FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Outdoor Inside-Out Linear Image Sensor Tracking

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Christian Prossenitsch, BSc.
Matrikelnummer 0925433

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Mitwirkung: Dipl.-Ing. Dr. Christian Schönauer

Wien, 1. Juni 2018

Christian Prossenitsch	Hannes Kaufmann

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Outdoor Inside-Out Linear Image Sensor Tracking

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Christian Prossenitsch, BSc.
Registration Number 0925433

to the Faculty of Informatics

at the TU Wien

Advisor: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Assistance: Dipl.-Ing. Dr. Christian Schönauer

Vienna, 1st June, 2018

Christian Prossenitsch

Hannes Kaufmann

Erklärung zur Verfassung der Arbeit

Christian Prossenitsch, BSc.
Nordbahngasse 17C/1/4, 2253 Weikendorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Juni 2018

Christian Prossenitsch

Acknowledgements

I want to thank Hannes Kaufmann for giving me the opportunity to work in such an interesting field.

I am particularly grateful for the assistance given by Christian Schönauer, for the guidance throughout the different phases of this thesis, for his knowledge and patience.

Finally, I want to express my gratitude to my family and my girlfriend Viktoria, for providing me with support and encouragement throughout my years of study and through the process of implementing and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Kurzfassung

Tracking Systeme sind eine weit verbreitete Technologie in digitalen Anwendungen. Ohne Positionsbestimmung könnten viele computer-gestützte Anwendungen wie z.B. Robotersysteme nicht funktionieren. Vor allem aber durch die zunehmende Verbreitung von Virtual Reality / Augmented Reality Brillen wird eine genaue und schnelle Positionsbestimmung immer wichtiger. In diesen Anwendungen ist es äußerst wichtig, dass die Kopfposition des Benutzers genau und latenzfrei erfasst wird, um Nebenwirkungen wie Übelkeit vorzubeugen. Weiters müssen Tracking Systeme portabel und billig verfügbar sein, um auf dem Massenmarkt bestehen zu können.
In dieser Arbeit wird eine neuartige Tracking-Methode vorgestellt, die auf der optischen Fokussierung und Brechung von Licht basiert. Das Herzstück der Methode bildet ein linearer Bildsensor, der gemeinsam mit einem fix montiertem Linsensystem als Lichtsensor fungiert. Das Licht wird von im Raum verteilten Leuchtdioden (LEDs) ausgestrahlt, die von einem Mikrocontroller abwechselnd geschalten werden. Die Lichtkegel der LEDs müssen überlappend angeordnet sein, damit der lineare Bildsenor mindestens zwei Messungen unterschiedlicher Lichtquellen aufnimmt. Die Lichtstrahlen werden durch das Linsensystem auf den Sensor fokussiert. Durch Bewegung des Trackers ändert sich der Einfallswinkel des Lichts, wodurch die Lichtstrahlen in einem anderen Winkel gebrochen werden. Der Bereich, in dem Lichtstrahlen auf den Sensor treffen, ändert sich dadurch. Ein Mikrocontroller kann durch diese Abweichungen eine schnelle und akkurate Positionsänderung berechnen und ausgeben.
Das Projekt lässt sich in zwei Teile unterteilen: Der erste Teil besteht aus einer Simulation des Tracking Systems. Mittels Matrizenoptik wurde die Lichtbrechung durch verschiedene Linsenanordnungen simuliert. Dadurch konnten die Grenzen des Tracking Systems berechnet und die passenden Linsen mit korrekten Abständen zueinander angeordnet werden. Der zweite Teil besteht aus der Umsetzung der Simulation. Dabei wurde ein funktionierender Prototyp entwickelt, mit dem die Ergebnisse der Simulation in der Realität getestet und umgesetzt wurden. Ein Mikrocontroller übernimmt das Auslesen des linearen Bildsensors und die Ansteuerung der LEDs. Die gemessenen Intensitätswerte des Sensors werden an einen Raspberry Pi geschickt, der aus diesen Daten die Positionsänderung bestimmt. Das Tracking System kann zweidimensionale Translationen erfassen. Eine dritte Achse könnte durch einen zusätzlichen Sensor und LEDs hinzugefügt werden. Im Anschluss an die Entwicklung des Prototyps wurde die Genauigkeit und die Latenz des Tracking Systems evaluiert.

Abstract

Tracking systems are a widely used technology in physical computing. Without a positioning system, many applications like robotic systems could not function properly. Especially Virtual Reality / Augmented Reality headsets rely heavily on the position data of such systems. For this emerging technology, accurate and fast position data is crucial, in order to prevent side effects like motion sickness. Furthermore, tracking systems have to be cheap and portable to be competitive on the mass market.
In this thesis, a new kind of tracking system is presented, based on the refraction of light and optical focus. The main part of the system forms a linear image sensor together with a 4-piece lens system. Light originates from Light Emitting Diodes (LEDs) placed in the tracking environment. These LEDs are controlled by a microcontroller one at a time. The LED light cones need to be overlapping, in order to get at least two intensity measurements by different light sources. Light rays travel through the lens system, which focuses light on the sensor. When the sensor moves, the angle of the refracted light changes, thus reaching a different part of the sensor. A microcontroller can then compute the tracking system's change of position.
The project can be split into two parts: The first part consists of a tracking system simulation. The simulation makes use of Ray Transfer Matrix Analysis, a ray tracing technique utilized in the design of optical systems. Using the simulation, the size of the tracking area has been calculated and the right lenses with correct distances to each other have been chosen. The second part consists of the actual implementation. A functional prototype has been developed, to turn the insights of the simulation into a real-life solution. A microcontroller is used to read the linear image sensor and control the LEDs. The intensity values are sent to a Raspberry Pi, which calculates the position change of the tracker. With only one linear image sensor, the system is able to track translations in two dimensions. However, through adding an additional sensor with corresponding LEDs, tracking of a third axis is possible in theory. Finally, the accuracy and latency of the tracking system prototype have been evaluated.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Tracking systems are a widely used technology in physical computing applications. Many interactive systems need to be aware of their position in space to perform meaningful computations and return useful results. The size of tracking systems ranges from small scale, room sized environments, to environments of global scale. Probably the most known and widely used tracking system operating globally is the Global Positioning System (GPS) [1]. It is responsible for the functioning of turn-by-turn navigation in cars, geolocation in smartphones and flight tracking, to name only a few [2]. Areas, where indoor tracking systems have proven themselves beneficial are robotic systems [3], motion capturing systems for movie or game animations [4] and Virtual Reality (VR) / Augmented Reality (AR) applications. There are also examples of very specialized applications of tracking systems: In [5], a laser-based tracking system for the construction industry is presented.

Especially in the field of VR / AR, tracking systems play a key role. Precise measurements and low latency data acquistion are crucial to prevent unwanted side effects such as motion sickness. In recent years, VR / AR headsets became commercially available and had a rise in popularity. This rise in popularity is mainly due to the fact that graphics hardware is getting steadily faster and the ambition of *Oculus* and *HTC* to release consumer VR products to the market. HTC has presented the *Lighthouse* tracking environment for their *Vive* head-mounted display [6], based on infrared technology. With this system it is possible not only to look, but also to move around in a virtual world with the size of about $5\,\mathrm{m} \times 5\,\mathrm{m}$. Although Optitrack is a general purpose tracking system, specific hardware for VR tracking is also available [7]. Optitrack claims high fidelity at low latencies and motion capturing of large areas.

However, these aforementioned tracking systems are proprietary technologies. None of its inner workings have been disclosed to the public. Furthermore, the Optitrack system

comes with a huge price tag only big companies or large research institutions can afford. Although the HTC Vive tracking system is more affordable, it cannot be used in every environment. Particularly when exposed to direct sunlight, the system seems to have problems to keep tracking accurate [8].

1.2 Problem Statement

Tracking systems are complex technologies and relatively expensive in purchase. Not all of them are working under every lighting condition. The HTC Vive, for example, has problems working in outdoor environments, directly exposed to sunlight [8]. Moreover, not every tracking systems tracking area can be extended easily. At the day of this writing, the tracking space of the HTC Vive is limited to room-sized environments. Especially in outdoor VR tracking, there is still room for improvement: No tracking system can be found which is cheap in purchase, easily expandable and working under outdoor conditions, but still giving good tracking accuracy and latency. Tracking systems often depend on specialized hardware, which can only be bought as proprietary components.

1.3 Aim of the Work

The aim of this thesis is to develop a new kind of tracking system, based on linear image sensors (line scan cameras) as light sensing devices. Linear image sensors are readily available and affordable hardware. They have high resolution in one dimension and can be read out very fast. The tracking system is based on optical data to gather movement information: Light rays originate from Light Emitting Diodes (LEDs) placed in the tracking environment. The light cones of the LEDs need to be overlapping, in order to get at least two measurements of different light sources on the sensor. The LEDs, as well as the linear image sensor which provides a one-dimensional image, are read and controlled by a microcontroller. The light rays travel through a lens system which is directly located before the linear image sensor, converging the light on the sensor. When the sensor moves, the angle of the refracted light changes, thus reaching a different part of the sensor. With two measurements of different light sources, an algorithm can calculate the change of distance to the LEDs and parallel to them, resulting in a two-dimensional tracking space. Figure 1.1 shows a schematic overview of the functional principles.

The functioning principles of the system are not overly complex and depend on commercial off-the-shelf components. Due to the small amount of sensor data aggregation, an embedded system has enough processing power to compute the tracking output. By adding LEDs, the system can be extended easily.

The main part of this work describes the development of a prototype of the previously suggested tracking system. In order to focus light on the sensor, optical lenses have to be arranged in the right order and with the correct distances to each other. Thus, a light ray simulation has been done prior to the actual prototype implementation, to calculate the proper lens characteristics. Afterwards, the insights of the simulation have been

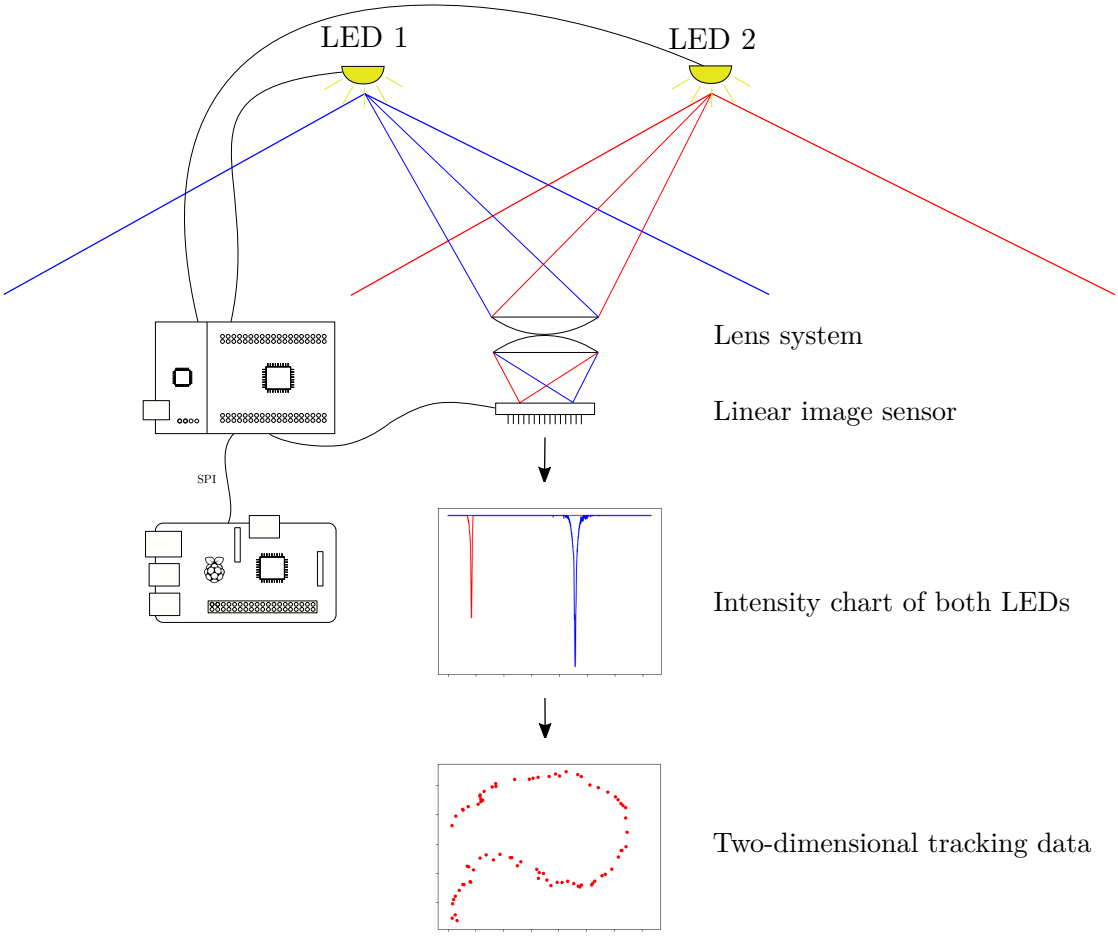


Figure 1.1: Overview of the functional principle of the tracking system.

turned into a functional prototype. A STM32 microcontroller is used to read out the linear image sensor and control the LEDs. Finally, the characteristics of the tracking system have been evaluated.

1.4 Outline

This work is split into six chapters. Chapter 1 gives an introduction to the topic and states the goal of the project. Chapter 2 gives an overview of related work and explains concepts similar to the system presented in the thesis. Chapter 3 explains the theoretical background which is necessary to understand the underlying principles of the tracking system. In Chapter 4, the simulation, implementation and prototyping stages are described. Chapter 5 examines the results of the developed prototype with regard to the systems accuracy and latency. Finally, Chapter 6 summarizes the most important aspects of this work, gives an overview of unsolved problems and leaves room for improvements in the future.

CHAPTER 2

Related Work

2.1 Tracking Systems

Position data produced by tracking systems is crucial information for physical computing applications. Many interactive systems like robotic applications and VR / AR headsets need to be aware of their position in space to perform meaningful computations. There is a wide variety of different tracking systems, each one is made for a different purpose. Tracking methods can range from large area, global systems to small scale, room sized environments.

2.2 Taxonomy of Tracking Systems

Ishii [9] divides tracking systems into inside-out and outside-in methods. The difference between these two methods lies in the location of the tracking sensor. In inside-out tracking, the sensor is attached to the tracked object. Distinctive features like fiducial markers or other active or passive markers have to be placed in the surroundings of the tracking space. In outside-in tracking, on the other hand, the sensors are placed around the tracking environment. The tracked object has to be equipped with markers to be detected by the sensors. Figure 2.1 depicts the difference between inside-out and outside-in tracking methods.

2.3 Satellite Tracking Systems

The most prominent example of a global tracking system is GPS. There are also other systems like the russian GLONASS, the chinese BEIDOU and GALILEO, operated by the european union. *Hofmann-Wellenhof et al.* [11] give an overview of global tracking systems.

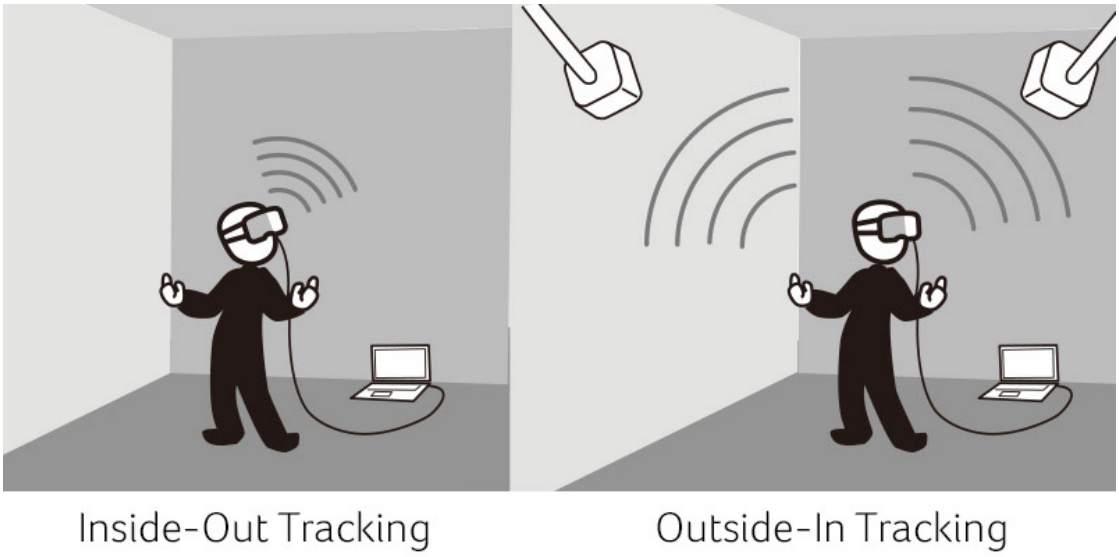


Figure 2.1: Inside-out vs. outside-in tracking systems (retrieved from [10]).

GPS calculates the user's position by estimating the distance of multiple satellites. At least four signals from different satellites are necessary to calculate a three-dimensional pose. Satellites send out their current time and position. The tracker calculates his own position from the received information and by measuring the signal transit times. GPS is developed and operated by the US Department of Defense.

2.4 Inertial Sensor Tracking

Inertial sensor tracking utilizes gyroscope and acceleration sensors to provide position information [9]. No outside markers have to be deployed for this kind of tracking method, which makes it very convenient. This alleged advantage is also its biggest disadvantage: Due to the fact that no fixed markers are deployed, the system inevitably starts to drift. Tracking errors accumulate over time which decreases the tracking accuracy. Without any base stations to reset the accumulated drift, inertial sensor tracking systems are not suited for extensive tracking sessions. Another drawback of inertial based tracking systems is that only relative tracking data can be obtained. To get absolute tracking positions, some additional method must be deployed.

Foxlin presents in [12] an invention called *NavShoe*, a tracking system for pedestrians based on inertial sensing. The system doesn't depend on external markers or installations. It can be used to find and rescue members of emergency teams or as a personal navigation assistant. The NavShoe system uses a tiny inertial / magnetometer bundle package which can be attached to the user's shoes. The system can then be wirelessly coupled to a personal digital assistant to visualize the navigation data. To reduce drift, the tracking system applies zero-velocity updates to the Extended Kalman Filter (EKF)

error corrector during the stance phase of a walking person. The EKF can then correct the velocity error to be linear in the number of steps. To provide more accurate data, the tracking system can be supported by information from satellite positioning systems.

Inertial sensors are frequently used in sensor-fusion systems together with visual tracking algorithms. In such systems, the inertial sensors are responsible for capturing fast movement data, while the vision-based algorithms provide absolute reference measurements to compensate for the drift accumulating in the inertial sensors [13, 14].

2.5 Ultrasonic-based Tracking Systems

According to *Ishii* [9], ultrasonic tracking systems are accurate and stable. However, ultrasonic sensors and sources have to be deployed in the tracking environment. It is possible to obtain a full 6 Degree-of-Freedom (DOF) pose, consisting of position and rotation data. Ultrasonic sensors don't cover a high range, therefore a lot of sensors have to be used to cover a tracking area of reasonable size. The drawbacks of such a system are unwanted reflections and costs: Ultrasonic equipment is expensive. Also, tracking accuracy will decrease in complex environments, because of echoes caused by reflecting surfaces.

Foxlin1998 et al. present in [15] a system called *Constellation*, an inside-out tracking system based on inertial sensors and ultrasonic measurements. Ultrasonic beacons have to be placed in the tracking environment. In the author's test arrangement, the beacons were placed on the ceiling. Sensor fusion is used to filter out corrupt measurements and perform predictions for the ultrasonic system, while at the same time the ultrasonic tracker corrects the inertial drift. The system claims to achieve resolutions of 1 mm to 3 mm and an effective latency of 2.5 ms.

2.6 Vision-based Tracking Systems

Vision-based tracking systems use cameras as their main sensors [9]. Image processing algorithms try to detect distinct features on the camera images. Distinct features are distinguishable image characteristics like corners, edges or planes. The aim is to infer the change of the camera's position and orientation from the recognized features. This is done by solving the perspective n-point problem.

Vision-based tracking systems can be classified into two categories:

- Tracking based on natural features [16]: Pose prediction is solely done on the visual input and the detected image features thereof. Neither markers have to be placed in the tracking space nor is prior knowledge of the scene necessary. These systems are referred to as visual odometry. Visual odometry provides only information on relative movements. The tracking motions can be fused with other systems (inertial sensors, GPS) to provide better accuracy. Camera tracking is only stable as long

as the environment remains unchanged. If the environment starts changing, the features change their position too, leading to an unstable tracking performance. Several visual odometry algorithms have been proposed, focussing on different techniques for feature extraction and pose estimation [17, 18].

- Tracking based on markers [9]: Marker-based tracking systems detect features on specifically designed markers. Fiducial markers consist of primitive shapes that can be easily recognized, such as circles and squares. The dimensions and positions of these shapes have to be known in advance. The markers have to be placed in the environment to ensure stable tracking. In [19], a marker-based tracking system for AR conferencing applications is presented.

Simultaneous Localization and Mapping (SLAM) techniques build a map at the same time as localization happens [20]. Through sophisticated methods such as loop closure, the accuracy of the built map can be enhanced drastically.

Various Visual SLAM methods, operating on different features, have been proposed. Parallel Tracking and Mapping (PTAM) [21], for example, uses Features from Accelerated Segment Test (FAST) corners for feature detection.

There are also outside-in tracking systems using more than one camera: In [22], an infrared-optical tracking system is presented. The system consists of 4 - 8 shutter-synchronized cameras. Each camera has an optical filter and an infrared illuminator built in. The system tracks targets consisting of passive spheres, which reflect infrared light. The spheres distances to each other are known by the system. The tracking system provides 6-DOF tracking with a latency of 20 ms to 40 ms and an absolute accuracy of about 0.5 cm.

Ribo et al. [23] present a stereo vision tracking method for VR / AR applications. The system is based on two cameras, tracking retroreflective targets which can be attached to any device. By using epipolar geometry, the system computes a 6-DOF pose from the 2D image points. 25 targets positions can be tracked at a rate of 30 Hz.

2.7 Linear Image Sensor Tracking

In [24], an outside-in motion capture system based on linear image sensors has been proposed. In this system, light point devices have a unique identifier by providing a unique sequence of light pulses. The light point devices are in sync over a wireless communication channel. Two linear image sensor devices are placed in a room, orthogonal to each other. Each sensor delivers one-dimensional information. A processing device computes three-dimensional positions of the light point devices, based on the one-dimensional sensor information. The motion capture system is comprised of a four-piece cylindrical lens system, aimed to focus light on the sensor. The lens system is divided into a collimating and telecentric subsystem. The aim of the collimating lens system is to get as much light as possible through the telecentric system and on the sensor. However, the collimating

subsystem enhances only the range of motion of the non-tracked axis. The telecentric system converges the light on the sensor, thus providing the actual tracking information. It is aligned perpendicular to the collimating subsystem.

In [25] and [26], *CoSTEL* is presented, a 3-dimensional outside-in motion capturing system, conceived for the analysis of whole body movements. The system consists of three linear image senors with 2048 pixels, each one is equipped with a cylindrical lens focussing light on the sensor. The linear image sensors lie on the focal planes of their corresponding lenses. The tracked person is equipped with active infra-red light emitting markers, casting light through the lenses and on the linear image sensors. The markers are toggled one after each other, sychronized with the operating system. The positions of the markers can then be calculated. 20 markers can be tracked at a frequency of 125 Hz and a spatial resolution of 0.025 %. This would yield a resolution of 1 mm in a 4 m wide tracking area. Figure 2.2 illustrates the constellation of the three linear image sensors.

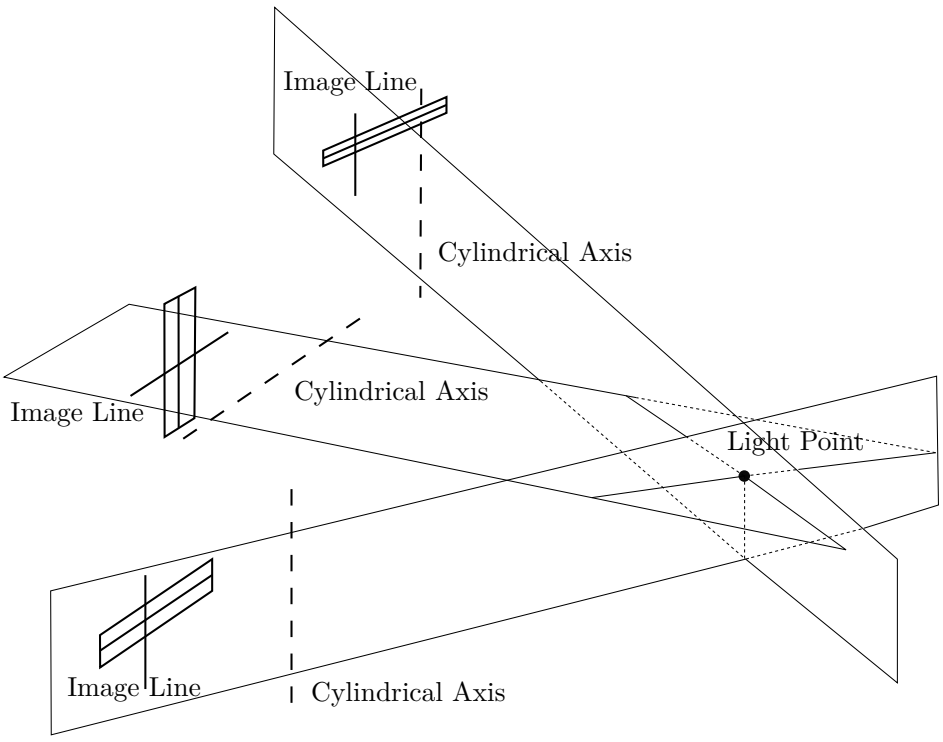


Figure 2.2: The tracking method of CoSTEL, utilizing three linear image sensors. The annotated cylindrical axis is the axis of the cylindrical lenses mounted in front of the linear image sensors. The cylindrical lens converges light to a line on the linear image sensor, marked as "Image Line" in the sketch (retrieved from [25]).

Bucholz et al. [27] present a tracking system for computer aided medical devices, similar to the CoSTEL system. It depends on three linear image sensors and infrared LEDs. A base ring is attached to the head of the patient during surgery. LEDs, fixed to the base ring and the hand-held instruments, are used as active markers for the outside-in tracking system. The position of the instruments are then visualized on three dimensional computer tomography, positron emission tomography or magnetic resonance imaging scans, relative to the head's position. The tracking system has an error of less than 1.5 mm.

In the patent filed by [28], a system for tracking whole-body movements is presented. Taking a similar approach to CoSTEL, the system consists of three linear image sensors and LED markers. The tracked person is equipped with LED markers, which are lit sequentially, synchronized with a controller. Therefore, only one infrared marker is lit during an image capturing process. Using triangulation, the absolute position of each marker can be calculated.

Bishop [29] presents *SELF-TRACKER* in his dissertation: An inside-out tracking system based on custom integrated, one-dimensional sensors, aimed at room-sized tracking of head-mounted displays. The sensors operate at rates of 1000 - 4000 frames per second. They operate under the assumption of small changes in the image when the frame rate is high. Therefore, small deviations can be detected by a simple algorithm. The sensors are fitted in a cluster, the minimum size of the cluster is given by the focal length of the lenses. Lenses and sensors are mounted on opposite sides of the cluster box. The tracking system can be used with artificial markers like blinking LEDs or reflective material. A natural feature based tracking approach is also described, which eliminates the use of artifical markers. In this approach, natural intensities of the room are captured by the linear image sensors and used to measure the user's motion. This approach doesn't measure absolute positions, thus an error accumulates over time, resulting in drift. The author is also one of the inventors of the *HiBall* tracker, a tracking system utilizing lateral effect photodiodes [30].

Fuchs et al. [31] present a lensless outside-in tracking system based on one-dimensional optical sensors. A number of linear image sensors have to be placed in a room, to detect the position of a light source. The sensors are partially shadowed by a "knife edge" covering parts of the sensors. When the user moves the light source, the shadowed and the illuminated region change. The change of the illuminated region can be detected by a linear image sensor. When multiple sensors are illuminated, the light source's three-dimensional position can be calculated.

In [32], a method for a lane keeping driving assistant using linear image sensors is described. The linear image sensor and an infrared flash are mounted on a car. On the side of the road, periodically spaced reflectors are used as road marks. The flash illuminates the reflectors every 10 ms while the linear image sensor captures an image. A lens system is attached in front of the linear image sensor, to focus the three-dimensional world intensities to a single line the sensor can read. The system is capable of recognizing

reflectors up to a distance of 40 m. Using an EKF and the information from the linear image sensor, the longitudinal axis of the vehicle can be measured.

Wu & Wen [33] introduce a positioning system for computer aided surgery systems based on linear image sensors. Similar to the CoSTEL system, it is based on three linear image sensors with corresponding cylindrical lenses. Two of the linear image sensors are oriented horizontally, the third one vertically. To be independent of computer processing power, the algorithm is implemented on Digital Signal Processors (DSP). The results are compared with results calculated by computer hardware and showed no difference within 0.01 mm.

Hu et al. [34] present an attitude measurement system for flying objects. The system consists of two Charge-Coupled Device (CCD) arrays and three linear image sensors. The tracked object has six LEDs attached in a triangular shape. Three of the six lights emit blue light, while the other three emit red light. The linear image sensors have a telecentric lens system and red filters attached, to filter out blue LEDs. The two CCD array sensors perform binocular vision tracking, supported by the linear image sensors. The linear image sensors improve the systems attitude measurements as well as speed and precision.

2.8 Other Systems depending on Linear Image Sensors

In [35], different types of Coordinate Measuring Machines (CMM) are examined. These machines perform dimensional inspection tasks, measuring the dimensions of various objects. When measuring large objects, mobile CMMs are an ideal solution. Mobile CMMs are aligned to the reference points on the objects, giving exact object measurements. Existing mobile CMM solutions use linear image sensors, to measure three angles which uniquely define the position of a point in space. Each linear image sensor measures one plane containing the point to be measured.

Linear image sensors are also used in the design of digital spectrometers, measuring spectral components of physical phenomena. *Dominec* [36] built a spectrometer using a *NEC μPD3799* linear image sensor, in which the light components are separated by a lens system and a diffraction grating component. The light spectrum is then recorded by the linear image sensor.

2.9 Commercially available Tracking Systems

The *HTC Vive* is a well-known, commercially available VR system. In [37], the output of the Vive's position and orientation data is examined, as well as its system latency measured. Although the HTC Vive has high precision tracking measurements and a low system latency of around 22 ms, it has a tilted coordinate system with respect to the physical ground plane. Therefore, the authors claim that the HTC Vive VR system

is currently not suited for scientific experiments, where accurate visual stimulation is important.

CHAPTER 3

Theoretical Background

The aim of this chapter is to give the reader an overview of relevant topics needed for understanding the concepts of the thesis.

3.1 Optics

Optics is a part of physics, investigating the theories behind light and vision [38]. Putting it in more broad terms, the field of optics studies the phenomena behind electromagnetic radiation in the optical spectrum, from the generation, to the transmission and detection of electromagnetic waves. Wavelengths range from one nanometer to approximately one millimeter in the optical spectrum.

At the beginning of modern science in the 16th and 17th centuries, light has been seen either as a particle or a wave [39]: In the phenomena of interference, diffraction and propagation, light behaves like a wave. However, when light exchanges energy with matter, it shows a particle-like behavior. Each of these seemingly incompatible models enjoyed a phase of popularity in the scientific community. To solve this contradictory state of light, referred to as the wave-particle duality, the theory of quantum electrodynamics was invented. Different fields of physics like electricity, magnetism and atomic physics all come together in this unified theory. One of the results of quantum electrodynamics is that light and subatomic particles are both considered as forms of energy, governed by the same principles.

3.2 Geometrical Optics

In geometrical optics, the wave characteristics of light are ignored, because the wavelength is considered negligibly small compared to the size of optical components. Therefore, phenomena like the diffraction of light around the edges of barriers are not considered in

geometrical optics. Within this approximation, light disperses along straight lines, which are also referred to as rays. A light ray travels through an optical system comprised of different homogeneous media along a sequence of straight lines. This section is based on [39] and [40].

There are two laws guiding the direction of the ray:

- **Law of Reflection**: When a light ray, travelling through a medium, is reflected at the interface to another medium, the angle of incidence θ_1 is equal to the angle of the reflection θ_r.

- **Law of Refraction** (Snell's Law): When a light ray, travelling through a medium, is refracted at another medium, the sine of the angle of incidence θ_1 is directly proportional to the sine of the angle of the refraction θ_2:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{3.1}$$

  n_1 and n_2 are the refractive indices of two different media.

Figure 3.1 depicts the reflection and refraction of light rays between different media.

There are two important principles conveying the movement of light:

- **Huygens' Principle**: Christian Huygens envisioned light as a series of waves. Every point on a wavefront propagating light in space is the source of secondary spherical wavelets. These wavelets themselves form the envelope of another wavefront at some later instant in time. The wavelets have the same speed and frequency as the wavefront they are originating from. Augustin Fresnel modified Huygens' principle to add the concept of interference, explaining the diffraction of light at obstacles or slits. Figure 3.2 shows the principle of wavefront propagation.

- **Fermat's Principle**: Fermat's principle is based on the idea of Hero of Alexandria, who stated that light always takes the shortest path between two points. Fermat generalized Hero's idea to provide an explanation for the refraction of light between different media. Fermat's principle claims that light travels the path of least time instead of least distance. When light would take the shortest path, the angle of incidence wouldn't change between different media, yielding a straight line and violating Snell's law of refraction. However, according to Fermat's principle, light tries to minimize the time it travels in a slower medium, therefore bending at the interface to follow an optimal path time-wise.

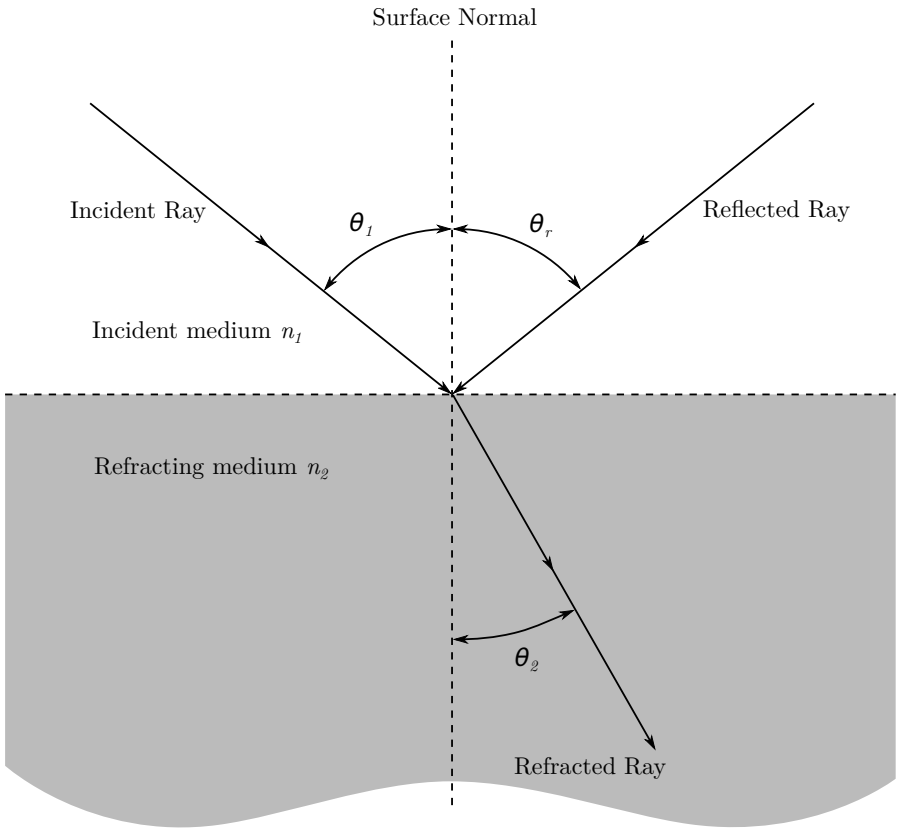


Figure 3.1: The reflection and refraction of light rays at the interface of different media. θ_1 is the angle of the incident ray, θ_r is the angle of the reflected ray and θ_2 is the angle of the refracted ray (retrieved from [39]).

3.2.1 Optical Systems

An optical system consists of different refracting and/or reflecting media, with surfaces of arbitrary curvature. Each medium in the optical system is assumed to be homogeneous and isotropic, therefore it can be described by its own refractive index. Light rays leaving a source point O are altered by the optical system and arrive in image space, converging to an image point I. Figure 3.3 shows a sketch of an optical system, consisting of light rays with their corresponding wavefronts emerging at object space and arriving at image space.

According to Fermat's principle, every ray starting at O and ending at I has the same transit time through the system. Light rays that have these property are called *isochronous*. By the principle of reversibility, exchanging the light source with the image point I would

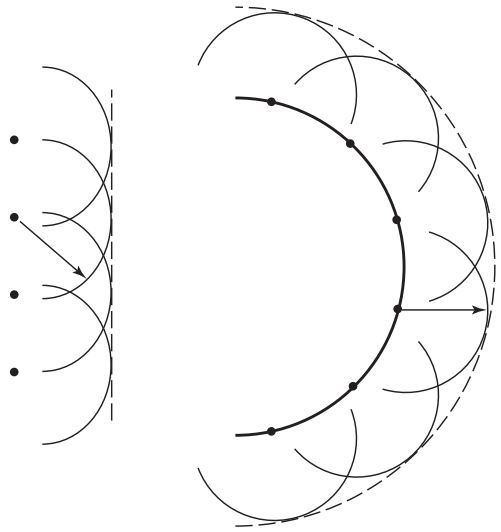


Figure 3.2: Huygens principle of wavefront propagation. The propagation of plane waves (a) and spherical waves (b) can be seen (retrieved from [39]).

reverse all rays and result in a new image point O, the previous light source.

Non-perfect images arise mainly because of three reasons:

- **Light Scattering or Reflections**: Light scattering can occur due to inhomogeneities in transparent media. A loss of light rays because of reflections on refracting surfaces leads to a decreased brightness of the image.

- **Diffraction**: Diffraction effects which arise from the wave nature of light limit the sharpness of the image. The effect occurs because only a fraction of the wavefront goes through the optical system. Only in a perfect geometrical optics scenario with wave amplitudes approaching zero, diffraction effects can be avoided. This scenario, however, can never be entirely reached.

- **Aberrations**: Aberrations occur due to imperfect optical components. If not every light ray that emerges at a point light source and travels through an optical system, reaches the same image point in image space, the resulting deviation is called spherical aberration. In contrast, if all light rays reach the exact same point in image space, the refracting or reflecting surfaces of the optical system are called cartesian surfaces. However, cartesian surfaces need a hyperboloid shape and are hard to produce. Therefore, most optical surfaces are spherical and the introduced

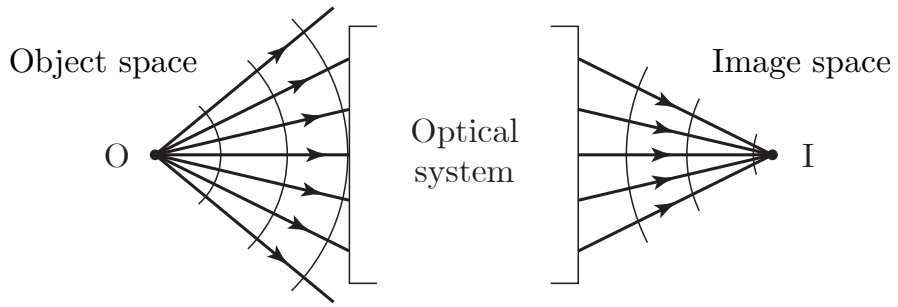


Figure 3.3: A typical optical system, with the object source O lying in object space. The light rays emerge from the object source O, travel through the optical system and arrive in image space at the point I. The wavefronts of the light rays can be seen as well (retrieved from [39]).

spherical aberrations are accepted, as a compromise between optical quality and ease of production.

3.2.2 Cylindrical Lenses

Cylindrical lenses, although not as common as spherical lenses, are important parts of optical systems. Cylindrical lenses are used for correcting astigmatism, a visual defect, and for cases where points need to be imaged as lines. A spherical lens can be rotated around its axis of symmetry without changing the resulting image. In contrast, a cylindrical lens has asymmetric focusing characteristics, an axis of symmetry doesn't exist. Another difference between these two lens types is the type of mapping they produce. While a spherical lens produces a point image of a point source, a cylindrical lens refracts a point source to a line image. This property of a cylindrical lens is called astigmatism.

Figure 3.4 shows a variety of convex and concave cylindrical lenses of different types and sizes. In Figure 3.5, the focusing characteristics of a convex cylindrical lens are shown, when light rays are emitted from a light source at infinity. When a light source is close to the lens, the cylindrical lens converges light only in a single dimension: Vertical rays are refracted by the lens, horizontal rays are left untouched, with respect to this image.

3.3 Ray Transfer Matrix Analysis

Ray transfer matrix analysis is a ray tracing technique, mainly used in the design and analysis of optical systems [42]. It consists of ray transfer matrices, which describe the optical system. Each matrix is comprised of a single lens property. Ray transfer matrices can be multiplied together, yielding a single matrix for a whole optical system. A light

Figure 3.4: Different types of convex and concave cylindrical lenses (retrieved from [41]).

ray can then be traced through the system by multiplying the resulting matrix by a vector representing the light ray. This section is based on [42] and [40].

The results of the ray transfer matrix analysis are only valid within the limits of two approximations:

- The first one is the underlying assumption of all geometric optics: Wave properties of light are ignored (see section 3.2).
- Secondly, ray transfer matrix analysis is making use of paraxial optics. Paraxial optics only considers rays which are close to the optical axis and remain nearly parallel to it. Thus, first-order approximations are used for the sines and tangents of the involved angles. Using this approximation, optical effects like spherical aberrations, field curvature and distortion will be ignored.

In mathematical terms, paraxial optics can be written als follows:

$$\sin\theta \approx \theta, \tan\theta \approx \theta \text{ and } \cos\theta \approx 1 \tag{3.2}$$

In paraxial optics, there are three pairs of cardinal points determining the behavior of an optical system: focal points, principal points and nodal points. If the initial image

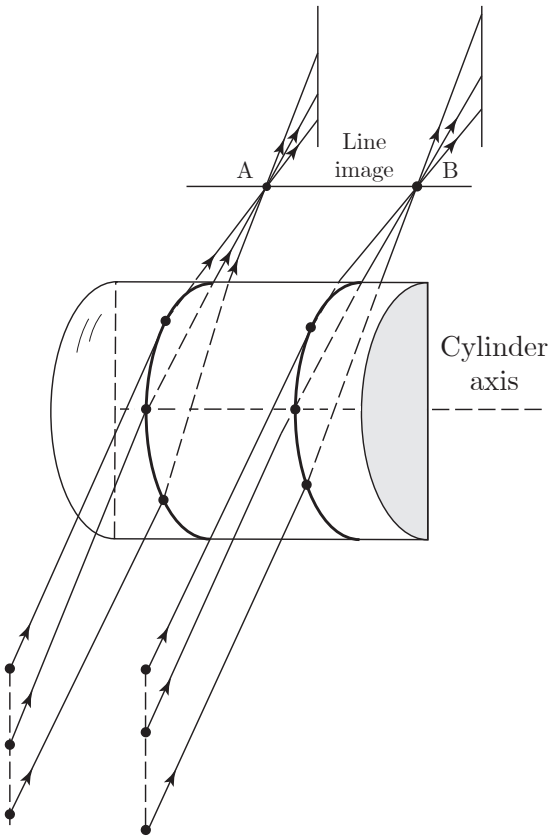


Figure 3.5: The focusing characteristics of a convex cylindrical lens. Light rays emerging from a section normal to the cylinder axis are converged to a common focus point, while rays from a section parallel to the cylinder axis are not. Parallel rays converge to a line image parallel to the cylinder axis of the cylindrical lens (retrieved from [39]).

location and the six cardinal points are known, the final image can be calculated for any system of refracting spherical surfaces.

3.3.1 Thin Lens and Thick Lens Equations

A thin lens can be seen as a single refracting surface seperating two spaces with the same refracting index. The thickness of a thin lens is negligible small compared to the curvature radii of the lens surfaces. Ignoring the thickness will not lead to serious errors. This section is based on [39] and [40].

The thin lens is defined by the lensmaker's equation:

$$\frac{1}{f} = \frac{n_2 - n_1}{n_1}\left(\frac{1}{R_1} - \frac{1}{R_2}\right) \tag{3.3}$$

In this equation, f is the focal length, that is the distance where perfectly parallel rays are brought into focus. The longer the focal length, the higher the lens' magnification. The focal length f is negative for diverging lenses and positive for converging ones. R_1 and R_2 are the corresponding curvature radii of the lens, n_2 and n_1 are the refractive indices of the lens material and its surrounding medium, respectively. The lens' curvature radii indicate whether the corresponding surfaces are concave or convex. While $R_1 > 0$ and $R_2 < 0$ indicate convex lens surfaces, $R_1 < 0$ and $R_2 > 0$ indicate concave lens surfaces. In most cases, the surrounding medium is air, thus $n_1 = 1$.

A thick lens, on the other hand, is a lens where the thickness cannot be ignored without producing significant errors.

The lensmaker's equation for a thick lens is defined as follows:

$$\frac{1}{f} = (n_2 - 1)\left(\frac{1}{R_1} - \frac{1}{R_2} + \frac{(n_2 - 1)\, d_t}{n_2 R_1 R_2}\right) \tag{3.4}$$

In this formula, it is assumed that the surrounding medium is air, therefore, n_1 is set to 1. All other variables have the same meaning as in the thin lens equation (see 3.3). Additionally, with the variable d_t the thickness of the lens is taken into account.

Further, the distance from the object to the lens s_o and the distance from the image to the lens s_i are connected with the focal length f via the following relation:

$$\frac{1}{s_o} + \frac{1}{s_i} = \frac{1}{f} \tag{3.5}$$

Figure 3.6 shows light rays traveling through a convex lens. The relations between the different lens variables can be seen.

3.3.2 Ray Transfer Matrices

The elegance of the ray transfer matrix analysis lies in the simplicity of its notation. An entire optical system can be represented by a simple 2 x 2 matrix. This section is based on [39].

A light ray travelling through a system of lenses can be expressed by a two-dimensional vector:

$$V = \begin{bmatrix} y \\ \alpha \end{bmatrix} \tag{3.6}$$

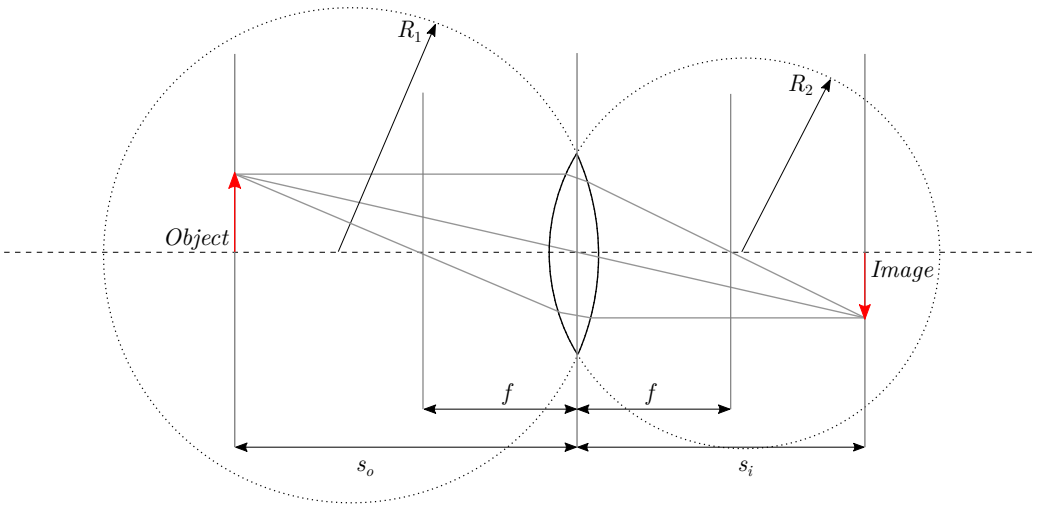


Figure 3.6: A convex lens with light rays traveling through it, showing how the variables of different lens equations relate to each other.

In this vector, y represents the distance of the ray from the optical axis. α is the angle the light ray forms with the optical axis.

A light ray can be translated in 2D by the translation matrix T:

$$T = \begin{bmatrix} 1 & L \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

In this matrix, L is the translation distance on the optical axis. Furthermore, the paraxial approximation $\tan\alpha \cong \alpha$ is used in the translation matrix.

The refraction matrix gives the result of a ray refracting at a spherical interface. The resulting vector is the angle and distance from the optical axis directly after the refraction. The matrix incorporates the paraxial form of Snell's law:

$$n_1\theta_1 = n_2\theta_2 \tag{3.8}$$

n_1 and n_2 are the refractive indices of the incident and refracting media, respectively. θ_1 and θ_2 are the angles of the incoming and outgoing rays with the normal vector of the hitting point. The refraction matrix S is defined as follows:

$$S = \begin{bmatrix} 1 & 0 \\ \frac{n_1 - n_2}{Rn_2} & \frac{n_1}{n_2} \end{bmatrix} \tag{3.9}$$

n_1 and n_2 are the refractive indices of the surrounding medium and the lens, respectively. R is the radius of the refracting surface.

The equations for thin and thick lenses result from multiplications of the previously described matrices. A thick lens can be approximated by using a combination of refraction and translation matrices:

$$E = S_2 \cdot T \cdot S_1 \tag{3.10}$$

The first matrix S_1 represents the refraction of the ray first hitting the lens. The second matrix T corresponds to a translation of the ray within the lens. The third matrix S_2 represents the outgoing ray refraction. The thin lens equation is a special case of the thick lens equation, where the length of the translation matrix T has a value of 0:

$$F = \begin{bmatrix} 1 & 0 \\ \frac{n_2-n_1}{R_2 n_1} & \frac{n_2}{n_1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{n_1-n_2}{R_1 n_2} & \frac{n_1}{n_2} \end{bmatrix} \tag{3.11}$$

This equation can be reduced to the following matrix:

$$F = \begin{bmatrix} 1 & 0 \\ \frac{n_2-n_1}{n_1}\left(\frac{1}{R_2} - \frac{1}{R_1}\right) & 1 \end{bmatrix} \tag{3.12}$$

The element in the first column and second row of the matrix equals the lensmaker's equation (see 3.3). Thus, the thin lens ray transfer matrix can be simply written as:

$$F = \begin{bmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{bmatrix} \tag{3.13}$$

3.4 Linear Image Sensors

Linear image sensors, sometimes also referred to as line scan cameras, linear CCDs or linear pixel arrays, are linear arrays of light detectors [43]. Compared to a two-dimensional image sensor, linear image sensors are comprised of only one dimension of pixels, i.e. one row of a two-dimensional sensor. Linear image sensors are used in applications where either the sensor or the scanned object is moved perpendicular to the pixel line of the sensor. Rigid control has to be maintained on the moved objects. Typical applications of linear image sensors are image scanners, barcode scanners or cash machines. A linear image sensor is depicted in Figure 3.7. This Section gives an overview of photosensitive technology and common issues of it. The linear image sensor utilized in the implementation, the *Toshiba TCD1304*, is described in more detail.

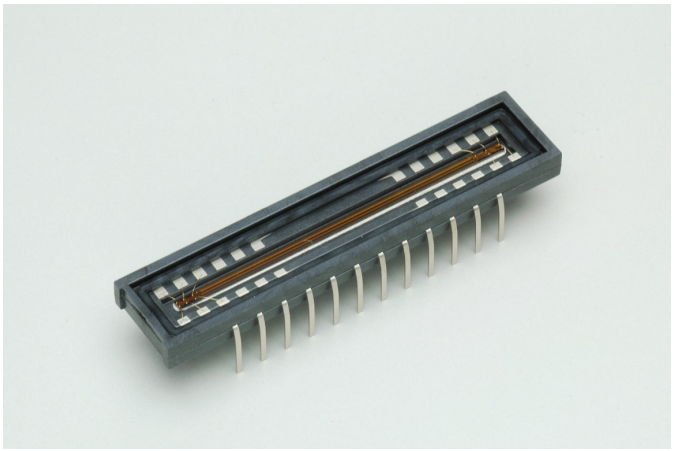


Figure 3.7: A linear image sensor (retrieved from [44]).

3.4.1 Charge-Coupled Device (CCD) vs. Complementary Metal-Oxide Semiconductor (CMOS)

CCD and CMOS image sensors convert light into charge by using the photoelectric effect [45]. Both technologies collect and measure electrical charge and turn it into an analog signal proportional to the amount of light they receive. The difference between these technologies lies in the way charge is collected and carried to the output.

The CCD is a semiconductor, transferring charges through storage areas. It was invented by Willard Boyle and George Smith at Bell Labs in October 1969 [45]. Boyle and Smith received the 2009 Nobel Prize in Physics for the invention of the CCD technology. A CCD combines the three tasks of collecting, transferring and converting charge to a measurable voltage [43]. Each pixel consists of a Metal-Oxide Semiconductor (MOS) capacitor with an electrode attached to its silicon dioxide layer [46]. To measure the intensity values of all pixels, charges are shifted like buckets sequentially to the output of the CCD, where they are converted to voltage.

In CMOS devices, on the other hand, each pixels converts its charge to a voltage directly [45]. The pixel outputs the voltage signal when it is selected by row and column busses. The CMOS technology was invented by Frank Wanlass in 1963.

Until lately, CMOS technology could not rival CCD technology in terms of image quality [45]. However, due to recent improvements in CMOS manufacturing like lithography and process control, CMOS sensors are able to compete with CCD sensors qualitywise. CMOS sensors are found increasingly in Digital Single Lens Reflex (DSLR) cameras and professional camcorders. Smartphones are one of the main drivers behind increasing sales of CMOS sensors lately [47].

3.4.2 Rolling Shutter Effect

The rolling shutter effect characterizes image errors in exposures of moving subjects [45]. The errors occur due to the way camera sensors are exposed and read out. In CMOS sensors, the pixel rows are read out with a small delay after each other. Because of this delay, the moving object has time to move further, inducing a shear effect in the resulting image. While CMOS sensors suffer from the rolling shutter effect, CCD sensors are not affected, because all pixels are exposed at the same time in CCD technology.

3.4.3 Dark Current

Dark currents are defined as unwanted charges accumulating in the pixel arrays of a CCD sensor [45]. Dark currents emerge because of thermal processes occuring at any temperature above absolute zero. Especially during longer integration times, which are used in low light situations, dark current leakage can arise [43]. Since dark currents are temperature dependent, they can be reduced by cooling the CCD sensor.

3.4.4 The TCD1304 Sensor

The TCD1304 is a linear CCD sensor with 3648 pixels [48]. The sensor weighs 2.7 g and has a dimension of 10 mm × 42 mm, while each pixel of the sensor has a size of 8 µm × 200 µm. It can be driven at a maximum clock rate of 4 MHz. However, each pixel needs 4 clock cycles to be read out. This means that a full readout of the sensor takes 14 592 Hz ≈ 0.0036 ms. The TCD1304 sensor is suited for light within the visible spectrum (380 nm - 750 nm). Figure 3.8 shows the spectral response of the sensor. This section is based on [48] and [49].

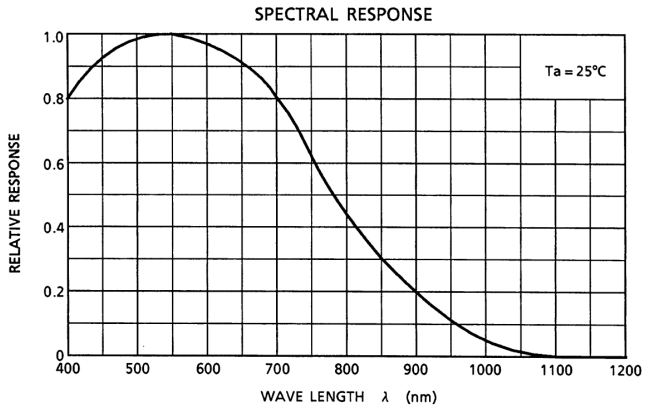


Figure 3.8: The spectral response of the TCD1304 linear image sensor (retrieved from [48]).

The sensor needs three driving pulses:

- The **Master Clock (fM)**, it must run at a speed of 0.8 MHz - 4 MHz.
- The **Shift Gate (SH)** controls the integration time. The minimum integration time is 10 µs.
- The **Integration Clear Gate (ICG)** controls the readout time of the sensor.

In Figure 3.10, a timing chart of the TCD1304 is shown. In this chart, it can be seen that the ICG and SH pulse must coincide in order to read pixel values from the sensor. Between two ICG pulses, pixel values can then be digitized by an external ADC. If the SH pulse has a shorter period than the ICG pulse, the sensor is in electronic shutter mode. In this mode, the SH pulse controls the integration time t_{INT}. An integration takes place between two SH pulses, when the second pulse coincides with the ICG pulse.

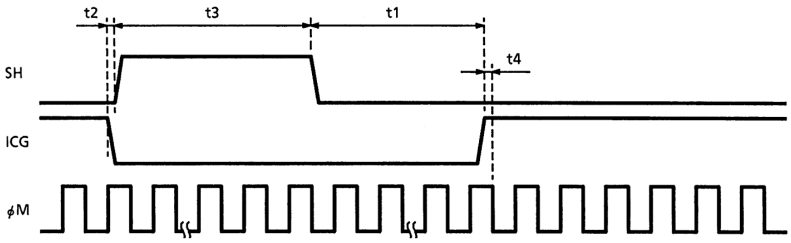
Looking at the sensor output marked as OS in the chart, it is shown that there are 3694 intensity values to be read out. These additional values are not actual pixels of the sensor, but dummy output values. There are 32 elements of dummy output before the actual 3648 intensity values, and 14 elements after them.

Figure 3.9 has a more detailed view on the timing requirements of the three pulses. Several rules have to be followed to get a proper signal from the sensor:

- **ICG Pulse Delay**: When the SH pulse is low, the ICG pulse goes high with a minimum delay of 1 µs.
- **Pulse Timing of ICG and SH**: When the ICG pulse is low, the SH pulse must go high with a delay of 0.1 µs to 1 µs.
- **SH Pulse Width**: The SH pulse width has to be at least 1 µs wide.
- **Pulse Timing of ICG and fM**: The ICG pulse can only go high when fM is high.

As can be seen on the bottom part of Figure 3.10, the intensity values are reversed. This means that high light intensities have low output values and vice versa. The baseline is marked as OS in the chart. It corresponds to the sensors supply voltage and indicates an unilluminated pixel. Pixel values are in the range of OS and the pixel with the highest intensity.

Figure 3.11 shows the electrical diagram to drive the linear image sensor. It consists of resistors and capacitors with different characteristics. The driving pulses are going through an inverter before arriving at their corresponding pins. The output signal marked as OS passes through a transistor, before it can be read by an Analog-to-Digital Converter (ADC). The V_{DD} and V_{AD} pins are connected to the source voltage, which has to be around 4 V. All other pins are pulled to ground.



CHARACTERISTIC	SYMBOL	MIN	TYP.	MAX	UNIT
ICG Pulse DELAY	t1	1000	5000	?	ns
Pulse Timing of ICG and S H	t2	100	500	1000	ns
SH Pulse Width	t3	1000	?	?	ns
Pulse Timing of ICG and ƒ M	t4	0	20	*	ns

Figure 3.9: A more detailed view of the TCD1304 timing requirements (retrieved from [48]).

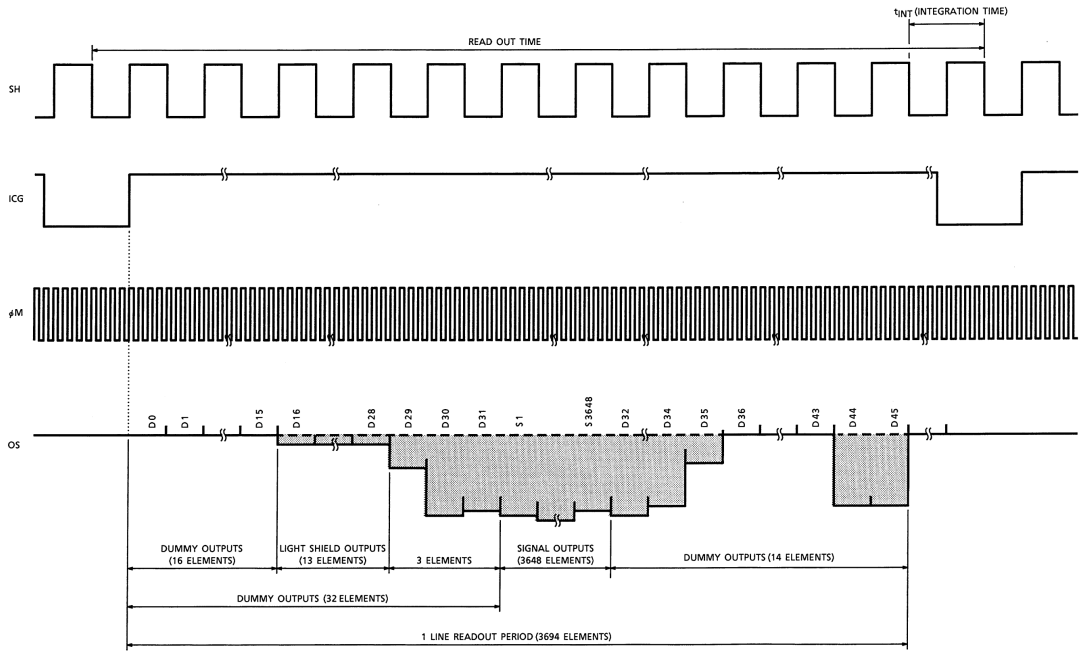


Figure 3.10: An overview of the driving pulses of the TCD1304 linear image sensor and their relation to each other (retrieved from [48]).

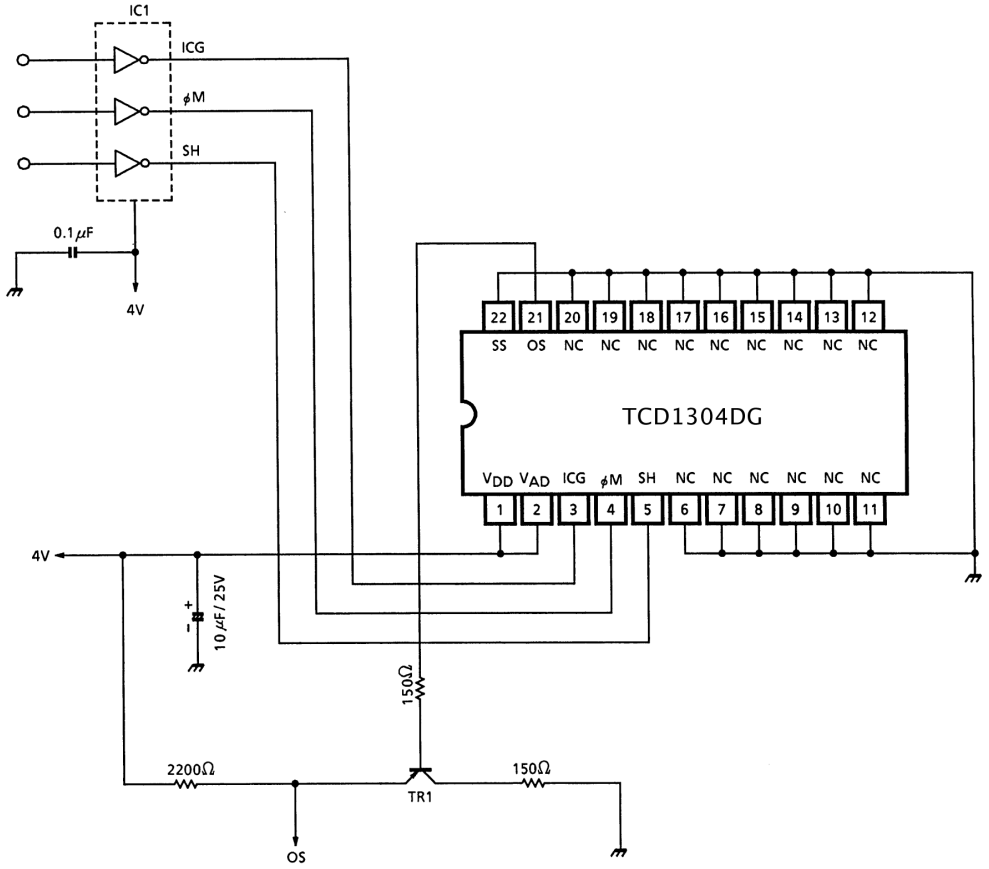


Figure 3.11: The electrical diagram needed for the TCD1304 linear image sensor. IC1 stands for a TC74HC04 inverter chip, TR1 for a 2SA1015-Y transistor (retrieved from [48]).

3.5 LED Technology

An LED is a light-emanating semiconductor component. A diode is an electric component which lets current pass in one direction but blocking it in the other. A semiconductor consists of a layer with positive carriers consisting mostly of holes (p-layer) and a layer with negative carriers consisting mostly of electrons (n-layer). This section is based on [50].

An LED can be in one of two states:

- **Unbiased State**: In this state, a depletion region is created which extends mainly into the p-layer, resulting in an insulating condition.
- **Biased State**: When a forward voltage is applied to an LED, electrons can flow from the n-layer to the p-layer. Holes are moving in the opposite direction. A hole and an electron can recombine if they have the same momentum, emitting optical radiation. The wavelength of the radiation depends on the energy bandgap of the semiconductor material used.

LED technology provides better energy efficiency and longer life compared to the tungsten light source. Furthermore, it has a more adjustable spectral power distribution and a better illuminance. While most semiconductors are made out of silicon, LEDs consist of some kind of gallium compound. The type of compound controls the spectral wavelength of the LED. LEDs made out of Indium Gallium Nitride (InGaN) for example are usually used for ultraviolet, blue and green light with high radiant efficiency. In Figure 3.12, a high power LED on a breakout board can be seen.

3.6 Microcontrollers

Microcontrollers are small integrated pieces of hardware [52]. They are built into countless objects of everyday use like smartphones or washing machines. Microcontrollers are single chip computer systems integrating the periphery needed for a working system. Components like the processor core, Random-Access Memory (RAM) and flash memory are all fitted in a single enclosure. Figure 3.13 shows the essential functions of a basic microcontroller. Some important microcontroller features are described in the following paragraphs. Advantages of microcontrollers compared to processors are lower manufacturing cost, higher reliability and lower power consumption. These advantages come at the expense of slower processing speeds and lower expandability. This section is based on [52] and [53].

3.6.1 Interrupts

Interrupts are a fundamental mechanism for processors and microcontrollers. They handle non-periodic, non-deterministic events like keyboard input. Every event that has

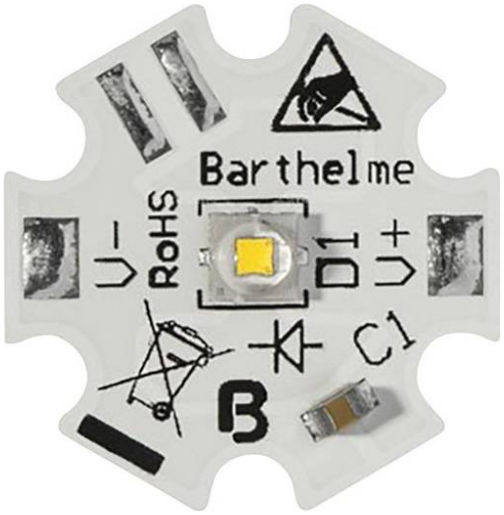


Figure 3.12: A breakout board for a high power LED (retrieved from [51]).

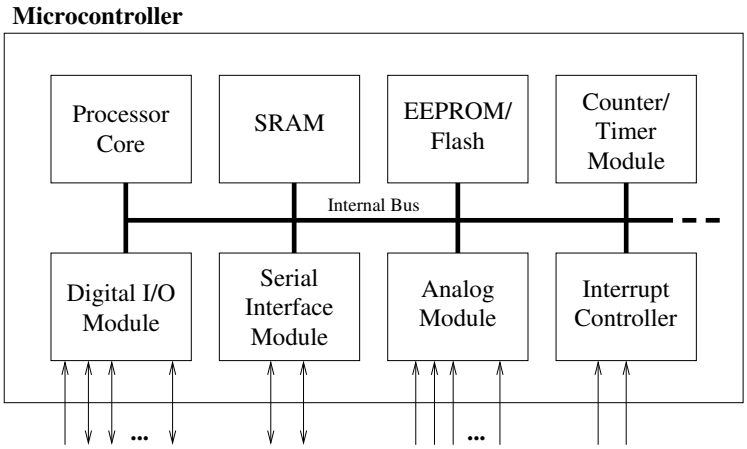


Figure 3.13: An overview of the basic components of a microcontroller (retrieved from [52]).

occurred signifies a state change and requires some kind of reaction. When an event triggers an interrupt, an Interrupt Service Routine (ISR) is called, which handles the event. The current program flow is stopped and the code from the ISR is executed. The program execution can be interrupted at any given machine instruction.

The opposite of interrupt handling would be simple signal polling: Periodically checking

for state changes of the input signal. However, polling has its drawbacks. Central Processing Unit (CPU) cycles are wasted unnecessarily when the event does not occur frequently. Furthermore, writing and extending a program which has to poll signals at a certain period can be a cumbersome endeavor. The polling function has to be called from several places in the code in order not to miss the event.

A Nested Vectored Interrupt Controller (NVIC) manages situations where multiple interrupts occur simultaneously. Since it is possible to disable interrupts during critical situations, an accumulation of interrupts waiting for processing is not unlikely. Determining the interrupt to process first is the task of the NVIC. The NVIC also handles nested interrupts, i.e. when an interrupt tries to interrupt an active ISR. By giving interrupts priorities, the NVIC can determine if an interrupt has the privilege to abandon an ongoing ISR.

3.6.2 Timers

Timers are important parts of microcontrollers. Timers are essentially counters that generate or measure various signals. They can generate Pulse-Width-Modulated (PWM) output or trigger events with a certain amount of delay. Timers raise an interrupt when they reach a user-defined count value. The ISR can then process periodic tasks. Timers can have different counter resolutions and clock frequencies.

3.6.3 Analog-to-Digital Converter (ADC)

The ADC converts analog inputs to binary values. The STM32 microcontrollers generate the corresponding digital value by successive approximation. Essentially by performing binary search, the ADC generates a discrete set of voltages and compares them against the sampled input voltage.

3.6.4 Direct Memory Access (DMA)

DMA controllers can handle memory-to-memory, peripheral-to-memory and memory-to-peripheral transfers. With DMA enabled, the CPU is relieved of the costs of transferring data between memory and peripherals and can perform other tasks simultaneously.

3.6.5 Serial Peripheral Interface (SPI)

The SPI is a sychronous interface bus using a master-slave architecture. It enables full-duplex communication between a master, which is normally a microcontroller, and one or more slave devices. The SPI bus consists of the following four communication lines:

- **Master Out Slave In (MOSI)**: The master transfers data to the slave on this line.

- **Master In Slave Out (MISO)**: The slave transmits data to the master on this line.
- **System Clock (SCK)**: This line gives a master clock signal to the slave.
- **Slave Select (SS)**: On this line, the master selects a slave for communication. Every participating slave needs a separate SS-line to the master.

To initiate a communication with a slave, the master sets the corresponding SS-line low. Master and slave transmit data over the MOSI and MISO line at the same time.

3.6.6 STM32F401RE

The STM32F401RE is a microcontroller based on a 32-bit *ARM Cortex-M4* CPU clocked at 84 MHz [54]. It has 512 kB of flash memory and 96 kB of Static Random-Access Memory (SRAM). Figure 3.14 shows a picture of the microcontroller.

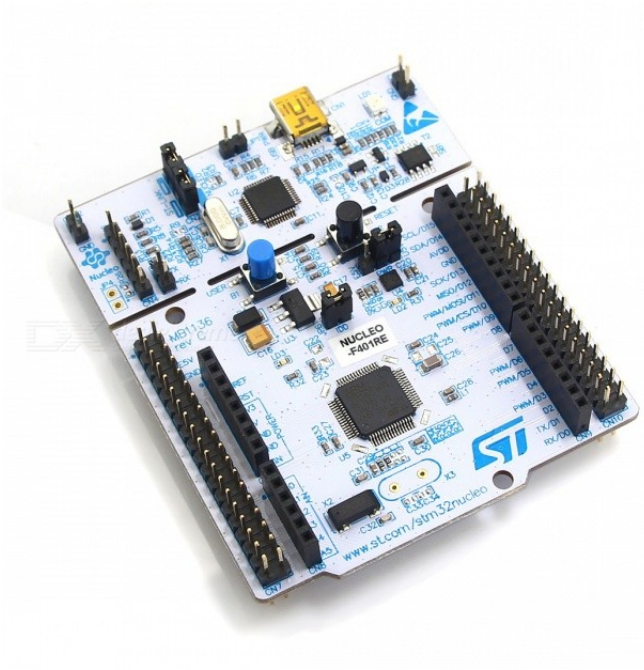


Figure 3.14: The STM32F401RE Nucleo microcontroller (retrieved from [55]).

The STM32F401RE microcontroller features up to four SPI communication channels in slave and master mode. The maximum data transfer rate is 42 Mbit/s. On the STM32F401RE microcontroller, there are eight multi-purpose timers. They have a counter resolution of 16 or 32-bit and a maximum timer clock of 84 MHz. The microcontroller features a single ADC. The ADC has a maximum clock frequency of 36 MHz and a resolution of 12 bits. With a typical clock frequency of 30 MHz, the ADC has a minimum conversion and sampling time of 0.5 µs.

3.7 Raspberry Pi

The Raspberry Pi is a credit card-sized single-board computer based on embedded Linux [56, 57]. The first Raspberry Pi was released in 2012. The main idea behind the project is to promote computer programming and information technology to young children. By keeping the price low, the Raspberry Pi was a huge success and quickly became popular outside its target market. The current version is the Raspberry Pi 3 Model B+. It features a 1.4 GHz ARM quad-core 64-bit CPU with 1 GB of RAM. Figure 3.15 shows a picture of the microcomputer.

Figure 3.15: The Raspberry Pi single-board computer (retrieved from [57]).

The Raspberry Pi has a High Definition Multimedia Interface (HDMI) port, four Universal Serial Bus (USB) ports, an ethernet connector as well as onboard Wi-Fi and Bluetooth. It also features 40 General Purpose Input/Output (GPIO) pins. These pins can be used for low level tasks like reading sensors or control LEDs. Furthermore, the GPIO pins can be used to communicate with other devices over the Inter-Integrated Circuit (I2C) or SPI bus.

There are several flavours of Linux running on the Raspberry Pi. The main Linux distributions are Raspbian, Ubuntu, OpenELEC and Arch Linux. Raspbian is a Debian-based distribution specifically developed for the Raspberry Pi and officially supported by the Raspberry Pi Foundation. The Raspbian distribution is optimized for the Raspberry Pi's ARM processor. It features PIXEL, a lightweight desktop environment as well as several other open-source applications.

CHAPTER 4

Implementation

4.1 Basic Functionality of the Tracking System

This chapter gives an overview of the basic functionality of the tracking system and the implementation part.

The system is primarily based on the physical phenomenon of refraction, light's property to change its direction at an interface to a different medium. The angle of refraction depends on the media the light travels through and the incident angle. Light is emitted by LED light sources, placed in the tracking environment. When the tracking device alters its position and therefore the refraction angle changes, the change of position can be computed. The incident light is measured by a linear image sensor. In order to toggle the LEDs at the right instances in time and read-out the linear image sensor, a microcontroller is used.

To get a valid measurement, only a single light source can be active at any point in time. At least two measurements of different lights sources are needed to calculate position data. Figure 4.1 depicts the tracking space from the xy-plane. Multiple light source are illustrated, which are arranged parallel to the x-axis. The area where complete tracking is guaranteed is marked in red, provided that the light sources are near enough to each other.

To get as much information as possible from light refraction, a sophisticated optical system is necessary. In order to find the right set of lenses for the optical system, a light ray simulation has to be conducted. Other parameters, like the distance between two LEDs and the corresponding beginning of the area where continuous tracking is guaranteed, can be calculated as well. After the simulation, the prototype can be realized. It is based on a microcontroller which controls the LEDs and reads the linear image sensor. The data of the linear image sensor is sent to a Raspberry Pi, which can then calculate the absolute position based on the intensity values of the sensor. Figure 4.2

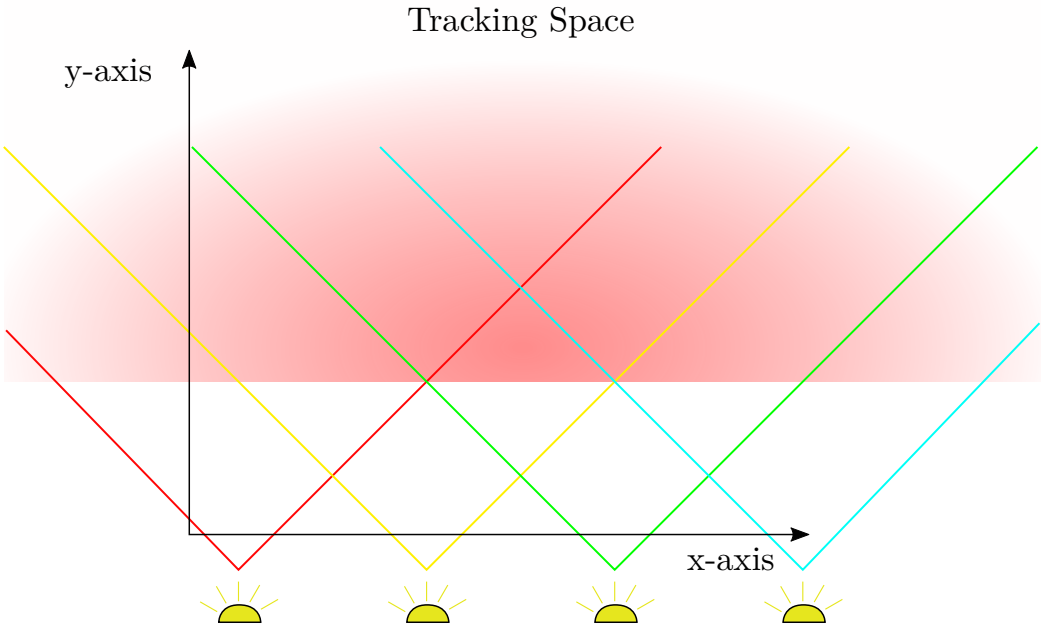


Figure 4.1: The tracking area is illuminated by multiple light sources, shown in a view from above. At each position, at least two light sources have to cast light on the linear image sensor, to ensure continuous tracking.

shows an intensity chart of two separate light source measurements. The chart's x-axis consists of the number of pixels the linear image sensor provides. The higher the amount of pixels of the linear image sensor, the better the tracking resolution gets. The y-axis gives the intensity value of each pixel. To compute the current position, the midpoint of each spike has to be calculated. The distance between these midpoints yields the position on the y-axis, while the distance's midpoint itself results in the x-axis value.

When only a single light source is mapped on the tracking device and every other light source is out of range, the resulting width and position of the spike can be sufficient to calculate the position data. The tracking quality depends on the proximity of the tracking device to the light source. The nearer the tracking device is to the light source, the more accurate the y-axis can be tracked. This approach can be used to enhance the tracking area closer to the line of LEDs.

4.2 Optics

The optical system is split into two independent parts, each one consisting of two lenses:

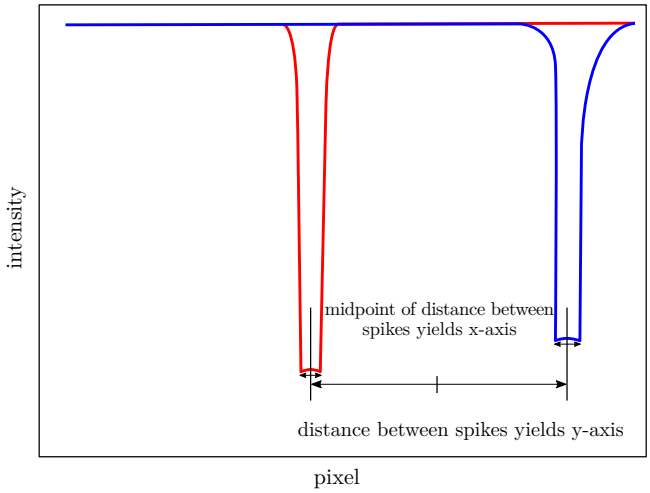


Figure 4.2: Intensity measurements of two different light sources and their corresponding relations to each other.

- **Telecentric Subsystem**: This system is responsible for the refraction of light and therefore for the actual position estimation. It consists of two convex cylindrical lenses, arranged with their curved surfaces pointing to each other. The aim of the subsystem is to converge light to a thin line, which is then digitized by a linear image sensor. The more extreme the angles of the light rays are, which are still deflected onto the sensor, the less LEDs have to be placed in the tracking environment. The focal length must remain constant over the whole tracking area. Two convex cylindrical lenses in series decrease the overall focal length. Therefore, the lenses can be placed closer to the sensor and different focal lengths can be designed more easily, by combining convex lenses. The telecentric subsystem also provides equal focal lengths for all incident angles, so that light at extreme angles is still in focus at the sensor [24].

- **Collimating Subsystem**: The purpose of the subsystem is to deflect as much light as possible through the telecentric subsystem and onto the linear image sensor. It extends the range of the z-axis, which is not tracked by the system. The tracking device can therefore be moved more freely without loss of tracking. Moving the tracking device along the z-axis does not change its position in 2D. The collimating subsystem consists of a concave and a convex cylindrical lens. The concave lens diverges the light, while the convex lens converges it. The combination of these two lenses results in an increase of light hitting the linear image sensor, which otherwise would not even reach it. Figure 4.3 shows a light ray simulation of the concave and convex cylindrical lens, forming the collimating subsystem. A side view of the tracking system is shown, with the z-axis pointing upwards.

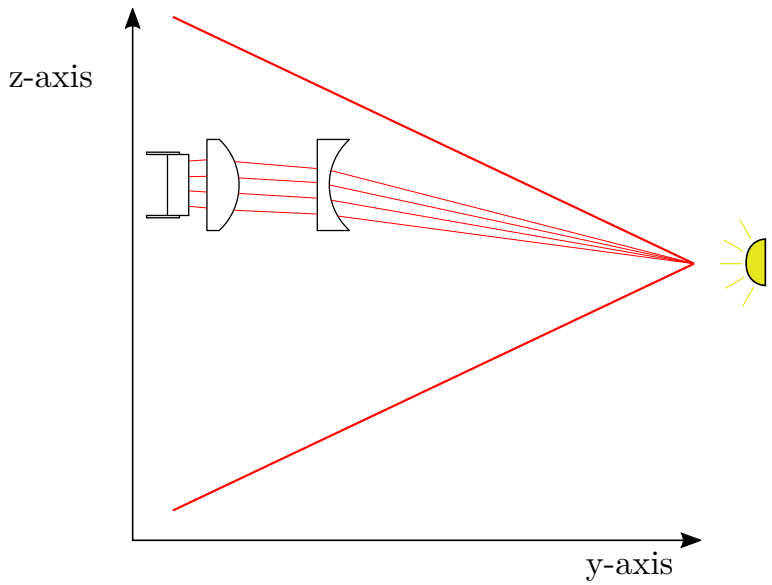


Figure 4.3: The effect of the collimating subsystem, consisting of a concave and a convex lens, is simulated by light rays. The simulation is shown in a side view with the z-axis pointing upwards, compared to the top view of the tracking area in Figure 4.1.

In order to get the right set of lenses for the optical subsystems, extensive calculations and simulations had to be done. Simulation starts with the establishment of the tracking area size. The optical system has to be in perfect focus exactly at the borders of the tracking space. The nearer the tracking system gets to the light source, the wider the converging light beam gets. Figure 4.4 depicts this characteristic.

The first picture shows the tracking system exactly at the border of the preconfigured tracking space. The light rays are in perfect focus on the sensor. In the second sketch, the tracking system is shown somewhere between the light source and the border of the tracking space. The width of the light rays impinging on the sensor depends on its distance to the light source. In the third picture, an out-of-bounds tracking system is depicted. The light rays are out of focus and produce a noisy result on the linear image sensor.

Jupyter Notebook based on Python 3 has been used as a development environment for the simulation. In the prototype simulation, a tracking system width of 5 m has been chosen. This means that the chosen lens characteristics and distances of the lenses to each other should yield a optical system which converges light originating 5 m from the system perfectly on the sensor. Trying to track beyond the 5 m tracking system width results in noisy intensity data and therefore undefined tracking behavior.

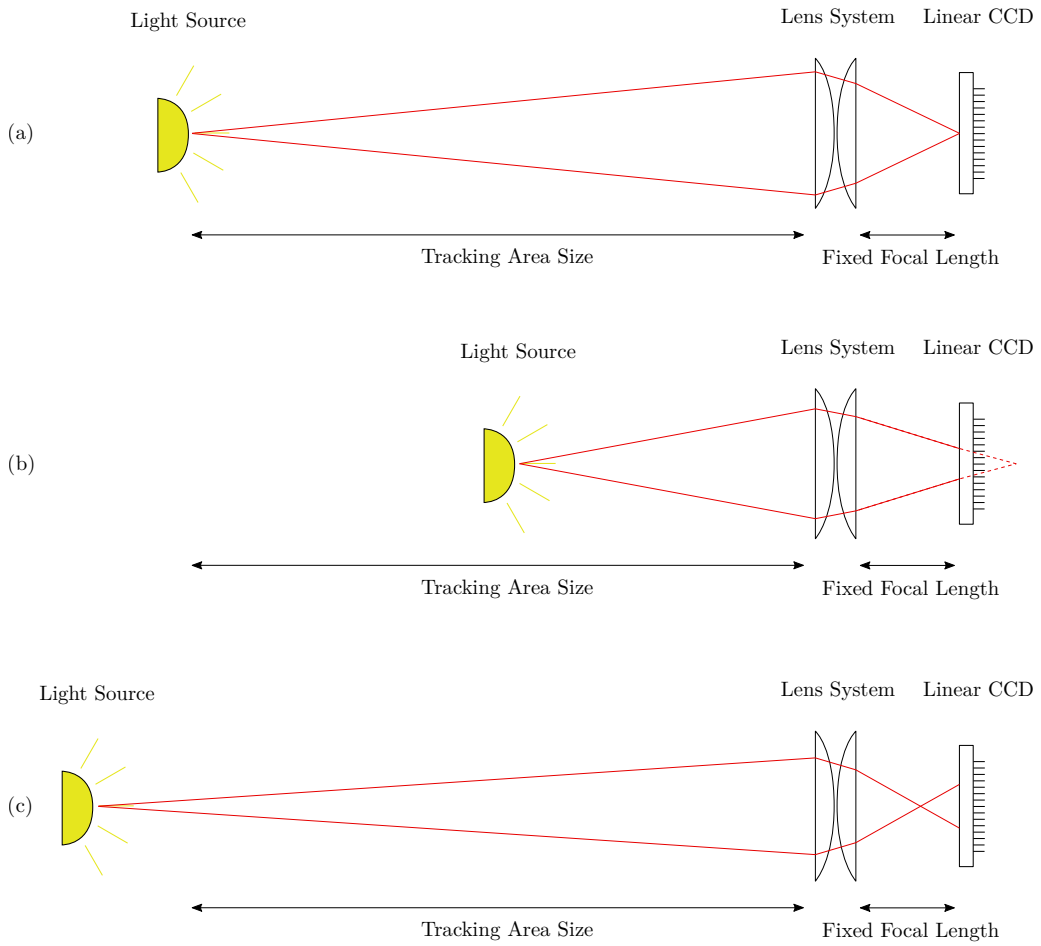


Figure 4.4: Different distances of the optical system from the light source produce different focal points. Three possible focal lengths are depicted: (a) the focal point lies exactly on the sensor, (b) the focal point lies after the sensor, (c) the focal point lies before the sensor.

Lenses were obtained from *Thorlabs*, an online shop for optical equipment. They offer a variety of plano-convex and plano-concave cylindrical lenses. Due to the fact that cylindrical lenses are only available in certain sizes with different focal lengths, the lens' dimensions had to be determined prior to the simulation. The TCD1304 linear image sensor has an effective sensor length of 29.1 mm. This is the minimum length one side of a lens has to cover. For the telecentric subsystem, the side perpendicular to the cylindrical axis, the lens' height, has to measure this length. Thorlabs offers cylindrical lenses with a height of 30 mm, a length of 32 mm and a focal length between 50 and 1000 mm. For the collimating subsystem, the lens' length axis, which is parallel to its cylindrical axis, has

to measure at least 29.1 mm. There is one lens available with a length of 30 mm and a height of 15 mm. It has a focal length of 20 mm and is available as a convex and concave lens.

In the `ray_transfer.ipynb` file, several functions based on ray transfer matrix analysis (see section 3.3) have been implemented:

- **`trans_mat( d )`**: This function returns a translation matrix with distance `d`.
- **`refract_mat( n1, n2, R )`**: This function returns a refraction matrix with the refractive indices `n1` and `n2` and radius `R`. `n1` is the refractive index of the incoming medium while `n2` is the refractive index of the outgoing medium.
- **`flat_mat( n1, n2 )`**: This functions returns a refraction matrix of a plane surface. A plane surface can be imagined as a curved surface with $R = \infty$, therefore

$$\lim_{R\to\infty} \frac{n_1 - n_2}{R n_2} = 0 \tag{4.1}$$

  and the corresponding element of the refraction matrix is set to 0. The parameters `n1` and `n2` are the refractive indices of the incoming and outgoing media, respectively.
- **`input_vec( offset, angle )`**: The function returns a two-dimensional vector, representing a light ray. The parameter `offset` is the distance of the ray from the optical axis, the parameter `angle` is the angle the ray forms with the optical axis.
- **`cast_rays( lens_width, lens_mat, source_dist, axis_off )`**: This function casts rays to the edges of the optical system and calculates the position of the resulting focal point. It is shown in Listing 4.1. Parameter `lens_width` is the lens width of the optical system. `lens_mat` is the lens system matrix, which represents the optical system to be tested by the function. It is composed of several ray transfer matrices, concatenated by matrix multiplication. Parameter `source_dist` represents the distance from the light source to the lens on the optical axis. Parameter `axis_off` is the offset of the light source from the optical axis. Figure 4.6 shows all variables in context to each other.

  The function uses `sympy` algebraic equation solvers to solve systems of equations. The angle needed for setting up the ray has to be determined. By separately solving the equations

$$\begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ x \end{bmatrix} - \begin{bmatrix} l/2 \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.2}$$

  and

$$\begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ x \end{bmatrix} - \begin{bmatrix} -l/2 \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.3}$$

for x, the angle of both rays is returned. All variables except for x are given. d is the parameter `source_dist`, the distance between the light source and the optical system. p is the parameter `axis_off`, the offset between the optical axis and the light source. l is the parameter `lens_width`. The lens width has to be divided by two and negated for the second equation, in order to send rays through the edges of the optical system. In geometrical optics, however, all rays are travelling through the same focal point, so the point of the ray hitting the lens does not matter. After the equation is solved, the rays travel through the optical system by multiplying them with the lens system matrix, `lens_mat`. Next, the focal point has to be found. This is again done by an equation solver:

The equation

$$\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} k_1 \\ \alpha_1 \end{bmatrix} - \begin{bmatrix} y \\ \beta_1 \end{bmatrix} - \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} k_2 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} y \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.4}$$

has to be solved, with two unknown variables x and y. The variable x corresponds to the `focal_length`, which is the distance between the optical system and the focal point. The vectors $\begin{bmatrix} k_1 \\ \alpha_1 \end{bmatrix}$ and $\begin{bmatrix} k_2 \\ \alpha_2 \end{bmatrix}$ are the two rays originating at the light source and going through the optical system. The vectors $\begin{bmatrix} y \\ \beta_1 \end{bmatrix}$ and $\begin{bmatrix} y \\ \beta_2 \end{bmatrix}$ are the rays going out of the optical system. The rays' angle is known, but the offset from the optical axis y is unknown. By solving this equation, the resulting `focal_length` is used to calculate the `focal_offset` by multiplying one of the outgoing rays with the translation matrix of the focal length. Both the `focal_length` and `focal_offset` values are returned by the function.

- **`cast_rays_to_focal_length(lens_width, lens_mat, source_dist, axis_off, focal_length)`**: This function casts two rays through a optical system given by the parameter `lens_mat`, from a light source of `source_dist` distance from the optical system and `axis_off` from the optical axis. The parameter `lens_width` corresponds to the width of the optical system. The parameter `focal_length` is the distance between the end of the optical system and the plane where the ray tracing ends. On this plane, the distances of the two rays from the optical axis are calculated and returned by the function.

- **`plot_ray_cast(lens_width, lens_length, lens_mat, source_dist, axis_off)`**: This function plots three rays originating at a given point in space, travelling through the optical system and coming together at the focal point (if a focal point exists). The rays are sent through the edges and the middle of

the optical system. The optical system is depicted as a hatched box. In Figure 4.5, a plot of the light ray simulation can be seen. The parameters are the same as in the previously described `cast_rays()` function. There is an additional `lens_length` parameter, which describes the length of the optical system.

The function utilizes the `matplotlib` library, which offers *MATLAB*-like plotting functionality. Similar to the previously described functions, an equation solver is used to calculate the angles of the light rays from the light source to the optical system. After the optical system, the rays end at the distance of the `tracing_end` variable.

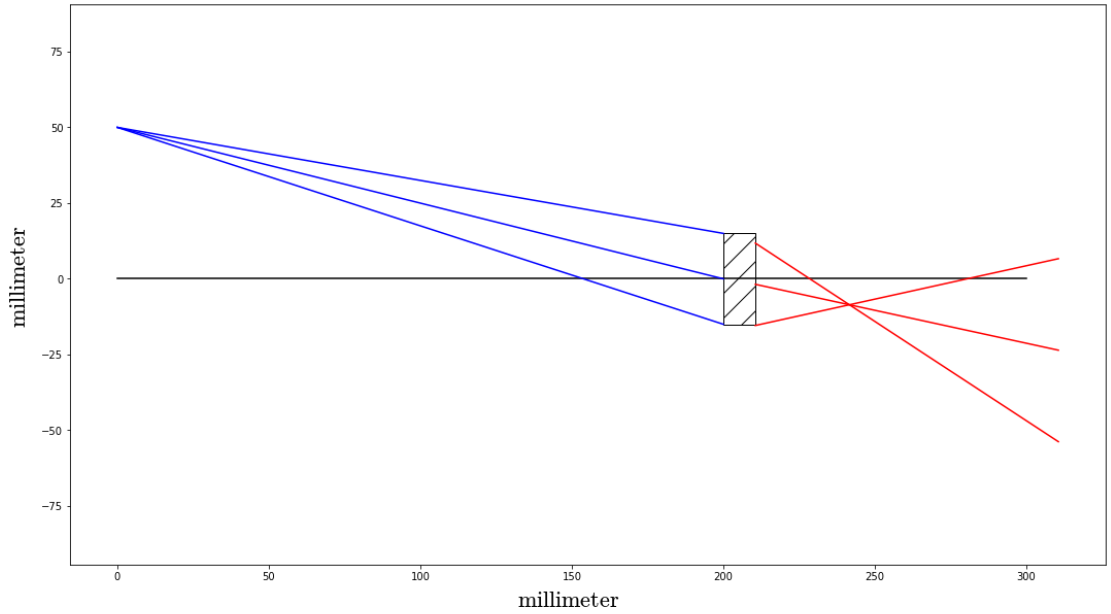


Figure 4.5: A plot resulting from the `plot_ray_cast()` function. In this example, the telecentric subsystem, consisting of two convex 30 mm × 32 mm lenses with a focal length 60 mm, was simulated. The light source has a distance of 200 mm from the optical system and 50 mm from the optical axis.

```
1  def cast_rays(lens_width, lens_mat, source_dist, axis_off):
2    # defines the symbols used for the solver
3    x,y = symbols('x y')
4
5    # solve for angle of top and bottom ray
6    inp = input_vec(axis_off, x)
7    top_ray = input_vec(lens_width/2, x)
8    bottom_ray = input_vec(-lens_width/2, x)
9    translation = trans_mat(source_dist)
10   top_ray_solve = (translation * inp - top_ray).applyfunc(solve)
11   bottom_ray_solve = (translation * inp - bottom_ray).applyfunc(solve)
```

```
12    top_ray_alpha = top_ray_solve[0][0]
13    bottom_ray_alpha = bottom_ray_solve[0][0]
14
15    # send both rays through the optical system
16    top_ray = input_vec(axis_off, top_ray_alpha)
17    bottom_ray = input_vec(axis_off, bottom_ray_alpha)
18    ray1 = lens_mat * trans_mat(source_dist) * top_ray
19    ray2 = lens_mat * trans_mat(source_dist) * bottom_ray
20
21    # solve for focal_length and focal_offset
22    focal_length_mat = trans_mat(x)
23    ray1_fo_vec = input_vec(y, ray1[1])
24    ray2_fo_vec = input_vec(y, ray2[1])
25    offset_solve = (focal_length_mat * ray1 - ray1_fo_vec - \
26      focal_length_mat * ray2 + ray2_fo_vec).applyfunc(solve)
27    focal_length = offset_solve[0][0]
28    focal_offset = trans_mat(focal_length) * ray1
29
30    # return focal_length and focal_offset
31    return (focal_length, focal_offset[0])
```

Listing 4.1: The function `cast_ray_to_edges()` calculates a focal point of light rays going through an optical system

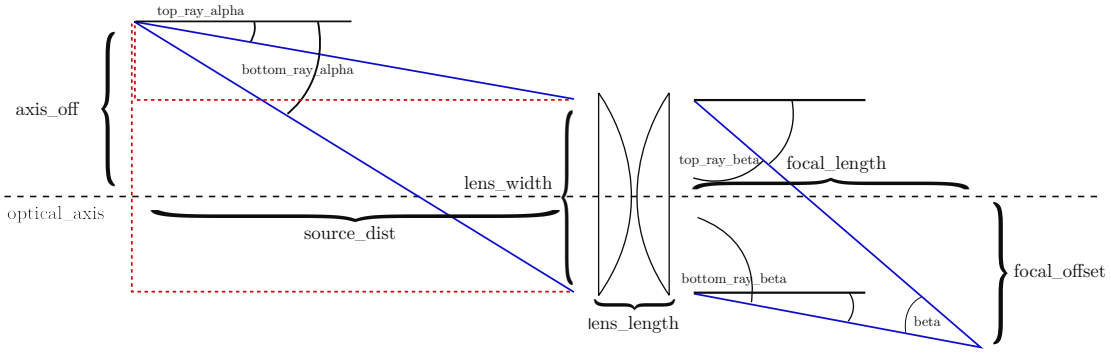


Figure 4.6: The parameters and variables of the `cast_rays()` function are shown in the context of a light ray simulation.

4.2.1 Selection of the Lenses

The lenses of the simulated optical systems are all modeled as thick lenses, which means that the lens' thickness is taken into account, in contrast to the thin lens equation. For the telecentric subsystem, all available lenses of size 30 mm × 32 mm with a focal length between 50 and 100 mm have been modeled and simulated by the functions described above.

The model of the telecentric subsystem with a 60 mm focal length of the convex lenses is shown in Listing 4.2. Each part of the lens is modeled as a distinct section, which consists of a translation or refraction part. In the case of the telecentric subsystem, a single convex lens consists of the concatenation of the refraction matrix (function `refract_mat()`) with the refractive indices 1.0 for air and 1.52 for N-BK7 material and a radius of 31 mm, the translation matrix with a width of 5.9 mm and a refraction at a straight interface (function `flat_mat()`). The matrices are concatenated from right to left, which means that the corresponding matrix of the part the light rays traverses first has to be on the rightmost side.

```
1 PCX_30_32_60 = \
2   flat_mat(1.52, 1.0) * trans_mat(5.9) * refract_mat(1.0, 1.52, 31)
3 PCX_30_32_60_reversed = \
4   refract_mat(1.52, 1.0, -31) * trans_mat(5.9) * flat_mat(1.0, 1.52)
5
6 lens_system_mat = PCX_30_32_60 * PCX_30_32_60_reversed
```

Listing 4.2: The model of the telecentric subsystem with a focal length of the convex lenses of 60 mm

For the tracking system's telecentric subsystem, two planar-convex cylindrical lenses with a dimension of 30 mm × 32 mm and a focal length of 60 mm were evaluated as the best choice. These two lenses are arranged as close as possible, both curved surfaces facing each other. The focal length results in a value of ≈ 26.1 mm when the light point is 5 m away from the optical system. When the light source is 0.5 m away from the tracking system, the focal length is ≈ 27.8 mm. At a distance of 0.5 m, the light beam has a width of ≈ 1.6 mm on the linear image sensor.

By using lenses with different focal lengths, the global focal length of the optical system changes. Convex lenses with a focal length of 100 mm would change the distance to the linear image sensor to ≈ 46.8 mm. Convex lenses with a focal length of 50 mm, on the other hand, would change the distance to the sensor to ≈ 20.5 mm. The shorter the focal length, the more extreme incident angles are still refracted onto the sensor. This means that the light sources can be placed further apart of each other. At a distance of 5 m, lenses with a focal length of 100 mm still refract light on the sensor when the light source is ≈ 1.45 m away from the optical axis. Simulating the same experiment with a focal length of 60 mm, light is still refracted when the light source is ≈ 2.42 m away from the optical axis. Another advantage of lenses with a shorter focal length is the smaller change of focus when the object gets nearer or farther away from the lens.

The collimating subsystem consists of a plano-concave and a plano-convex cylindrical lens. Lenses with dimensions of 15 mm × 30 mm were primarily chosen because they have the same length as the height of the telecentric subsystem, and can therefore be easily fitted into a lens mounting. The only choice of the focal length for both the convex and concave lens is 20 mm. The concave lens is mounted 40 mm above the linear image sensor and 2.63 mm above the telecentric subsystem. The convex lens is mounted 3 mm above the linear image sensor, focussing as much light as possible on the sensor. The

Type	Dimension H x L	Focal Length	Radius
Plano-Concave Cylindrical Lens	15 mm × 30 mm	−20 mm	−10.3 mm
Plano-Convex Cylindrical Lens	30 mm × 32 mm	60 mm	31 mm
Plano-Convex Cylindrical Lens	30 mm × 32 mm	60 mm	31 mm
Plano-Convex Cylindrical Lens	15 mm × 30 mm	20 mm	10.3 mm

Table 4.1: The list of lenses utilized in the optical system.

collimating subsystem can be simulated by the ray transfer matrix functions. Listing 4.3 shows the simulation model of the collimating subsystem.

```
1 PCX_15_30_20 = \
2   flat_mat(1.52, 1.0) * trans_mat(5.2) * refract_mat(1.0, 1.52, 10.3)
3 PCV_15_30_20 = \
4   flat_mat(1.52, 1.0) * trans_mat(2) * refract_mat(1.0, 1.52, -10.3)
5
6 lens_system_mat = PCX_15_30_20 * trans_mat(32) * PCV_15_30_20
```

Listing 4.3: The model of the collimating subsystem with a focal length of 20 mm for both convex and concave lenses

The lens simulation can also be used to simulate the beginning of the tracking area when the distance between the LEDs is given. For example, when the distance between the LEDs is 0.5 m, continuous tracking starts at ≈ 1.05 m from the line of LEDs, given that a TCD1304 sensor is used. It is calculated by using the `cast_rays(lens_width, lens_mat, source_dist, axis_off)` function, with the parameters `lens_width` and `lens_mat` given from the actual optical system. The parameter `axis_off` is the known distance between the LEDs. By experimenting with the `source_dist` parameter, the function shall return a `focal_offset` value of ≈ 14.55 mm, which is half the length of the TCD1304's pixel array. Thus, the optical system refracts light exactly to one end of the linear image sensor when it is 1.05 m from the light source (y-axis) and 0.5 m from the middle of the light source (x-axis).

Figure 4.7 shows the distances of the lenses from the linear image sensor. The distances are measured from the top cover of the linear image sensor. The space between the top cover and the actual sensor is 0.7 mm. This extra space has to be taken into account when designing the lens mounting.

Table 4.1 lists the utilized lenses in the optical system from top to bottom. The telecentric subsystem with the two plano-convex 30 mm × 32 mm lenses lies between the collimating subsystem with the 15 mm × 30 mm plano-concave and plano-convex lenses.

4.3 Software

The implementation is split between the STM32F401RE microcontroller and the Raspberry Pi 3. The microcontroller is responsible for low-level tasks like reading out the

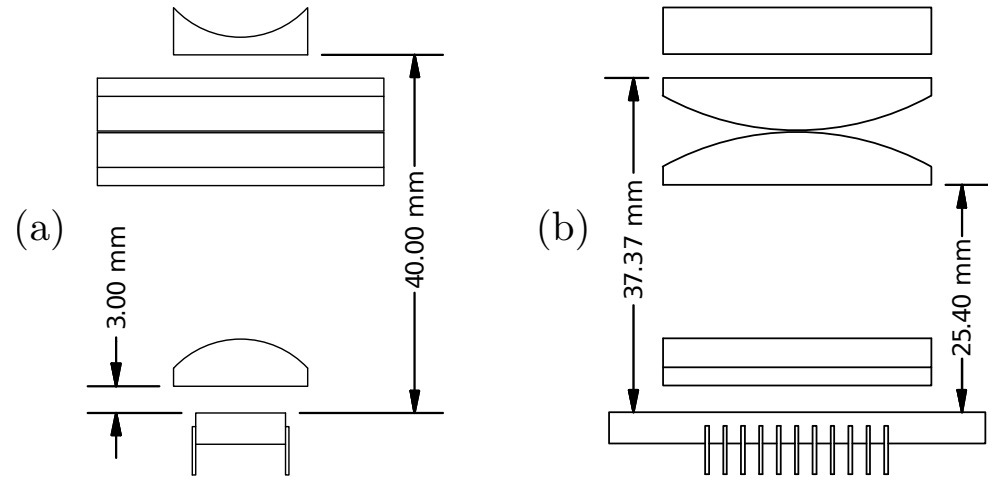


Figure 4.7: The distances of the lenses from the linear image sensor. The side view (a) depicts the distances of the collimating subsystem, while the front view (b) displays the distances of the telecentric subsystem.

linear image sensor and controlling the LEDs. The Raspberry Pi performs high-level tasks like executing the algorithm and returning an absolute 2D position.

In the following sections, the implementation of the low- and high-level functions are described.

4.3.1 STM32F401RE microcontroller firmware

The firmware for the STM32F401RE microcontroller is based on the firmware of [49], who utilizes the TCD1304 sensor for a do-it-yourself spectrometer.

Due to the strict timing requirements of the TCD1304 sensor, the required pins are all interrupt driven. The STM32F401RE microcontroller is fast enough to read the TCD1304 sensor at a full speed of 4 MHz. The maximum speed a GPIO port can be clocked is 42 MHz [54]. The ADC is also fast enough to read the ADC at a speed of 1 MHz and a full resolution of 12-bits.

The implementation utilizes the STM32 standard peripheral library [58]. It provides functions and data structures to access all important peripherals of the microcontroller. Furthermore, porting the program code to another microcontroller can be done only by reconfiguring the compiler, because the library is the same for all STM32 devices. As an alternative to using the standard peripheral library, the registers of a microcontroller can be changed directly.

Initializing peripherals with the standard peripheral library consists mainly of creating a predefined data structure, changing the appropriate fields and committing the changes by handing the data structures to an initialization function. The main part of the firmware implementation consists of setting up the peripheral devices correctly.

The corresponding functions are explained in the following sections.

Setting up the LEDs

The function `setup_LEDBlink()` initializes the GPIO pins `C4` and `C5` for controlling the LEDs. Each pin operates a transistor, which in turn toggles an LED. By setting the pin `HIGH`, the LED is turned on, while setting it `LOW`, the LED is turned off. The data structure `GPIO_InitTypeDef` is used to set up the pins. The pins are initialized as digital output pins by setting the `GPIO_Mode` to `GPIO_Mode_OUT`.

Interrupt Configuration

The `NVIC_InitTypeDef` data structure is used to configure the priority of nested interrupts. With the `NVIC_IRQChannel` field, the appropriate interrupt channel can be selected. The `NVIC_IRQChannelPreemptionPriority` field assigns the interrupt priority. It can hold an integer value between 0 and 15. A lower priority value corresponds to a higher interrupt priority. Interrupt Requests (IRQs) with a higher priority can interrupt IRQs with a lower priority.

Table 4.2 lists the priorities of the interrupts used in the firmware. `TIM5` has the highest priority while `TIM2` has the lowest. The NVIC is initialized by the `NVIC_Init ( &NVIC_InitStructure )` function call.

Interrupt	Priority
TIM5	0
DMA2	1
DMA1	2
TIM2	3

Table 4.2: The interrupt priorities of the STM32F401RE firmware.

Configuring the Timer for the Master Clock

In the function `TIM_CCD_fM_conf()`, the master clock timer is configured. The master clock of the TCD1304 sensor is served by `TIM3`, timer 3 of the microcontroller. The function call `RCC_APB1PeriphClockCmd ( RCC_APB1Periph_TIM3, ENABLE )` enables the low speed 42 MHz `APB1` clock for the `TIM3` peripheral. Next, a `GPIO_InitTypeDef` is created and initialized as pin `PB0`. Setting the `GPIO_Mode` to `GPIO_Mode_AF` configures the pin as an alternative function. The function call `GPIO_PinAFConfig ( GPIOB, GPIO_PinSource0, GPIO_AF_TIM3 )` connects `TIM3` to pin `PB0`.

The timer itself is initialized by the TIM_TimeBaseInitTypeDef data structure. The TIM_Prescaler field is set to 0, therefore the maximum timer frequency of 84 MHz is used. The function get_Timer_clocks() calculates the APB1 frequency and assigns it to the global integer variable apb1_freq. The TIM_Period field is set to apb1_freq / CCD_fm - 1 which yields 20 when CCD_fm is set to 4000000. This means that the fM clock signal needs 21 clock cycles for a full period.

The TIM_OCInitTypeDef structure initializes the output compare signal, needed for PWM. The TIM_Pulse field is set to apb1_freq / (2 * CCD_fm), resulting in a value of 10 and a duty cycle of $\approx 50\%$. By calling the function TIM_OC3Init (TIM3, &TIM_OCInitStructure), the output compare device is enabled. The function TIM_Cmd (TIM3, ENABLE) enables timer 3. After this function call, the pin PB0 outputs a PWM signal serving the master clock of the TCD1304 sensor.

Setting up the ADC

The function TIM_ADC_conf() initializes the timer of the ADC. TIM4 is used as the ADC timer. By calling the function RCC_APB1PeriphClockCmd (RCC_APB1Periph_TIM4, ENABLE), timer 4 is enabled and clocked by APB1. Similar to the setup of the master clock, the GPIO_InitTypeDef, TIM_TimeBaseInitTypeDef and TIM_OCInitTypeDef data structures are used. Pin PB9 is used as an optional output. The timer structure has a TIM_Period of 4 * apb1_freq / CCD_fm - 1, resulting in a value of 83, or a frequency of 1 MHz. The TIM_Pulse in the TIM_OCInitTypeDef is set to apb1_freq / (2 * CCD_fm), yielding a duty cycle of $\approx 12.5\%$.

The function ADC1_conf() configures the actual ADC pin and initializes DMA, to free the processor of unnecessary workload. The function calls RCC_AHB1PeriphClockCmd (RCC_AHB1Periph_DMA2, ENABLE) and RCC_APB2PeriphClockCmd (RCC_APB2Periph_ADC1, ENABLE) configure the AHB1 clock for DMA2 and the APB2 clock for ADC1. The DMA_InitTypeDef structure initializes DMA_Channel_0 for the ADC. The DMA_PeripheralBaseAddr field is set to (uint32_t) &ADC1->DR, the base adress of the ADC. DMA_Memory0BaseAddr is set to (uint32_t) &aTxBuffer, an array of length 3694, which corresponds to the number of pixels of the TCD1304 sensor. DMA automatically moves each pixel value to its position in the array. Although the ADC provides only a resolution of 12 bit, the array has a data-size of 16 bit. Therefore, 4 bits are unused for each array item. The DMA controller is set to DMA_Mode_Circular, which means that once the aTxBuffer array is full, the DMA writes new values automatically to the beginning of the array. By calling the function DMA_ITConfig (DMA2_Stream0, DMA_IT_TC, ENABLE), an interrupt is generated when the DMA transfer is complete.

The ADC input uses pin PC0. The actual ADC configuration is done with the ADC_InitTypeDef and ADC_CommonInitTypeDef data structures. ADC_TwoSamplingDelay is set to ADC_TwoSamplingDelay_5Cycles, yielding a

delay of 5 clock cycles between two samples. The `ADC_Resolution` is set to 12 bits. The field `ADC_ExternalTrigConv` is set to `ADC_ExternalTrigConv_T4_CC4`, therefore the ADC is triggered by timer 4.

The ADC runs with a frequency of 30 MHz, which means a cycle takes 33 ns. The ADC has a length of 12 bits, the total conversion time therefore takes $12 \cdot 33\,\text{ns} \approx 0.4\,\mu\text{s}$. The data rate of the TCD1304 is 1 MHz, if the master clock is set to 4 MHz. Thus, the ADC's total processing time must not exceed 1 µs. Because the conversion time takes 0.4 µs, there is 0.6 µs sampling time left. The equation $0.6\,\mu\text{s}/33\,\text{ns} \approx 18$ results in a maximum of 18 cycles left for sampling. Therefore, the function call `ADC_RegularChannelConfig ( ADC1, ADC_Channel_10, 1, ADC_SampleTime_15Cycles )` configures the ADC sampling time to 15 cycles.

Setting up the SPI Communication

The function `SPI2_conf()` initializes the SPI bus. `GPIO_InitTypeDef` is used for the initialization of the GPIO pins, `DMA_InitTypeDef` configures DMA and `SPI_InitTypeDef` initializes the SPI communication. The `APB1` clock drives the SPI2 interface at 42 MHz, while the `AHB1` clock runs DMA at 84 MHz. Pins `B13`, `B14` and `B15` are used for the SPI communication.

In the `SPI_InitTypeDef` structure, the `SPI_Mode` is set to `SPI_Mode_Slave`, because the Raspberry Pi insists on being the master in an SPI interaction. Since the Raspberry Pi can only handle a word length of 8 bits, the `SPI_DataSize` field is set to `SPI_DataSize_8b`.

The `DMA_BufferSize` of the `DMA_InitTypeDef` data structure is set to `2 * CCDSize`. The buffer needs to be twice as big as the number of pixels of the TCD1304, because each 12-bit ADC value is stored in a 16-bit array item, but the SPI configuration has an 8-bit word length set. The `DMA_PeripheralBaseAddr` is set to `SPI2->DR`, the base address of the SPI bus. The `DMA1_Stream4` is used to transfer data from the `aTxBuffer` array over to the SPI buffer, while `DMA1_Stream3` transfers data received from the SPI communication to the `aRxBuffer` array. The DMA interrupt is enabled by the `DMA_ITConfig ( DMA1_Stream4, DMA_IT_TC, ENABLE )` function call.

Configuring the ICG and SH Timers

In the `TIM_ICG_SH_conf(void)` function, the peripherals for driving the SH and ICG pulses are initialized. Timer 2 and timer 5 are used for the SH and ICG pulses, respectively. Both are driven by the `APB1` clock and have a frequency of 84 MHz, as well as 32-bit counter size. The SH signal is emitted on pin `PA1`, while the ICG signal is emitted on pin `PA0`. The `TIM_Prescaler` field is set to `apb1_freq / CCD_fm - 1` for both outputs, resulting in a value of 20 and a clock frequency of `84 MHz / 21 = 4 MHz` after prescaling.

The TIM_Period of timer 5 is set to ICG_period - 1, on timer 2 it is set to SH_period - 1. Both values can be changed by the application running on the Raspberry Pi. The PWM duty cycle for timer 5 is set to (5 * CCD_fm) / 1000000, resulting in a value of 5 µs. For timer 2, the duty cycle is set to (2 * CCD_fm) / 1000000, yielding a value of 2 µs. Afterwards, the timers and interrupts are enabled and all pending flags are cleared. In the last step, the counter values are modified directly. The counter of the ICG timer is set to ICG_period - ICG_delay, with a ICG_delay value of 11. The counter of the SH timer is set to SH_period - SH_delay and a SH_delay value of 12. The counters are set near to their expiration value to accelerate the process of reading-out the sensor.

Main Loop

After all peripherals are initialized, the main loop is executed. In Listing 4.4, the code of the main loop is shown.

```
1  while(1)
2  {
3    if (change_exposure_flag == 1)
4    {
5      change_exposure_flag = 0;
6      flush_CCD();
7
8      /* set new integration times */
9      ICG_period = \
10       aRxBuffer[6]<<24|aRxBuffer[7]<<16|aRxBuffer[8]<<8|aRxBuffer[9];
11     SH_period = \
12       aRxBuffer[2]<<24|aRxBuffer[3]<<16|aRxBuffer[4]<<8|aRxBuffer[5];
13
14     /* set number of integrations to peform */
15     Integration_count = aRxBuffer[11]<<8|aRxBuffer[12];
16
17     /* disable TIM2 and TIM5 */
18     TIM_Cmd(TIM2, DISABLE);
19     TIM_Cmd(TIM5, DISABLE);
20
21     /* reconfigure ICG and SH timers */
22     TIM_ICG_SH_conf();
23   }
24 }
```

Listing 4.4: The main loop of the TCD1304 firmware.

The variable change_exposure_flag is set to 1 in the DMA1_Stream4_IRQHandler() ISR. This ISR is invoked when the SPI transmission is completed and the new parameters of the ICG, SH and integration counter values have been received from the Otterly-rpiCLI application. These values are at first stored in the aRxBuffer array, afterwards the body of the main loop is executed.

The variable change_exposure_flag is immediately set to 0 in the main loop, therefore the code is executed only once after an SPI transmission. The flush_CCD() function flushes the TCD1304 by executing multiple integrations at a very low integration time with the ADC turned off (see Listing A.1). The ICG_period, SH_period and Integration_count variables are set to the values received from the Raspberry Pi. ICG_period controls the time between two read-outs while SH_period adapts the integration time. Integration_count controls the number of integrations to be captured and sent to the Raspberry Pi. The remaining processing steps are adjusted according to these values.

The timers TIM2 (SH timer) and TIM5 (ICG timer) are disabled before they are reconfigured with the new ICG and SH variables (see section 4.3.1). After the interrupts have been configured, the ISRs are responsible for reading out the TCD1304 sensor.

Interrupt Sequences

The first interrupt that is invoked is the ICG ISR (TIM5_IRQHandler(), see Listing A.2). The variable ICG_pulse_counter is incremented each time the interrupt is invoked. During CCD flushing, the variable is already incremented four times and has a value of 4.

When the ICG_pulse_counter variable has a value of 6, the SH ISR (TIM2_IRQHandler()) gets important (see Listing A.3): The variable SH_pulse_counter counts how often the ISR has been triggered. Because the SH_period value has to be an integer factor of the ICG_period, the SH interrupt triggers simultaneously with the ICG interrupt once. The LEDs are turned on 5 interrupts before the ICG and SH interrupts coincide. Which LED is turned on depends on the value of the Integration_count variable. In the last SH phase before ICG and SH signals coincide, the integration takes place. After the integration, the pixels are shifted to the output register (see Figure 3.10) and are read by the ADC. Therefore, in the seventh ICG interrupt, the boolean expression ICG_pulse_counter == 6 evaluates to True in the ICG ISR: The SH_pulse_counter is reset and the LEDs are turned off. Afterwards TIM4 is restarted, which in turn triggers the ADC. After an integration, the Integration_count variable is decremented.

When the ADC has finished reading the TCD1304, the DMA1_Stream4_IRQHandler() ISR is invoked. In this ISR, TIM4 together with the ADC is stopped, and the PB1 pin which intializes a communication with the Raspberry Pi is toggled. On the Raspberry Pi side, the SPI communication is initiated and the intensity values of the TCD1304 are sent from the STM32F401 microcontroller to the Raspberry Pi. This last part of integrating, reading the ADC and sending data via SPI to the Raspberry Pi is repeated as often as the Integration counter number is configured.

Figure 4.8 presents the whole read-out procedure in a sequence diagram. Two integrations take place during this exemplary process.

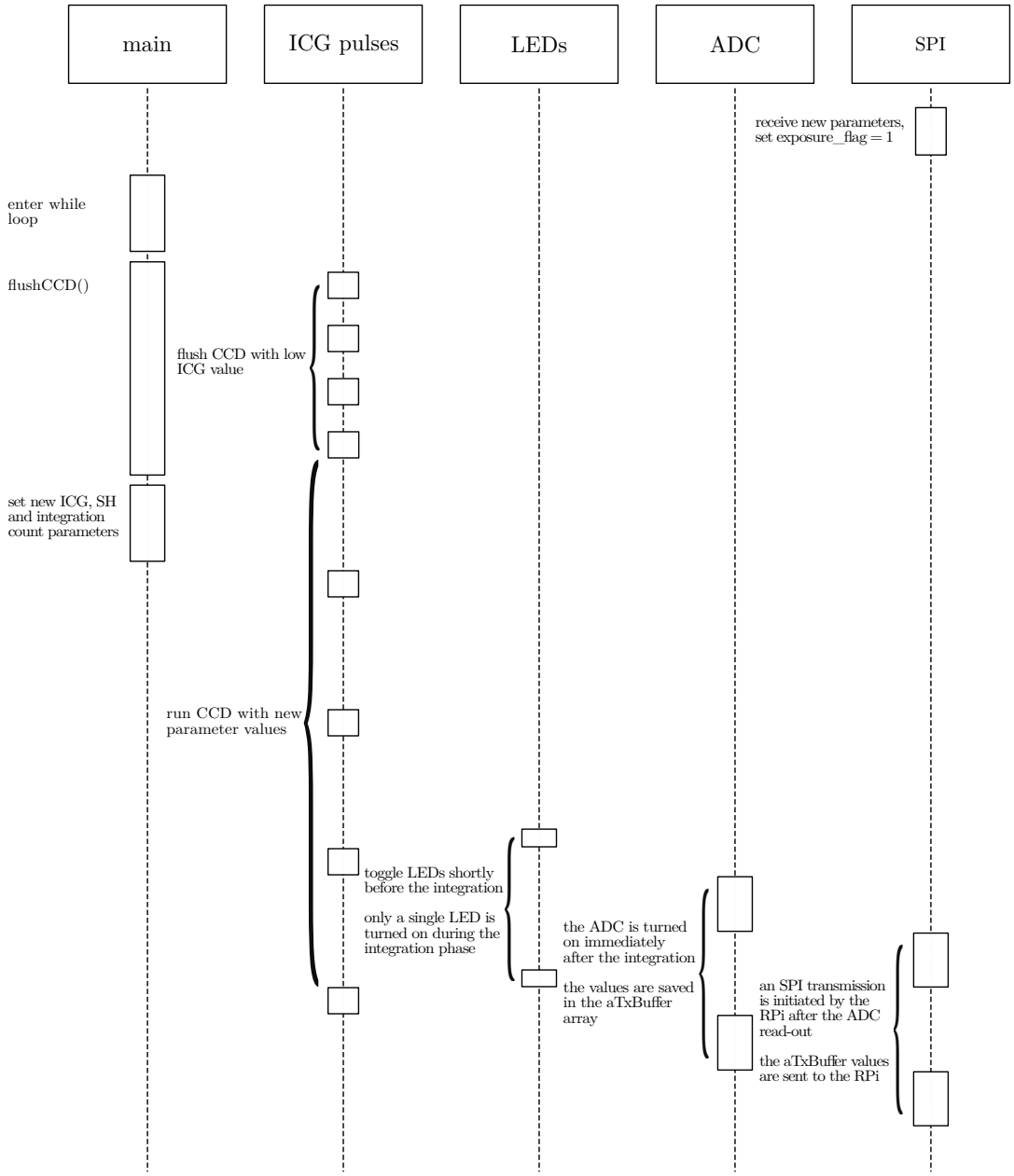


Figure 4.8: A sequence diagram of the whole read-out process is depicted. Two integrations take place at the end of this sequence, therefore light toggling, integration, ADC reading and SPI transmission occur twice. The blocks' dimensions and proportions to each other do not reflect the actual timings of the interrupts.

Compilation and Flashing of the Firmware

Ubuntu Linux is recommended as the platform of choice. *Ubuntu 14.04* has been used in the implementation of the tracking system. For the firmware compilation of the STM32F401RE microcontroller, the `gcc-arm-none-eabi` cross compiler is necessary. On Ubuntu platforms, the package can be installed by the `apt-get` package management system.

With the command

```
$ apt-get install gcc-arm-none-eabi
```

the necessary software will be installed.

To compile the firmware, the Makefile included in the source code has to be invoked. With a simple

```
$ make
```

command in the root folder of the source code, the firmware is compiled. It results in the creation of several files starting with the filename `F401_FW_TCD1304`.

To flash the firmware onto the STM32F401RE, the microcontroller has to be connected to the computer via USB. The system recognizes it as an external USB drive. By copying the file `F401_FW_TCD1304.bin` to the USB drive, the firmware is flashed onto the microcontroller.

4.3.2 Data Evaluation on the Raspberry Pi

On the Raspberry Pi side, the `Otterly-rpiCLI` application, implemented by [49], is used to control the STM32F401RE microcontroller and read out the TCD1304 sensor. It handles the complete SPI transfer and initializes the communication when a `RISING_EDGE` is detected on GPIO pin 22. The application takes three command line arguments:

- **ICG Period**: Using the `-i` option followed by an integer passes the ICG period to the microcontroller. The minimum value for this variable is 14776, because the TCD1304 has 3694 pixels to read out and every pixel takes 4 fM cycles, therefore $3694 \cdot 4 = 14776$. In the implementation a value of 30000 is used, to save time for the data transmission after the read-out.
- **SH Period**: Using the `-s` option followed by an integer passes the integration period to the microcontroller. The minimum value for this variable is 80. The minimum integration time is 20 µs, thus $4000000 \cdot 0.00002 = 80$. The SH value has to be an integer divider of ICG, in order to guarantee that both interrupts occur simultaneously at the ICG phase. In the implementation, a value of 100 is used.

- **Integration Count**: This value can be passed by using `-n` option. The value controls the number of integrations that will be taken in a row without flushing the TCD1304 sensor.

The application saves each integration in a separate file, starting with the name `output`, a consecutive number consisting of a four-digit integration count value and the ending `.dat`.

In the prototype implementation, the `Otterly-rpiCLI` application is invoked in the while loop of the `run.sh` bash script, continuously taking measurements of the TCD1304 sensor.

The application is invoked with the following arguments:

```
$ ./Otterly-rpiCLI -i 30000 -s 100 -n 2
```

Therefore, an ICG value of 30000, an SH value of 100, corresponding to an integration time of 25 µs, and 2 integrations are used as command line arguments.

The measurements are saved as `output00001.dat` and `output00002.dat` files, corresponding to read-outs with different LEDs turned on. Afterwards, both files are processed by the `spike_calc.py` python script. Several functions are implemented in this script:

- **`tracking_2d()`**: This function reads both output files with the `read_data( path )` function, processes them with the `simple_spike_tracking( data1, data2 )` function and prints the result. It filters out bad data, which can occur during the SPI communication, by printing an "invalid data" warning.
- **`read_data( path )`**: The function takes the path of the file to read in as an argument and returns a `list` data structure of intensity values.
- **`simple_spike_tracking()`**: This function is responsible for the tracking data calculation. It provides a very simple approach for 2D tracking. The minimum value's pixel position of each intensity chart is extracted. It corresponds to a spike in the intensity chart. The extraction only works under the assumption of the existence of a single spike, which has the least value of the intensity array. The approach does not eliminate noise, which may occur during the integration process. By subtracting `spike2` from `spike1`, the y-axis is calculated. In the next step, the midpoint value between the two spikes is calculated, yielding the x-axis value. Both values are returned by the function. Figure 4.2 depicts an exemplary intensity chart of two LED measurements.
- **`print_tracking_area()`**: This function evaluates the intensity charts of the folders `lf` (an abbreviation for left front), `lb` (left back), `rf` (right front) and `rb` (right back). The files in these folders are measurements from the edges of a

predefined tracking space. Tracking within the borders of the premeasured tracking space maps the tracking data to absolute positions in space.

Numerical differentiation has also been tested for spike extraction, by finding the biggest value change in the intensity chart. Gaussian filters have been used to filter out noisy data. By passing paths of measurement files to the `out_plot.py` python script, an intensity chart is plotted of all given arguments. Figure 4.9 shows two spikes of an actual intensity measurement during tracking.

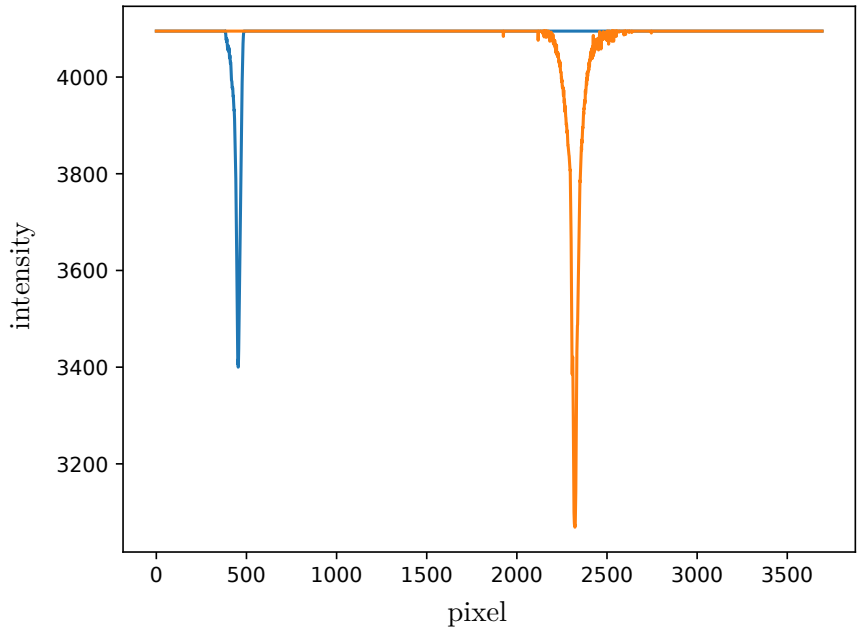


Figure 4.9: Two read-outs of the TCD1304 sensor are displayed in an intensity chart. Both measurements were taken during a tracking session and merged into a single diagram. The spikes of the LEDs can be seen easily.

4.4 Hardware

In this part of the implementation, the prototyping part of the thesis is explained. 3D prints and Printed Circuit Boards (PCBs) have been designed and manufactured in this step. The designs are shown in different illustrations and pictures, to give the reader an insight into the prototyping process.

4.4.1 3D Printing

Different parts have been designed with *Autodesk Inventor*, a Computer-Aided Design (CAD) application for 3D design, simulation and visualization. For the manufacturing of the actual prototyping parts, a Kossel XL 3D printer has been used. The Kossel XL is a Fused Deposition Modeling (FDM) 3D printer, a type of additive Computer-Numerical Control (CNC) machine. The printer fabricates plastic 3D parts from a thermoplastic filament. The material is heated by a nozzle and deposited on the growing part, yielding a manufactured part consisting of many layers of molten plastic. The thinner the layer size is chosen, the more accurate the resulting part will be.

Prior to the printing operation, the CAD model has to be preprocessed or "sliced". The slicing process results in a list of commands for the 3D printer, telling it step by step where the nozzle has to be moved to produce a plastic part out of the 3D model. In the slicing software, a number of settings like layer height, printing speed or wall thickness can be tweaked, affecting the outcome of the 3D print.

In the 3D prints fabricated for the tracking system prototype, a layer height of 150 µm was used, producing an accurate result in a reasonable amount of time. *Verbatim* PLA transparent filament was used in the printing process, with a diameter of 1.75 mm.

The lens mounting model was designed according to the results of the simulation. The telecentric subsystem has a distance of 25.4 mm to the sensor. The lenses of the telecentric subsystem have their curved sides facing each other, leaving nearly no space between them. The lenses of the collimating subsystem enclose the lenses of the telecentric subsystem. The concave lens has a distance of 40 mm to the sensor, to focus as much light as possible onto the sensor. The convex lens of the collimating subsystem has a distance of 3 mm to the sensor.

Figures 4.10 and 4.11 show illustrations of the lens mounting CAD model. It has dimensions of 45 mm × 45 mm × 59 mm. The lens mounting is kept together by four M3 screws. After the lenses are in place at their corresponding positions, the linear image sensor PCB was screwed to the lens mounting with four M2.5 screws, together with the linear image sensor which is attached to the pin connectors.

The motherboard connects all necessary pieces of hardware to get a portable prototype. It holds the Raspberry Pi, the STM32F401RE microcontroller, the PCB controlling the LEDs and the lens mounting. The motherboard consists of four seperate parts. The base part mounts the Raspberry Pi and the STM32F401RE microcontroller, which are connected by seven M2.5 screws. The feet of the motherboard are all connected by M2.5 screws. One is holding the PCB controlling the LEDs, another holds the lens mounting system. Figure 4.12, 4.13 and 4.14 depict the motherboard with its connected components.

The high power LEDs are mounted to a pole with a diameter of 14.1 mm. The LED breakout boards are attached to the 3D printed mount with two M3 screws. Between these

two objects, a heatsink with thermal compound is placed, for optimal heat dissipation. Figure 4.15 shows the LED mounting system.

Figures 4.16, 4.17 and 4.18 show the complete tracking device in action. The LEDs are connected to the tracking device with a cable. The tracking session can be started by logging into the Raspberry Pi via Secure Shell (SSH) and running the `run.sh` shell script.

4.4.2 PCB Design

KiCad, an open source Electronic Design Automation (EDA) software suite, was used for designing the PCBs of the prototype. The design process is split into two parts: In the first part, the electronic schematic is laid out in a diagram. In this diagram, the functioning principles of the circuit can be seen. Standardized symbols for electrical components are used together with their corresponding interconnections, to design a simplified version of the actual circuit. In the second part, the actual PCB is designed, which can then be fabricated. The footprints of the electrical components are put out within the boundaries of the board. Afterwards, the connections between the electronic components have to be routed, to ensure correct functionality. The finished PCBs can be manufactured, by sending *Gerber* files of the board to a PCB manufacturer. The PCBs that were designed for the prototype were fabricated at *elecrow*, a chinese PCB manufacturing service. They have a thickness of 1.6 mm and consist of two copper layers. Only Through Hole Technology (THT) parts were used in the assembly of the PCB, except for the LT1761 voltage regulator, which was only avaible as a Surface-Mount Device (SMD) component.

For the tracking system prototype, two PCBs were designed and fabricated: The first one is the circuit needed to read out the linear image sensor. The electronic schematic was obtained from [49], which corresponds to the schematic of the Toshiba TCD1304 sensor (see Figure 3.11). The electronic schematic is shown in Figure 4.19. In addition to the circuit necessary for reading-out the TCD1304 sensor, a circuit for an LT1761 voltage regulator was designed. The voltage regulator's purpose is to stabilize an unregulated Direct Current (DC) power supply. However, the voltage regulator is not necessary for driving the TCD1304 sensor and can be bypassed. Bypassing can be done by soldering a wire from the 5 V pin of the connector P1 to the VCC pin of the capacitor C1. Figure 4.20 depicts the laid out and routed components of the PCB, ready to be manufactured. The PCB was designed with the linear image sensor in the middle on one side, and all other necessary electronic components on the other side. Due to this layout it is possible to attach the PCB together with the linear image sensor directly to the lens mounting system. The Bill of Materials (BOM) of the TCD1304 PCB is listed in Table 4.3. Optional parts (only needed for the voltage regulator) are marked with an asterisk.

The second PCB is responsible for controlling the LEDs placed around the tracking space. It consists of six transistors and sockets to connect the LEDs, as well as a socket to connect an external power supply and pins to control the LEDs by the microcontroller's

Type	Amount	Price (Euro)
150 Ω Resistor	2	0.32
2200 Ω Resistor	1	0.12
2700 Ω Resistor*	1	0.19
1200 Ω Resistor*	1	0.12
0.1 µF Capacitor	2	0.30
10 nF Capacitor*	1	0.14
10 µF Capacitor*	1	0.14
2SA1015 Transistor	1	0.04
SN74HC04 Hex Inverter	1	0.28
TCD1304 Linear image sensor	1	4.30
6-Pin strip	1	0.40
LT1761IS5-BYP LDO*	1	3.49

Table 4.3: BOM of the TCD1304 PCB.

GPIOs. The BOM is listed in Table 4.4. The electronic schematic is illustrated in Figure 4.21 and the routed PCB is shown in Figure 4.22.

Type	Amount	Price (Euro)
IRLZ34N Transistor	6	5.16
2-Pin strip	7	2.03
7-Pin strip	1	0.70

Table 4.4: BOM of the LED control PCB.

The BOM for assembling a single LED is shown in Table 4.5. The wire has to be soldered on one end to the LED and on the other to the pin socket. Thermal compound has to be applied between the LED and the LED cooler.

Type	Amount	Price (Euro)
Barthelme High-Power LED 1800 mA	1	3.89
fischer elektronik LED cooler	1	1.13
5 m two-core wire	1	1.40
2-Pin socket	1	0.52
Thermal compound	1	0.90

Table 4.5: BOM of the LED assembly.

4.4.3 Component Wiring

The different components of the prototype have to be connected together to ensure proper functionality. For this task, standard jumper wires were used. Table 4.6 lists the pin associations between the STM32F401RE microcontroller and the Raspberry Pi.

Table 4.7 show the connections between the STM32F401RE microcontroller and the LED control PCB. Table 4.8 shows the connections between the TCD1304 PCB and the STM32F401RE microcontroller.

STM32F401	**Raspberry Pi**
PB15 MOSI	GPIO10
PB14 MISO	GPIO09
PB13 SCLK	GPIO11
PB2	GPIO18
GND	GND

Table 4.6: Pin connections between the STM32F401RE microcontroller and the Raspberry Pi.

STM32F401	**TCD1304 PCB**
PB0	fM
PA1	SH
PA0	ICG
PC0	OS
+5V	+5V
GND	GND

Table 4.7: Pin connections between the STM32F401RE microcontroller and TCD1304 PCB.

STM32F401	**LED control PCB**
PC5	1
PC4	2
GND	GND

Table 4.8: Pin connections between the STM32F401RE microcontroller and the LED control PCB.

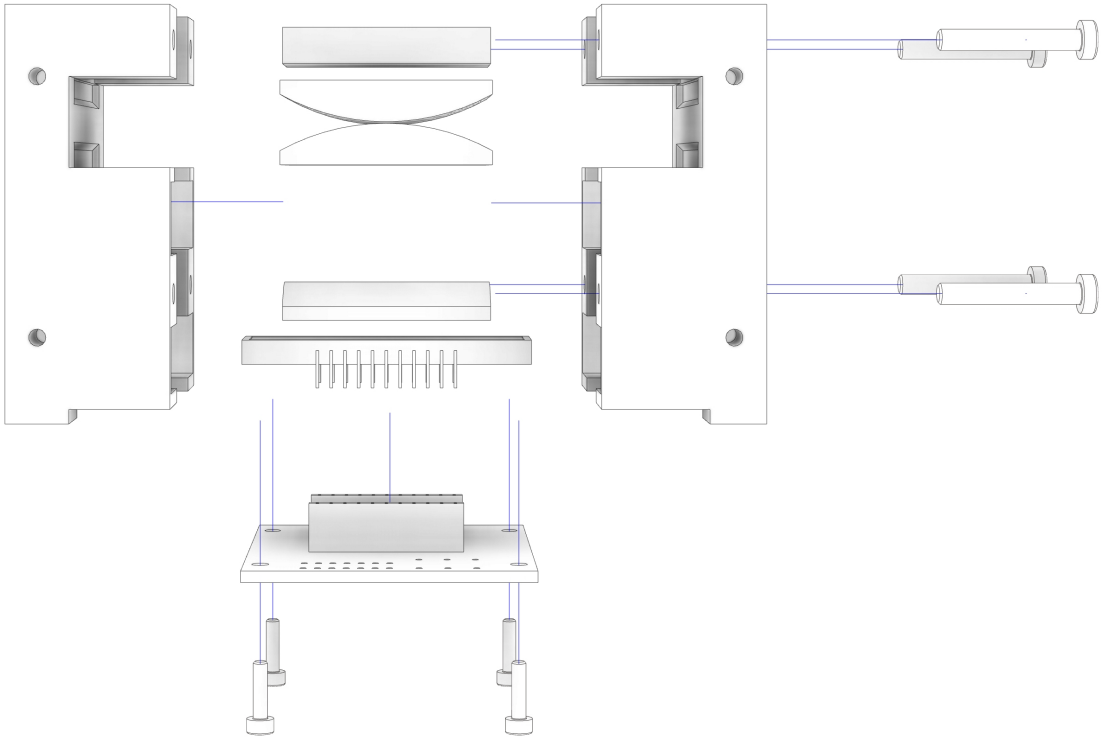


Figure 4.10: A front view of the lens mounting system.

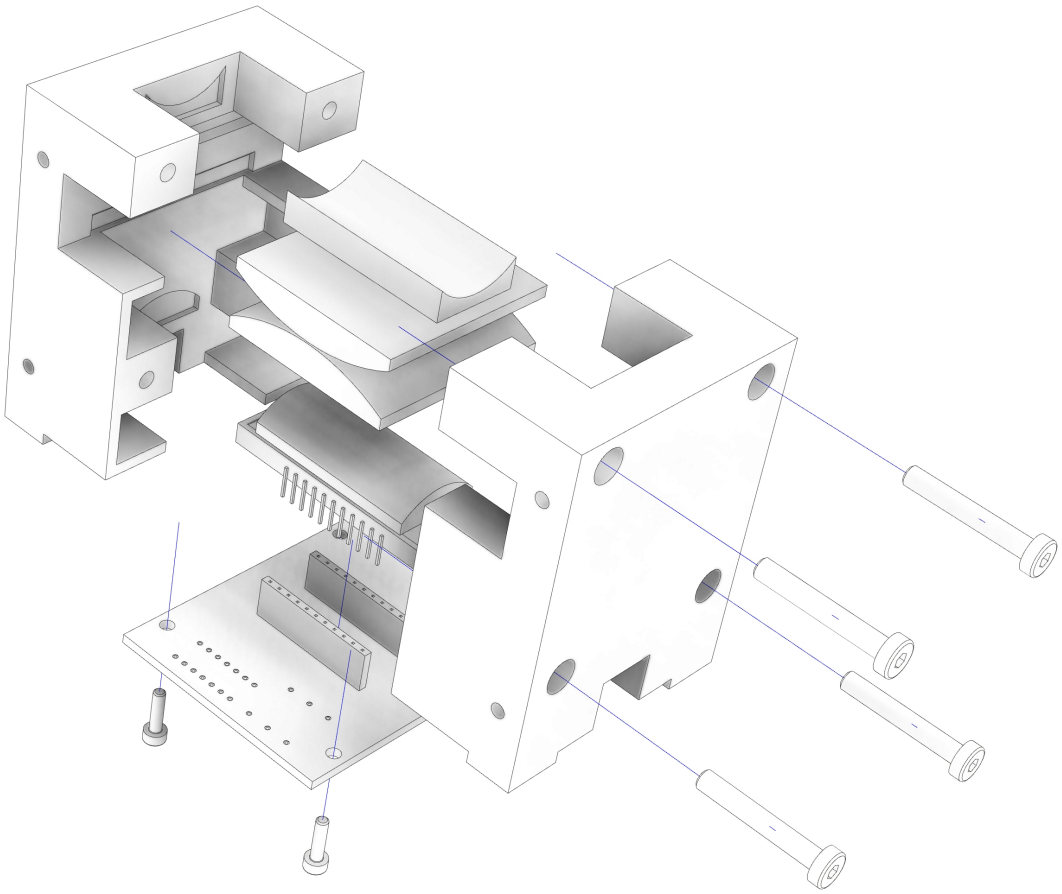


Figure 4.11: A perspective side view of the lens mounting system.

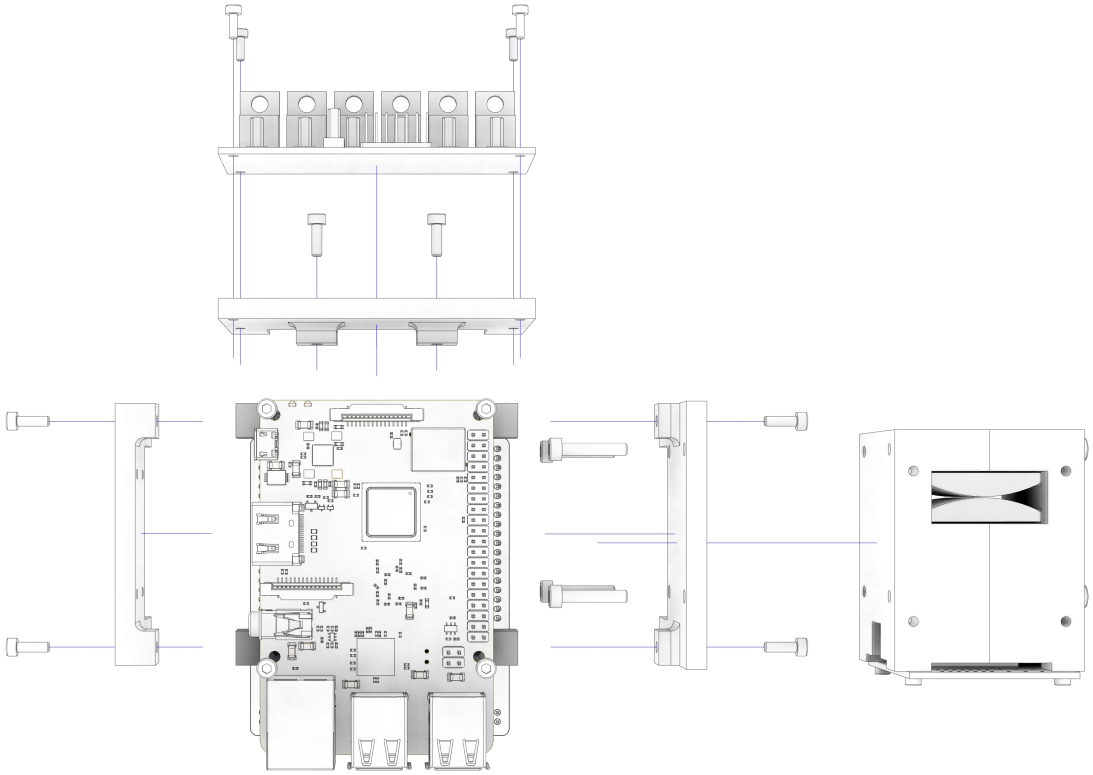


Figure 4.12: A top view of the motherboard.

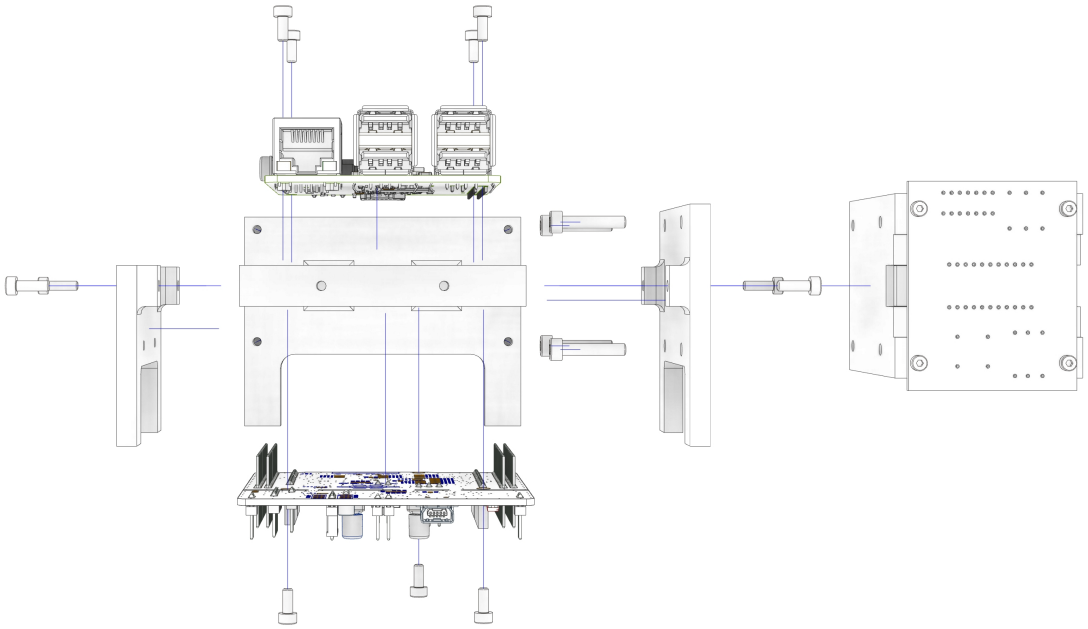


Figure 4.13: A front view of the motherboard.

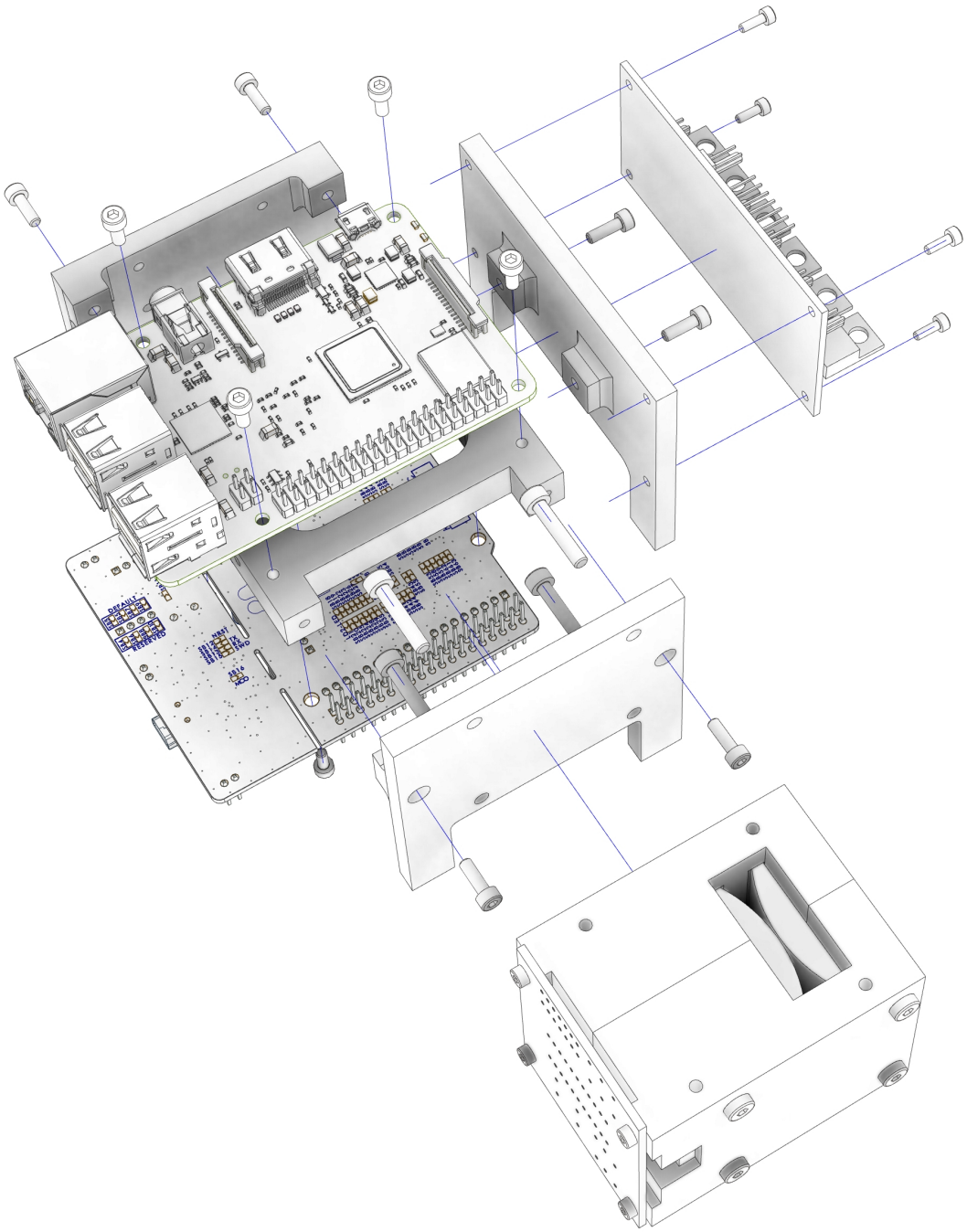


Figure 4.14: A perspective side view of the motherboard.

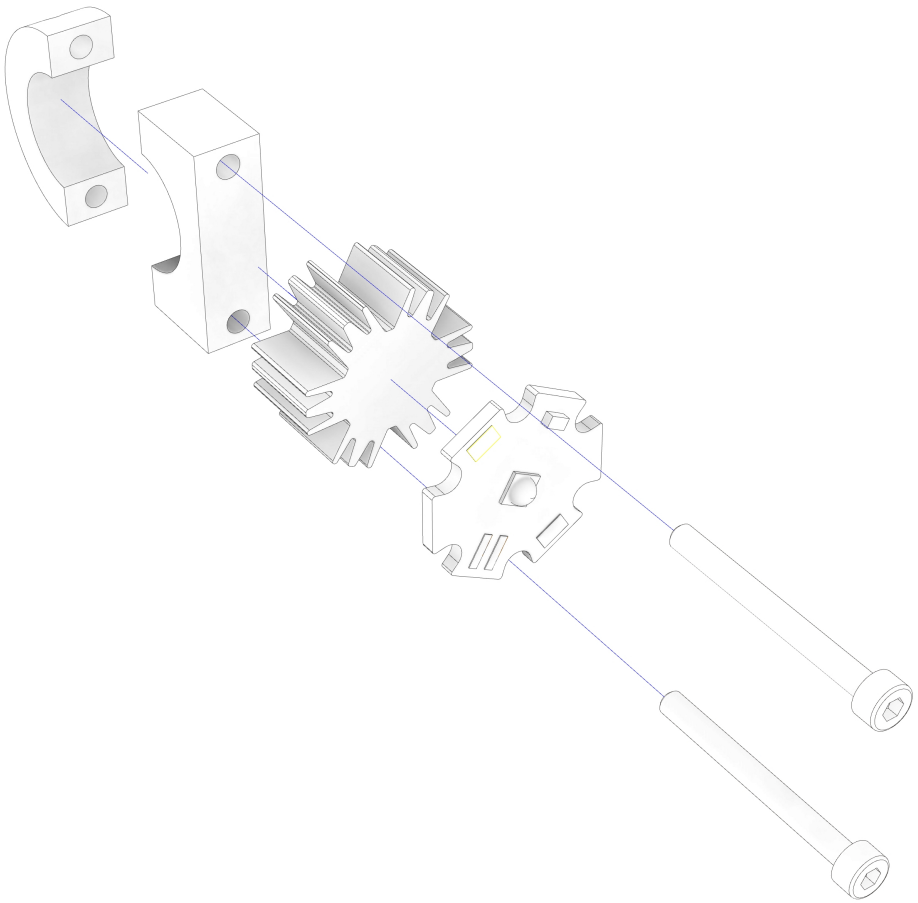


Figure 4.15: A perspective side view of the LED mounting system.

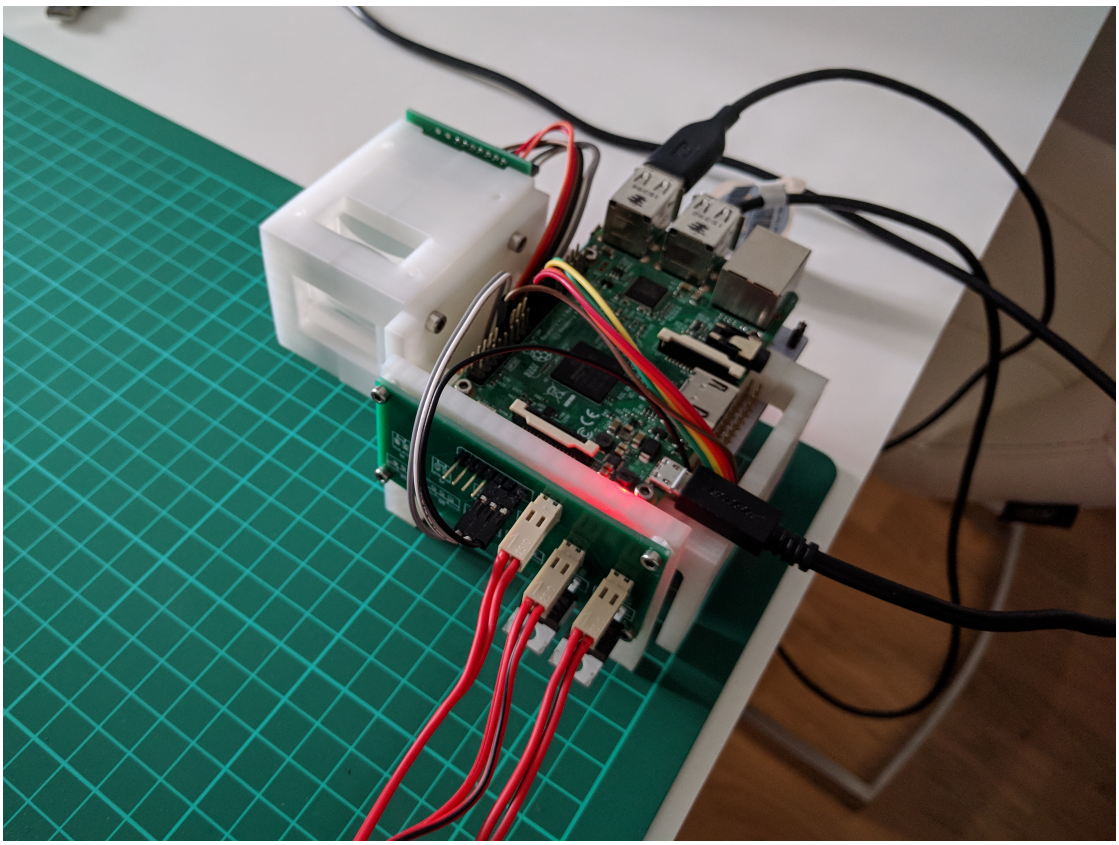


Figure 4.16: A front view of the tracking system prototype. The Raspberry Pi, the LED control PCB with 2 LEDs connected and the lens mounting can be seen.

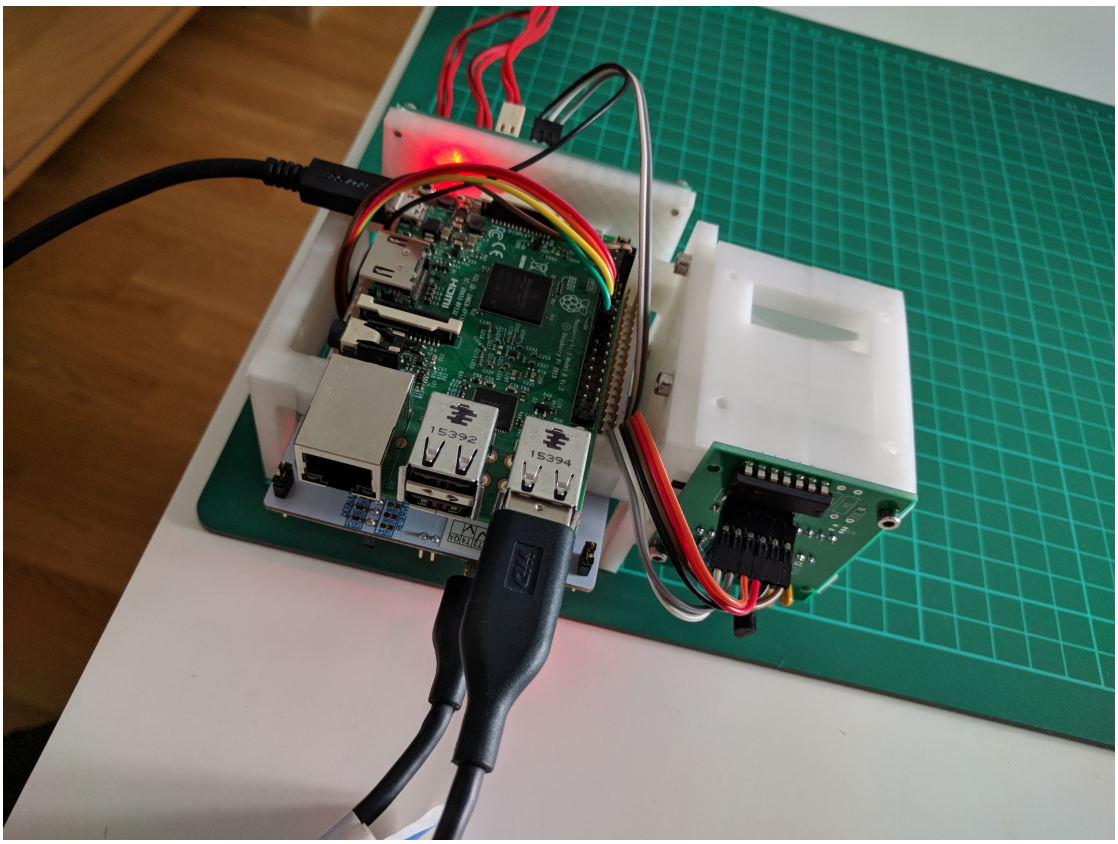


Figure 4.17: A view from the back of the tracking system prototype. The lens mounting with the TCD1304 PCB can be seen.

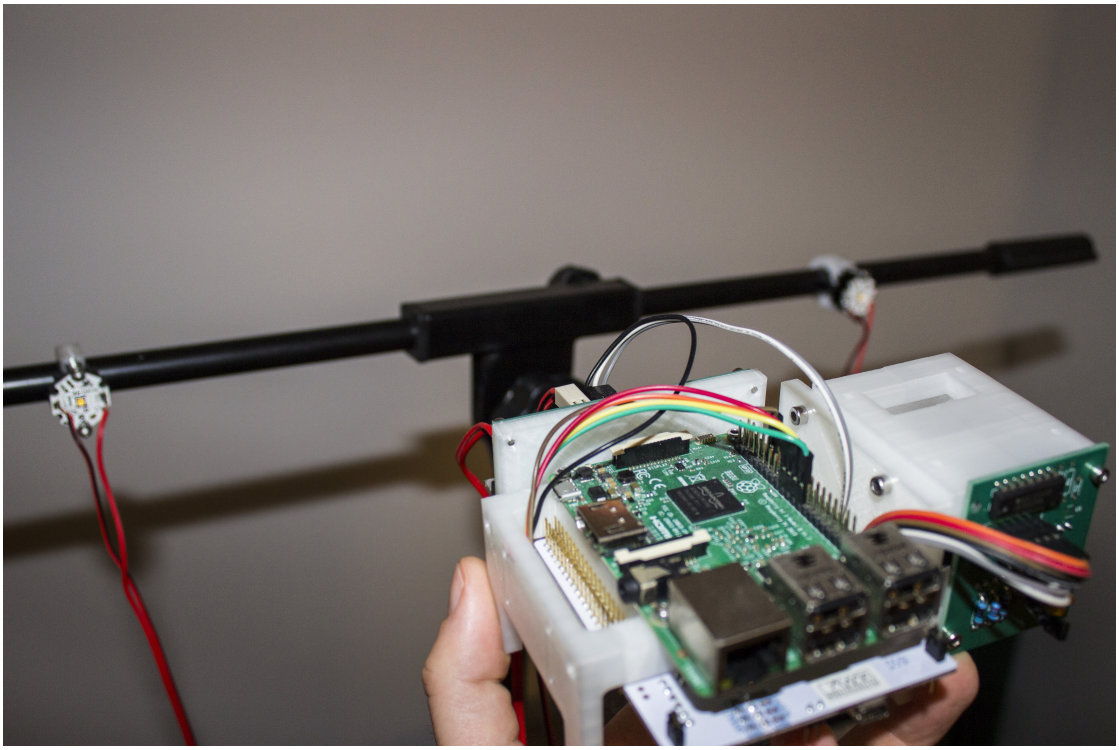


Figure 4.18: A view of the complete tracking system. The two LEDs mounted on a pole are connected to the tracking device via cable.

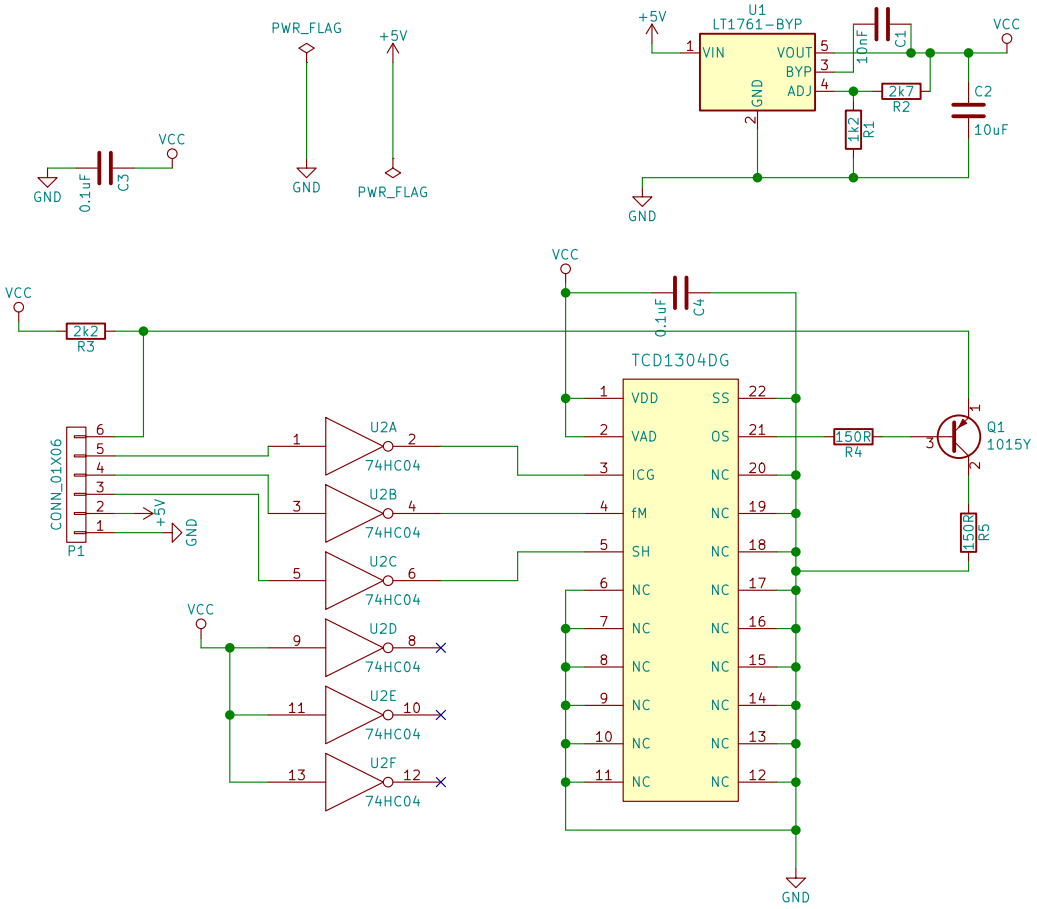


Figure 4.19: The electronic schematic of the TCD1304 linear image sensor. The circuit for reading out the linear image sensor can be seen in the bottom part. In the upper right corner, the circuit for the LT1761 voltage regulator component is shown (retrieved from [49]).

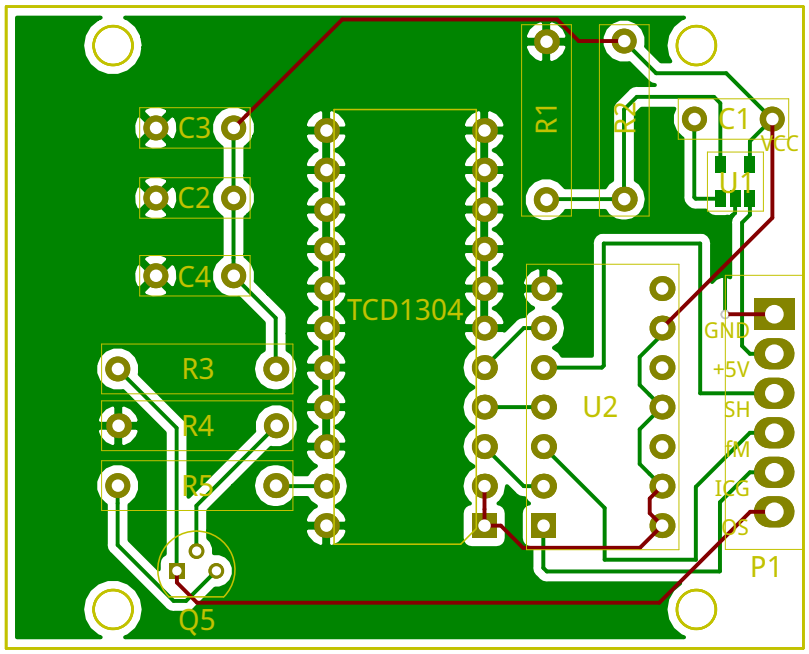


Figure 4.20: The laid out and routed PCB of the TCD1304 linear image sensor, ready for fabrication. The PCB is based on the electronic schematic shown in Figure 4.19.

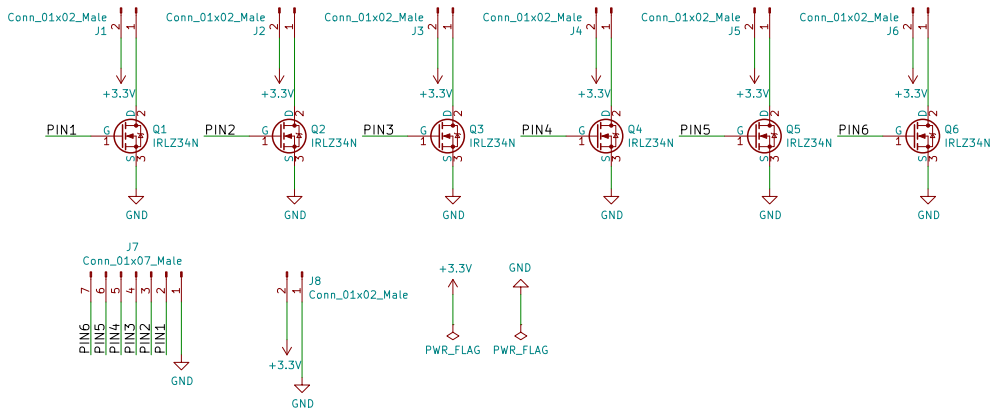


Figure 4.21: The electronic schematic of the LED control circuit.

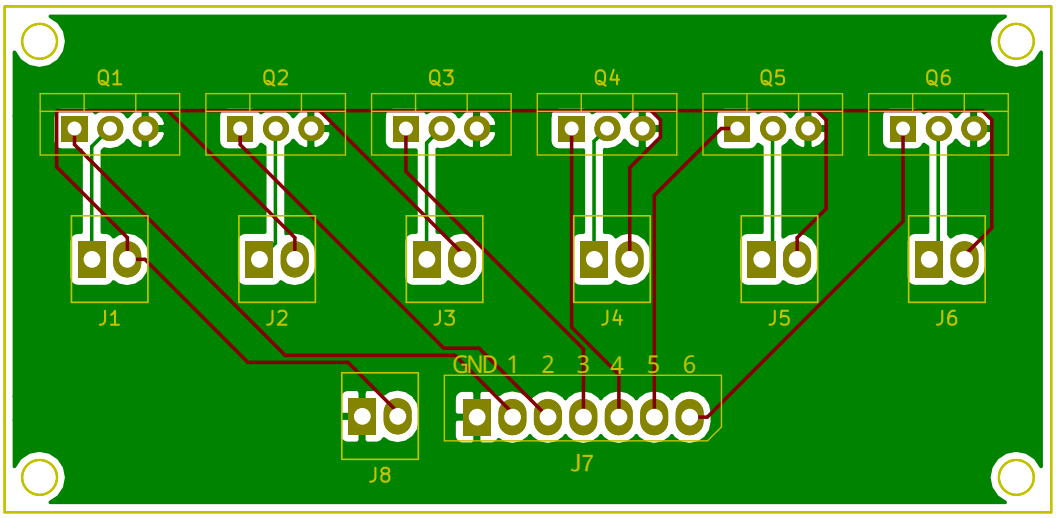


Figure 4.22: The laid out and routed PCB of the LED control circuit, ready for fabrication. The PCB is based on the electronic schematic shown in Figure 4.21.

CHAPTER 5

Evaluation

This Chapter is based on theoretical and practical results from the tracking system prototype and the light ray simulation.

5.1 Definition of the Evaluation Parameters

According to [59], the quality of tracking systems is defined by several parameters:

- **Resolution** is defined as the smallest change a tracking system is able to detect.
- **Accuracy** is the difference between the actual position and the position reported by the tracking system. The better the accuracy, the closer the tracking system captures the actual movements of the user. Accuracy usually degrades with increasing distance of the sensors to the emitters.
- **Jitter** represents the change in the tracking system's output when the tracking device remains stationary. In a perfect system with no jitter, the tracking system would always report the same value when the tracking device is static. The mean value $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ and standard deviation $\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N-1}}$ can be calculated. N is the number of samples and x corresponds to a single sample value.
- **Latency** characterizes the time period between a movement of the tracking device and its recognition by the tracking system. Low latencies are mandatory in VR / AR tracking systems, in order to give immediate feedback and prevent the user from simulation sickness.

5.2 Latency

The TCD1304 sensor is clocked at 4 MHz, but every pixel needs 4 clock cycles to be read-out. There are 3648 px (pixel) in the pixel array, therefore the linear image sensor needs at least $4 \cdot 3648 = 14\,592$ Hz, or $14592/4000000 = 3.648$ ms for a full read-out. Thus, the TCD1304 sensor can be read-out at a maximum frequency of ≈ 274 Hz per second. Prior to reading-out the sensor with the ADC of the microcontroller, the sensor is flushed, to reduce signal noise. The flushing process takes 3 read-out cycles, and another 3 cycles elapse before the actual two read-outs with the corresponding LEDs turned on take place. A total of 8 read-outs are currently needed to compute the absolute position. With the ICG set to 30 000 Hz, yielding a read-out time of 7 ms, 8 read-outs take a total time of $3 \cdot 3.75 + 5 \cdot 7 = 46.25$ ms. The $3 \cdot 3.75$ term in the previous equation yields the flushing procedure, which is executed at an ICG value of 15000 or a speed of 3.75 ms.

Timing a call of the `Otterly-rpiCLI` application with the

```
$ time ./Otterly-rpiCLI -i 30000 -s 100 -n 2
```

shell command results in a timing value of ≈ 50 ms. This means that the remaining ≈ 4 ms are taken by the SPI communication and writing the measurement files to disk. A value of ≈ 50 ms per position estimate yields a tracking frequency of 20 Hz. The execution of the tracking algorithm consumes also time, but is neglectable compared to the time the read-out of the sensor and the SPI transmission takes.

However, by implementing the algorithm on the microcontroller itself, the tracking frequency can be greatly improved. The flushing process of [49] has been adopted in the prototype implementation. However, a total of 6 flushing cycles may be unnecessary for the tracking system. Due to the fact that the spikes of the intensity measurements are easily distinguishable, some signal noise may not deteriorate the tracking process. Therefore, the number of cycles prior to the actual ADC read-out could be greatly reduced. The maximum tracking frequency is $274/2 = 137$ Hz, because at least two light source measurements have to be taken for a single position computation.

5.3 Theoretical Tracking Resolution

Using the light ray simulation, the resolution of the tracking system can be approximated. The TCD1304 sensor has 3648 px. Each Pixel has a length of 8 µm, yielding a total length of $3648 \cdot 8 = 29\,184$ µm or ≈ 29.2 mm. The question is how much a light source has to be moved to trigger a movement of 8 µm on the sensor. By calling the `cast_rays ( lens_width, lens_mat, source_dist, axis_off )` function of the simulation with the right arguments, the deviation dimension can be computed.

The deviation's value depends on the distance of the light source to the optical system. The nearer the tracking system is to the light source, the more accurate it works. The parameters for the `cast_rays()` function are 30 for the `lens_width` parameter, the optical system matrix of two convex lenses with a focal length of 60 mm for the `lens_mat`

parameter and 5000 for the `source_dist` parameter, because the accuracy at 5 m from the optical system is measured. The aim is to find a `focal_offset` value of 0.008 mm. This is achieved by varying the axis_off parameter. Figure 4.6 shows all parameter in an optical system sketch.

An `axis_off` value of 1.35 mm yields a `focal_offset` of 0.008 mm. Therefore, the tracking systems simulated resolution at a distance of 5 m is ≈ 1.35 mm.

The impact of the collimating subsystem can also be simulated. At a distance of 5 m from the optical system, the tracking system is still able to capture light when the light source is moving ≈ 2.7 m up or down the z-axis (non-tracking axis). At a distance of 0.5 m to the tracking system, this margin is narrowed down to ≈ 0.28 m in both directions.

5.4 Tracking Accuracy and Jitter

To test the actual tracking resolution of the prototype, several tests in the field were conducted. The prototype was aligned manually on millimeter paper, in order to take accurate measurements.

At a distance of ≈ 130 cm from the line of LEDs, the prototype was moved separately 100 mm on the x- and on the y-axis. On the x-axis, a 100 mm movement resulted in a x-value difference of 253 px. Therefore, every millimeter step in the real world yields a 2.53 px x-value displacement in the tracking system. Every value on the x-axis is mapped to ≈ 0.4 mm. In the middle of the two measurements, the tracking system computes a value of 2224 px. The expected value is 2217 px. Thus, the accuracy of the the tracking system corresponds to $2224 - 2217 = 7$ px or ≈ 2.8 mm on the x-axis. Figure 5.1 shows the refracted light sources of the base measurement on the TCD1304 sensor. Figure 5.2 shows the refracted light sources at a 100 mm x-axis displacement, compared to the base measurement.

On the y-axis, a 100 mm distance results in a difference of 82 px. Each value is mapped to a y-axis value of ≈ 1.22 mm. The 50 mm midpoint of the 100 mm distance lies at a y-axis value of 1536 px, although it should result in a value of 1541 px. Therefore, the accuracy on the y-axis results in 6.1 mm.

Also, tests at a distance of ≈ 415 cm to the LEDs were conducted. The x-axis test resulted in a difference of 83 px between two measurements taken 100 mm apart. This results in a x-axis tracking resolution of ≈ 1.2 mm. The y-axis test showed a value difference of 12 px in a 100 mm displacement, resulting in a y-axis resolution of ≈ 8.3 mm. At a distance of ≈ 415 cm, the tracking system yielded an accuracy of ≈ 6.8 mm on the x-axis and ≈ 16.6 mm on the y-axis.

Figures 5.3 and 5.4 show the tracking system's jitter. 93 samples were taken, with the tracking device fixed at the same position. On the x-axis, the mean value μ is equal to a value of ≈ 0.05 mm. The standard deviation σ has a value of ≈ 0.436 mm. On the y-axis, the mean value μ is equal to ≈ -0.46 mm. The standard deviation σ results in ≈ 1.11 mm

5.5 Discussion

Borrego et al. [60] evaluated the tracking systems of the HTC Vive and the Oculus Rift VR headsets.

For the HTC Vive, they estimated an accuracy of (0.75 ± 0.69) cm and a jitter of (0.23 ± 0.33) mm at a tracking height of 1.3 m. The Oculus Rift supplies similar values. It has an accuracy value of (0.76 ± 0.53) cm and a jitter of (0.12 ± 0.18) mm, tested at the same height as the HTC Vive. Both values were measured in the inside of the tracking area, leading to more accurate results.

Comparing these values to the results of the presented prototype, it can be seen that the prototype implementation delivers competitive results. Even at a distance of 415 cm from the line of LEDs, the tracking prototype yields a better accuracy on the x-axis than the HTC Vive and Oculus Rift tracking systems. The x-axis jitter is also superior to the jitter of the reference tracking systems, but there is still room for improvements. A more robust spike-detection algorithm would reduce noise and further improve tracking jitter. However, the accuracy of the y-axis decreases drastically with the distance to the LEDs. At a distance of 415 cm to the LEDs the resolution is only a seventh of the resolution at 130 cm distance. Also, the tracking system is very sensitive to rotation. Small rotational misalignments of the tracking device lead to severe tracking errors.

Although the prototype is only able to track two dimensions without rotation, a theoretical system with full 6-DOF pose tracking would fulfill the qualitative requirements of a VR tracking system, based on the results of the evaluation. The latency can be reduced drastically by implementing the algorithm on the microcontroller side, and the accuracy and resolution values are comparable to state-of-the-art tracking technology. A second sensor could provide three-dimensional tracking and rotation information.

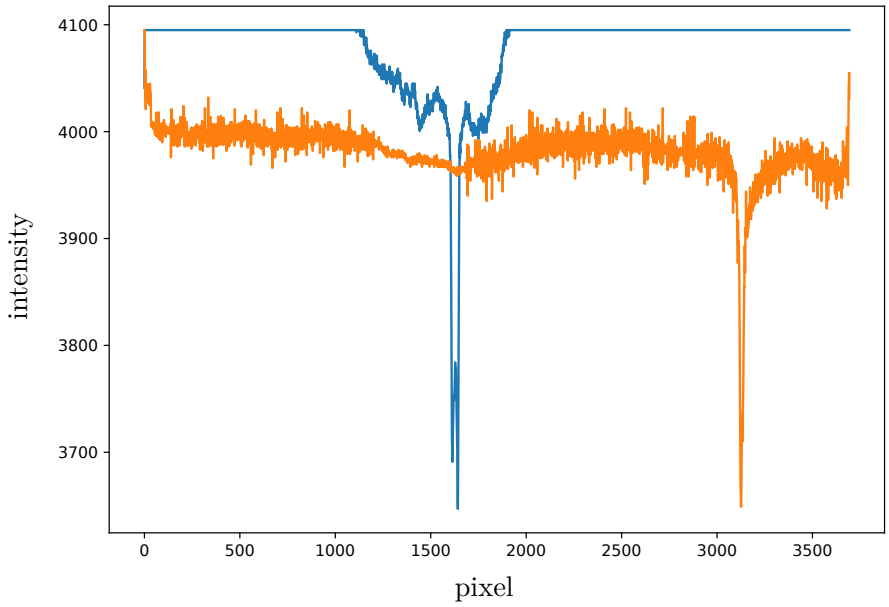


Figure 5.1: The light intensity chart of the base measurement.

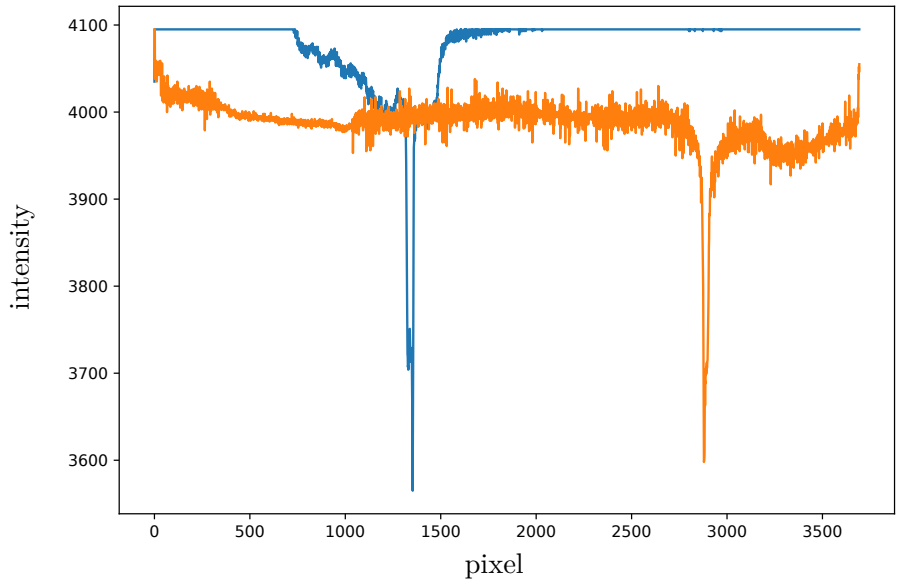


Figure 5.2: Light intensities measured at a 10 cm displacement on the x-axis relative to the base measurement.

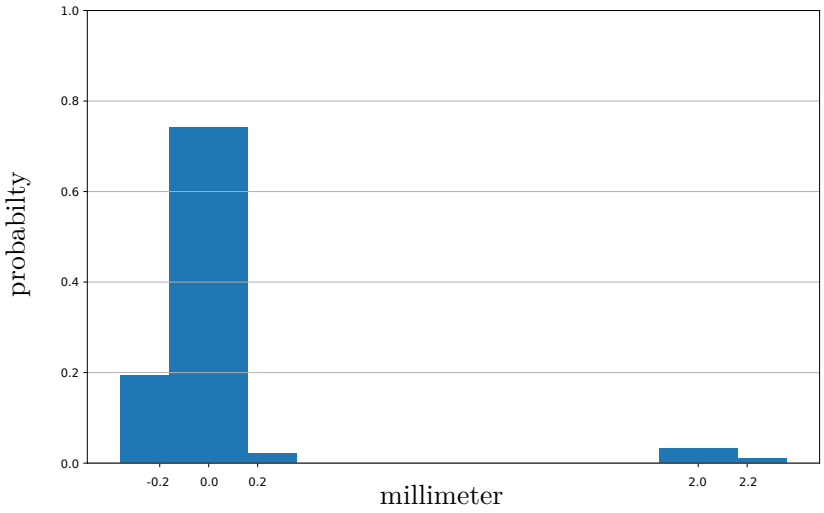


Figure 5.3: The histogram shows the tracking system's jitter on the x-axis, with a sample size of 93 measurements.

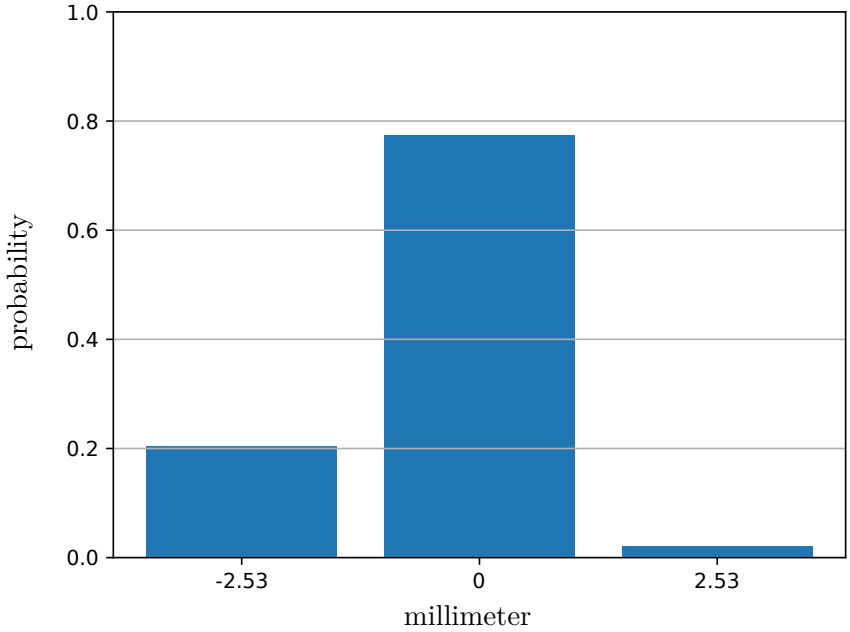


Figure 5.4: The histogram shows the tracking system's jitter on the y-axis, with a sample size of 93 measurements.

CHAPTER 6

Conclusion and Future Work

In this thesis, a new kind of tracking system has been designed, simulated and implemented. The resulting prototype is an inside-out tracking system based on a linear image sensor and a lens system. These parts are readily available and affordable. Positions are computed by fundamental optical principles. When the tracking device moves, the refraction angle changes and therefore illuminates a different part of the linear image sensor. With a single linear image sensor, two dimensional tracking can be realized.

In the simulation part, a light ray simulation using ray transfer matrices has been implemented. With this simulation, the lens characteristics of a given tracking area can be calculated and different lens arrangements can be tested. The simulation makes it also possible to compute other important parameters like the necessary distance between LEDs and the resolution of the tracking system.

After the necessary components of the optical system were selected and ordered, a functional prototype has been implemented. The prototype is based on a STM32F401RE microcontroller, which is able to do the low level tasks like reading-out the linear image sensor and controlling the LEDs. The LEDs have to be toggled at the sensor's integration period. Only one light source is turned on at a given moment in time. A linear image sensor has been used to read the light intensities of the LEDs. A microcontroller has been used to drive the linear image sensor and read its pixel values. The pixel values are sent to a Raspberry Pi via the SPI interface, to run higher-level algorithms and calculate the current absolute position.

A functional prototype has been designed and fabricated. It consists of several 3D printed parts and PCBs. The lenses were aligned in a lens mounting system, with the distances to each other calculated in the simulation step. 3D printed mounting parts were also designed for the microcontroller, the Raspberry Pi and the PCBs, to build a tracking device that is easily usable. Two PCBs were designed, one for the electronic components needed to drive the TCD1304AP linear image sensor and one to toggle the LEDs with

transistors. Two LEDs, wired to the prototype, provide the light source for the tracking device. The tracking device is able to compute absolute two dimensional position data on a premapped tracking area.

6.1 Possible Improvements

The actual qualities of the tracking system have to be evaluated and tested systematically with external tracking hardware, to get accurate ground truth data.

Due to fact that the tracking system is still in an early prototyping phase, there is a lot of room for improvements. To improve the speed of the tracking device, the algorithm should be implemented on the microcontroller directly. This step removes the need for the Raspberry Pi and shortens the tracking latency by the time the SPI communication process takes. The flushing process of the TCD1304, which is implemented in the current firmware, should also be examined. Flushing is required to get noiseless intensity values from each reading. However, it might not be necessary to flush the sensor three times before every sensor read-out.

The spike recognition itself has to be made more robust, potentially by utilizing Gaussian filters and numerical differentiation instead of just picking the smallest intensity value. A more sophisticated algorithm would also improve the tracking system's jitter and its overall accuracy. Using numerical differentiation for spike recognition would have the advantage of calculating the width of a single spike (see Figure 4.2), which can be used as an additional variable in the tracking system. Single spike tracking can also be utilized to enhance the tracking area to spaces where only single light source measurements can be taken. When the tracking area is large, many LEDs are needed to get a continuous tracking experience. Toggling many LEDs, some of them may not even illuminate the tracking device, reduce the system's latency and responsiveness. Therefore, a sophisticated strategy is needed to toggle only the LEDs wich are actually needed for the position computation. This approach can only work under specific assumptions, like restraints in the speed the tracking device is allowed to move. The system has to know in which direction the tracking device is moving, in order to turn on the appropriate LEDs. Implementing such a feature would result in low latency tracking even in large tracking areas.

6.2 Different Hardware Components

To make the tracker more independent of wires, the LEDs should be toggled by a wireless communication protocol. The protocol has to be reliable and communicate fast, in order not to increase the system's latency. Fresnel lenses are more compact than their thicker counterparts, but have the same optical features. These type of lenses can be used to make the lens system and the overall tracking device more compact.

There are many different linear image sensors on the market. Some of them, like the *Toshiba TCD1703C* sensor, have more pixel elements than the TCD1304, but are also bigger in size. By using such a sensor, the accuracy of the tracking system can be increased. The tracking area could also be enhanced without sacrificing tracking precision.

6.3 3D Tracking and Rotation

In order to extend its usefulness, the tracking system has to be enhanced to three dimensions. For three dimensional tracking, a second linear image sensor is required. It has to be mounted perpendicular to the existing sensor, with an additional optical system. An additional sensor will be able to track the z-axis and the y-axis. Additional LEDs have also to be placed in the environment, perpendicular to the LED line of the 2D tracking environment. This would result in a two dimensional LED wall. The integration of rotation tracking, although tricky to implement, should be possible with the data of two linear image sensors and enough tracking information from different light sources.

APPENDIX A

Listings

```
1  void flush_CCD()
2  {
3    /* set exposure very low */
4    ICG_period = 15000;
5    SH_period = 20;
6
7    /* disable ICG (TIM5) and SH (TIM2) before reconfiguration */
8    TIM_Cmd(TIM2, DISABLE);
9    TIM_Cmd(TIM5, DISABLE);
10
11   /* reset flags and counters */
12   CCD_flushed = 0;
13   ICG_pulse_counter = 0;
14
15   /* reconfigure TIM2 and TIM5 */
16   TIM_ICG_SH_conf();
17
18   /* block until CCD is properly flushed */
19   while(CCD_flushed == 0);
20 }
```

Listing A.1: The `flush_CCD()` function

```
1  void TIM5_IRQHandler(void)
2  {
3    if (TIM_GetITStatus(TIM5, TIM_IT_Update))
4    {
5      /* clear TIM5 update interrupt */
6      TIM_ClearITPendingBit(TIM5, TIM_IT_Update);
7
8      if (ICG_pulse_counter == 6)
9      {
10       /* reset SH_pulse_counter */
11       SH_pulse_counter = 0;
12
13       /* reset LEDs */
14       GPIO_ResetBits(GPIOC, GPIO_Pin_4);
15       GPIO_ResetBits(GPIOC, GPIO_Pin_5);
16
17       /* restart TIM4 as this gets the ADC running again */
18       TIM4->CR1 |= TIM_CR1_CEN;
19     }
20     else if (ICG_pulse_counter == 3)
21     {
22       CCD_flushed = 1;
23     }
24     ICG_pulse_counter++;
25
26     /* prevent overflow */
27     if (ICG_pulse_counter > 10)
28       ICG_pulse_counter = 10;
29
30     /* keep track of integrations to perform */
31     if ((Integration_count > 0) && (ICG_pulse_counter > 6))
32     {
33       /* decrease counter */
34       Integration_count--;
35
36       /* restart ADC at next ICG-pulse*/
37       ICG_pulse_counter = 6;
38     }
39     /* flash the led to the beat of ICG */
40     GPIOA->ODR ^= GPIO_Pin_5;
41     /* reset raspberry pi communication pin */
42     GPIOB->BSRRH = (1<<2);
43   }
44 }
```

Listing A.2: The ISR of the ICG interrupt (timer 5)

```
1  void TIM2_IRQHandler(void)
2  {
3    /* calculate the number of SH pulses */
4    uint16_t SH_pulses = ICG_period / SH_period;
5    if (TIM_GetITStatus(TIM2, TIM_IT_Update))
6    {
7      /* clear TIM2 update interrupt */
8      TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
9      if(ICG_pulse_counter == 6)
10     {
11       SH_pulse_counter++;
12     }
13
14     /* turn on LEDs 5 interrupts before read out */
15     if(SH_pulse_counter == (SH_pulses - 5))
16     {
17       /* toggle PC4 LED */
18       if(Integration_count == 1)
19       {
20         GPIO_SetBits(GPIOC, GPIO_Pin_4);
21       }
22       /* toggle PC5 LED */
23       if(Integration_count == 0)
24       {
25         GPIO_SetBits(GPIOC, GPIO_Pin_5);
26       }
27     }
28   }
29 }
```

Listing A.3: The ISR of the SH interrupt (timer 2)

List of Figures

List of Tables

Bibliography

[1] U.S.A. Department Of Defense. Global Positioning System Standard Positioning Service Performance Standard. 2008.

[2] Ahmed El-Rabbany. *Introduction to GPS - The Global Positioning System*. 2002.

[3] Henrik Hautop Lund, Esther de Ves Cuenca, and John Hallam. A Simple Real-Time Mobile Robot Tracking System. pages 1–8, 1996.

[4] Markus Windolf, Nils Götzen, and Michael Morlock. Systematic accuracy and precision analysis of video motion capturing systems-exemplified on the Vicon-460 system. *Journal of Biomechanics*, 41(12):2776–2780, 2008.

[5] Ghang Lee, Hong Hyun Kim, Chi Joo Lee, Sung Il Ham, Seok Heon Yun, Hunhee Cho, Bong Keun Kim, Gu Taek Kim, and Kyunghwan Kim. A laser-technology-based lifting-path tracking system for a robotic tower crane. *Automation in Construction*, 18(7):865–874, 2009.

[6] HTC Vive. `https://www.vive.com`. Online; Accessed: 2018-07-12.

[7] Optitrack. `https://optitrack.com`. Online; Accessed: 2018-07-12.

[8] Using HTC VIVE outdoors - Will it work? `https://www.youtube.com/watch?v=920S62fDALA`. Online; Accessed: 2018-07-12.

[9] Hirotake Ishii. Augmented Reality : Fundamentals and Nuclear Related Applications. *Nuclear Safety and Simulation*, 1(4):316–327, 2010.

[10] Acer. `http://www.acer.com`. Online; Accessed: 2018-08-06.

[11] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS – Global Navigation Satellite Systems - GPS, GLONASS, Galileo, and more*. 2007.

[12] Eric Foxlin. Pedestrian Tracking with Shoe-Mounted Inertial Sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[13] Gabriele Bleser and Didier Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Computers and Graphics*, 33(1):59–72, 2009.

[14] Jonathan Kelly and Gaurav S. Sukhatme. Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration. *International Journal of Robotics Research*, 30(1):56–79, 2011.

[15] Eric Foxlin, Michael Harrington, and George Pfeifer. Constellation: A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, 98:371–378, 1998.

[16] David Nistér, Oleg Naroditsky, and James Bergen. Visual Odometry. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:652–659, 2004.

[17] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. *IEEE International Conference on Robotics and Automation*, 2014.

[18] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. 2016.

[19] Hirokazu Kato and Mark Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94, 1999.

[20] An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015.

[21] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[22] Thomas Pintaric and Hannes Kaufmann. Affordable Infrared-Optical Pose-Tracking for Virtual and Augmented Reality. *Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop, IEEE VR 2007*, pages 44–51, 2007.

[23] Miguel Ribo, Axel Pinz, and Anton L. Fuhrmann. A new optical tracking system for virtual and augmented reality applications. *Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference*, 3:1932–1936, 2001.

[24] Tracy D. McSheery, John R. Black, Scott R. Nollet, Jack L. Johnson, and Vinay C. Jivan. Distributed-processing motion tracking system for tracking individually modulated light points, 1997. US Patent 6324296B1.

[25] Velio Macellari. CoSTEL: a computer peripheral remote sensing device for 3-dimensional monitoring of human motion. *Medical and Biological Engineering and Computing*, 21(3):311–318, 1983.

[26] Giuseppe Bianchi, Fabio Gazzani, and Velio Macellari. The COSTEL system for human motion measurement and analysis. *Image-Based Motion Measurement*, 1356, 1990.

[27] Richard D. Bucholz, Kurt R. Smith, Jaimie Henderson, Lee McDurmont, and Dean Schulz. Intraoperative localization using a three-dimensional optical digitizer. *Proceedings of SPIE - The International Society for Optical Engineering*, 1894, 1993.

[28] Nestor Voronka and Charles J. Jacobus. Optical body tracker, 1999. US Patent 6801637B2.

[29] Gary Bishop. *Self-Tracker: A Smart Optical Sensor on Silicon.* PhD thesis, 1984.

[30] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments. *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 1 – 10, 1999.

[31] Henry Fuchs, Joe Duran, and Brian Johnson. A System for Automatic Acquisition of Three-Dimensional Data. *Proceedings of the 1977 National Computer Conference*, pages 49–53, 1977.

[32] David Gualino, Michel Parent, and Michael Uchanski. Autonomous Lateral Control of a Vehicle Using a Linear CCD Camera. *IEEE International Conference on Intelligent Vehicles*, pages 69–73, 1998.

[33] Jian Wu and Qiuting Wen. The method of realizing the three-dimension positioning based on linear CCD sensor in general DSP chip. *IEEE Engineering in Medicine and Biology Society*, 30:2302–2305, 2008.

[34] Yinghui Hu, Feng Yuan, Kai Li, and Yan Wang. Linear CCD Attitude Measurement System based on the Identification of the Auxiliary Array CCD. *Proceedings of SPIE - The International Society for Optical Engineering*, 9671, 2015.

[35] W. Cuypers, N. Van Gestel, A. Voet, J. P. Kruth, J. Mingneau, and P. Bleys. Optical measurement techniques for mobile and large-scale dimensional metrology. *Optics and Lasers in Engineering*, 47(3-4):292–300, 2009.

[36] Filip Dominec. Design and construction of a digital CCD spectrometer. Technical report, 2010.

[37] Diederick C. Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research. *i-Perception*, 8(3):1–23, 2017.

[38] McGraw-Hill. *McGraw Hill Encyclopedia of Science & Technology.* McGraw-Hill Professional, 2007.

[39] Frank L. Pedrotti, Leno M. Pedrotti, and Leno S. Pedrotti. *Introduction to Optics*. 2006.

[40] Eugene Hecht. *Optics*. 2016.

[41] Wikipedia - The Free Encyclopedia. `https://www.wikipedia.org`. Online; Accessed: 2018-08-28.

[42] Anthony Gerrard and James M. Burch. *Introduction to Matrix Methods in Optics*. 1975.

[43] Gerald C. Holst. *CCD Arrays Cameras and Displays*. 1998.

[44] Imaging & Microscopy. `https://www.imaging-git.com`. Online; Accessed: 2018-07-17.

[45] Alen Lustica. CCD and CMOS Image Sensors in new HD Cameras. *Proceedings ELMAR-2011*, (September):133–136, 2011.

[46] Toshiba. Application Notes and Technical Articles - The CCD Image Sensors, 1970.

[47] Ray Fontaine. The State-of-the-Art of Mainstream CMOS Image Sensors. *International Image Sensor Workshop*, page 4, 2015.

[48] Toshiba. TCD1304AP Datasheet, 1997.

[49] The linear CCD module (Firmware). `https://tcd1304.wordpress.com`. Online; Accessed: 2018-07-20.

[50] Tran Quoc Khanh, Peter Bodrogi, Qunag Trinh Vinh, and Holger Winkler. *LED Lighting - Technology and Perception*. 2015.

[51] Conrad Online Store. `https://www.conrad.at`. Online; Accessed: 2018-08-05.

[52] Günther Gridling and Bettina Weiss. *Introduction to Microcontrollers*. 2007.

[53] Geoffrey Brown. *Discovering the STM32 Microcontroller*. 2013.

[54] STM32. STM32F401xD STM32F401xE Datasheet - production data, 2015.

[55] DX - deal extreme. `http://www.dx.com`. Online; Accessed: 2018-07-20.

[56] Derek Molloy. *Exploring Raspberry Pi - Interfacing to the real world with embedded Linux*. Wiley, 2016.

[57] The Raspberry Pi Foundation. `https://www.raspberrypi.org`. Online; Accessed: 2018-07-26.

[58] STM32. *Description of STM32F2xx Standard Peripheral Library*. 2011.

[59] Grigore C. Burdea and Philippe Coiffet. *Virtual Reality Technology*. Wiley-IEEE Press, 2003.

[60] Adrián Borrego, Jorge Latorre, Mariano Alcañiz, and Roberto Llorens. Comparison of Oculus Rift and HTC Vive: Feasibility for Virtual Reality-Based Exploration, Navigation, Exergaming, and Rehabilitation. *Games for Health Journal*, 7(2), 2018.