have made a number of recordings with particles on the inside and outside of the glass container. Figure A.13 illustrates how a number of frames can be processed for segmentation of particle blobs. The segmented frames should then be used as input for tracking of particles floating in the liquid or adhered to the glass surface. Since the physical geometry of the cylinder and the rotation speed are known, it should be possible to track and estimate the position of the particles.

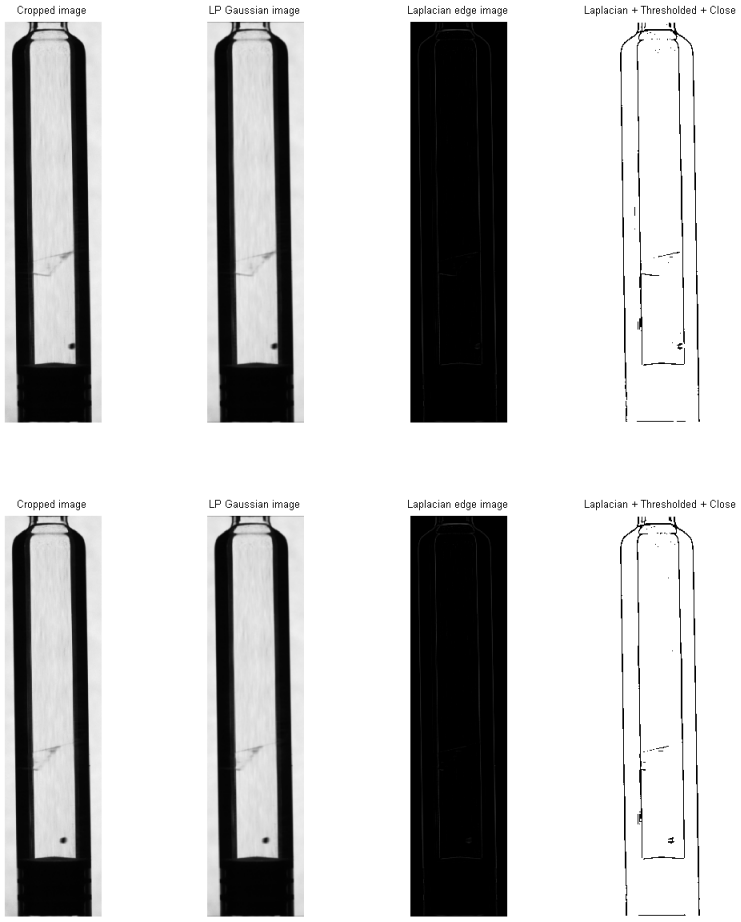


Figure A.13: Two video frames are low pass filtered (Gaussian) followed by a Laplacian edge detection, thresholding and morphological close operation. A small particle is floating inside the cartridge and a piece of tape is adhered on the glass surface. We can observe how the particle and tape have moved comparing the top and bottom images.

B

Appendix B - Particle Observations

B.1 Particles and Refraction

If an object is transparent and background lighting is used as described in chapter 3.2.3, refraction will occur [21]. When light passes a boundary between two transparent materials with different refraction indices, the direction of light propagation is deviated. The velocity of light propagation depends on the medium where the light rays pass through. In all other mediums than vacuum the light propagates slower. Hence a change in the direction of light rays is the consequence and the light is refracted. When rays go from a medium with refractive index η_1 to one with refractive index η_2 with an incidence angle to the surface normal of Θ_1, the transmission angle Θ_2 can be calculated from Snell's law of refraction

$$\eta_1 sin\Theta_1 = \eta_2 sin\Theta_2 \tag{B.1}$$

If there is no angle Θ_2 fulfilling Snell's law, the light cannot be transmitted through the medium and will instead undergo total internal reflection:

$$\frac{\eta_1}{\eta_2} sin\Theta_1 > 1 \tag{B.2}$$

Using ray tracing and Snell's law we can construct a model for how particles on the backside of the glass container will be observed on the front side. By using paraxial approximation [79] Snell's law can be rewritten as:

$$\eta_1 \Theta_1 = \eta_2 \Theta_2 \tag{B.3}$$

Paraxial approximations are potentially very useful. To use them, the angles need to be small. If this is not the case, the results cannot be relied upon for total accuracy. However, formulas based on equation B.3 will generally give a good prediction of where the focal planes will be and how the image is transformed. Therefore paraxial approximation is used for optical lens theory to create paraxial aberration models. Optical paraxial aberration models exist for cylinder and ball lenses that behave similar to our glass container filled with liquid.

In figure B.1 it is illustrated how only a portion of the particles on the backside will be visible on the front side due to refraction. The refraction index of glycerol is 1.47 and for glass it is 1.52. Since they are nearly the same, we only model light rays passing from air to liquid. By assuming that light rays pass the container surface perpendicular to the center, we can find the

transmission angle passing through the glass container with liquid. Particles from the backside will be magnified approximately 2.6 times due to a ball-lens effect and only some of them will be visible. The model corresponds with our observations where we are able to see 15 out of 50 particles on the backside for a cartridge filled with glycerol. The rest of the particles are hidden due to refraction and their 2D position and size will not be correct since they are magnified due to the lens effect.

The law of refraction will also apply for particles inside the liquid. For particles on the inside of the glass surface we will observe that their position is refracted resulting in an increased distance between them as illustrated for the red particles refracted on the front side in figure B.1. Thereby an estimate of the observed radii will probably be slightly bigger than in reality.

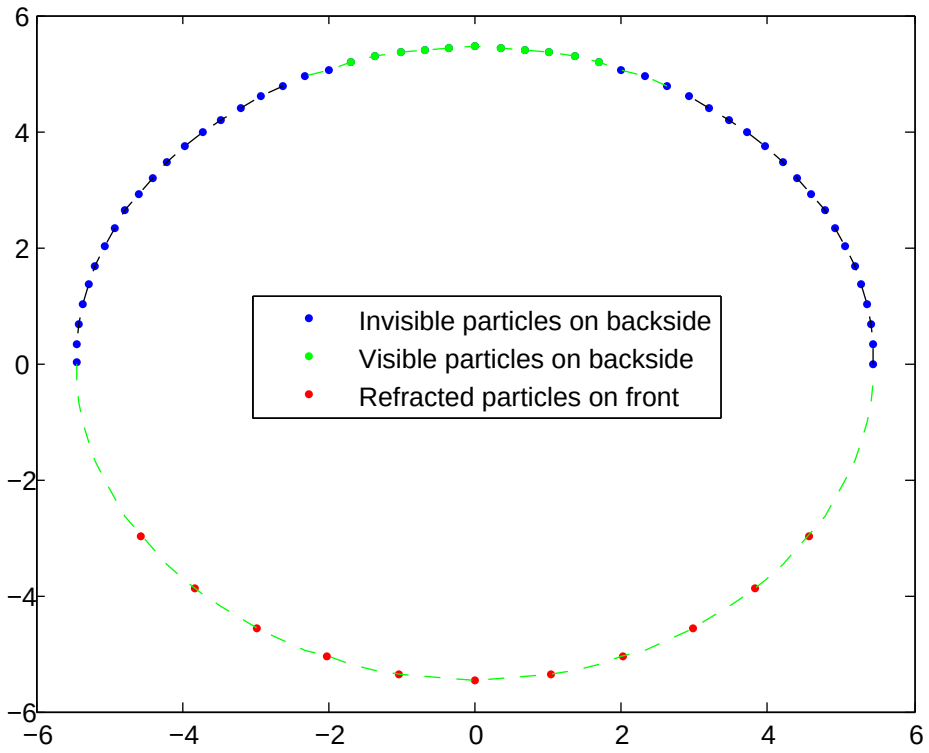


Figure B.1: Top-view of a container. Blue dots denote particles on the backside that are not visible from the camera. Green dots denote particles on the backside that are visible from the camera but distorted due to refraction. On the front side these are seen as the red dots. Particles will be magnified due to the ball-lens effect.

B.2 Particles and Fluid Mechanics

The properties of fluid mechanics in relation to particles floating inside the liquid of the container must be examined. There are three forces influencing the particle movement inside the liquid: gravity, centripetal force caused by rotation and friction between particle and glass surface. From Stokes' law a gravitational velocity on the particle is introduced [15]. The velocity U_g in direction of the gravity depends on the density of fluid ρ and the particle density ρ_p:

$$U_g = D_p^2 \frac{(\rho_p - \rho)}{18\mu} g \tag{B.4}$$

where g is the acceleration of gravity, μ the dynamic viscosity of the fluid and D_p is the diameter of the particle. When the container is rotating we have a circular motion which requires the presence of a centripetal force. We then have a rotating frame of the container where particles appear stationary because of a balance between the real centripetal force, the frame-determined centrifugal force and the frictional force. The frictional force between a particle and the glass surface will be proportional to the centripetal force. At a certain high rotation speed the frictional force will be higher than the impact of gravity, and thus the particle will move at the same speed as the rotating container.

From equation B.4 it can be seen that the required rotation speed and friction depend on the liquid viscosity and difference between densities. From the same equation we observe that for a liquid with a high dynamic viscosity the velocity U_g will be small, which corresponds to observations about particles in liquids with high viscosities being difficult to move around.

Appendix C - First Algorithm

This appendix contains an elaborate evaluation of the two matching methods described in chapter 4.3.1 as well as a comprehensive explanation and an algorithm for correcting particle positioning for perspective projection as described in chapter 4.4.

C.1 Matching Evaluation

In the first algorithm described in chapter 4 particle tracking is based on different matching algorithms. In evaluating these matching algorithms and deciding on whether to use matching by correlation or by area the first algorithm is analyzed based on the data set (F-J) as described in chapter 3.4.

This is done by manually inspecting a sequence of images that are equally distributed around the glass container. Different number of images are used ranging from 11 to 87 in an image sequence. A particle is detected by the algorithm if it is successfully tracked in a successive number of frames on the front side of the glass container. For every particle that is detected by the algorithm information whether it was observed as being inside or outside the surface of the glass container is manually recorded.

Data is also provided enabling statistics for investigating points of weaknesses within the system and algorithms. We would like to know if it is the segmentation, matching or tracking algorithm that in each case is the reason for not being able to track a manually observed particle correctly. If the manual inspection finds that a particle tracked by the algorithm is non-existing, information about where it was detected is recorded. There are two locations where we would like to determine false particle detections – either in the bottom of the container where we have a dark background due to the rubber stopper, or on the glass surface caused by small particles of dust. The list below summarizes the information recorded for every particle.

- **Observed position**: Particle observed on the inside, outside or non-existing. Non-existing means that the particle has not been labeled manually. If non-existing, information is recorded of where the particle is detected – for instance on the glass surface or in the bottom of the container.
- **Observed reason**: Reason that the particle is not detected – for instance due to the segmentation, matching or tracking algorithm. The reason is rated subjectively by the person inspecting the results.

Looking at the result for matching we can see from figure C.1 and C.2 that the best result is achieved using matching by area. Here we achieve the **best mean particle detection rate of 94.1% for matching by area** compared to 90.1% when using matching by correlation for varying numbers of frames (see figure C.1).

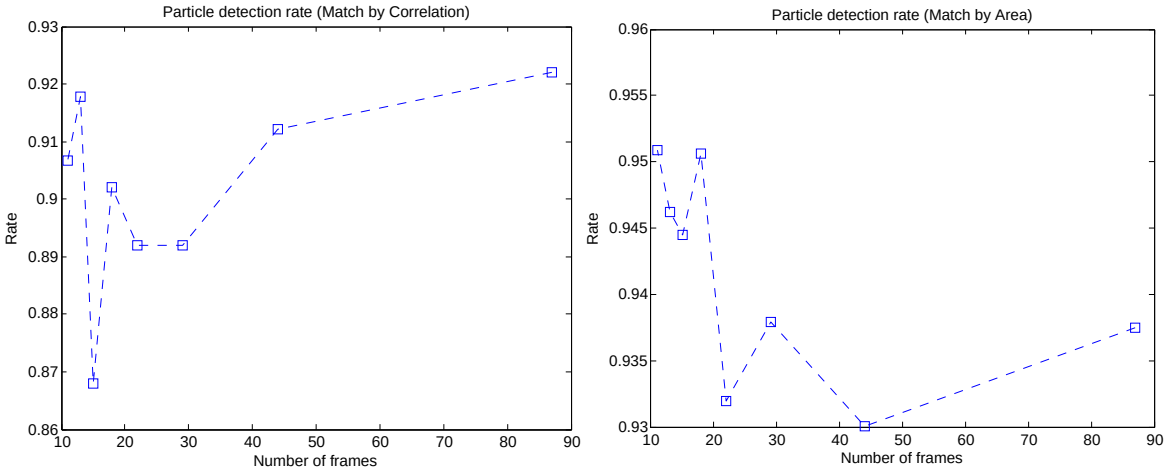


Figure C.1: Particle detection rate using matching by correlation (left) or area (right) as a function of the number of frames distributed around the glass container.

During manual inspection we observe that match by correlation fails especially for big particles and decreasing number of frames. In this case the similarity of pixel intensities around the particle centers fail to correlate by increasing distance between centers in two successive frames. In figure C.2 (left) this is seen as a rise in missing particle detections when using less than 44 frames.

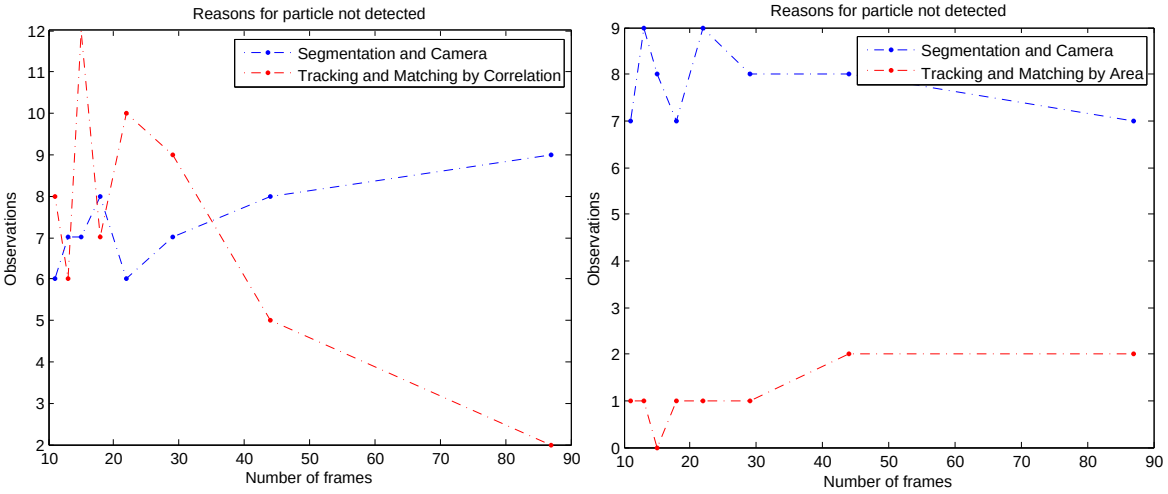


Figure C.2: Observed reason for why particles are not detected using match by correlation (left) or area (right) as a function of the number of frames distributed around the glass container.

From figure C.3 we observe for matching by correlation that the number of false particle detections on the container glass surface increases when only a few number of frames are used for inspection. During manual inspection it is observed that the reason is related to an increase in false matches in both horizontal and vertical directions. Since we are searching for matches in a circular distance to the particle center, matches are found in many directions instead of limiting the

search using a rectangular window, as used in the match by area algorithm. To conclude, match by area is the best algorithm achieving the best detection rate with the fewest number of false particle detections on the glass surface.

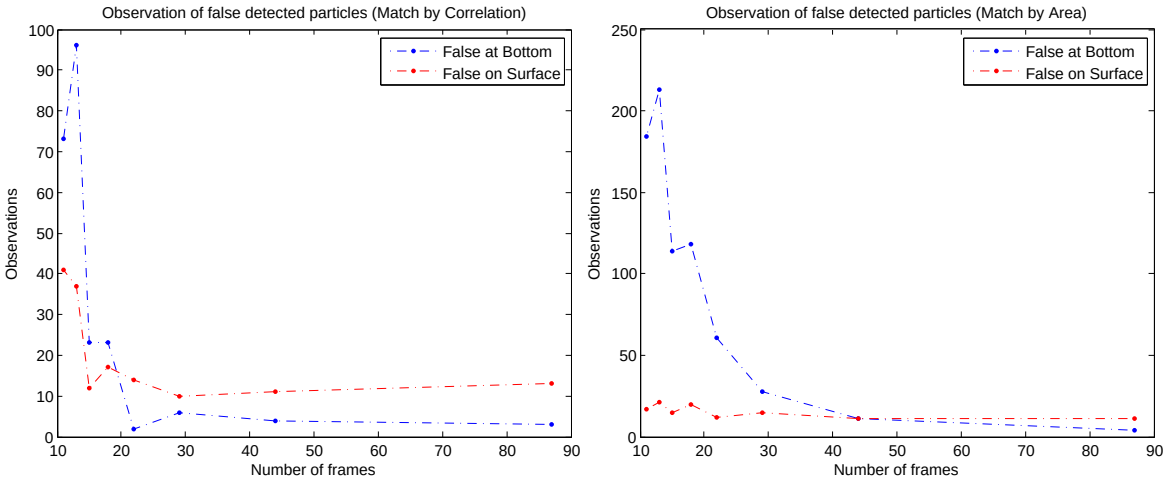


Figure C.3: Observed reason for false particle detections using matching by correlation (left) or area (right) as a function of the number of frames distributed around the glass container.

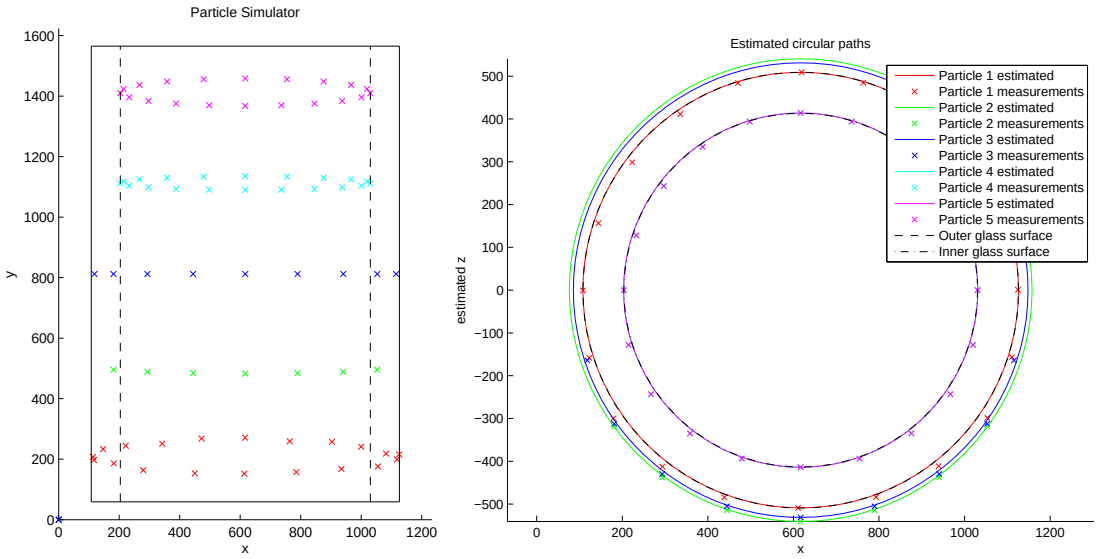
In figure C.3 we also observe an increasing number of false detections in the bottom area of the container. The reason relates to a combination of segmentation as discussed in chapter 4.2.1.2 and tracking. Since the tracking algorithm accepts a particle as being detected when linking only a few number of sequential matches, we observe the increase in false detections when fewer number of frames are use.

C.2 Correcting for Perspective Projection

As seen in figure C.4 (copies of figures 3.9 and 4.18) a small offset is present for the estimated radii of the two particles that only have measurements on the front side of the container. The reason for the offset is the perspective projection that distorts the x-coordinates. The perspective projection depends on the distance between the particle and the camera.

Figure C.5 illustrates the problem using an extremely small distance d between the camera and the container to emphasize the distortion for visualization purposes. A particle located at a position (x_k, z_k) will intersect the image plane at (x'_k, z'_k) when viewed from a center of projection (COP) at $(0, d)$. However, a particle located at $(x_k, -z_k)$ will intersect the image plane with a larger x-coordinate (the red dashed line) and thereby introduce a distortion of the utilized coordinates in the positioning algorithm. Actually, particles with a negative z-coordinate will make the x-coordinates distort outwards resulting in larger estimated circles, whereas particles with a positive z-coordinate will make the x-coordinates distort inwards resulting in smaller estimated circles.

The situation depicted in figure C.4b also illustrates this phenomenon. Both the red and the magenta colored particles experience an outward distortion at negative z-coordinates and an inward distortion at positive z-coordinates. Luckily these two phenomena cancel out their contributions to the erroneously estimated radii, and the best fit is actually the correct radius. However, for the blue and green particles only points on the front side of the container with negative z-coordinates are available, and the radius is therefore estimated too large.



(a) Example with 5 particles generated by the particle simulator.

(b) Estimated circles from measurements.

Figure C.4: Example from the particle simulator illustrating the problem of perspective projection.

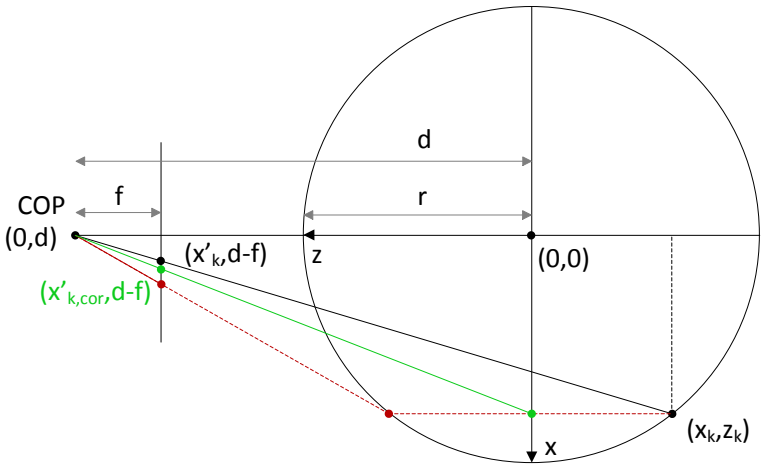


Figure C.5: Model for perspective correction of distorted x-coordinates.

A method for correcting the distortions of the x-coordinates and thereby preventing erroneously estimated radii can be derived from figure C.5. The ratio between the legs of the two similar triangles must be the same:

$$\frac{(x'_k - x'_c) \cdot s_x}{f} = \frac{x_k}{d + z_k} \tag{C.1}$$

where x'_c is the center of the container in pixels and s_x is the scale factor relating pixels to meters on the camera CCD chip. In this equation both x_k and z_k are unknown. However, from the segmentation we know the diameter of the container in pixels and if we also know the diameter in meters, we can write:

$$x_k = (x'_{k,cor} - x'_c) \cdot \frac{r_{container,m}}{r_{container,px}} \tag{C.2}$$

where $x'_{k,cor}$ is the corrected x-coordinate in the image and $r_{container}$ is the container radius in either meters or pixels. A similar expression can be written to substitute z_k:

$$z_k = z'_{k,est} \cdot \frac{r_{container,m}}{r_{container,px}} \tag{C.3}$$

Here $z'_{k,est}$ is an estimate of the radius in pixels. The problem is that the overall goal is to estimate this radius, and it is therefore unknown at this stage. As an estimate however, the radius of the position without perspective correction can be used.

Substituting equation C.2 and C.3 into equation C.1 and rearranging gives us:

$$x'_{k,cor} = \frac{d + z'_{k,est} \cdot \frac{r_{container,m}}{r_{container,px}}}{f} \cdot (x'_{k,cor} - x'_c) \cdot \frac{r_{container,m}}{r_{container,px}} \cdot s_x + x'_c \tag{C.4}$$

Using this formula the estimated radius can be corrected for perspective correction in two iterations of the positioning algorithm. First it estimates the radius roughly in order to obtain a value for $z'_{k,est}$. A second iteration corrects the x-coordinates using equation C.4 before re-estimating the radius.

Figure C.6 shows the same reconstructions as in figure 4.18, but now with perspective correction. The blue and the green particles are now estimated correctly and all measurements fit with the circles, resulting in smaller values of the error-function.

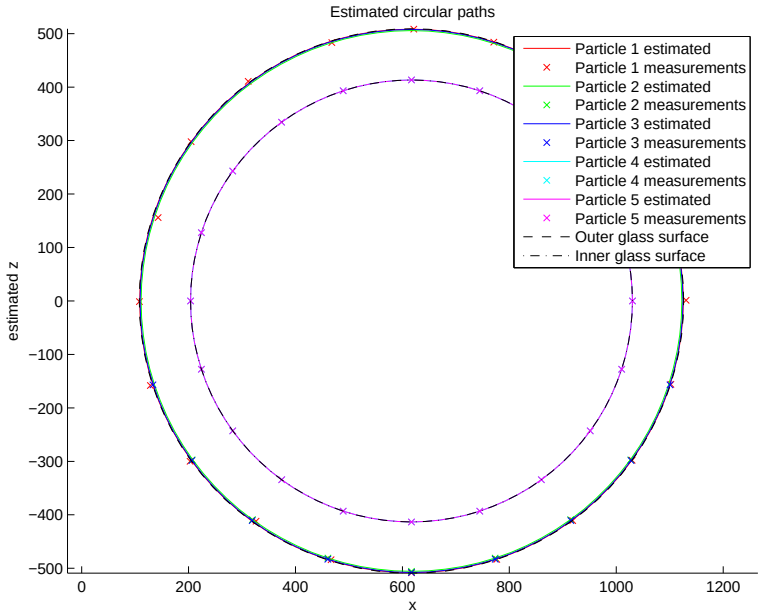


Figure C.6: Estimated circles from measurements using perspective correction. Together with the circles the corrected measurements are plotted, where the z-coordinates have been reconstructed from the estimated radii.

The phenomenon can also be visualized by simulating with a varying visible angle range. Figure C.7 shows the estimated radii relative to the visible angle range, going from 0 to 2π. The left graph shows the simulation without perspective correction where an "overshoot" is present when only points of the front side of the container are available. The right graph shows the simulation with perspective correction where the radii are estimated correctly. This is the case when at least two points are available for the positioning algorithm.

From equation C.4 the perspective correction assumes that we know the values of the focal length, f, the scale factor, s_x, and the working distance, d, between the camera and the container.

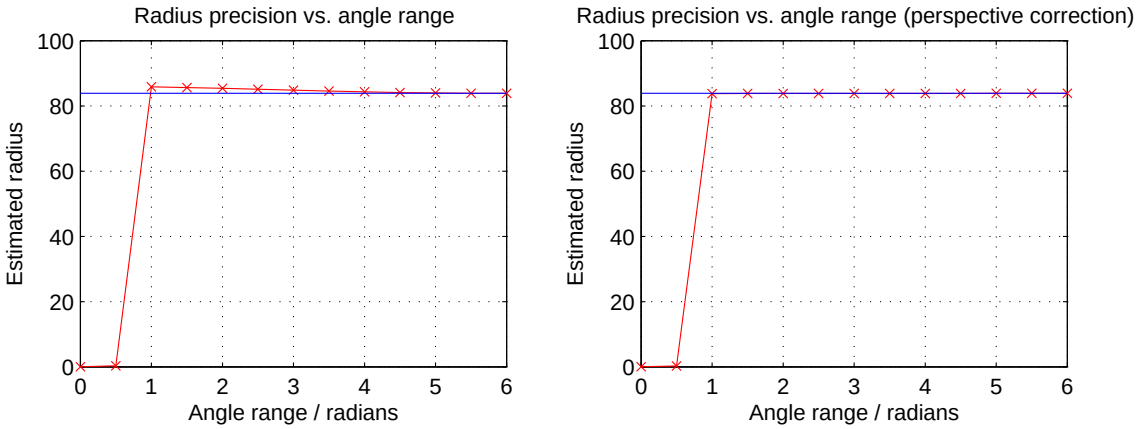


Figure C.7: Estimated radii as a function of the visible angle range. At the left image no perspective correction is made, but at the right image the x-coordinates are corrected achieving exact estimates of the radii.

This may not always be the case, and if the parameters are not precise, the perspective correction can actually worsen the estimates. Additionally the estimate of θ_0 from the first iteration of the positioning algorithm decides whether the points are located on the front side or the back side of the container, which directly affects whether $z'_{k,est}$ is negative or positive - determining whether to correct for inwards or outwards distortion.

During testing it has been found that θ_0 is often estimated with an error of π. If this is not corrected by a priori knowledge about the segmentation and tracking of points, so that only points located on the front side of the container are used, the perspective correction might have the opposite effect of what was intended – aggravating the errors of the estimated radii.

Therefore, in practice the perspective correction is not used in this first version of the system design. Also, since it utilizes an estimate of the radius in order to correct it, it is arguable whether anything can be gained by the algorithm.

Appendix D - Data Set

Test unit	Container	Liquid	Description
A	Cartridge	Glycerol	2 marker points, 1 inside and 1 outside
B	Cartridge	Glycerol	2 aluminum particles inside and 1 marker point outside
C	Cartridge	Insulin	3 aluminum particles inside
CS	Cartridge	Insulin	3 aluminum particles inside and many dust particles outside
D	Cartridge	Empty/air	6 dots of correction fluid, 3 inside and 3 outside
E	Cartridge	Insulin	1 particle inside and 3 marker points on the outside

Table D.1: Detailed description of initial data set for algorithm development.

Test unit	Container	Liquid	Description
F6	Cartridge	Insulin	3 glass balls inside
F23	Cartridge	Insulin	1 hair particle inside and 2 dust grains outside
F48	Cartridge	Insulin	1 rubber plug particle inside
F85	Cartridge	Insulin	1 large aluminum particle and 1 small particle inside
F101	Cartridge	Insulin	3 small glass balls inside
F149	Cartridge	Insulin	3 elongated particles inside
G301	Cartridge	Insulin	3 marker points on outer surface
G302	Cartridge	Insulin	5 marker points on outer surface
G305	Cartridge	Insulin	3 marker points on outer surface
G306	Cartridge	Insulin	3 dots of correction fluid on outer surface
G308	Cartridge	Insulin	6 dots of correction fluid on outer surface
H1	Syringe	Gel	Empty
H2	Syringe	Gel	Damage to the piston, but no particles
H4	Syringe	Gel	1 fiber particle in piston groove but not visible inside
H5	Syringe	Gel	1 glossy particle inside
H6	Syringe	Gel	1 glossy particle inside
H8	Syringe	Gel	1 white fiber particle inside
I1	Syringe	Gel	2 marker points on outer surface
I4	Syringe	Gel	2 marker points located closely and 1 elongated dust grain on outer surface
J403	Cartridge	Glycerol	3 small marker points and 7 air bubbles inside

Table D.2: Detailed description of training set.

Test unit	Container	Liquid	Description
F8	Cartridge	Insulin	3 glass balls inside
F45	Cartridge	Insulin	1 paper particle inside
F68	Cartridge	Insulin	1 small particle inside
F103	Cartridge	Insulin	1 glass ball inside
F119	Cartridge	Insulin	1 particle inside
F126	Cartridge	Insulin	1 thin wire inside
G303	Cartridge	Insulin	3 marker points on outer surface
G304	Cartridge	Insulin	4 marker points on outer surface
G307	Cartridge	Insulin	3 large dots of correction fluid on outer surface
G309	Cartridge	Insulin	3 dots of correction fluid on outer surface
G310	Cartridge	Insulin	3 dots of correction fluid on outer surface
H3	Syringe	Gel	1 particle in piston groove but not visible inside
H7	Syringe	Gel	1 bright particle inside
H9	Syringe	Gel	1 white fiber particle inside and 1 dust grain outside
H10	Syringe	Gel	1 white particle inside
H11	Syringe	Gel	Empty
I2	Syringe	Gel	2 marker points on outer surface
I11	Syringe	Gel	2 small marker points on outer surface
J401	Cartridge	Glycerol	2 marker points, rubber and 11 air bubbles inside
J404	Cartridge	Insulin	Empty

Table D.3: Detailed description of test set.

Appendix E - Images and Videos

With this report a DVD is attached containing a number of images and video recordings. Here a number of samples are generated for a selected number of containers from the test set described in appendix D:

- A sequence of images concerning container recordings used for developing the first algorithm as described in table D.1.

- Video recordings concern tracking and positioning results of the first algorithm using 87 or 44 frames described in table D.3.

In every video of the rotating container particles found by manual hand-labeling and detected by the algorithm are marked being on the inside or outside of the container. Hand-labeled particles on the inside or outside are marked with respectively a magenta or yellow dot indicating the center of the particle blob. The result of tracking and positioning is illustrated with a green or red bounding box for particles detected respectively on the outside or inside.

The following videos of containers are found on the DVD recorded with 87 or 44 frames:

- F8_87, F126_87, G309_87, G310_44, G310_87, H7_87, H9_87, I11_87 and J401_87

The following images of containers are found on the DVD concerning:

- A1, CS1, D1 and E1

Appendix F - Code and Platform

This appendix describes the processing platform used for measuring execution times. The MATLAB script and CUDA Kernel for optimized background extraction are listed.

F.1 Processing Platform

Computer processing platform for real-time performance evaluation:

Software

- MATLAB R2013b
- CUDA version 5.5
- Windows 7 Professional, Service Pack 1

Hardware

- Processor: Intel Core i7 CPU 980, 3.33 GHz, 6x2 kernels
- Memory: 16 GByte, 64-bit
- GPU: NVIDIA GeForce GT 610, 0.81/1.62 GHz, 1 GByte

F.2 Background Extraction

Below, the MATLAB script and the CUDA Kernel for extracting the background image by median filtering of 9 input images are listed.

The listed backgroundKernel.cu file must first be compiled using the NVIDIA CUDA Toolkit (nvcc -ptx backgroundKernel.cu) to generate the binary PTX file that will be downloaded to the GPU and executed from the 'feval' invocation in MATLAB. The Parallel Computing Toolbox [41] contains a complete guide for executing CUDA kernels from MATLAB. The GPU architecture and programming guides for C++/CUDA are found in [45, 40, 42].

The MATLAB code listing below creates a CUDA kernel for median filtering of 9 images in extracting the background image. A thread grid size of 32x32 is selected. Input parameters in terms of images will automatically be copied from CPU memory to GPU memory. The result is copied back to the CPU memory by the MATLAB **gather** function.

```
% MATHLAB script that executes the GPU Background Extraction Kernel

threadSize = 32;
w = size(imgArr{steps(1)},1); % Width of image
h = size(imgArr{steps(1)},2); % Height of image
% Create CUDA Kernel
k = parallel.gpu.CUDAKernel('backgroundKernel.ptx', 'backgroundKernel.cu');
% Specify thread grid size
k.ThreadBlockSize = [threadSize, threadSize, 1];
k.GridSize = [ceil(w/threadSize), ceil(h/threadSize), 1];
% Create result image
GRI = zeros(w, h);
% Copy data to GPU memory and executes kernel
GRI = feval(k, GRI,...
        imgArr{steps(1)}, imgArr{steps(2)}, imgArr{steps(3)},...
        imgArr{steps(4)}, imgArr{steps(5)}, imgArr{steps(6)},...
        imgArr{steps(7)}, imgArr{steps(8)}, imgArr{steps(9)},...
        w, h);
% Gather result from GPU memory, copied to CPU
img = gather(GRI);
```

The CUDA code listing below contains the kernel to perform median filtering of 9 input images. Remark that no for loops are needed since threads are instantiated on a huge number of parallel GPU cores. Each thread then computes its position in the two dimensional grids of threads. It performs sorting and median filtering on one pixel in every 9 frames and stores the result in the same position of the returned destination matrix.

```
/**
 * @file backgroundKernel.cu
 *
 * CUDA code to calculate the background image using median filtering.
 * Median filtering of pixel intensities from 9 images.
 *
 * Copyright 2013, Kim Bjerge
 */

__device__ void swap (unsigned char *x, unsigned char *y)
{
        unsigned char tmp;
        tmp = *x;
        *x = *y;
        *y = tmp;
}
```

```
__device__ void bublesort (unsigned char *a, int depth)
{
        int i, j;
        for (i = 0; i < (depth-1); i++)
                for (j = 0; j < (depth-(i+1)); j++)
                        if (a[j] > a[j+1])
                                swap(&a[j], &a[j+1]);
}

/** Main entry point.
 * Works out where the current thread should read/write to global memory
 * and calls bublesort to sort pixel values before the median value is returned
 */
__global__ void backgroundKernel(double * dst,
                const unsigned char * i1, const unsigned char * i2,
                const unsigned char * i3, const unsigned char * i4,
                const unsigned char * i5, const unsigned char * i6,
                const unsigned char * i7, const unsigned char * i8,
                const unsigned char * i9,
                const int width,
                const int height)
{
  unsigned char median[9];

  // Computes current thread position
  int row = blockIdx.y * blockDim.y + threadIdx.y;
  int col = blockIdx.x * blockDim.x + threadIdx.x;

  // Test for thread inside image area
  if (row < height &&  col < width)
  {
    // Work out which thread we are
    int globalThreadIdx = row * width + col;

    // Read in pixel valued from same position in each 9 images
    median[0] = i1[globalThreadIdx];
    median[1] = i2[globalThreadIdx];
    median[2] = i3[globalThreadIdx];
    median[3] = i4[globalThreadIdx];
    median[4] = i5[globalThreadIdx];
    median[5] = i6[globalThreadIdx];
    median[6] = i7[globalThreadIdx];
    median[7] = i8[globalThreadIdx];
    median[8] = i9[globalThreadIdx];

    // Sort pixel intensity values
    bublesort(median, 9);

    // Return median value
    dst[globalThreadIdx] = median[4];
  }
}
```

Appendix G - Circular Trajectory Tracker

This appendix contains a proof related to the Circular Trajectory Tracker.

G.1 Proof of Adaptive Window Size

In the following, equation 6.2 (repeated below) is proven.

$$c_k(i) - c_{k-1}(h) < c_{k+1}(j) - c_k(i) \qquad if \qquad c_k(i) < c_{cl} \tag{G.1}$$

$$c_k(i) - c_{k-1}(h) > c_{k+1}(j) - c_k(i) \qquad if \qquad c_k(i) > c_{cl} \tag{G.2}$$

By assuming a perfect harmonic motion of the x-coordinate, $c_k(i)$ can be described in polar coordinates as $r \cdot cos(\theta)$ where a potential offset is ignored. Because we assume a constant angular velocity, $c_{k+1}(j)$ can be described as $r \cdot cos(\theta + \Delta\theta)$. In this way equation G.1 can be rewritten as

$$\begin{aligned}
r \cdot cos(\theta) - r \cdot cos(\theta - \Delta\theta) < r \cdot cos(\theta + \Delta\theta) - r \cdot cos(\theta) \qquad & if \qquad r \cdot cos(\theta) < r \cdot cos(\frac{3\pi}{2}) \\
\Leftrightarrow cos(\theta) - cos(\theta - \Delta\theta) < cos(\theta + \Delta\theta) - cos(\theta) \qquad & if \qquad cos(\theta) < 0 \\
\Leftrightarrow 2cos(\theta) < cos(\theta + \Delta\theta) + cos(\theta - \Delta\theta) \qquad & if \qquad cos(\theta) < 0 \\
\Leftrightarrow 2cos(\theta) < \frac{cos(\theta)cos(\Delta\theta)}{2} \qquad & if \qquad cos(\theta) < 0 \\
\Leftrightarrow 4 > cos(\Delta\theta) \qquad & if \qquad cos(\theta) < 0
\end{aligned}$$

In the last step the 'less than' sign changes to a 'greater than' sign because $cos(\theta) < 0$. The same proof can be made for equation G.2 that also results in $cos(\Delta\theta) < 4$ which is always true.

Appendix H - Multiple Hypothesis Tracking

This appendix contains theory related to Multiple Hypothesis Tracking.

H.1 Clustering

As mentioned in chapter 6.2.3, clustering is performed in order to decompose the tracking problem into smaller problems. The goal is to achieve a computationally efficient algorithm that scales linearly with the number of targets [53]. If every target can be separated into its own cluster, parallel computing approaches could potentially guarantee real-time execution irrespective of the number of targets present.

A cluster is defined as a set of targets and measurements and the different association hypotheses (the hypothesis matrix) linking them. In the simple example in figure 6.11 the targets $T1$ and $T2$ share the measurement $\mathbf{x}_k(2)$. Therefore, these targets are related and must be within the same cluster. However, if a third target $T3$ was known and associated with a fourth measurement $\mathbf{x}_k(4)$ as in figure H.1, a second cluster containing target $T3$ and a potential new target $T6$ would be created.

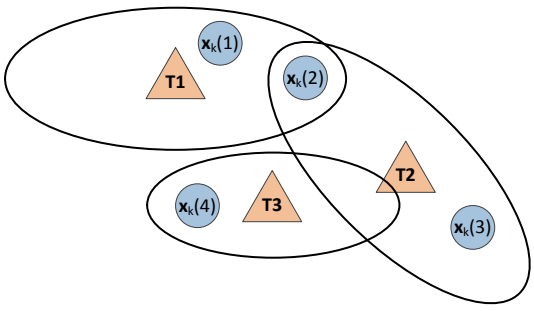


Figure H.1: Example of multiple hypothesis clustering. Three prior targets, $T1$, $T2$ and $T3$, and a measurement set at time k consisting of four measurements are present. Two clusters are created because target $T1$ and $T2$ are related by measurement $\mathbf{x}_k(2)$ creating the first cluster, whereas target $T3$ is unrelated to $T1$ and $T2$ thereby creating the second cluster.

In the above example two clusters are formed. In the next frame, the measurement set $\mathbf{x}_{k+1}$ is available. Each measurement within this set is associated with one of the two known clusters if it is within the gate of one of the prior targets. If a measurement connects two known clusters, these are combined/merged into a supercluster. In figure H.1 this would happen if a measurement is observed within the joint gate area of target $T2$ and $T3$. However, if a measurement cannot be associated with any prior targets, a new cluster is created.

H.2 Track-Oriented MHT

The described MHT algorithm and hypothesis generation in this thesis is generally known as the Hypothesis-Oriented MHT (HOMHT). This is the original algorithm proposed by Reid in 1979 also known as the conceptual MHT. It is hypothesis-oriented in the sense that it generates hypotheses for all measurement-to-track associations. Reid originally chose the hypothesis-oriented approach because he was convinced that it appeared simpler than the track-oriented approach [53]. However, today Track-Oriented MHT (TOMHT) [69] is the preferred method because is tends to require less memory.

The principle behind TOMHT is that instead of constructing a hypothesis tree as in figure 6.12a a track tree is constructed for every known track. The nodes of each track tree denote the measurements that are associated with the track, completely opposite the convention used in HOMHT. A track tree consists of N incompatible nodes, because each node represents different measurement-to-track associations for the track. Figure H.2 shows the track trees of the simple example presented in figure 6.11. A zero denotes that the track is undetected in the current frame. Since target $T1$ and $T2$ are prior targets, these are associated with hypothetical measurements 1 and 2 in the previous frame $k-1$ that initiated them, whereas target $T3$, $T4$ and $T5$ were not detected in frame $k-1$ and thus are indicated by zeroes.

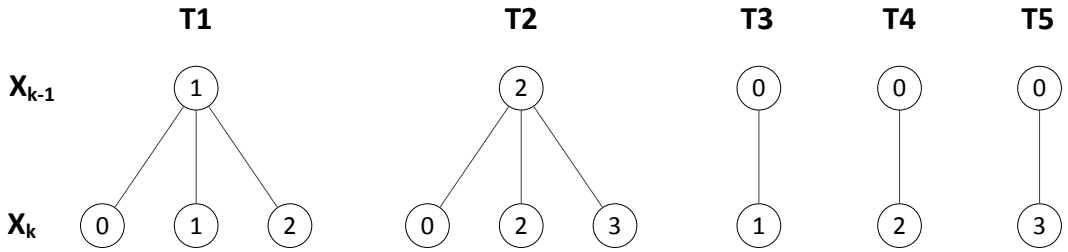


Figure H.2: Simple example with Track-oriented MHT. The track trees are generated from the example presented in figure 6.11. The nodes of each track tree denote the measurements that are associated with the track. Since target $T1$ and $T2$ are prior targets, these are associated with hypothetical measurements 1 and 2 in the previous frame $k-1$, whereas target $T3$, $T4$ and $T5$ were not detected in frame $k-1$ and thus are indicated by zeroes.

Basically the data representation is the same for HOMHT and TOMHT. If no pruning and track maintenance is performed, the required memory and computation is the same. However, when pruning is performed and combined with keeping only the m-best hypotheses, it is argued that TOMHT has to store fever combinations of tracks than HOMHT has to store combinations of hypotheses [48].

The process of pruning and track management can be explained quite simply with TOMHT:

1. When the track trees are formed by the measurements at time k, the first step is to perform n-scan-back pruning as described in chapter 6.2.2.5 above. For a track tree this corresponds

to choose a unique origin (measurement-to-track association) of the track tree at the level $k-n$. In figure H.2 $n = 1$ and there is already a unique origin for simplicity. However, if $T1$ had been associated with more than just measurement 1 in frame $k-1$, n-scan-back pruning with $n = 1$ would have to choose either of these measurements as the unique origin of $T1$. This is done by choosing only the branch that is present in the most probable hypothesis calculated by equation 6.24 as previously. If no branches of a track are present in the most probable hypothesis, we look at the second most probable hypothesis [48].

2. The second step is to find the m-best hypotheses as done with Murty's algorithm in HOMHT. Here we assume that we know the probabilities of all tracks. The probability of a track can be calculated by summing the probabilities of all hypotheses that contain the track. However, in practice a dedicated track probability equation can be derived to replace equation 6.24 [48]. Hypotheses are formed by arranging compatible tracks and calculating the combined probabilities. This can be formulated as an optimization problem with the goal of maximizing the hypothesis probabilities with the constraint of using only compatible tracks. The Lagrangian relaxation algorithm can be used to replace the constraint by Lagrange multipliers in order to solve a more simple problem [80]. In this way, the m-best hypotheses can be generated.

H.3 Extended State Space Model

An extended state space model includes also the two-dimensional area of the particles seen from the camera. Since the particles undergo a rigid transformation when rotating with the container, the visible area is assumed to follow a harmonic motion given by the equation:

$$A_k = A_{max} \cdot sin\left(\theta_k\right) \tag{H.1}$$

As for the radius of the circular path, A_{max} must also be included as a state, while A_k is introduced as a measurement. The extended process model can still be described with a linear state transition function:

$$\mathbf{z}_k = \begin{bmatrix} r_k \\ \theta_k \\ y_k \\ A_{max,k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{k-1} \\ \theta_{k-1} \\ y_{k-1} \\ A_{max,k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \omega \cdot T_s + \mathbf{w}_{k-1} \tag{H.2}$$

The measurement vector is extended with a measurement of A_k related to the polar coordinate description in the same way as the measured x-coordinate. In addition, a rough estimate of θ_k can be obtained from the measured x-coordinate. By assuming a particle position on the glass surface, an estimate of θ_k can be obtained as:

$$\hat{\theta}_k = cos^{-1}\left(\frac{x_k}{d_c/2}\right) \tag{H.3}$$

where d_c is the measured container diameter as described in chapter 4.1. Assuming again that we have purely additive measurement noise, the extended measurement model can be written as:

$$\mathbf{x}_k = \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{A}_k \\ \hat{\theta}_k \end{bmatrix} = \begin{bmatrix} r_k \cdot cos\left(\theta_k\right) \\ y_k \\ A_{max,k} \cdot sin\left(\theta_k\right) \\ \theta_k \end{bmatrix} + \mathbf{v}_k \tag{H.4}$$

Appendix I - McNemar's Test

In this appendix the statistical foundation is described for assessing which of the three proposed algorithms performs the best.

For convenience the graph illustrating the accuracies (on system level) for the three algorithms is repeated below.

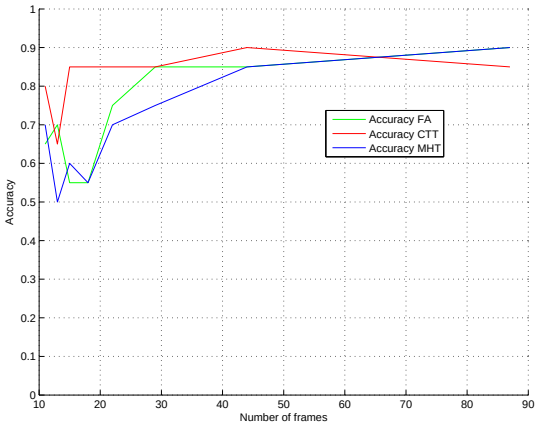


Figure I.1: Accuracy plotted as a function of the number of frames used for all algorithms. FA denotes the first algorithm, CTT denotes the Circular Trajectory Tracker and MHT denotes Multiple Hypothesis Tracking.

Due to the small test set, a single tracking error in a container can contribute with a 5% decrease in the accuracy because the test set consists of only 20 containers. Therefore, statistically a variation in the accuracy between two algorithms might not necessarily mean that one is better than the other. In order to find the statistical foundation for determining this, hypothesis testing can be performed in which we define the null hypothesis H_0 to be that two algorithms are equal in their performances. By doing this we can calculate the probability of H_0 using McNemar's test to compare the paired results of the two algorithms [81]. If this probability is smaller than 5% we reject H_0 in favor of an alternative H_1 hypothesis stating that one algorithm performs better than the other.

The McNemar test is used to evaluate categorical data of two paired proportions. In our case the categories come from confusion matrices as the values true positive (TP), true negative (TN), false positive (FP) and false negative (FN), whereas the paired proportions stem from containers in

the test set common for both algorithms. Based on the results of the two algorithms on the shared test set, a contingency table is formed as in table I.1. Here p_1 refers to algorithm 1 correctly classifying a container, that is either a TP or TN. n_1 on the other hand refers to algorithm 1 incorrectly classifying a container, that is either FP or FN. The same goes for p_2 and n_2 concerning algorithm 2.

	p_1	n_1
p_2	a	b
n_2	c	d

Table I.1: Contingency matrix.

The table is filled by evaluating all responses from the two algorithms. If a container is correctly classified by both algorithms, we add one to a. However, if both algorithms fail, we add one to d. If algorithm 1 classifies correctly while algorithm 2 fails, we add one to c, whereas in the opposite case we add one to b.

The test statistic for the McNemar test describes a chi-squared distribution given by:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{I.1}$$

However, for small sample sizes, the chi-squared distribution does not approximate a binomial distribution particularly well. Therefore, instead the exact probability distribution (the binomial distribution) can be calculated.

Restating the above definition of H_0 and H_1 we can write:

$$H_0 \ : \ \pi_b = \pi_c \tag{I.2}$$
$$H_1 \ : \ \pi_b > \pi_c \tag{I.3}$$

where π is a random variable denoting a proportion. Here H_1 is a directional alternative hypothesis, and the test is therefore one-tailed.

Because we have constructed H_0 such that the proportion of b is equal to the proportion of c, we must have the same probabilities of drawing these numbers. That is, $\pi_b = \pi_c = 0.5$. The probability of H_0 is calculated by finding $P(x \geq b)$ – the likelihood of obtaining a value that is equal to or greater than the observed b:

$$P(H_0) = P(x \geq b) = \sum_{r=b}^{m} \binom{m}{r} (\pi_b)^r (\pi_c)^{(m-r)} \tag{I.4}$$

where $m = b + c$.

An example for figure 7.18a presented in chapter 7.4 is given below. Here the first algorithm (FA) is compared to the Circular Trajectory Tracker (CTT) on the accuracy of classifying containers. Table I.2 shows the contingency matrix comparing the two algorithms for 18 frames.

	p_1	n_1
p_2	10	7
n_2	1	2

Table I.2: Example of contingency matrix comparing FA (1) and CTT (2) using 18 frames.

From the table we see that both algorithms have made 10 correct and 2 false classifications. However, CTT has classified 7 containers correctly where FA has failed, whereas the opposite case has only happened once. Therefore, the probability of the two algorithms being equal is:

$$P(H_0) = P(x \geq b) = \sum_{r=7}^{8} \binom{8}{r} 0.5^r 0.5^{(8-r)} = 0.0352 \tag{I.5}$$

Because this is below a significance level of 5%, we can reject H_0 in favor of H_1. This means that, in this case, CTT performs better than FA.

Appendix J - Future Improvements

In this appendix we have tried to illustrated some of the algorithm improvements suggested in future work.

J.1 Segmentation

In this section we have tried to illustrate the impact of improving the segmentation algorithm at the bottom of the container. In the bottom area of the rotating container the robber stopper creates a dark background. The background subtraction algorithm as described in chapter 4.2.1.2 uses a threshold value to find dark blobs on a bright background. The result is that a high number of false blob detections are found, resulting in false particle detections that have an impact on the results and especially the precision of the particle detection.

Below, we have inserted a fixed filter of 45 pixels to remove many of these false blob detections below the area of real particles. This is to illustrate how an improved background subtraction would improve the overall result as illustrated in figure J.1. Here we observe an increase in precision for both the first algorithm and the Circular Trajectory Tracker.

For the accuracy on container level in figure J.2 we observe a high accuracy down to 22 frames with the first algorithm. The CTT algorithm performs well with 18 frames still having a fairly high precision. These results indicate that improving the segmentation algorithm could give a considerably better result than presented in this thesis.

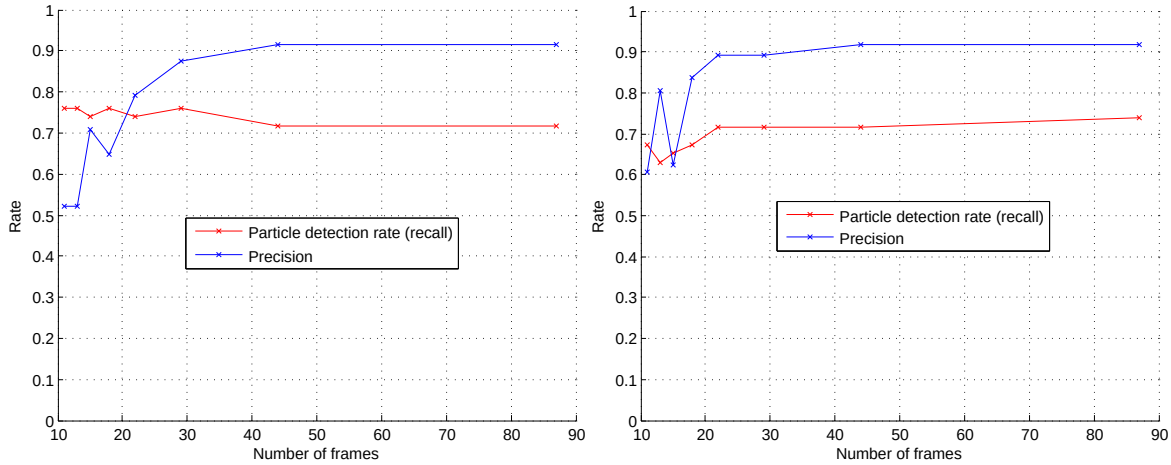


Figure J.1: Particle detection rate after filtering of noise blobs in the bottom area of the container. Left figure - first algorithm. Right figure - Circular Trajectory Tracker (CTT).

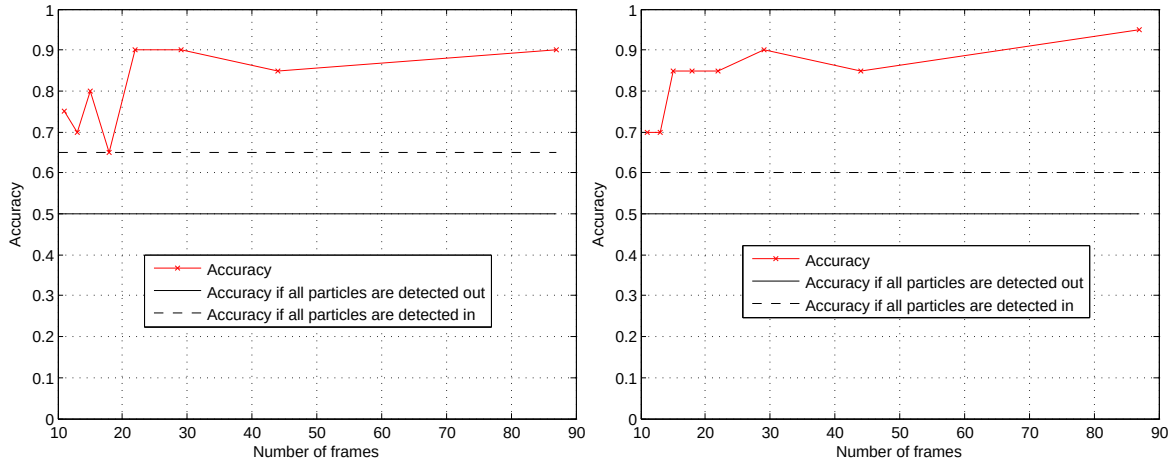


Figure J.2: Accuracy on container level after filtering of noise blobs in the bottom area of the container. Left figure - first algorithm. Right figure - Circular Trajectory Tracker (CTT).

J.2 Circular Trajectory Tracker

Parameters of the Circular Trajectory Tracker need to be investigated to find an optimal setting concerning performance and accuracy. Requiring a sequence of at least 4 blobs for 22, 29 and 44 frames results in a higher performance and precision as illustrated in figure J.3 without filtering of the bottom area. Future work needs to focus on finding optimal parameter settings based on a training set.

An alternative complex version of the Circular Trajectory Tracker could be investigated in the future when a more comprehensive test set is obtained. This version of the algorithm uses a cost function based on minimizing the direction of movement instead of change in blob area. The first two kernels are different compared to the presented CTT algorithm and seem to gain a slightly higher detection rate when using few frames. However, the complexity and memory requirements are fairly high.Figure J.4 illustrates the performance using this approach, when the same fixed filter of 45 pixels as described above is used. It performs slightly better than the original CTT when only a few frames are available.

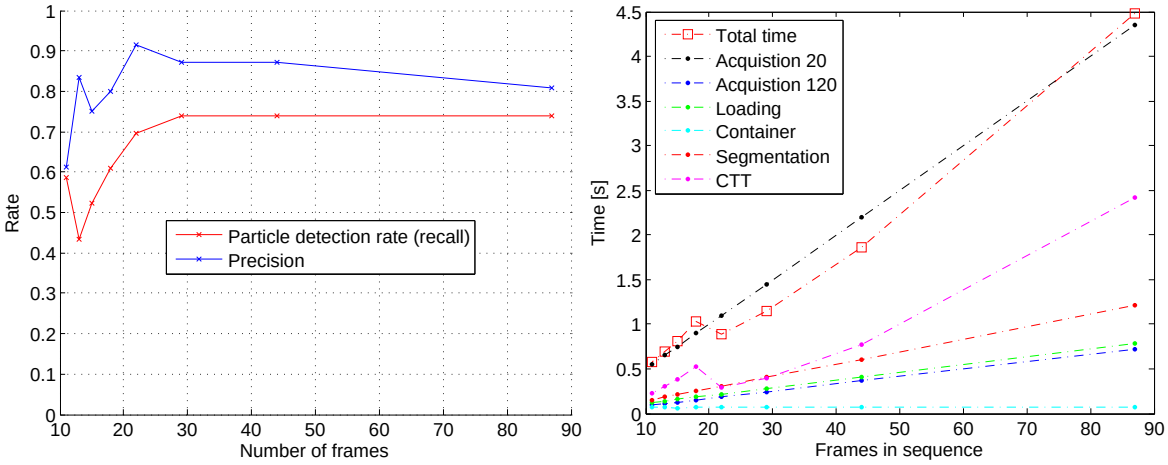


Figure J.3: Left - Particle detection rate and precision with sequence length > 3 for 22, 29 and 44 frames. Right - Total execution time with adjusted parameters.

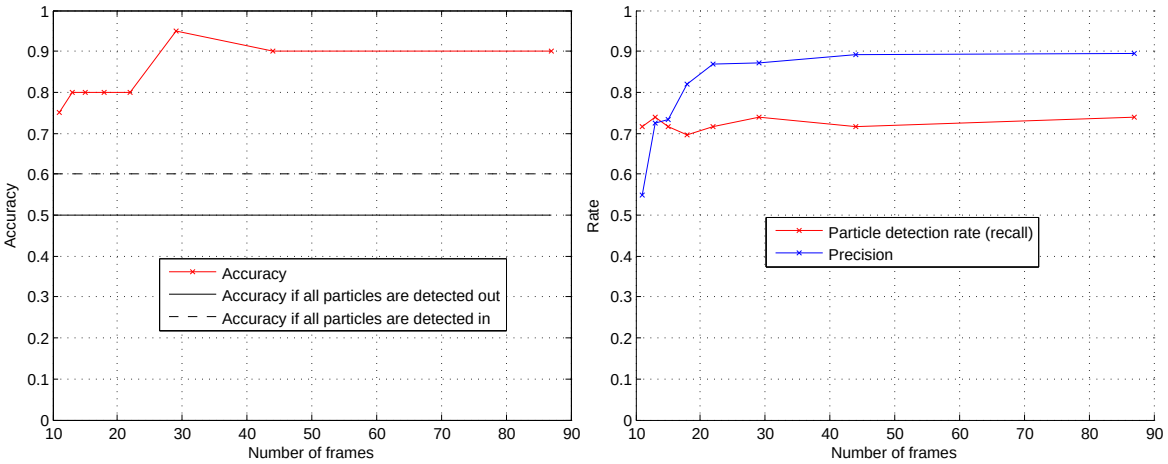


Figure J.4: Left - Accuracy for alternative CTT algorithm. Right - Particle detection rate for alternative CTT algorithm.