Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

BGIA

Institut für Arbeitsschutz der
Deutschen Gesetzlichen Unfallversicherung

Master Thesis

”Industrial jointed arm robot evading dynamic objects”

by:

Dipl.-Ing. (FH) Björn Ostermann

A thesis submitted to the
University of Applied Sciences Bonn-Rhein-Sieg
for the degree of
Master of Science in Autonomous Systems

Hochschule Bonn-Rhein-Sieg
Department of Computer Science
Master of Autonomous Systems
Grantham-Allee 20
53757 Sankt Augustin

Referee and Tutor:	**Dr. Michael Huelke**
Referee:	**Prof. Dr. Dietmar Reinert**
Referee:	**Prof. Dr. -Ing. Gerhard K. Kraetzschmar**

Submitted: Sankt Augustin, 13. January 2009

Acknowledgement

Several people have contributed to the success of this work, whom I’d like to thank at this point.

I’d like to thank my colleagues at the BGIA, who contributed to this work by professional discussions and hints on important literature, as well as the construction of the robot’s workplace. Especially I’d like to thank my supervisor Dr. Michael Huelke for his constant support and Dr. Michael Schaefer for making this work possible.

Also I’d like to thank my supervising professors, Prof. Dr. Dietmar Reinert and Prof. Dr.-Ing. Gerhard K. Kraetzschmar, for their support and ideas on the improvement of the project.

I thank Jörn von der Lippe, who worked on a parallel master project on the same 3D camera, for tossing back and forth ideas on algorithms and programming (e.g. see chapter 5.2.1) for one and a halve year and for letting me reuse some of his code (e.g. see chapter 5.1.2).

Furthermore I thank the people at Reis GmbH & Co. KG Maschinenfabrik, especially Dipl.-Ing. Franz Som and Michael Pfeiffer, for their detailed information on their hard- and software and for contributing the robot, which was used in this project.

In addition I thank my proofreaders for their comments and suggestions.

Statutory Declaration

With this I declare that the present Project Report was made by my self. Prohibited means were not used and only the aids specified in the Master Thesis were applied. All parts which are taken over word to word or analogous from literature and other publications are quoted and identified.

Björn Ostermann

St. Augustin, 13. January 2008

Abstract

Most industrial robots used today work behind fences. Even in those cases where the fences are substituted by electro sensitive protective equipment (ESPE), the position of the separation between human's and robot's workspace is invariable.

This work presents an integrated solution that allows flexible human-robot workspace separation. The algorithms, responsible for maintaining the separation, are based on the tracking of objects that enter the workspace. These algorithms are divided into proximity monitoring and path planning. The demonstrator, developed in this work, consists of a 3D Camera, an industrial jointed arm robot and a desktop computer.

The main part of this work focuses on the algorithms that evaluate the collected data and compute a collision free path for the robot. A software interface has been developed, that finds intrusions in the robot's vicinity and changes the robot's path accordingly.

It is also shown, which integration level has been reached and which problems could not be covered in the course of this project. The applicability of such a system in praxis is discussed, depending on the current safety standards and available technology.

Contents

1 Introduction

The first industrial jointed arm robot was invented in 1954 and first used in 1962 for industrial production [64]. At the end of the year 2008 the IFR Statistical Department, a group hosted by the VDMA (association of German machine and machinery manufacturer) [59], estimates the amount of operational industrial robots at over one million [58]. About 140,000 of those robots are estimated to be in use in Germany [58].

Due to the regulations of European law and standards, most of those robots work behind rigid fences to ensure the safety of human operators in the vicinity. With the approval of laser scanners for safety applications in 1998 [60] and SafetyEye [47], a camera system, in 2006, some fences have been replaced by electro sensitive protective equipment (ESPE), but the rigid borders between human and robot still remain.

The constantly increasing demand for the possibility of collaborative tasks, according to [70] a cooperative task between human and robot, also increases the demand on systems with less rigid separation mechanisms. Experimental research to free the robots from their fences started in the 80s [3], after the first death related to a robot accident occurred in 1982 [1]. Although the number of projects working on this topic is constantly increasing, until today no application is known to be approved for industrial production.

The work on the presented project started in 2007. Earlier studies [77] provided an insight into possible approaches towards the mechanisms of human robot collaboration. As a result from those studies a new approach to the problem was formulated and tested [78].

This project describes the new approach, a flexible borderline control of “human-robot workspace separation”, and shows its implementation into a demonstrator at the BGIA – Institute for Occupational Safety and Health. The demonstrator consists of a 3D time of flight camera, a jointed arm robot with a safety control system and a PC. The data collected from the camera and the robot are processed in the PC, with the result that the robot evades persons and objects.

1.1 Problem Formulation

Robots in today’s industries work behind fences. This reduces their field of application, especially when collaboration between humans and robots is needed.

Certain approaches towards a free robot have been made already, but until today, no practical solution usable in praxis is known.

The BGIA, hosting this project, is a leading research institute in the case of industrial safety related applications. One important goal of the division “Accident Prevention / Product Safety” is to find a solution for the problem that allows maximum flexibility of the robot with minimum risk to the human operators.

Humans usually do not feel comfortable working alongside a robot without physical borders. Therefore, as a secondary goal, a method has to be found, that increases the trust in a freely accessible system.

1.2 Task Description

The goal of the framework project, consisting of this project and previous work [77] [78], is to provide a demonstrative workplace, showing an industrial robot working in flexible boundaries while evading dynamic objects.

The workplace set up in this project is to be demonstrated with real human operators.

To achieve the necessary safety, the robot is only allowed to drive at a safely monitored speed under the constant supervision of a human operator. This allows the use of equipment that does not fulfil the safety standards needed in real industrial automatic production. Therefore this work is only a feasibility study and not designed at this stage for industrial production.

The required hardware, a robot, a sensor and a logical unit, was chosen in the previous work [78], where they were handled separately. In this work the hardware is to be combined in order to act as a unit. Therefore a controlling software program is to be developed, which implements and tests the desired flexible boundaries.

On the software side, algorithms of previous work [77], have to be implemented and tested. Where those algorithms lack feasibility or completeness, new algorithms have to be designed, implemented and tested. The performance of the developed solution has to be shown, with respect to the safety of the system.

On the hardware side, a 3D camera provides data about the workspace which have to be evaluated by a controlling program running on a PC. Based on the analysis of the acquired 3D data the program has to send commands to the attached robot, controlling its movements. The optimal spatial arrangement of the hardware has to be determined and realized. The spatial arrangement of the 3D camera thereby has to be flexible, to test different perspectives on the workplace.

Additionally to the functional demonstrator, a depiction of the acquisition and analysis of the data has to be programmed that allows a live observation of the single steps of the program.

All regulations and standards concerning safety have to be considered.

2 Overview on human-robot collaboration

2.1 State of the art

Since the invention of the joint arm robot in 1954 by George Devol, founder of the company Unimation, and its first usage in 1962 by General Motors [64], humans have been separated from industrial robots to assure their safety. Until the end of the 20th century this separation was achieved by rigid fences.

A major step towards a fenceless workplace was achieved in 1996, when the sourcecode SICK programmed for lasercannes was approved or safety related applications [60] by the BGIA, and later in 1998, when the first laser scanner, the PLS 101-312 by SICK, was approved for safety related applications [60] by the BGIA.

The next step was achieved in 2006, when the company Pilz GmbH & Co. KG introduced a system, called SafetyEye [47], which is based on cameras (see Figure 1).

With respect to safety, both systems, laser scanner and SafetyEye, can differentiate between three different situations. Therefore their field of view can be separated into non protective and protective fields. The protective field thereby is separated into a warning and a safety area. If there is no object in the protective field, the robot is fully functional. If an object enters the warning (yellow) area, the speed of the robot is reduced, and if an object enters the safety (red) area, the robot is stopped.

In case of the SafetyEye, no action is taken for objects in the working (blue) area, since intruding objects can not be distinguished from working material or the robot itself.

Although laser scanner and cameras allow easy changes of these boundaries between production cycles, during production the human and the robot are still separated by a nonflexible invisible fence, and no real collaboration is possible.

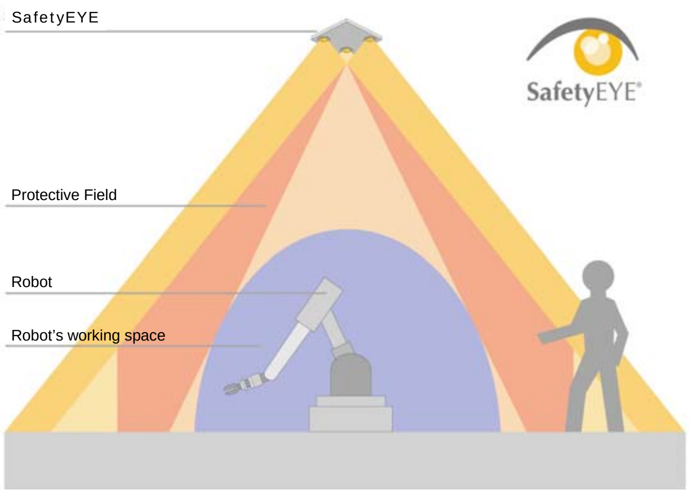


Figure 1: Pilz SafetyEYE - an invisible fence [47]

2.2 State of science – Related Work

The first approaches on safe interactions of humans and industrial robots were made in the 80s. After the first death of a worker related to a robot accident, reported in 1982 [1], safety studies in 1983 concluded that a robot could only be safe if the robot could detect the approach of a human and react accordingly [2].

In 1985 [3] Graham and Meagher of the Rensselaer Polytechnic Institute in New York investigated microwave, ultrasound, infra-red and capacitive sensors on their applicability in collaborative workplaces. They continued their work, together with Derby, in 1986 [4] resulting in the conclusion that "*much research still needs to be done on robotics safety*". The Rensselaer Polytechnic Institute is still working in the field of safe robot-human collaboration. In one of their latest projects from 2007 [5] Meisner, Isler and Trinkle used a galvanic skin response sensor, measuring the workers stress level, to generate robot trajectories.

In 1989 [6] Takakura et al. outfitted a robot with torque sensors and programmed an algorithm that worked on contact sensing and memorization of objects. Although this approach was not feasible in praxis, due to the reduced speed of the robot, in order to avoid any damage by collisions, and problems with dynamic objects, whose changed positions can not be tracked, the idea of using torque sensors persists until today (see chapter 2.2.2).

In 1992 and 1994 Novak and Feddema used capacity sensors for object avoidance [7]. The advantage of this approach is that the robot does not need to touch the obstacle, but can sense it in its vicinity. While increasing the sensor range from touch to close vicinity was an improvement, the problem in this approach is the different reactions of capacity sensors to different materials.

Ebert and Henrich presented in 2002 a solution that used a multiple camera system to detect collisions between the robot and obstacles [10]. The problems they reported for this approach existed in cells being reported as occupied while free and vice versa and in the robot blocking the view of the cameras in certain positions. In later projects by Henrich this problem has been reduced but not completely solved (see chapter 2.2.3).

Heiligensetzer and Wörn demonstrated in 2002 [20] [21] a Kuka Robot, equipped with capacity proximity sensors and haptic devices. Heiligensetzer continued the approach, using capacitive proximity and tactile sensors, in 2004 [19] and 2007 [57]. A problem in approaches using force sensors is always the reduced speed of the robot, which is necessary to avoid harm to humans. The robot presented in 2007 was therefore also equipped with collision absorbing material, making it possible to detect a collision before harm is done to the worker. Wörn continued his work in this field in 2004 [22], together with Yigit and Burgart, by developing a reflex based control for industrial robots, in which reflexes take over the control if a dangerous situation is detected.

In recent years the primary focus in the field of safe industrial robot collaboration was on the force sensors, which detect only collisions, and on camera based solutions. In 2005 [17] Lu and Chung presented a wrist force/torque sensors using robot, equipped with a weighted path planning algorithm. The DLR lightweight robot [16] presented in 2007 allows torque sensing in its joints, which makes fast collision based planning possible.

In 2007 F. Som, Manager Control Development Software – REIS Robotics, presented an approach with a 3D camera mounted to the ceiling that observes the roboter's working environment [56]. Little information was given on the used hardware and algorithms, as well as on the state of the project, but individual communication with Mr. Som showed that the hardware set-up of the approach from REIS is similar to the set-up of the approach developed in this master thesis. Differences exists in the handling of the acquired image data. For example REIS is trying to eliminate the robot for further calculations by knowledge of its position, while this master thesis identifies the robot but keeps it in the acquired data because of safety reasons.

Among other problems, camera based solutions suffer from occlusions of objects by other objects in the workplace. This is a problem independent from the chosen approach, whether 2D data or 3D data is acquired from the sensor system. Thus in recent years the number of studies on combining several data sources has increased. In 2008 Elshafie and Bone [14], Mure and Hugli [24] and Lenz et al. [25] presented solutions to this problem. Also the number of systems that try to anticipate human behaviour, as shown by Lenz et al. [25], is increasing.

Another approach to the problem that has come up is equipping the human with sensors, rather than the robot. This was shown by Meisner et al. [5] in 2007, as described above, and also by Tamei et al. [26] who used a surface electromyogram in 2008 to simulate force sensors on the robot from the data collected at the human body.

2.2.1 Morpha

From 1999 to 2003 the research project MORPHA was conducted [50]. The goal of this project was the interaction, communication and collaboration between humans and intelligent robot assistants. In the course of the project, several approaches on the given topic were presented.

As one of the first results on industrial robot arms from this project Wösch and Neubauer showed in 2001 an eight degree of freedom robot with a stereo camera system, a laser scanner and a tactile sensor skin that could evade humans [8]. They used real forces, measured by the tactile skin, as well as virtual forces, induced from the model of the environment, to move the robot arm away from objects.

In the MORPA project "rob@work" in 2002, Helms, Schraft and Hägele used a camera mounted at the ceiling and a 3D laser scanner behind the working place, to achieve a collaborative workplace of a human and an industrial robot arm [9]. The robot arm was mounted on a mobile platform and could thus be transported to other workplaces, while the camera and 3D laser scanner were stationary.

While the camera was used to detect humans in the workspace and reduce the robot's operation speed in the near vicinity of the human, the 3D laser scanner results were used for path planning. Due to the usage of only one 3D scanner, the robot arm induced blind spots in the environment.

Also a contribution to MORPHA was made in 2004 by Stopp et al. [11]. They used multiple laser scanners and a 360 degree overhead camera on the DaimlerChrysler Manufacturing Assistant, a normal industrial robot by Reis GmbH & Co. KG Maschinenfabrik, mounted on a mobile platform. The software created static and dynamic protective fields, based on the current task and position, which caused the robot to slow down if approached, up to the point of standing still without an emergency stop. The system also included a force sensor in the endeffector that allowed the program to detect, if a human was trying to acquire the object currently held by the robot, in which case the object was released.

2.2.2 Phriends

From October 2006 till September 2009 the Phriends (Physical Human Robot Interaction – depENDability and Safety) project [45] is conducting research on physical human-robot interaction. The DLR (German Aerospace Center) and KUKA Roboter GmbH, both participants in the MORPHA project, also take part in this project. Part of this work is the investigation into the field of safe interaction of humans and robots without any separation in time or place. To acquire this grade of collaboration the robot has to be outfitted with force sensors.

A first result of this project is the KUKA lightweight robot [53]. This seven degrees of freedom industrial joint arm robot is equipped with integrated sensitivity sensors, that allow the robot to follow human guidance and detect impacts. Impact tests at 2m/s were conducted, which show that the robot combined with its control does only minimal hurt to the human body.

At present the Phriends team is working on solutions using two 2D cameras, one to observe the area and one mounted to the robot for close observation.

2.2.3 Simero

Safety strategies for human-robot-cooperation (SIMERO [51]) is a project of the University of Bayreuth. This project is supervised by Prof. D. Henrich and based on his approach [10] from 2002.

The sensor used in this project is a multi 2D camera system, where all cameras are placed strategically in the working area. The different angles of the camera were chosen to minimize the occlusion by dynamic objects and the robot itself. This allows the robot to evade the human horizontally and vertically.

While some problems have been solved during the course of the project, in 2008 changes in the lighting conditions in the environment still cause difficulties. Along with pre-processing cameras and a new approach on path planning, these difficulties will be addressed in the future work of the project [13].

2.3 Overview of the project at the BGIA

Prior to this master thesis two project reports have been created [77] [78], which have laid the foundation of this thesis and their results are included in this report.

This chapter summarizes the complete framework project, consisting of both reports and this thesis.

The overall goal of the complete project is to develop a collaborative workspace, consisting of an industrial jointed arm robot, a sensor system and a control system that allows a human operator to work safely in close proximity of the robot.

In the R&D 1 report: “Development of a prototype of an autonomous motion control for assisting industrial robots” [77] several approaches to this topic were evaluated (see chapter 2.2) and a solution to the topic was developed.

At the beginning of the R&D 2 project the developed theoretical approach to the above mentioned solution was discussed on the point of feasibility. An optimal tradeoff between functionality and feasibility was found, which is briefly described in chapter 2.3.1, along with the different algorithms in chapter 2.3.2 that are needed for the completion of the project in this master thesis.

The hard- and software, discussed in the R&D 1 report, on which the project is implemented, was selected in the R&D 2. A description of this hard- and software, as well as its assembly in this master thesis, can be found in chapter 3.

The main attention of this report is on the designed program that shows how the robot can be controlled to evade dynamic objects in its working place. Additionally, in chapter 6.4, the performance of the system is evaluated and compared to an optimal system.

2.3.1 Path planning approach

As described in the previous R&D 1 report [77], there are several modes of safe human-robot collaboration that can be achieved. The easiest is to separate the human and the robot completely, thus allowing no cooperation at all. The most difficult is to allow the human and the robot touching each other, while still executing tasks. To collaborate with a human, robots have to sense their complete environment and act accordingly, because safety, mainly of the human operator, is always the biggest issue.

The approach described in the R&D 1 report has been designed to allow a high degree of collaboration between human and robot, enabling the human to touch and interact with the robot at all times. This approach not only puts high demands on the required hardware, using for example at least two high tech 3D cameras, but also needs a lot of computational effort, which in turn puts high demands on the processing capabilities of the logical unit.

Based on the knowledge, acquired in the first report, a practical approach was found. This approach is a tradeoff between the functionality of the previously developed algorithm and the simplicity of a fence. It was develop in the R&D 2 report [78] and can best be described as a flexible fence. Human and robot are still separated, but the human is allowed to enter the robot's working area. The robot is working around the human or, if no alternative path is available, waiting for the human to leave the occupied path.

The flexible fence is realized with only one 3D camera, mounted above the robot and looking downward at the working area, as depicted in Figure 2.

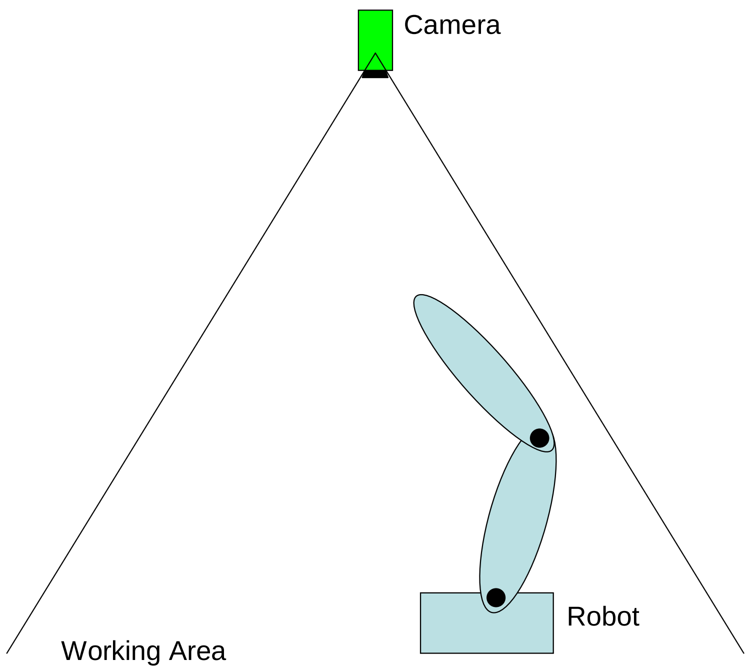


Figure 2: Assembly of the workspace

Since the camera delivers distance information for each observed pixel, the knowledge about an intruding dynamic object can be build by comparing the distances measured in the empty workspace (background image – see chapter 5.2.1) with the distances currently observed in the workspace. An intruding object is recognized through its lower distance values.

As long as there are no intruding objects, which might be dynamic or stationary, in the working area, the robot can execute its normal tasks. If objects intrude into this guarded working area, the space they occupy has to be excluded from the robot's possible movement. This reduced space is depicted by the red dashed line in Figure 3.

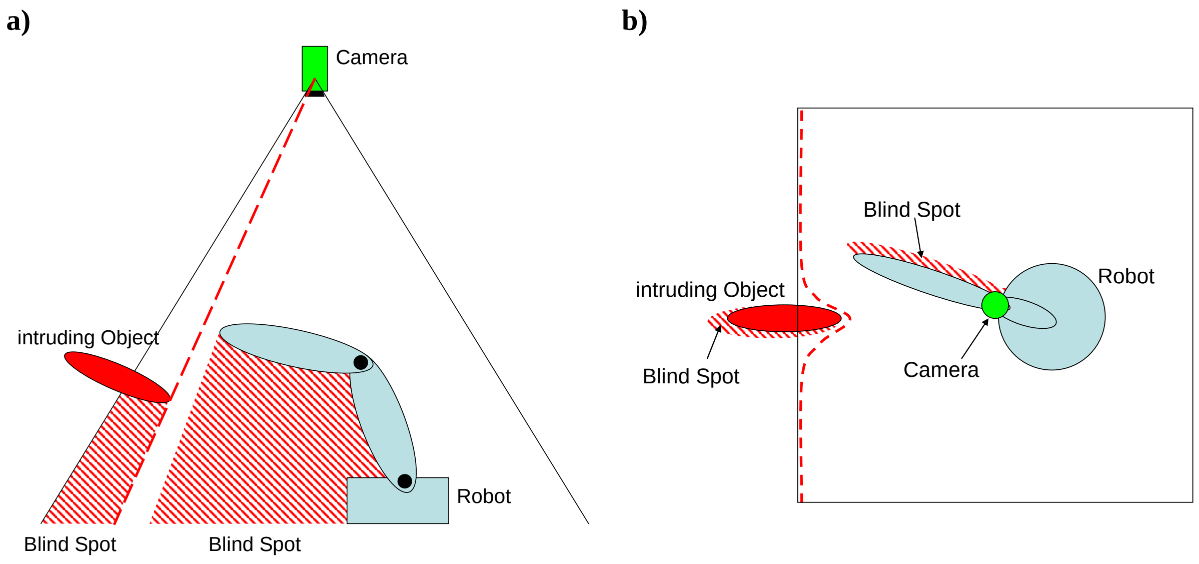


Figure 3: Camera's view on the working space a) from the side and b) from top

The blind spots below the intruding objects, as well as below the robot itself are problematic. The camera can not observe the area within those blind spots. From the knowledge of the robot's physical dimensions and the past distance values observed by the camera, showing the static background, it can be assumed, as long as the blind spot of the robot has not been connected to any other blind spot, that the space below the robot is still free of new objects. Therefore a path planning algorithm can plan the robot's path accordingly. If those blind spots would connect at any time, the knowledge about the static background can no longer be used, since a new dynamic object could have entered the robot's blind spot via the blind spot of the detected dynamic object.

For path planning this leads to the restriction that the robot may never work in the blind spot of a detected object, since this space can not be monitored. It also may not cast its own blind spot on intruding objects or touch the object, since the blind spots of the robot and the intruding object would otherwise merge. At this point the blind spot below the robot can no longer be considered free of objects, since other objects can pass from one blind spot to the other, without being detected. Theoretically, if this would happen, the robot could only be allowed to move upwards, towards the camera. The problem in this case would be that the monitoring of the workspace below the robot can no longer be deduced from history. The knowledge of the workspace below the robot is lost. To insure safety, the robot's motions are stopped in such a case. The robot's workspace needs to be checked manually by the human operator. After a successful inspection the operator can allow the robot to continue.

As an additional approach, it is possible not to control the robot's trajectory, but only its speed in accordance to the robot's distance to an intrusion. This approach can be based on the same data acquired by the 3D camera. It has a lower rate of productivity, since the robot is not avoiding obstacles but stopping and waiting in a safe distance from the intrusion. Its advantage is its simpler realisation, which can be easier verified for freedom from errors.

Both approaches can be combined. In this case, only the distance measurement needs to be approved for safety related work. The robot can not collide with any object due to path planning, if its approach on those objects is monitored. The functionality of the path planning algorithm is then only relevant for the performance of the process.

2.3.2 Necessary algorithms and features

To complete the given task several sub-goals had to be reached. The developed and applied solutions are described in chapter 4 and 5.

There is no interface available from the manufacturer that allows the control of the chosen industrial robot with a C++ program. To remotely control the robot, only a set of XML (extended markup language) commands are available (see chapter 3.5 and chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**) that have to be sent to the robot via TCP/IP. From those XML commands an own C++ interpreter had to be created. The XML commands are limited in their amount and can not be extended by the user. The robot needed to be controlled with relative and absolute positioning commands, but the relative positioning commands are not supported by the given XML commands. The solution to the relative control was achieved in the R&D 2 project [78]. In this master thesis the C++ library of the commands implemented in the R&D 2 was created (see source code on CD).

The chosen 3D camera delivers a set of image data, containing the distance of objects and their reflectivity in the used near infrared spectrum. The image data delivered by the 3D camera can not be interpreted by a human observer. The values of reflectivity are usually very close to each other, resulting in a mostly black image when depicted in greyscale, and thus have to be enhanced, in order to depict an understandable image. The distance data need to be interpreted by the program as well, to show an understandable image.

The optical analysis by a human observer is not only necessary for the development process of the final program. The illustration of the correctly working algorithm next to the robot can, especially at the introduction of this new technology, also enhance the operators trust in the developed concept. Algorithms for the reflection image as well as the distance image have been developed in previous work [78] and combined in this thesis with highlighting algorithms that show the robot, dynamic objects and the flexible border or the distance between robot and object, depending on the chosen algorithm.

From the distance data, delivered by the 3D camera, the background of the empty workspace has to be acquired as a reference image, in order to be able to find intrusions later. Those intrusions have to be distinguished into robot and other objects. Problems in this area are mainly induced by the fluctuation noise, present in the acquired distance data. In order to achieve the desired goal, filters had to be tested and implemented, which reduce the fluctuation noise as much as possible.

From the data about the robot’s actual position and the actual position of all other intrusions, a speed control and a path planning has to be developed.

3 Description of the hard- and software

To achieve the desired goal, a robot avoiding dynamic objects, in addition to the robot further hard- and software is needed, as Figure 4 shows.

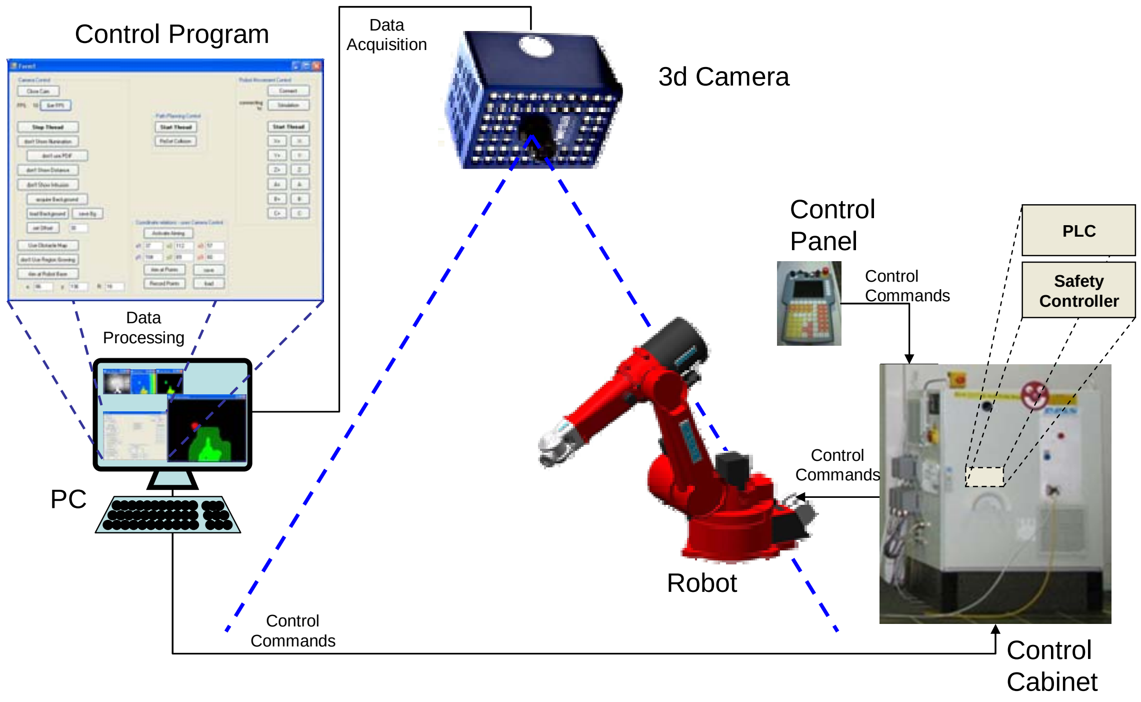


Figure 4: Hard- and software, necessary for the approach from chapter 2.3.2

The complete assembly consists of:

- robot
- control cabinet, housing the robot’s programmable logic controller (PLC) and a safety controller
- control panel
- 3D Camera
- PC
- controlling program

The controlling program, running on the PC, is collecting data from the 3D camera, build by Mesa Imaging AG, processing this data and sending controlling commands to a PLC according to the computed results. The PLC controls the robot, as long as the control panel sends an approval signal. All the time, the reduced speed of the robot is monitored by the Esalan-SafetyController, by ELAN, which was developed in cooperation with REIS and the BGIA [54].

A detailed diagram, showing the connection between the PC and the robot, respectively the PC and the camera, can be found in chapter 3.6.

To select appropriate hard- and software for the project, several considerations have been made:

- Robots, used in today's industry, exist in many shapes and sizes. Since most of the industrial tasks that require human-robot collaboration also require a high degree of freedom (DoF) of the robot's endeffector, a jointed arm robot was chosen for this project. Since a six DoF joint arm robot is the most common joint arm robot in the manufacturing industries, it is the most reasonable choice. The robot chosen for the project was a RV30-16 by REIS.
- If the wrong commands are given, this robot might be dangerous to its environment and itself. Therefore it is reasonable to simulate its behaviour prior to the implementation in praxis. For simulating the robot's behaviour, when controlled by the PC, the simulation-software RobOffice 20.3, developed by REIS, was used throughout the whole project. This software behaves in most points similar to the robot and gives a good anticipation of the final results. Since some deviations do exist, routines worked out with the simulation have to be carefully rechecked on the real robot.
- Since finding a path for a joint arm robot is a task in a three dimensional room, the sensor results should also be three dimensional.

  The sensor's range must be appropriate to cover the complete workplace.

  The safety distance between robot and human is directly connected with the measuring time of the sensor. Thus the measuring time should be as short as possible to reduce the required safety distance to a minimum.

  The best achievable results are yielded by 3D sensors like the 3D laser scanners used in previous approaches in this field of science or 3D cameras, whose results are more inaccurate, but which are considerably faster. Since the speed of the 3D laser scanners is not fast enough at this point, this project is based on a 3D camera. The chosen sensor is the 3D Camera by Mesa Imaging AG, the Swissranger SR-3100.
- It is common to communicate with a robot using an XML language (see literature in chapter 8.4). There are several common languages proposed by different applicants from the robotics industries, for example XIRP supported by the VDMA (see [38] and [39]) and RoboML (see [41] and [43]), but until today no common structure for this language is standardized. The XML-language used in this project, RSV COMMAND XML, is specified by the robot's manufacturer REIS.
- The camera's manufacturer only supports a C++ library for the connection of PC and camera. Since the functions contained in this library change with the version of the library, an interpreter was programmed during this project that allows exchanging the library with as little effort as possible.

- The program, developed in this project has to be able to acquire data from the camera and the robot and process this data in real time.

  The XML commands controlling the robot are send via TCP/IP and sockets, which gives a lot of freedom of choice for the programming language and the system.

  The camera's C++ library and the fact that C++ can work with sockets and is also known to be considerably fast in execution, makes C++ the logical choice as the programming language.

- To achieve the best possible result in case of real time execution the program was executed on a multicore processor. To make the best usage of the processor, the program was separated into several tasks. With C++ this can be achieved easily using the .NET framework, which is supported by MS Visual Studio 2008 Express.

- The operator at the robot has to be able to see the results of the algorithms, enabling him to visually control their operation. OpenCV, a free library containing algorithms for graphical analysis and image creation, was chosen for the depiction of the images, created by the program.

3.1 Jointed Arm Robot Reis RV30-16

A jointed arm robot resembles the arm of a human and has similar degrees of freedom (DoF). While the human arm has 7 DoF, most robot arms, like the RV30-16, only have 6 DoF, since those are enough for the robot's endeffector to reach any pose (position and orientation of the endeffector) within its working space. That means that all X, Y and Z coordinates and all Alpha-X, Beta-Y and Theta-Z angles can be reached.

Figure 5 shows the RV30-16 by Reis and Figure 6 shows the kinematics for this jointed arm robot.

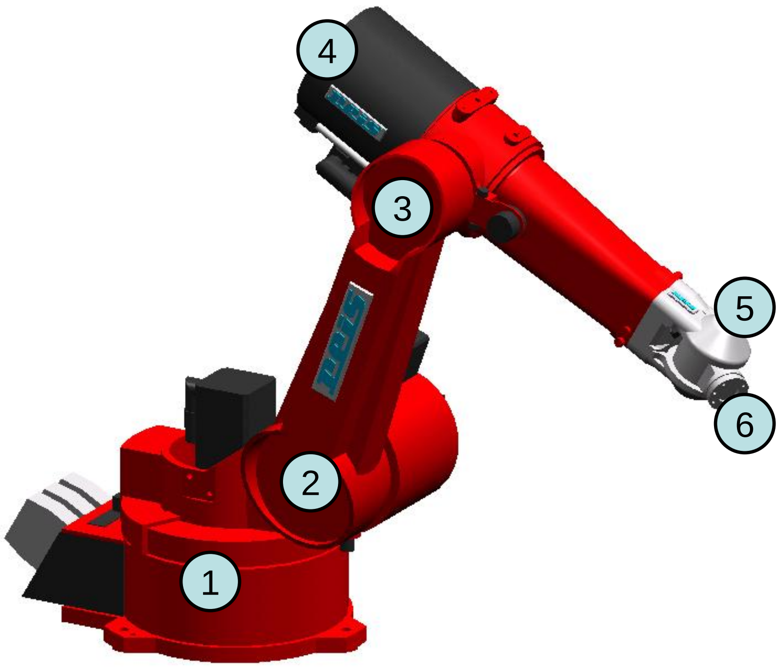


Figure 5: Reis RV20-16 [52] with numbered axes

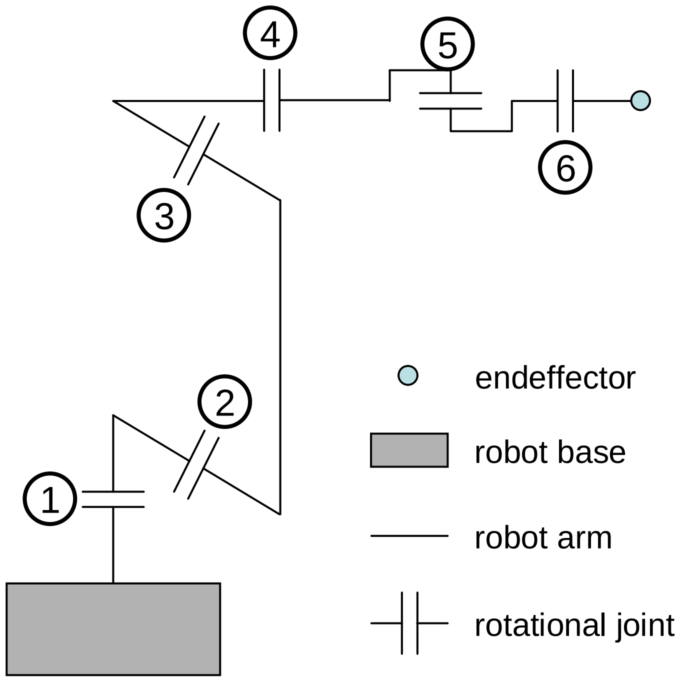


Figure 6: RV30-16 kinematics – upper right arm configuration

The six rotational joints in Figure 6, also called revolute joints, are grouped based on two conditions. One condition is whether their rotation changes the position of the next arms endpoint or not. The other condition is whether it changes the next arms rotational angle or not.

- Joints 2, 3 and 5 only change the position of the next arm's joint.
- Joint 4 only changes the angle of the next arm's joint.
- Joint 1 changes both, the position and the angle.
- Joint 6 depends on the endeffector (the tool), that is mounted on the robot's arm. If the tool centre point is centred on the axis of joint 6 only the angle is changed (same as joint 4). If the tool centre point is outside of the axis of joint 6, the position is changed as well (same as joint 1).

The robot's kinematics results in two methods for position calculation, the forward kinematics and the inverse kinematics.

The forward kinematics handles the question of the endeffector's position, given specific angles of the joints. This problem always yields one specific solution for each set of angles.

The inverse kinematic is a more complex problem. A six degree of freedom robot arm has up to eight different possibilities to reach most endpoints, in special points, singularities, a multitude of possibilities exist (see below). The six configurations are the right arm configuration or the left arm configuration, each combined with the upper or the lower arm configuration and the flip and no-flip position.

The robot depicted in Figure 5 and Figure 6 is positioned in an upper right arm position. Right arm position means his second joint is on the right side of the first one. The upper part is from the fact that joint three is above the direct line from joint two to five.

If the robot’s first axis is turned by 180° and flipped over by turning axes two and three, the same position could be reached with the left arm configuration. From both configurations the lower arm configuration can be achieved, by turning axes two and three. Figure 7 shows the kinematics for a robot in lower left arm configuration.

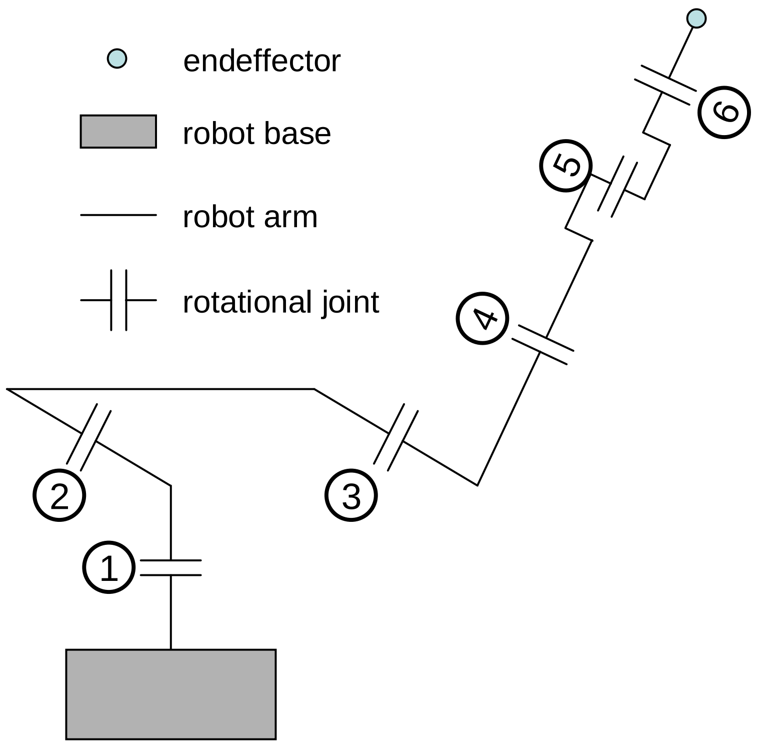


Figure 7: RV30-16 kinematics – lower left arm configuration

Flipping is achieved by the axes four, five and six. If axis four is turned by 180°, half a circle, axis five is flipped to the other side and axis six is turned by 180° as well, the same position as before is reached.

The path of the robot, defined by a start- and an endpoint, can be executed in two different ways. Since both positions are defined by the angles of the robot’s joints, the easiest way to reach the endpoint is to simply change every angle simultaneously, calculating the changing speed from the longest changing time in all angles. This calculation can be achieved by applying a factor to the maximum possible speed of the angle, as shown in Equation 1.

$$f(n) = \frac{t_{end}(n) - t_{start}(n)}{\max_{i=1}^{6}\left(t_{end}(i) - t_{start}(i)\right)}$$

Equation 1: Calculating the factor f for the speed of the individual angle n according to the description in [76]

$f(n)$ factor to be applied to maximum speed of angle n

$t_{start}(n)$ start time of movement for angle n

$t_{end}(n)$ end time of fastest movement for angle n

The other, more complex, method to move the robot's endpoint from one position to another is to calculate a straight path between those points. To achieve this, the desired line is separated into small changes and for every such change the angles of the axes are calculated, using inverse transformation. Afterwards the results are connected by interpolation. Thus, the more points are used, the higher the accuracy of the path.

Moving on a straight path can cause problems, if singularities exist in the calculated path. Singularities are special points in the robot's workspace where two or more joints are collinear (joint four and six in the shown configurations in Figure 6 and Figure 7). In those points a multitude of possible configurations exist (see [70] chapter 3.23). The singularities can cause problems when they exist in a calculated path, because the robot might have to change the arms configuration (e.g. from flip to no flip), which can not be done instantly. Thus the robot has to stop at such a point, which in turn can be bad for the process (e.g. in special welding cases).

While being one of the smaller Reis robots, the RV30-16 can still handle up to 16 kg, which should be enough for demonstrative purposes. Also the speed of the robot, which can be up to 4 m/sec is sufficient for this task.

The robot is also equipped with electronic safety trips [48], designed by REIS. These trips monitor the position of all of the robot's joints and shut down the robot, if any joint malfunctions. Malfunctioning can be a diversion from the parameters transmitted by the robot's PLC, a dangerous increase in the joints speed, e.g. while driving through a singularity, or entering of a zone that has been prohibited in the setup phase of the trips. Finally, the trips guaranty that the robot's endeffector is at the desired position.

3.2 3D Camera Mesa Swissranger

The chosen camera, the Mesa Swissranger SR-3100 (see Figure 8), is a Photonic Mixer Device (PMD).

Figure 8: Mesa Swissranger SR-3100 [55]

PMDs are based on the principle of measuring the distance by the phase difference of an additional amplitude modulation (see Figure 11) between the emitted and the received light.

All PMD devices are equipped with an own light source. Figure 9 shows the camera's own LEDs emitting light at a certain wavelength, near the infrared range. This light is reflected by objects, loosing intensity in the process dependent on the reflecting surface. A filter in front of the camera's sensor subtracts any but the emitted wavelength (see [55]).

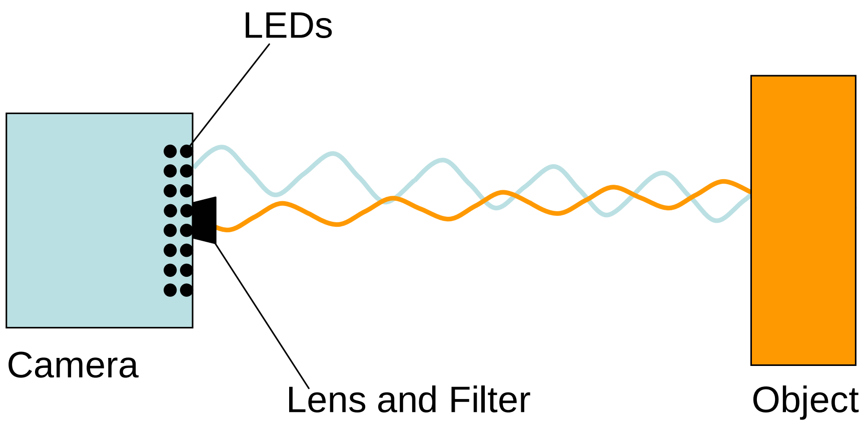


Figure 9: Camera emitting near infrared ray (blue) and receiving reflected ray (orange)

The camera's internal electronic can measure the phase shift between emitted and received light, depicted in Figure 10, resulting in a "Time of Flight" (ToF) measurement of the light wave.

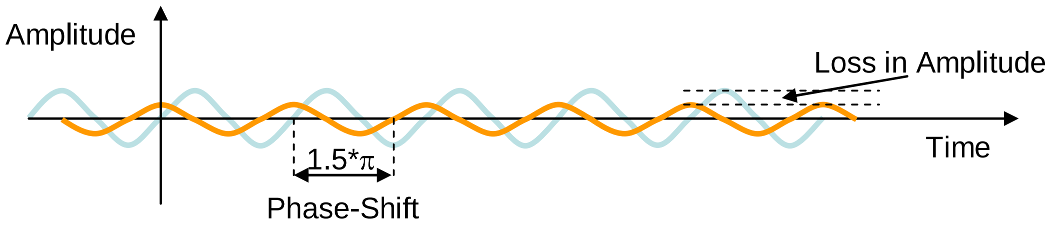


Figure 10: Example of a Phase Shift between two waves

By this ToF and the constant speed of light the distance from the camera to the object can be deduced. Since this measurement is done in every pixel of the camera's sensor chip, a two dimensional image of distance information is gathered, equal to the human depth perception.

As long as it is constant and above the measuring threshold, the percental loss in amplitude between the emitted and received light has no effect on the distance measurement. This loss can be used to generate additional information on the received objects, which are normally displayed as a black to white reflection image. Since the filter in front of the camera's sensor filters out any light but the emitted near infrared light, this image does not show the "normal" greyscale values, but the reflective properties of the viewed material to this specific wavelength. Thus the actual colour of the objects can not be recovered from this image, since there is no strong correlation between the object's colour and its reflectivity.

The disadvantage of using phase shift to measure the ToF is the limited range. Since the phase of the light wave repeats itself after approximately 800nm the range would be limited to 400nm, half a wavelength. This is why the amplitude of the emitted light is modulated as well (see Figure 11), generating a second wave, to increase the range to about 7.5 meters.

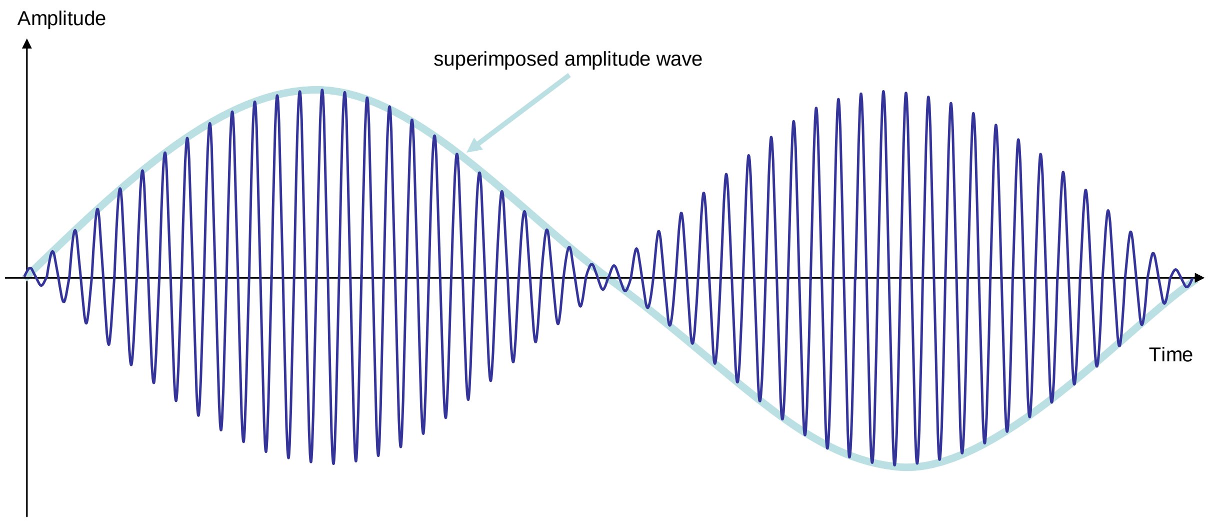


Figure 11: Generating a superimposed wave by amplitude modulation

To achieve a range of 7.5 meters, the modulation of the amplitude needs to be at 20MHz, as shown in the formula in Equation 2.

$$\frac{c}{f} * \frac{1}{2} = \text{camera range}$$

$$\frac{299\,792\,458\,\frac{\text{m}}{\text{s}}}{20MHz} * \frac{1}{2} \approx 7.5m$$

Equation 2: Calculation of the maximum range

c being the speed of light

f being the used modulation frequency

Longer ranges would theoretically be possible by choosing other modulation frequencies. Due to the physical restrictions of the used sensor chip (saturation time, etc.) 7.5 meters is the maximum range for which the camera is usable, according to the manufacturer [55].

The advantage in using near infrared light is that the every other part of the spectrum, including the visible light, can be filtered out before measuring.

The camera's main characteristics are summarized in Figure 12, additional information can be found in the first project report [77].

Model No.	SR-3100
Pixel array resolution	176 x 144 pixels
Field of view	47.5 x 39.6 degrees
Non-ambiguity range	7.5 m
Frame rate	Variable, max 50fps
Estimated Error	1% of measured distance

Figure 12: Table containing Camera Data, taken from [55]

As shown in the first project [77], recognition of the humans hand is assured for a distance of up to three meters.

Detailed studies on the accuracy of the camera have been done by another BGIA project [80]. Their result is that the camera's accuracy does not depend on temperature in a range between -10°C and +50°C. The camera delivers sufficiently reliable and repeatable results in a distance between 2 and 5 meters, provided that no material is present that does absorb most of the emitted light wave.

In the studies of this project only two materials (a grey jeans and a black shirt) have been found that do not reflect the emitted waves sufficiently. In the case of the given application, this problem can be handled as described in chapter 7.2.

3.3 Workspace

The Workspace used to implement and test the developed program is located in a laboratory of the BGIA, St. Augustin. The workspace is surrounded on three sides by walls which are more than 2.70m high (see Figure 13). According to the standard EN ISO 13857 [73], it is not required to have any distance between the robot and the walls, in order to secure human operators from reaching into the robot's working area.

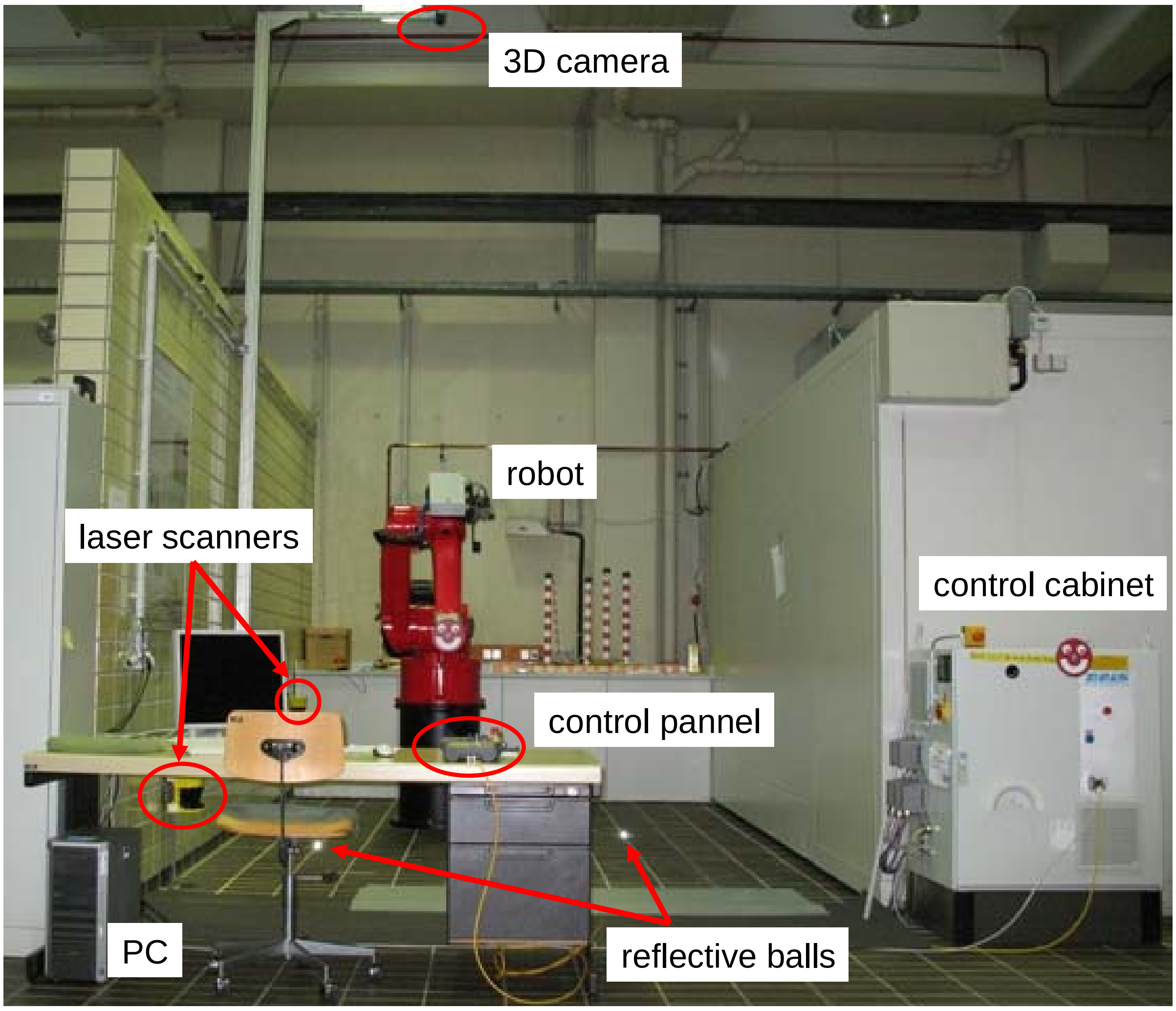


Figure 13: Workspace in the BGIA

Towards the frontal opening, the workspace is protected with two SICK laser scanners, which monitor the whole workspace. The laser scanners are positioned in such a way, that the robot is stopped, before an operator can reach the hazardous area. These scanners are active if the robot is controlled by the robot's PLC alone, i.e. during the automatic mode of operation. In this project, the PC can only take over the control of the robot, if the robot's control panel is set to "adjusting". In this mode the scanners are muted by the robot's so called safety controller (see chapter 3.1) and the safety is ensured by a hold to run button at the control panel during operation. Additionally in this mode of operation the speed of the tool centre point is reduced by its PLC to a speed of 250mm/sec which is safely monitored by the robot's safety controller.

To protect the robot from harming any other equipment, protective zones have been implemented in the robot's safety controller and in the configuration of the electronic safety trips.

The 3D camera is mounted to an arm that is flexible in its horizontal and vertical position. This allows fixating the camera in a height between 4m and 5m. Because of the given workplace setup in this project, where little demonstration can be done to the side and behind the robot, the camera was mounted in front of the robot, to increase the covered working area in the frontal location. This has to be noticed in the risk assessment of the project (chapter 6.3) since the camera does not cover parts of the working range behind the robot.

If the position of the camera was changed, a new setup is needed, in which the camera is adjusted to the robot (see chapter 5.5.2). For this adjustment three reference blocks are used, which are balls covered in highly reflective material. The exact positions for the reference blocks have been determined during the programming phase of the project. These positions have been marked on the ground, so that the reference blocks themselves can be removed from the working area after the adjustment is completed (see Figure 14).

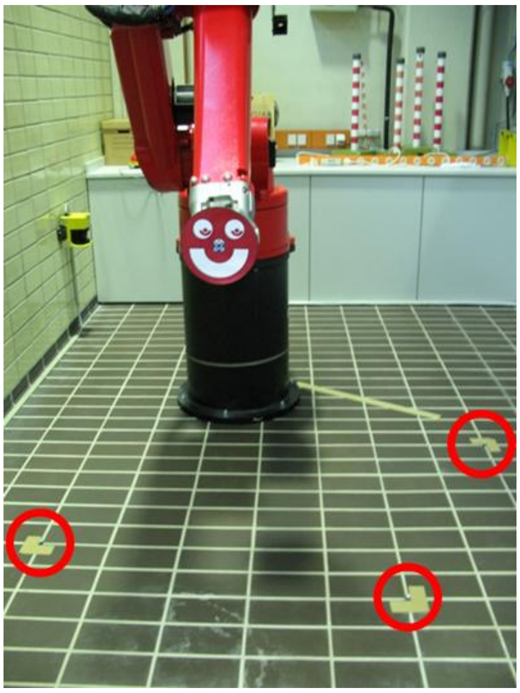


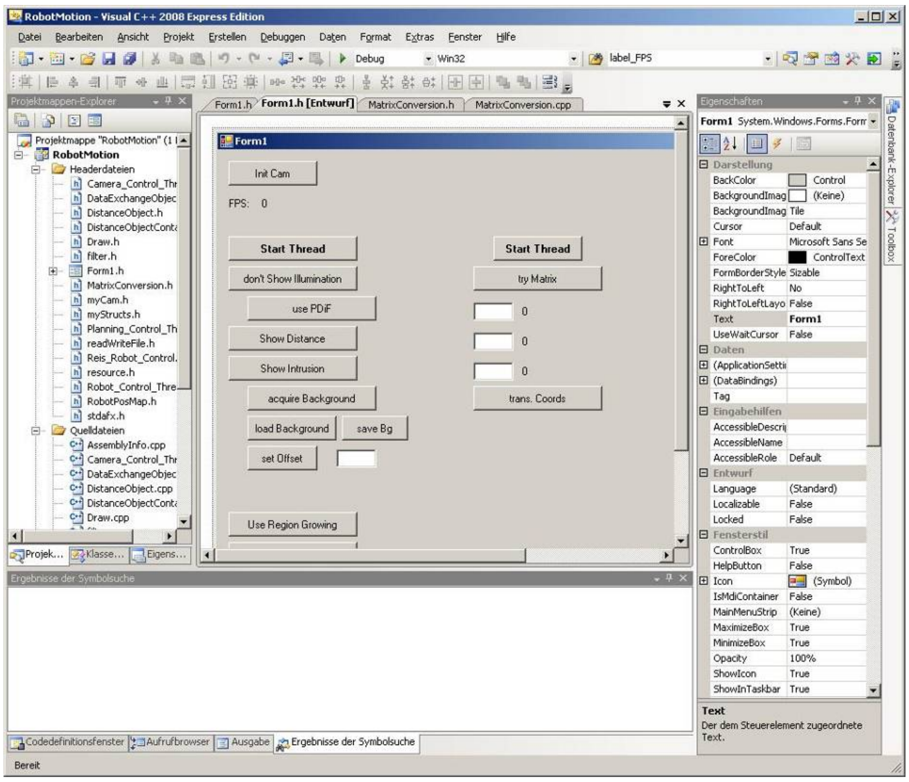
Figure 14: Robot working place with reflective balls (red circles)

Later versions of this project can use an adjustment program as described in chapter 7.2.

3.4 MS Visual Studio

Visual Studio is an integrated development environment designed by Microsoft. It includes several tools that help in programming, like IntelliSense – an autocompletion tool for symbol names – and a form designer tool for building graphical user interfaces (GUIs – see Figure 15).

While Visual Studio supports many languages, the actual program developed in this project only used the C++ environment. Thus it was possible, to implement the program in the 2008 Express Edition, which is freeware.



Form1 System.Windows.Forms.Form

Darstellung	
BackColor	Control
BackgroundImag	(Keine)
BackgroundImag	Tile
Cursor	Default
Font	Microsoft Sans Se
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayo	False
Text	**Form1**
UseWaitCursor	False
Daten	
(ApplicationSetti	
(DataBindings)	
Tag	
Eingabehilfen	
AccessibleDescri	
AccessibleName	
AccessibleRole	Default
Entwurf	
Language	(Standard)
Localizable	False
Locked	False
Fensterstil	
ControlBox	True
HelpButton	False
Icon	(Symbol)
IsMdiContainer	False
MainMenuStrip	(Keine)
MaximizeBox	True
MinimizeBox	True
Opacity	100%
ShowIcon	True
ShowInTaskbar	True

Text
Der dem Steuerelement zugeordnete Text.

Figure 15: MS Visual Studio form designer

3.5 Robot remote control language

The REIS robot’s control software is equipped with an extended markup language (XML) interface [75]. This interface can be used to remotely control REIS robots via a TCP/IP connection. The language used to communicate is RSV COMMAND XML, a language designed by REIS. A complete listing of all used commands can be found in Appendix I.

The usage of this mode of control is not the intended use of the robot in industrial environments. To prevent any PC connected to the robot from controlling it, this mode of operation is disallowed by default. Two conditions have to be met in order to be able to control the robot with a chosen PC.

Firstly the XML support of the robot has to be explicitly allowed in the control software and set to a specific IP address. Afterwards any device with that IP can establish a socket connection via TCP/IP to the robot’s webserver (see chapter 3.6) and use this socket connection to transfer XML commands and receive their appropriate responses.

Secondly the robot will only execute XML movement orders from a PC, if the robot is set to “External Mode”. This mode can be set by a program running on the robot’s PLC, or by the usage of the robot’s control panel.

The XML commands are sent as arrays of characters. They start with <RSVCMD> and end with </RSVCMD>. Each command has a client stamp, a number that can be freely chosen and that can be used to assign responses to their respective commands.

The different commands are separated into three different application programming interfaces (API), the transformation API, the symbol API and the API for the program memory.

- The transform API consists of commands used to convert coordinates into the different coordinate systems. It contains functions for the calculation of the forward and inverse kinematics, described in chapter 3.1.
- The symbol API is used to read and write values into variables. These variables can be used to control the robot's movement.
- The third API includes the commands necessary to read and write programs in the robot's PLC's memory.

In this project, only the second API, the symbol API, was used, since all necessary data of the robot are stored in global variables, which can be accessed and changed using the symbol API [76].

3.6 Control hierarchy

There are two control hierarchies to be considered in this project.

The first is the connection between the PC and the camera, which is provided via the universal serial bus (USB), both to send requests as well as to receive responses (see Figure 16).

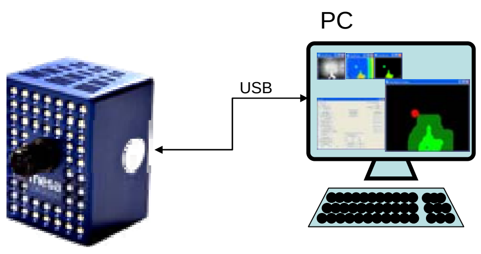


Figure 16: Control hierarchy between PC and Camera

The second, more complex, hierarchy involves the PC, the robot's control panel, the robot's control cabinet and the robot itself. The complete hierarchy, created with information taken from a presentation [56] and email correspondence, both by/with Franz Som, Manager Control Development Software – REIS Robotics, is depicted in Figure 17.

The communication interface for an external PC is handled by a webserver of a programmable logic controller (PLC). This interface is used by the developed program, described in chapter 5. The connection to the webserver is handled via TCP/IP and the commands are sent as XML.

A control panel is directly connected to the PLC with the command transfer handled by a CAN bus. For safety reasons, as described in chapter 3.3 and 3.5, the panel is needed to start the "external mode" of the robot. Afterwards it is continually needed to act as a hold to run button, while the robot is remotely controlled by the PC.

CAN busses are also used to connect the PLC to the servo controllers and the Safety Controller, which is charged with monitoring the correct execution of the position commands, as well as other safety related features.

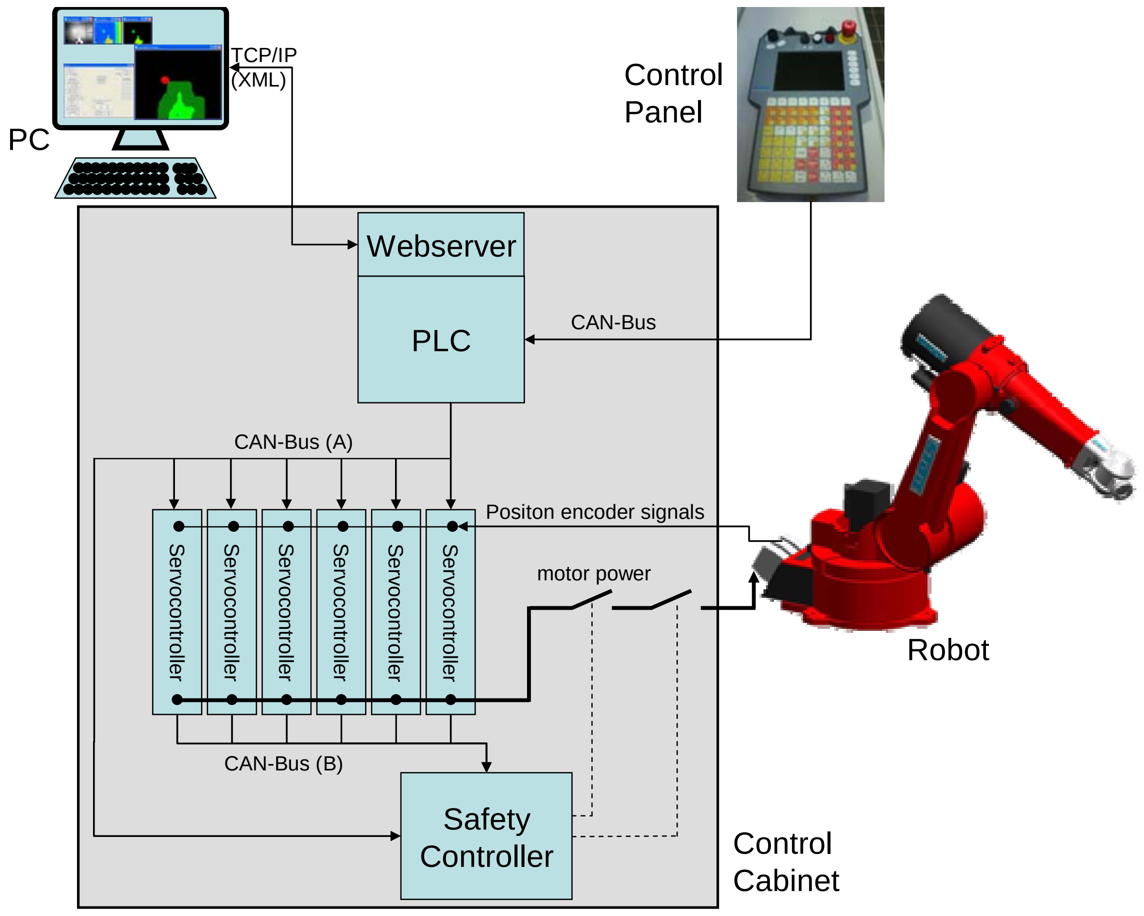


Figure 17: Control hierarchy from PC to robot – diagram created with the help of F. Som, REIS Robotics

If the safety controller detects an error in the execution of a command, it safely and redundantly cuts the power to the robot’s motors, causing the motor’s brakes to activate.

Direct communication between the safety controller and the PC is not possible. Thus any errors in the communication of PC and webserver can not be recognized by the safety controller. In a later industrial implementation, the safety controller would have to be included in the process of object avoidance, as described in chapter 7.2.

4 Overall evading concept

In this chapter an overview of the developed concept for evading dynamic objects is presented. The overall task is the acquisition of data from the sensors, the calculation of a collision free path for the robot and its execution.

This overview focuses on the interactions of the single components of the workplace and their interfaces. The different subprograms and algorithms themselves therefore are explained only briefly in this chapter. A detailed explanation of the internal mechanics of the single algorithms can be found in chapter 5. At this point, these detailed explanations are not necessary for the understanding of the overall concept, but rather disturbing to the fluency of reading. They are meant for the detailed study of the developed program.

4.1 Tasks hierarchy

The task of this project is a safety system. Safety systems need to be able to operate in real time, e.g. their lowest possible response time must be at least as fast as the task demands. Therefore the execution time of the developed program needs to be as fast as possible. It has to be considered that image processing and path planning are tasks that require a relatively large amount of computational effort.

Modern PCs are equipped with multicore CPUs that allow parallel processing of different tasks, if those tasks are separated in the executed program in different threads. To use this advantage, the developed PC-program has been divided into four separate tasks, organized in separate kernel threads.

In the used PC, a dual core processor was installed. By separation of the program into different threads the operating system has the possibility of allocating the threads to different CPUs, to better balance the computational load. This optimizes the execution time and contributes to the real time ability of the program.

The tasks are divided into the following threads:

- Graphical User Interface (GUI)	(chapter 4.2)
- camera control	(chapter 4.3)
- path planning control	(chapter 4.4)
- robot control	(chapter 4.5)

The flow of the data, from the camera’s image acquisition to the robot’s PLC, is depicted in Figure 18.

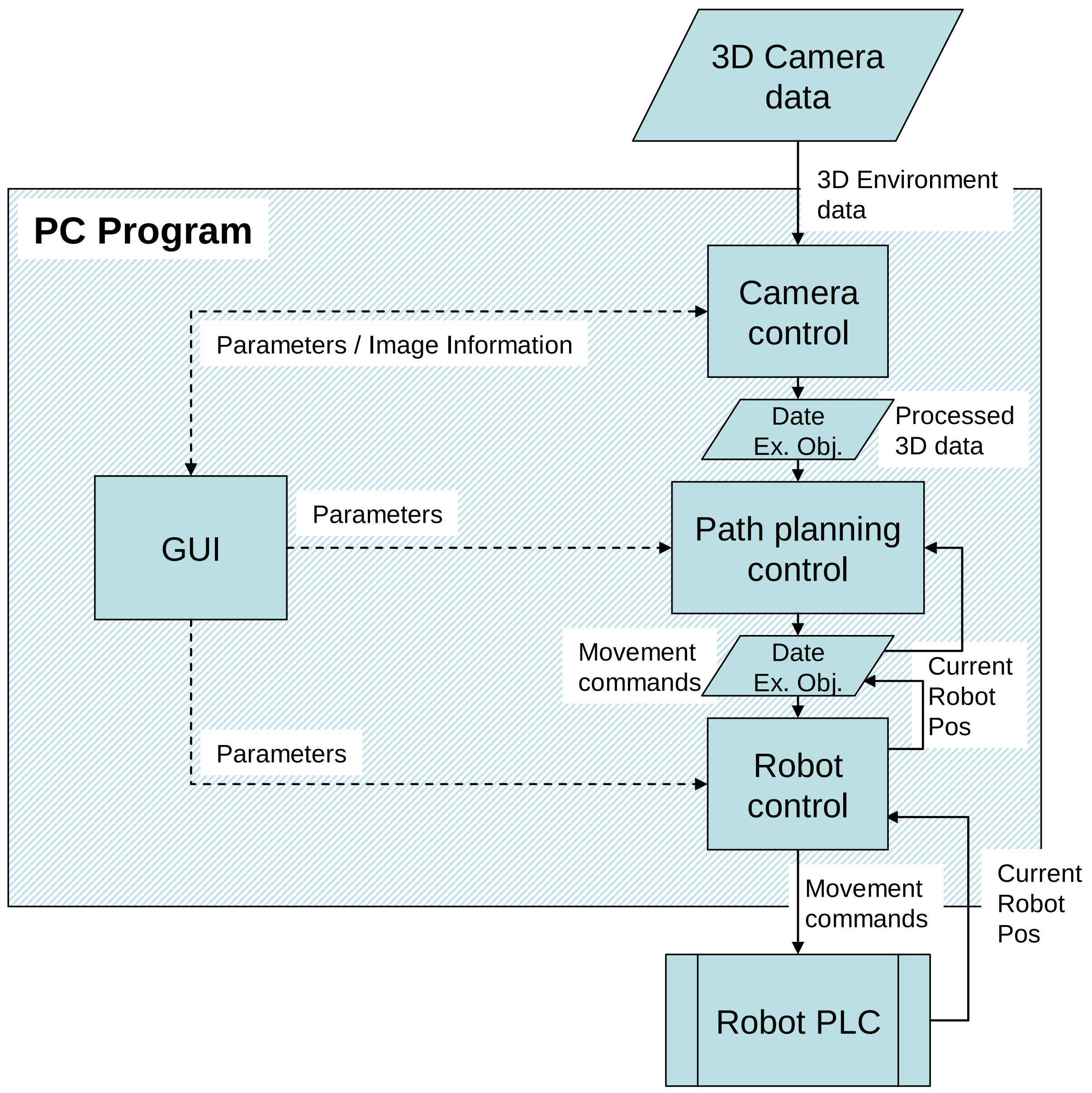


Figure 18: Data exchange diagram

While the GUI and the robot control thread contain only few computations, the camera control and path planning control thread share the main computational load.

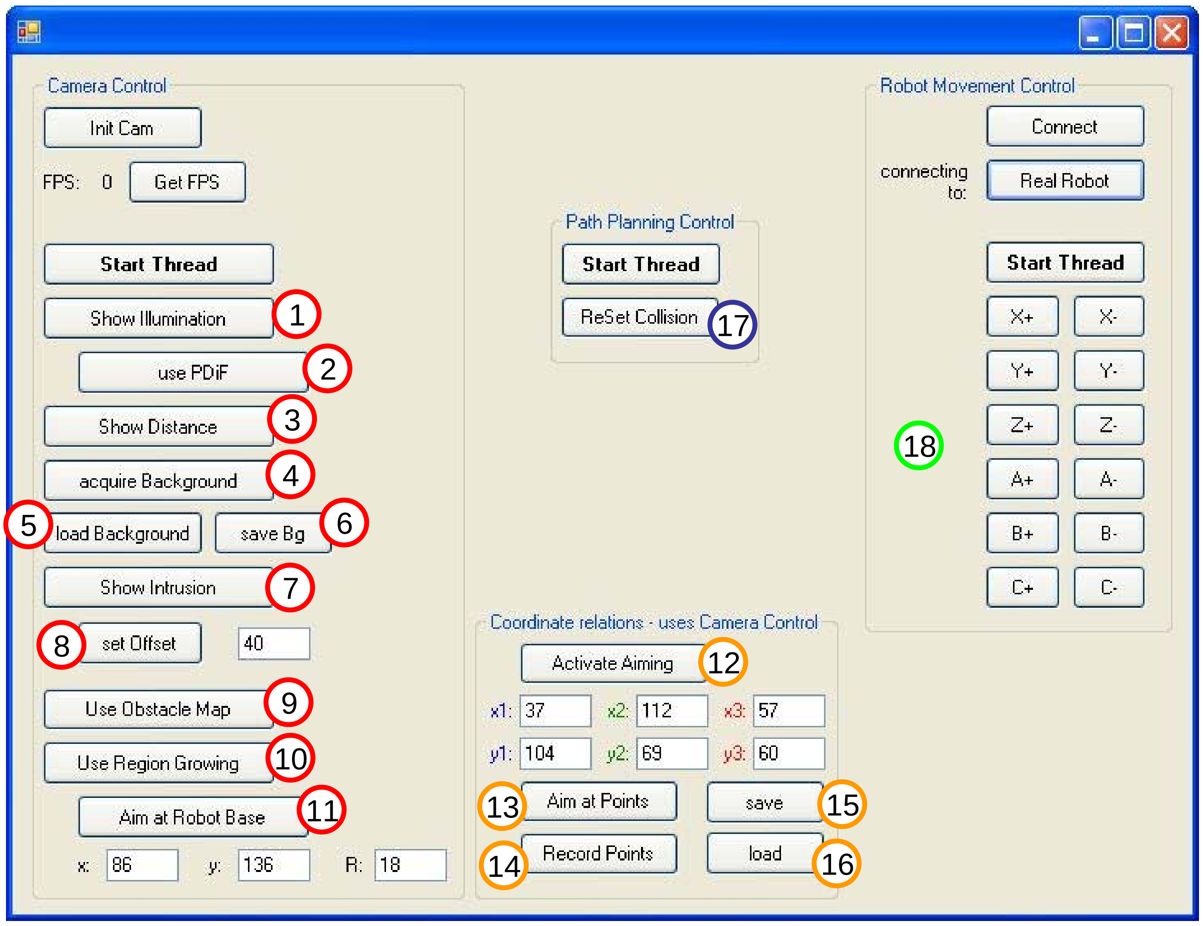
Each thread contains a setup phase, in which important variables are initialized, and a continuous processing loop, in which the computations are achieved. During this continuous loop certain variables are checked to switch sub-parts on and off (see following chapters). The values of those variables can be changed by the user using the GUI shown in Figure 19.

Exchanging information in variables accessed by different parallel tasks leads to consistency problems in those variables. To allow the safe data exchanged between the tasks, a data exchange object was created. This object handles the problem of dual access to single variables as well as the storage and retrieval of data from files (see chapter 4.6).

4.2 Graphical User Interface (GUI)

The GUI, handled by the main thread, allows the separate control of all threads involved (see Figure 19). At this stage of the development of this work a main switch, allowing every part of the program to be started simultaneously, was not programmed. The separate controls were necessary for a better control of the whole program during this testing phase. The GUI therefore is divided into four regions:

- Camera Control
- Coordinate relations
- Path Planning Control
- Robot Movement Control



FPS: 0

set Offset: 40

x: 86 y: 136 R: 18

x1: 37 x2: 112 x3: 57

y1: 104 y2: 69 y3: 60

connecting to: Real Robot

Figure 19: Main control window
numbering in accordance with chapter 4.3 ff.

It has to be noted (see chapter 4.1) that even if the control of the threads is separated in the GUI for testing purposes, they work together automatically, if all threads are started.

In the region “Camera Control” the connection to the camera can be established, respectively disconnected if it was already established. Also the camera control thread is handled here, allowing the single sub-programs of the camera control thread to be switched on and off.

In the region “Coordinate relations” additional controls for the camera thread are collected. They are only used during setup phase, if the camera’s position in the workspace has been changed (see chapter 3.3).

In the region “Path Planning Control” the path planning control thread is started / stopped. An additional function is to allow the user to reset the collision detection (see chapter 5.3), such that the robot can be allowed to continue working after an automatic stop induced by a collision.

In the region “Robot Movement Control” the connection between robot and PC-program is established, respectively disconnected. Also the robot control thread is controlled here, allowing among other things to manually change the robot’s position.

The single threads are described more precisely in the following chapters.

During the setup phases when the connection to the camera or to the robot is established, the GUI window blocks further user access to all its functions. This is due to the fact that the GUI thread handles those connections itself and does not delegate it to other threads. As described in previous work [78], the connection to the robot takes some time, depending on the responding hardware (robot or robot simulating PC).

Apart from those connection phases, the user can stop every thread separately at all times via the GUI. Threads are closed by the GUI using the transmission of a “thread stop” exception. This allows the user to stop the threads at every point during their execution, even if they are caught in a continuous loop.

4.3 Camera control thread

As described in chapter 3.6, the camera is connected to the PC via USB. The functions used to communicate with the camera are supplied by the manufacturer [55].

The camera controlling task handles the interpretation and display of the acquired images:

- Histogram equalization of the intensity image (algorithm see chapter 5.1.1)
- Display of the intensity image (see Figure 21a)
- Depth to colour conversion of the distance image (algorithm see chapter 5.1.2)
- Display of the distance image (see Figure 21b)

It also handles the processing of the acquired data and the display of the computed results:

- Acquisition of the workspace’s background (algorithm see chapter 5.2.1)
- Finding the intrusions in the distance image (algorithm see chapter 5.2.2)
- Displaying intrusions (see Figure 24)

- Region extraction (algorithm see chapter 5.2.4.1) and identification from the found intrusions (algorithm see chapter 5.2.4.3)
- Computation of the distance between robot and other regions (algorithm see chapter 5.4)
- Display of this distance (see Figure 27a)
- Computation of the space that can be reached by the robot (algorithm see chapter 5.5.3)
- Display of this space (see Figure 27b)

Since the computations and presentations of the results require a certain amount of processor time, each part of the processing loop can be switched on and off by the user using the GUI (see Figure 19). This allows for single testing of the routines as well as an optimized working process.

For better clarity, the flowchart of the continuous loop of the thread is split into Figure 20, Figure 22, Figure 23, Figure 25, Figure 28 and Figure 30. The numbers and their colours in the flowcharts are in accordance with the numbering of the GUI in Figure 19. Each numbered button of the GUI controls one subpart of the camera control thread loop.

4.3.1 Intensity and distance depiction

Figure 20 contains the first part of the continuous loop.

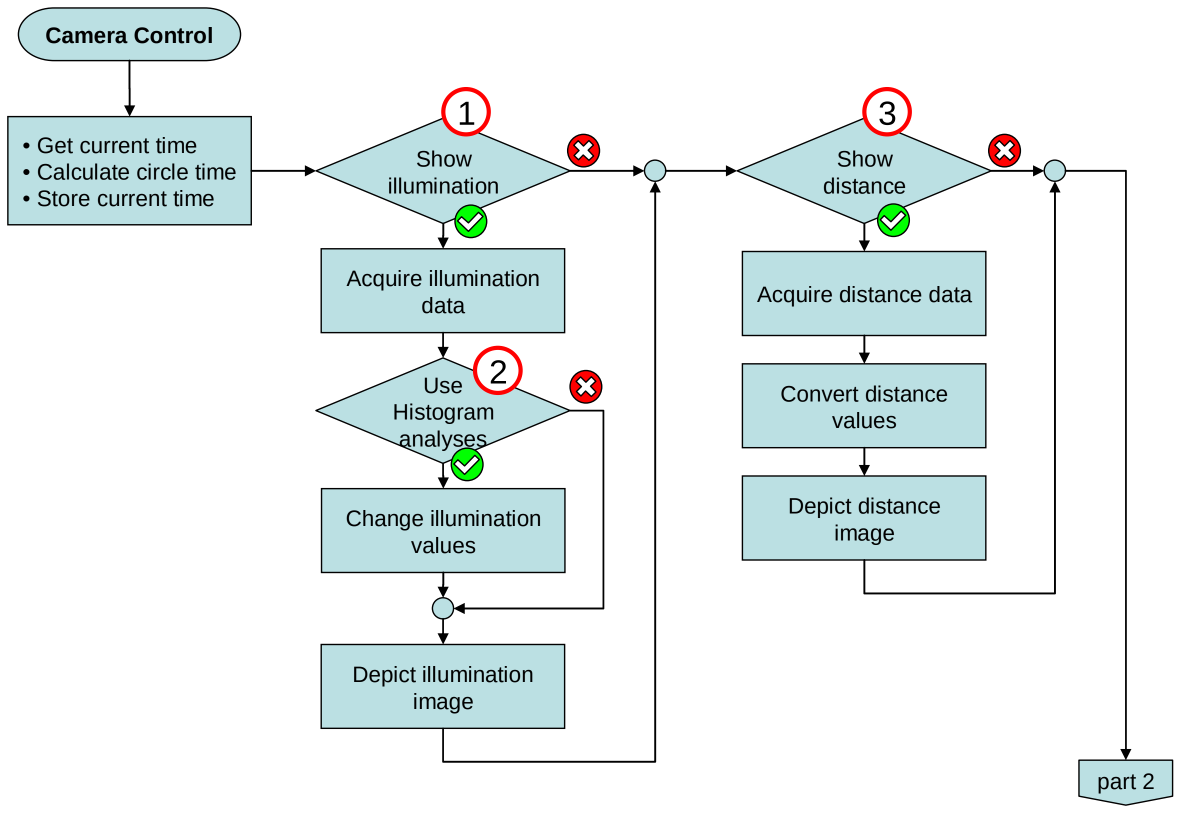


Figure 20: Camera control thread – part 1
Timer, intensity and distance depiction

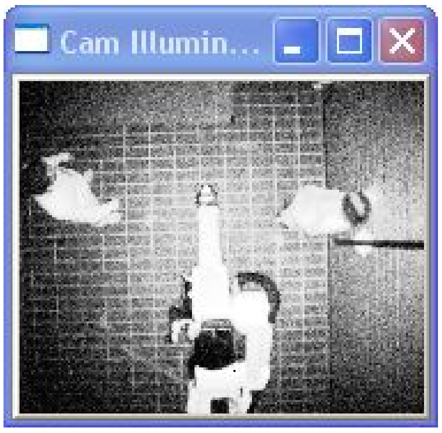
The performance of the program must be known, in order to compute the necessary safety distance the robot has to keep from the operator (see chapter 6.3). For measuring the performance of the single parts of the program it is necessary to acquire the time a single loop takes to compute. Thus, at the beginning of each loop the current processor time is taken and compared to the time of the last loop. This allows for an estimation of the frames that are handled in a certain amount of time.

The number of frames handled per second changes within a small range between two loops. Thus, in order to avoid problems in the depiction of the value, it is not continuously transferred to the thread of the GUI, but is acquired by the GUI if the user pushes the respective button ("Get FPS" – Frames Per Second) and displayed alongside the button. Also this has the advantage that the value can be better taken during specific situations.

The first subpart that can be activated by the user in Figure 20 is the depiction of greyscale values. These values represent the intensity of the light in the near infrared spectrum, reflected of observed objects. A sample image is depicted in Figure 21a, including two operators in the workspace. The depiction can be optionally enhanced by a histogram analyses that was developed during previous work [78] (see chapter 5.1.1).

The second subpart is the depiction of the distance data as shown in the example in Figure 21b. This algorithm (see chapter 5.1.2) was developed in another project [79] and already implemented in previous work of this project [78].

a)



b)

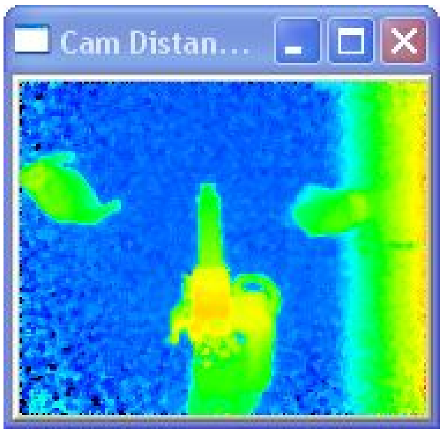


Figure 21: a) Intensity Image enhanced by histogram analysis and b) Distance Image

These two subparts of the program are mainly used during the setup phase of the workplace, when it is assembled or reassembled. In this phase the 3D camera has to be positioned and adjusted to the workplace. Both views are useful, since they show the camera's actual view. They show the region, covered from the camera in its current position, as well as problematic parts of the regions, where the distance values change during continuous measurements due to the camera's inconsistencies (see chapter 3.2 and report [80]). This can be used to manually optimize the camera's position.

Since the project was developed to run as a demonstrator, these images are also used to explain the camera's function during demonstrations. The enhancement of both images is not tied to the functionality of the program but is only done with respect to a human observer.

4.3.2 Background acquisition

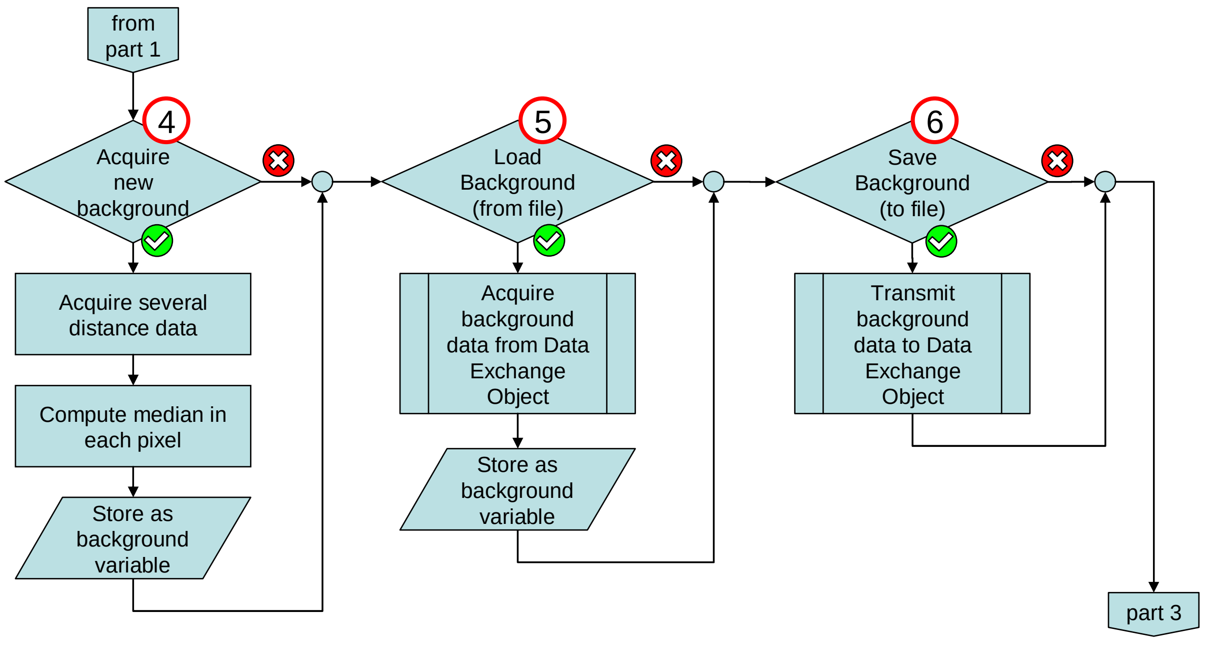

from part 1
4
Acquire new background
Acquire several distance data
Compute median in each pixel
Store as background variable
5
Load Background (from file)
Acquire background data from Data Exchange Object
Store as background variable
6
Save Background (to file)
Transmit background data to Data Exchange Object
part 3

Figure 22: Camera control thread – part 2
Recording, storing and retrieving the distance values of the workspace's background

The first subpart of the second part in Figure 22 (button 4) is the acquisition of the background of the workplace. This acquisition has to be activated, if the workplace has been altered, e.g. if a new static object was included in the workplace. During this acquisition the workplace has to be free of dynamic objects, which generally includes the robot. Since this is not possible with a robot fixed to the floor, a compensation algorithm has been programmed (see chapter 4.3.4) that later deals with the robot's unmovable base.

The distance data of the camera contain fluctuation noise, caused by inaccuracy in the distance measurement of the 3D camera. The manufacturer specifies an error of 1% of the measured distance in the camera's manual [55]. Thus, to achieve reliable data from the environment, several distance data samples are taken (500) to compensate for the camera's error. The median value in each pixel is computed (see chapter 5.2.1.1) during this measurement process. The resulting data is smoothened by a Gaussian function to further compensate errors in the measurement (see chapter 5.2.1.2). Upon completion of the background acquisition, the data is stored in memory.

As long as there are no changes to the workplace, the background data remains the same. Therefore this data does not need to be reacquired with every program start, but can be loaded from a stored set of data. This saves time during the setup phase of the standard program start. Also the background data remains consistent between two program calls, which is necessary during the testing and evaluating process of the program.

The loading of the background data from a file, which is stored on the computer alongside the program, is realized in the second subpart (button 5). This process is the standard procedure of loading the background data into the program. Thus this subpart is automatically activated once, alongside the thread’s activation. The user only needs to take action in acquiring new background data, if the workplace was changed.

If a change in the workplace occurred, new background data has to be acquired to guaranty a safe operation of the robot. The newly acquired background data has to be stored to the respective file by use of the third subpart (button 6). In this case, the currently stored background data in the file is overwritten with the background data in the program’s variable.

Since other threads may work on the background data simultaneously, which can cause problems, all file operations of the program are handled by the data exchange object that takes special measures to avoid those problems (see chapter 4.6).

4.3.3 Adjusting offset and intrusion filter

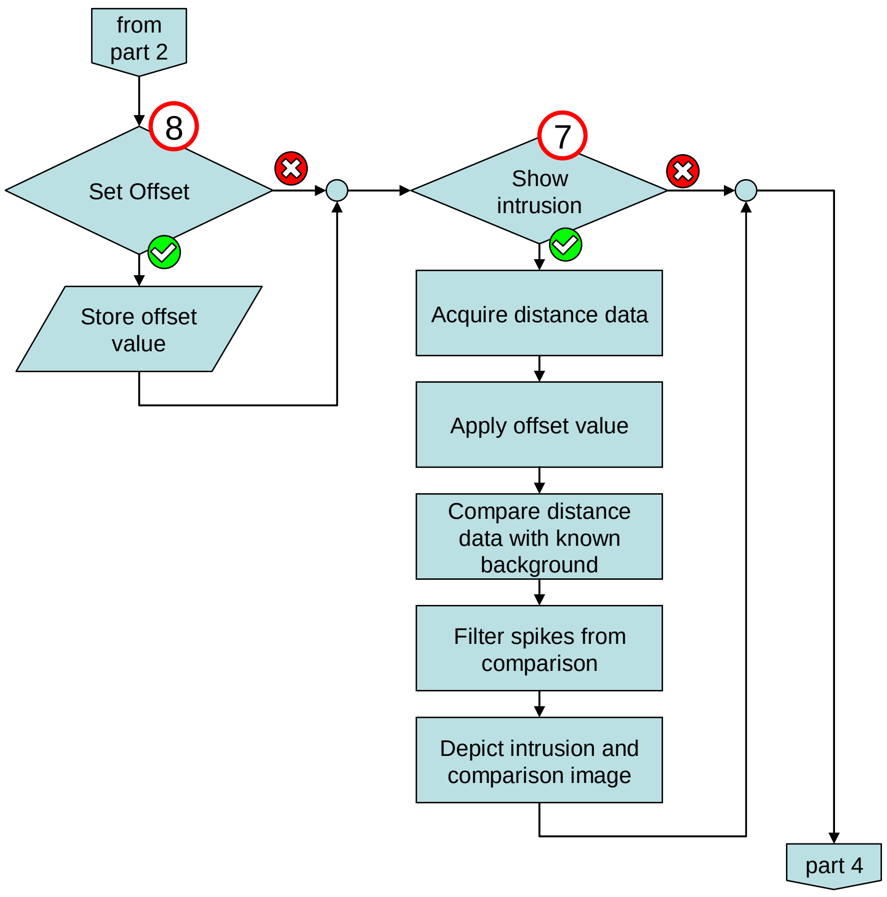

from part 2
8
Set Offset
Store offset value
7
Show intrusion
Acquire distance data
Apply offset value
Compare distance data with known background
Filter spikes from comparison
Depict intrusion and comparison image
part 4

Figure 23: Camera control thread – part 3
Setting the offset value and showing the detected intrusions

This part depicts an intrusion image, shown in Figure 24, which is to be used, to adjust filter parameters to the workplace and its environmental conditions.

As described in chapter 4.3.2, the distance measurement of the camera contains fluctuation noise that has to be compensated. Since the algorithms working on life data can not acquire a large amount of images in order to calculate a reliable median like with the background data, this inaccuracy has to be dealt with by other algorithms. Most of those algorithms that work on single images take a lot of computational effort. A simple, and computationally cheap, way of noise compensation is to add an offset to the background, causing intrusions only to be recognized, if a certain threshold is exceeded. Since a percental offset can not be used, due to the necessary safety analysis (see chapter 6.3) the offset distance has to have a specific distance value. Using this offset value allows the following compensatory algorithms (see Figure 24) to work with data that is much less noisy and thus easier and faster to filter further.

With the use of button 8 of the GUI (Figure 19), contained in the third part of the flowchart in Figure 23, the user transmits this offset value to the camera control thread. The offset then is continuously added to the background, whenever the background is accessed for comparison. This offset-filter (see chapter 5.2.2.1) needs to be adjustable in the demonstrator, since it influences the false positive and false negative detections.

- If the chosen value is too low, static noise is seen as intrusions. (false positive)
- If the chosen value is too high intrusions close to the background are lost. (false negative)

In industrial applications a value of more than 120 mm can not be allowed (see chapter 6.2.1).

The second subpart in Figure 23 (button 7) displays the intrusions, detected in the workplace. This subpart uses the algorithms described in chapters 5.2.2.1, 5.2.2.2 and 5.2.2.3. It starts by comparing the currently measured distance values of all pixels, acquired from the 3D camera, with those stored in the background values (see Figure 22). As described above, an offset is added to all currently measured distance values during this comparison. The next step of this subpart is a filtering of the remaining noise. This can be done either by the spike filter (see chapter 5.2.2.2) or the plausibility filter (see chapter 5.2.2.3). Both filters are contained in the source code of the program. As explained in the algorithms’ chapters, the plausibility filter yields better results and is thus activated in the program. The spike filter was kept for evaluation and comparison. In the end of the subpart, the results of the algorithms are displayed. Sample images of the remaining possible intrusions, using different settings are depicted in Figure 24.

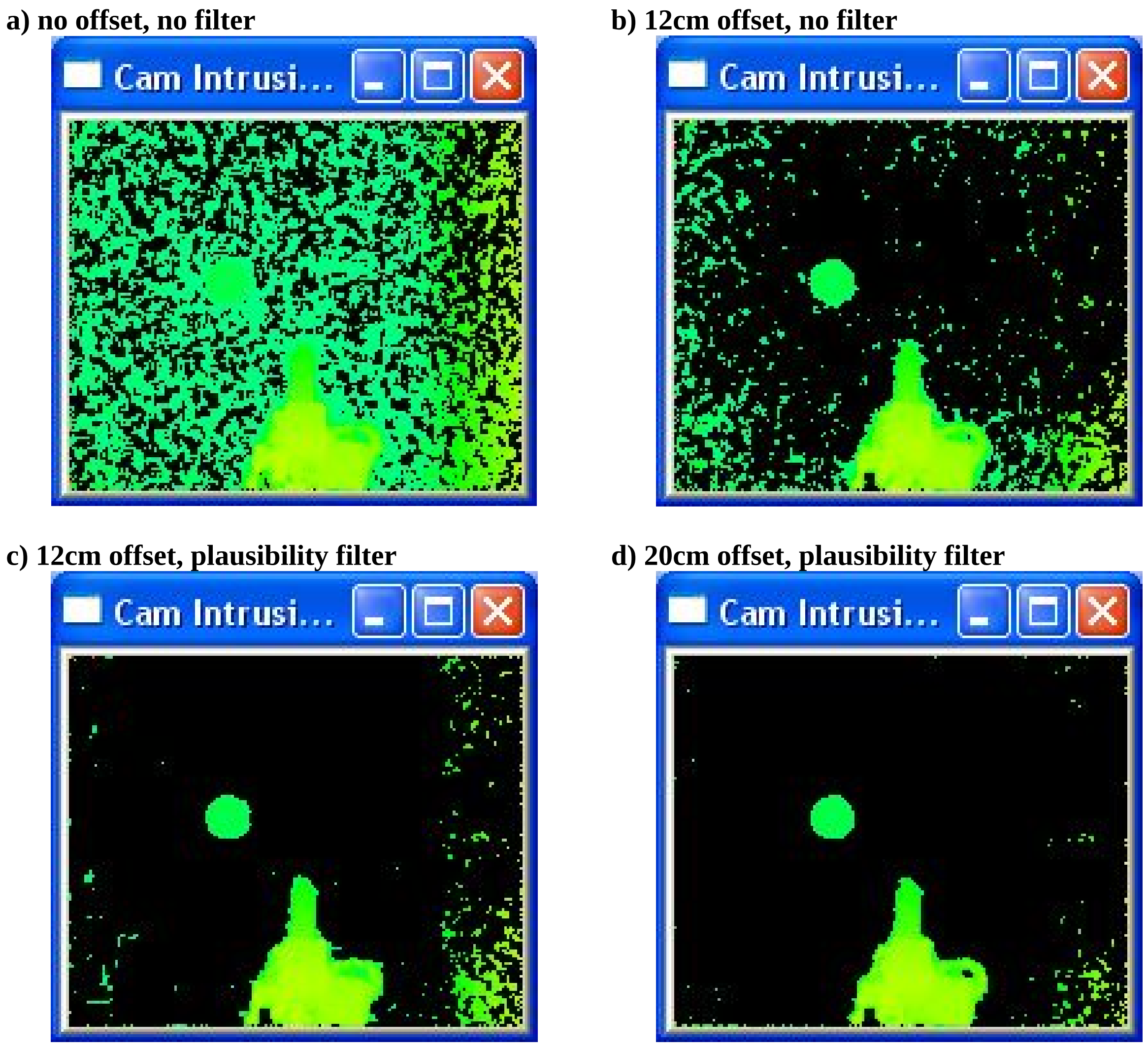

a) no offset, no filter
b) 12cm offset, no filter
c) 12cm offset, plausibility filter
d) 20cm offset, plausibility filter
Cam Intrusi...

Figure 24: Detected intrusions with different settings

Figure 24a shows the intrusions detected by the program, if no offset and no filter is used. The fluctuation noise is clearly visible. At this stage, a filtering algorithm based solely on the presence of intrusions would hardly be possible, because too much information would be lost along with the noise.

In Figure 24b, an offset of 12 cm was used. As described in chapter 6.4 this setting is still applicable in the industrial praxis, due to safety reasons. The remaining intrusions are reduced to such a point, that an additional filtering with the plausibility filter, shown in Figure 24c, leaves only small connected regions that can mostly be filtered by the region size filter described later in chapter 4.3.4.

In the demonstrator, the offset setting of 12 cm leads to the detection of small artefacts in every other frame, such that a setting of 20 cm of the offset filter was used in this project, its result shown in Figure 24d. In this setting the remaining artefacts were small enough to be later filtered out by their size. As described in chapter 6.3 this has to be improved for industrial application.

In these images of the detected possible intrusions problematic regions can be identified, in which strong fluctuation noise is present. Due to the nature of the used camera, which uses active spotlight intensity, the noise is worse in the corner of the image, where the intensity is lower. Another factor for noise is the vertical wall on the right side of the image, which causes problems due to its steep slope, because a small deviation in the horizontal measurement causes a large deviation in the vertical measurement, the distance measurement.

The following subparts that are used to create data for the robot's path planning are based upon the resulting intrusion image Figure 24d.

4.3.4 Region growing and reachable space computation

During the working process the program has to adjust the robot's movements in order to avoid collisions between the robot and other objects. Therefore the positions of the robot and the dynamic objects in the environment have to be detected and the objects have to be avoided. As described above, single images have to be used, to achieve a real time application. The intrusion detection, described in the previous chapter, delivers single images that still contain little noise. The noise needs to be reduced further to identify the real intrusions. This is achieved by the subparts in this chapter by region growing and size differentiation. Additionally the robot has to be identified in the image to differentiate between dynamic objects that are obstacles for the robot and the robot itself. This is achieved by manual targeting by the user.

To avoid dynamic objects two possibilities are implemented in this work, as described in chapter 2.3, and can both be activated by the user. One is the calculation of a flexible fence that defines the space the robot is not allowed to leave during its motion (reachable space) and the other is to control the robot's speed by its distance to dynamic objects (approach control). To enable the path planning thread, described in chapter 4.4, to control the robot accordingly, the reachable space as well as the distance between the robot and the nearest object has to be known.

The part four of the camera control thread, shown in Figure 25 contains the main subparts of the camera control thread that are active during the working time of the robot. These are the calculation of the reachable space (button 9) and the calculation of the distance between the robot and the nearest object (button 10).

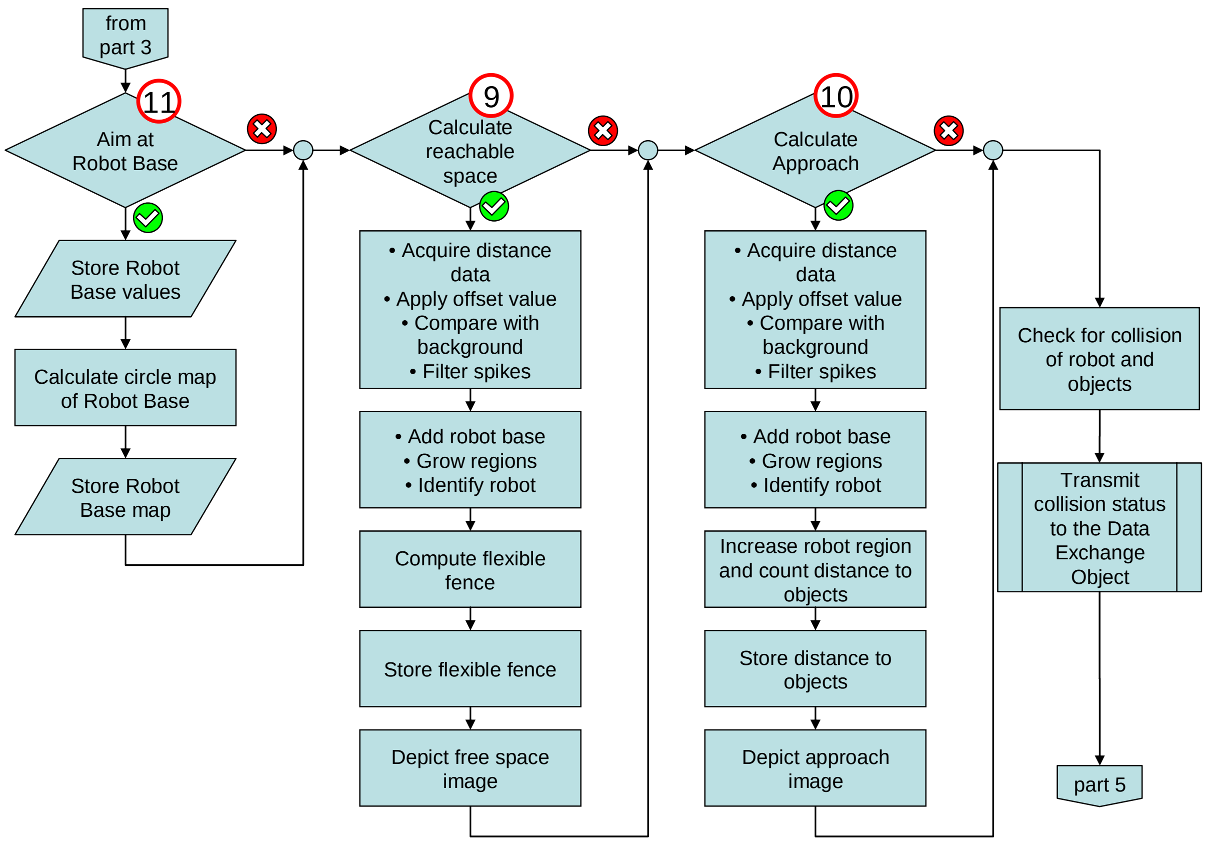


Figure 25: Camera control thread – part 4
Aiming at the robot base, calculating the reachable space
and the approach of the robot towards dynamic intrusions

Before those subparts can be executed, the program needs to be able to identify the robot in the intrusion image (see chapter 4.3.3). Since the robot's base is fixed to the floor, it can be identified by its position in the image. It is part of the workplace's setup and needs only to be redone if the relation between camera and robot has been changed.

Every object that is closer to the camera than the acquired background shows up as an intrusion. To later detect the robot, it has to completely show up as one intrusion. Since the robot's base can not be removed during the acquisition of the background data, it will not completely show up as an intrusion in the intrusion image, as shown in Figure 26a. To compensate this problem, a virtual robot base intrusion is calculated (see chapter 5.2.3) from a targeted position and a given radius and added to the filtered intrusions (see chapter 4.3.3). This position of the robot's base is visually targeted by the user and transferred to the camera control thread by the entering of its centre point and its radius. The result can be visually confirmed by the user in the resulting image, shown in Figure 26b. The base's position is used by the region growing algorithm, to identify the region that contains this position as the robot. This region is visually distinguished from all other reason by its green colour.

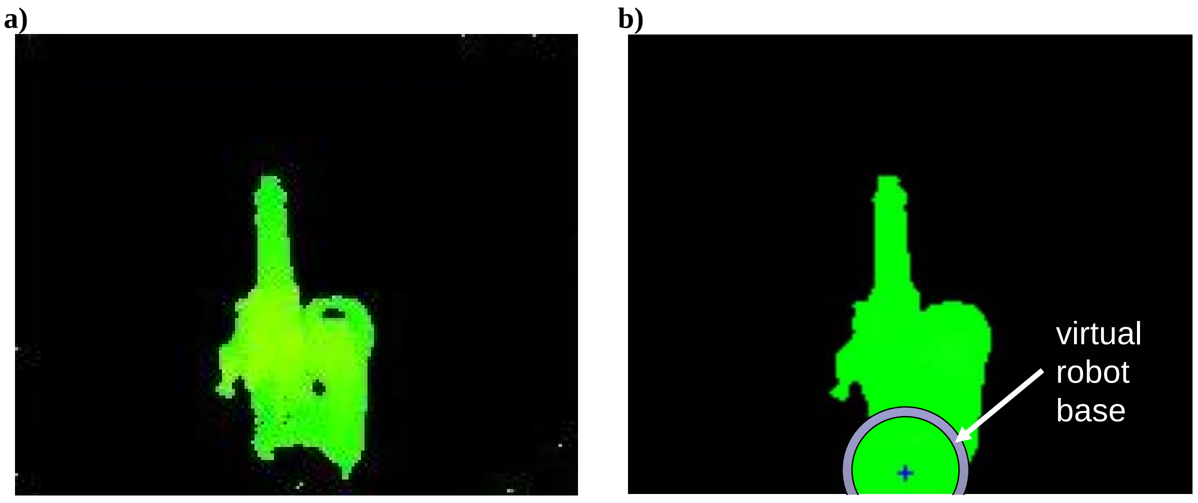
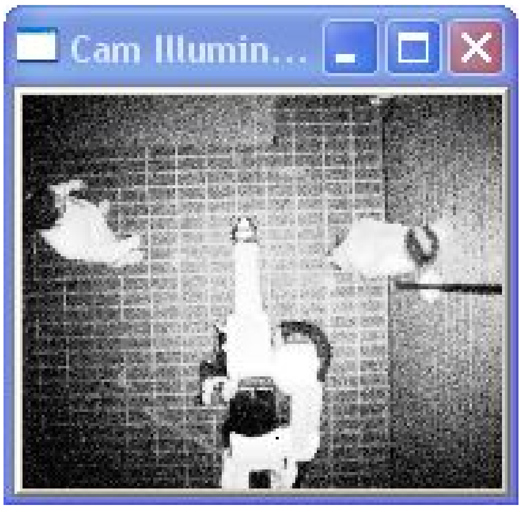


Figure 26: Detected intrusion a) before targeting and b) after targeting the robot base

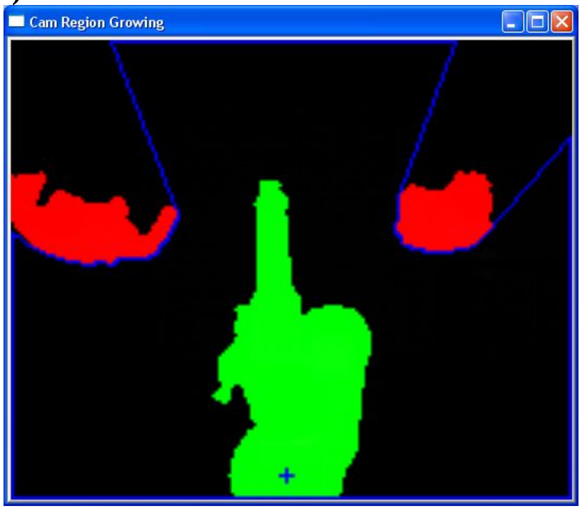
The calculation of the reachable space results in the flexible fence as displayed in Figure 27b. The computations start with the data of the intrusion detection (Figure 24) including the virtual robot base. Upon this data a region growing algorithm is executed (see chapter 5.2.4.1). The remaining noise is eliminated by a size filter (see chapter 5.2.4.2). Then the flexible fence, depicted as a blue line, is computed (see chapter 5.5.3). The resulting vectors of the blue line are transmitted to the data exchange object (see chapter 4.6) to be later used in the path planning control thread (see chapter 4.4).

The next subpart is the calculation of distance between the robot and intruding objects, for the implemented approach control. It can be activated by button 10, which automatically deactivates the reachable space calculation. The resulting image of this subpart is displayed as shown in Figure 27c. The green area shows the robot, surrounded by (green) distance lines which automatically extend around the robot's area, until they touch the nearest (red) intruding object. To achieve this image, again the data of the intrusion detection (Figure 24) containing the virtual robot base is used. As before, regions are grown, identified and filtered. The region identified as the robot is taken by the algorithm described in chapter 5.4, which creates the distance lines by uniformly enlarging the taken area, until one intruding region is touched. The steps of this algorithm are counted and transmitted to the data exchange object (see chapter 4.6) to be later used in the path planning control thread (see chapter 4.4), which controls the robot's speed according to the distance.

a)



b)



c)

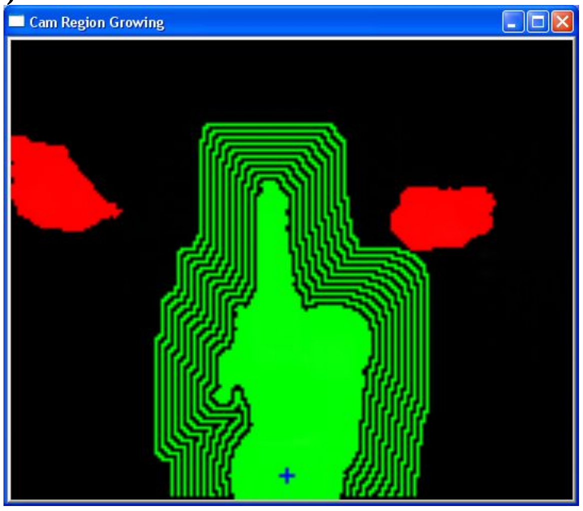


Figure 27: a) Based upon the situation in the workspace (see Figure 21a)
b) Image of the robot and the reachable space
c) Image of the robot and its distance to the nearest intrusion

A problem of both approaches up to this point is that if an object touches the robot, the object would appear to be part of the robot, due to the region growing algorithm. As described in chapter 2.3, the robot has to be stopped in this case because of safety reasons. Otherwise the robot would act as if no obstacle is present.

Therefore the next part of the camera control thread, a part that can not be deactivated, is to check, if a collision between any object and the robot itself took place. This algorithm that uses the regions found in the actual cycle and the cycle before is described in chapter 5.3. The result is transferred to the data exchange object, for later use in the path planning thread.

4.3.5 Determining the relations of the coordinate systems of robot and camera

The robot and the camera each use their own coordinate system, in which they operate. To be able to use the data acquired by the camera for the control of the robot, the relationship between those systems has to be known. This problem can be mathematically solved (see chapter 5.5.2) if the positions of three specific points in the workspace are known for both coordinate systems.

As described in chapter 3.3, the workplace therefore has been equipped with three reference blocks that are to be positioned at three fixed points. Thus their position in respect to the robot, whose position is fixed as well, is known. Since the camera is flexibly mounted, the positions of the blocks in respect to the camera change if the camera is repositioned.

This part of the camera control thread deals with the acquisition of the position of those points in the camera's coordinate frame and the calculation of the transformation matrix that describes the relationship between the coordinate systems.

As described in chapter 4.2, the region "Coordinate relations" contains additional controls for the camera thread, which need to be used only in the first setup of the workplace, or if the camera's position in the workspace has been changed (see chapter 3.3).

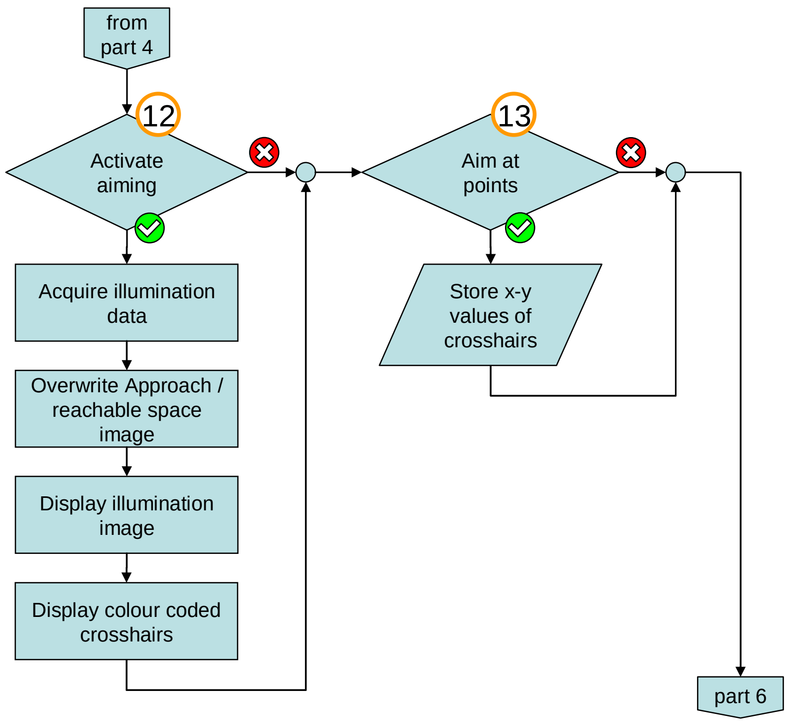

from part 4
12
Activate aiming
Acquire illumination data
Overwrite Approach / reachable space image
Display illumination image
Display colour coded crosshairs
13
Aim at points
Store x-y values of crosshairs
part 6

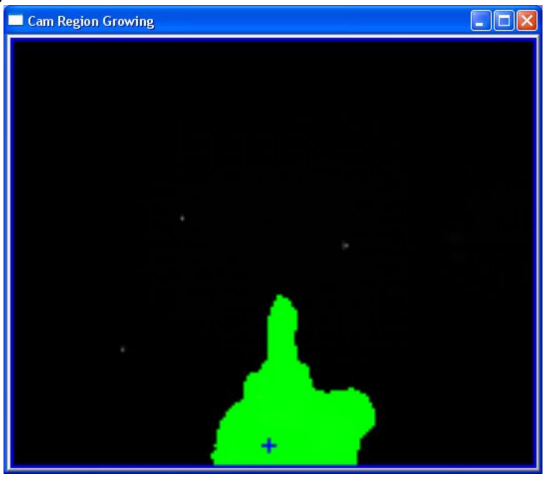
Figure 28: Camera control thread – part 5
Activating the image for the aiming at reference points and aiming at those points

The first subpart controlled in the "Coordinate relations" region, switched by button 12, acquires the current intensity data from the camera. It then exchanges the image, displayed by the approach / reachable space subparts (see Figure 27 and chapter 4.3.4), with the intensity image (see chapter 4.3.1), including the grown regions and colour coded crosshairs (see Figure 29b).

The three reference blocks, the reflective balls described in chapter 3.3, which have to be placed on their marked positions in the workspace, can be easily distinguished from the background. They are shown as white dots, as displayed in Figure 29a.

The user then has to manually enter the pixel coordinates of those dots in the GUI and transmit them to the camera control thread by using the button 13. The result of the manually transmitted coordinates is shown as three colour coded crosshairs at the manually entered positions (see Figure 29b), which have to be manually adjusted, until they match the white dots of the reference blocks.

a)



b)

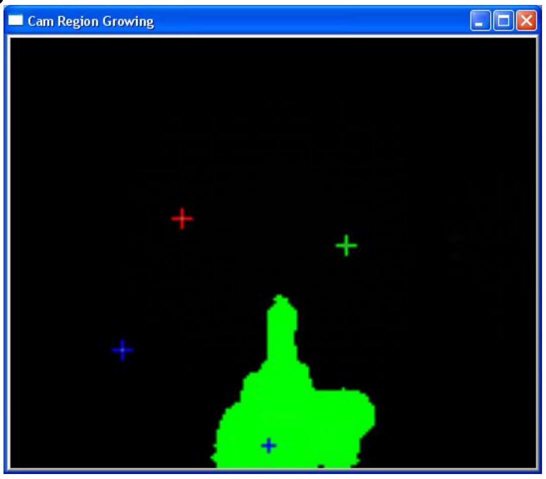


Figure 29: Image showing a) the highly reflective balls and b) the manual aiming process

The next part of the coordinate relation determination, and also the last part of the camera control thread, shown in Figure 30, deals with recording, storing and retrieving the voxels of the targeted reference blocks.

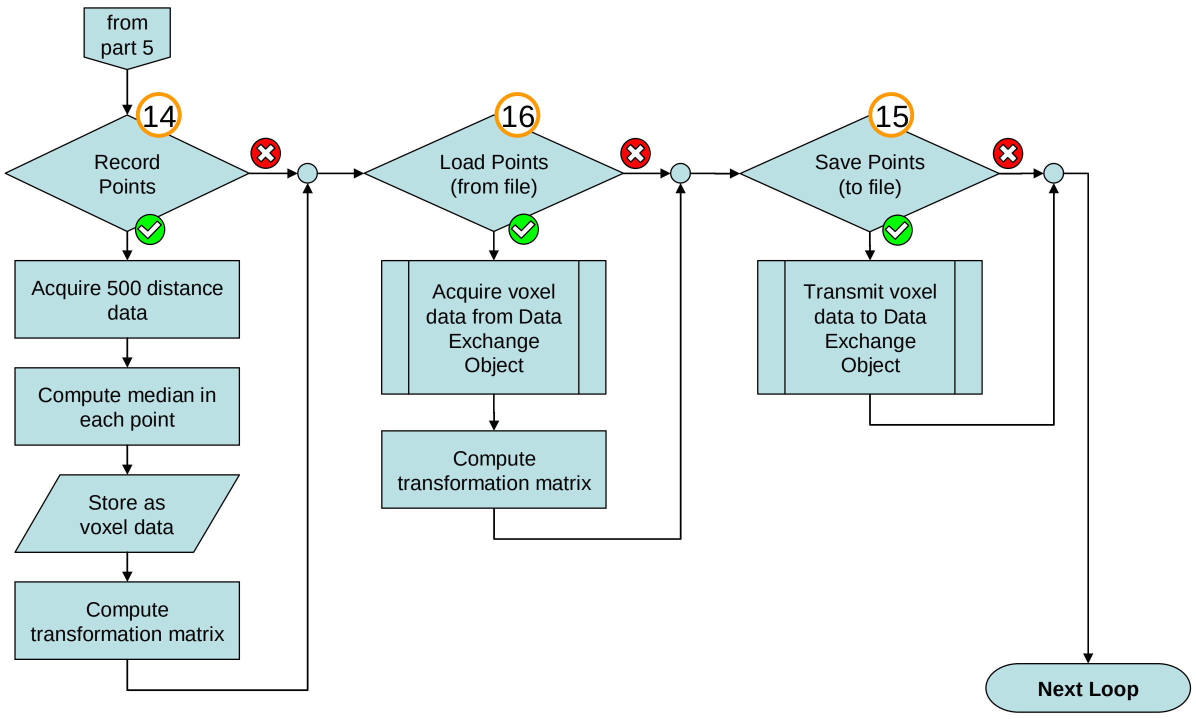

from part 5
14
Record Points
Acquire 500 distance data
Compute median in each point
Store as voxel data
Compute transformation matrix
16
Load Points (from file)
Acquire voxel data from Data Exchange Object
Compute transformation matrix
15
Save Points (to file)
Transmit voxel data to Data Exchange Object
Next Loop

Figure 30: Camera control thread – part 6
Recording, storing and retrieving the positions of markers in the camera coordinate system

To record the points, similar to the recording of the background image in chapter 4.3.2, 500 sample data are taken by the 3D camera. The medians of each targeted pixel is computed and stored. From those values and the respective values from the robot's coordinate frame, the transformation matrix between the camera's and the robot's frame is computed, using the algorithm described in chapter 5.5.2.

As with the background, the voxel data acquired in this part can be stored to disk and retrieved later. This shortens the start up time of the system. Only the voxel positions have to be stored, the transformation matrix can be easily recomputed after the retrieval of their values in the set up phase.

4.4 Path planning control

The path planning control thread has to compute the solutions described in chapter 2.3. That means it has to find a path for the robot, on which the working robot avoids the collision with intruding objects and is still able to achieve its desired goal as far as possible. The primary goal in this task is the avoidance of collisions.

As shown in Figure 18, the path planning control thread acquires pre-processed data from the camera control thread. This data contains among others the distance between the robot and objects intruding in the workspace and the outline of the reachable space.

The path planning control thread combines those data with a set of defined goal positions the robot has to drive through while working. In this project, to demonstrate the path planning and approach detection, the robot was given four goal positions that result in a rectangle in front of the robot, by which it attempts to drive. From the combination of the data the thread calculates the next pose (position and orientation) the robot has to reach respectively the speed in which the robot can do this, dependant on the chosen approach (see chapter 4.3.4 – reachable space or distance measurement). The result is transmitted to the robot control thread (see chapter 4.5) via the data exchange object.

4.4.1 Basic work routine

Figure 31, Figure 32 and Figure 33 contain the flowchart of the loop that is started in the path planning control thread. As with the camera control thread, all button numbers correlate with the colouring and numbering in Figure 19.

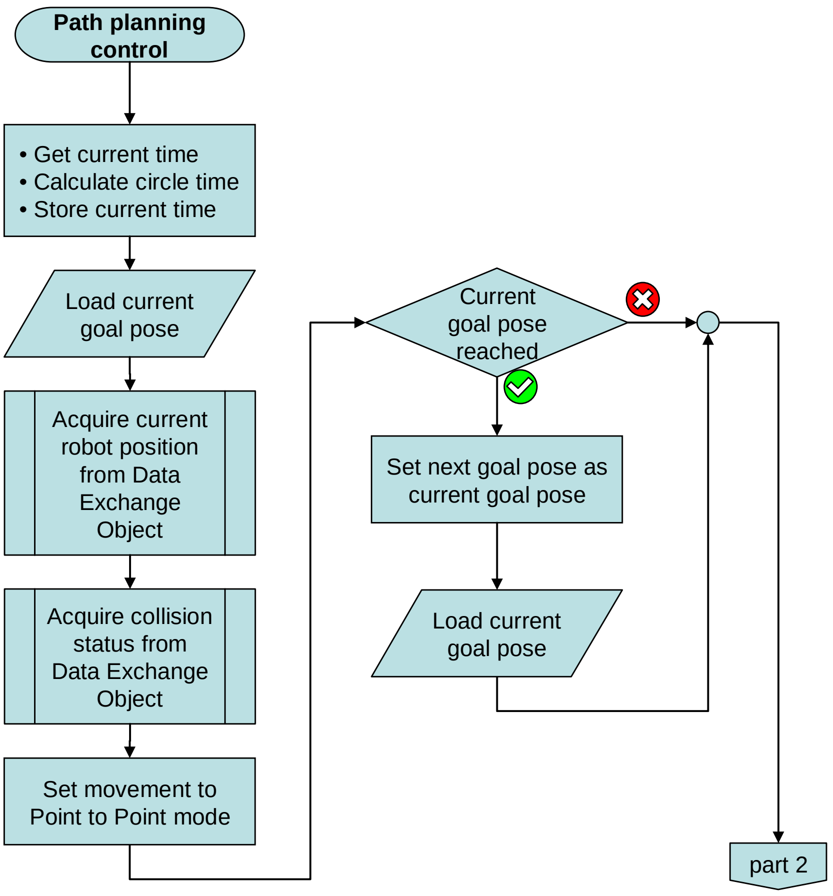

Path planning control
• Get current time
• Calculate circle time
• Store current time
Load current goal pose
Acquire current robot position from Data Exchange Object
Acquire collision status from Data Exchange Object
Set movement to Point to Point mode
Current goal pose reached
Set next goal pose as current goal pose
Load current goal pose
part 2

Figure 31: Flowchart path planning control thread – part 1
loading the necessary data and checking the current goal position

Like the camera control thread, the path planning control starts by calculating the duration of each loop, which is required for the evaluation of the program in chapter 6.3. After this, the loop executes data acquisition routines.

- The current pose (position and orientation of the endeffector) is loaded, the goal, towards which the robot is headed due to its working process.
- The current robot pose is loaded from the data exchange object, in which it was stored by the robot control thread (see chapter 4.5).
- The collision status, calculated by the camera control thread (see chapter 4.3.4), is loaded from the data exchange object.

Then the movement mode is set to “Point to Point mode” (see chapter 3.1), because this is the most used mode by the program. In those cases where the “Direct Drive mode” (see chapter 3.1) is needed, the movement mode is changed.

If the robot would be forced to reach each goal pose exactly, this would cause trouble in the operation cycle. Positions of the robot can be defined much more exactly in the robot’s variables than the robot can drive in praxis, due to the robot’s mechanical accuracy. This has to be compensated. Therefore in the next step the current pose of the robot, acquired earlier, is compared to the current goal position that has to be reached. This is done with the algorithm described in chapter 5.5.1. If the robot is within a given error margin of the current goal position, the next goal position is activated and loaded. In this project an error margin of 5cm between desired and reached goal was allowed. This results in a smooth driving of the robot through the goal positions of the demonstrator. In industrial applications the value needs to be adjusted to the desired path’s accuracy and the possible robot’s accuracy.

4.4.2 Speed control and object avoidance

As described in chapter 2.3, two different approaches of controlling the robot with respect to dynamic objects are implemented in this work. Those are the control of the robot’s speed, dependant on the proximity of dynamic objects, and the control of the robot’s path, dependant on the position of those objects.

The data about the distance between the robot and the closest intruding object in the workspace, as well as the data about the space the robot is able to reach without a collision with dynamic objects, is collected and evaluated in the camera control thread, described in chapter 4.3.4. This data needs to be combined with the robot’s goals, in order to calculate a safe speed, respectively a safe trajectory for the robot.

This is achieved in the second part of the path planning control thread, depicted in the flowchart in Figure 32.

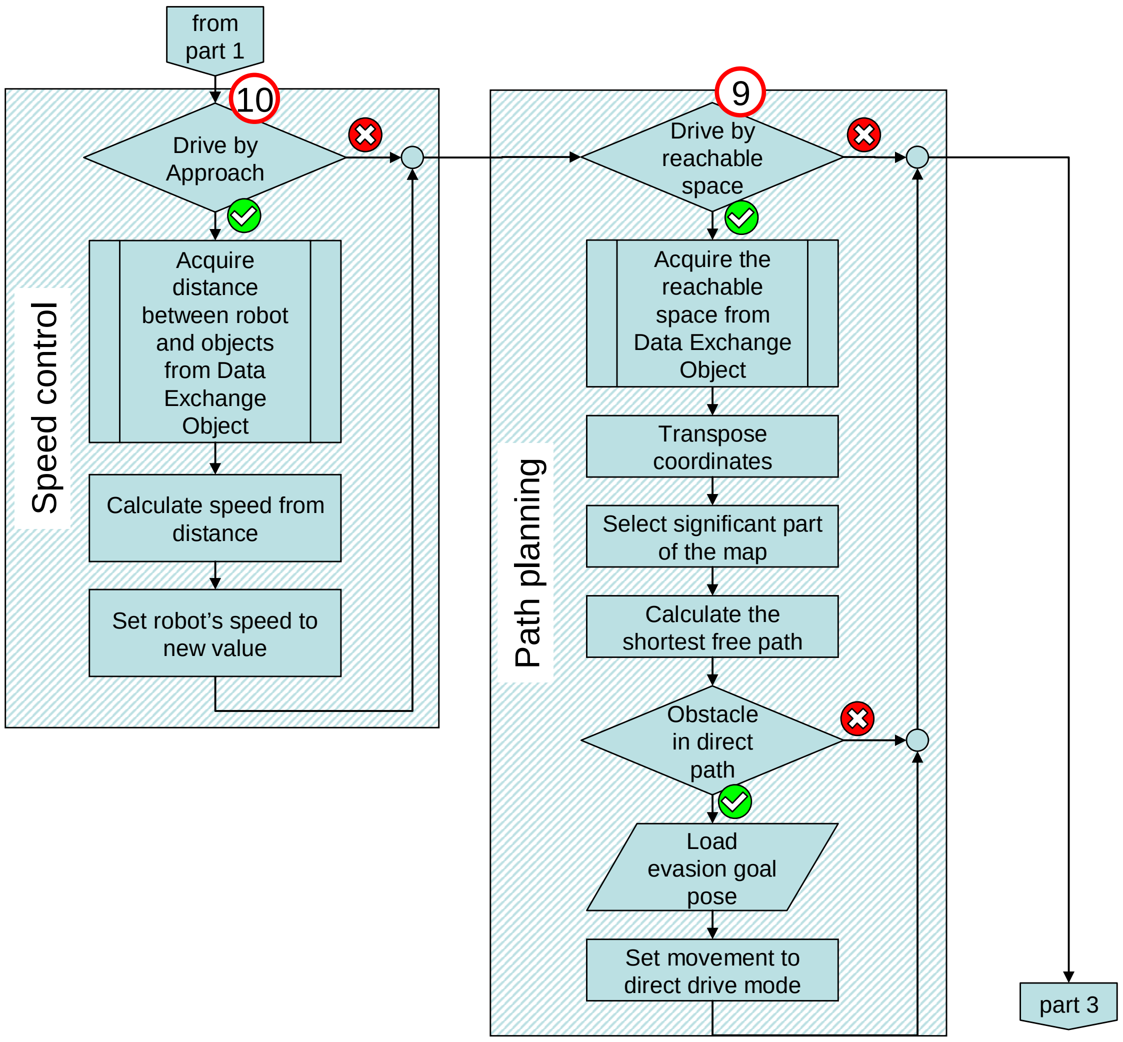

from part 1
10
Drive by Approach
Acquire distance between robot and objects from Data Exchange Object
Calculate speed from distance
Set robot's speed to new value
Speed control
9
Drive by reachable space
Acquire the reachable space from Data Exchange Object
Transpose coordinates
Select significant part of the map
Calculate the shortest free path
Obstacle in direct path
Load evasion goal pose
Set movement to direct drive mode
Path planning
part 3

Figure 32: Flowchart path planning control thread – part 2 processing the environmental data (see Figure 25) in path planning routines

The first subpart is the speed control, which is dependant on the distance between the robot and dynamic objects intruding into the workspace. It can be activated by the user (button 10), who automatically activates the acquisition of the required distance data as well. The data is transmitted via the data exchange object. This subpart acquires the data and changes the robot's current speed accordingly, using the algorithm described in chapter 5.4.

The next subpart, which requires the most computational effort in this task, is the path planning. It is activated by the user (button 9) if the robot is meant to evade intruding objects. As above, the activation of this subpart automatically activates the respective subpart of the camera control thread as well. The space the robot is able to reach, calculated by the camera thread, is acquired from the data exchange object. Since the coordinates of this space are still in the camera's coordinate system, they have to be transposed into the robot's coordinate system (see chapter 5.5.2). After this step, the significant part of the workspace, the part in which the movement takes place, can be chosen by an algorithm (see chapter 5.5.4). Based on the data about this significant part, the current pose of the robot and the next goal pose, another algorithm (see chapter 5.5.5) calculates the shortest free path, divided into sub-goals connected by straight lines, from the current pose to the goal pose around the detected dynamic objects (see flowchart in Figure 72). In this calculation a required safety margin (see chapter 6.3) is added to the objects. If this algorithm does not result in one straight line, a dynamic object is present between the current pose and the goal pose. In this case, the first sub-goal of the calculated free path is stored as the next pose to be reached and the robot's movement mode is set to "direct drive" (see chapter 3.1). The "direct drive mode" is necessary, since the trajectory of a "point to point" motion is not straight (see chapter 3.1). As with the basic work routine, described in chapter 4.4.1, positions can not be approached as exactly as they can be programmed. This algorithm thus sets a point as approached if the robot's tool centre point is within 3 cm of the given position.

4.4.3 Collision reset

Although the robot control thread is trying to prevent the robot from colliding with objects, collisions may happen, since the robot is only moving at a slow speed and can thus be easily reached by persons. This causes safety problems, as described in chapter 2.3, which have to be avoided. The detection of collisions between the robot and intruding objects is handled in the camera control thread (see chapter 4.3.4), and passed on to the path planning thread, which deals with the problem by causing the robot to stop.

For safety reasons after a collision is detected the robot is only allowed to "restart by voluntary actuation of a control device" [66]. Additionally the necessary information on the workspace below the robot is lost, as described in chapter 2.3, and has to be regained with the user's help. Therefore a detected collision has to be manually reset by the user, after the user inspected the workplace.

This user reset is handled in this part of the path planning thread (Figure 33, button 17). The reset of the collision status is stored and also transmitted to the data exchange object.

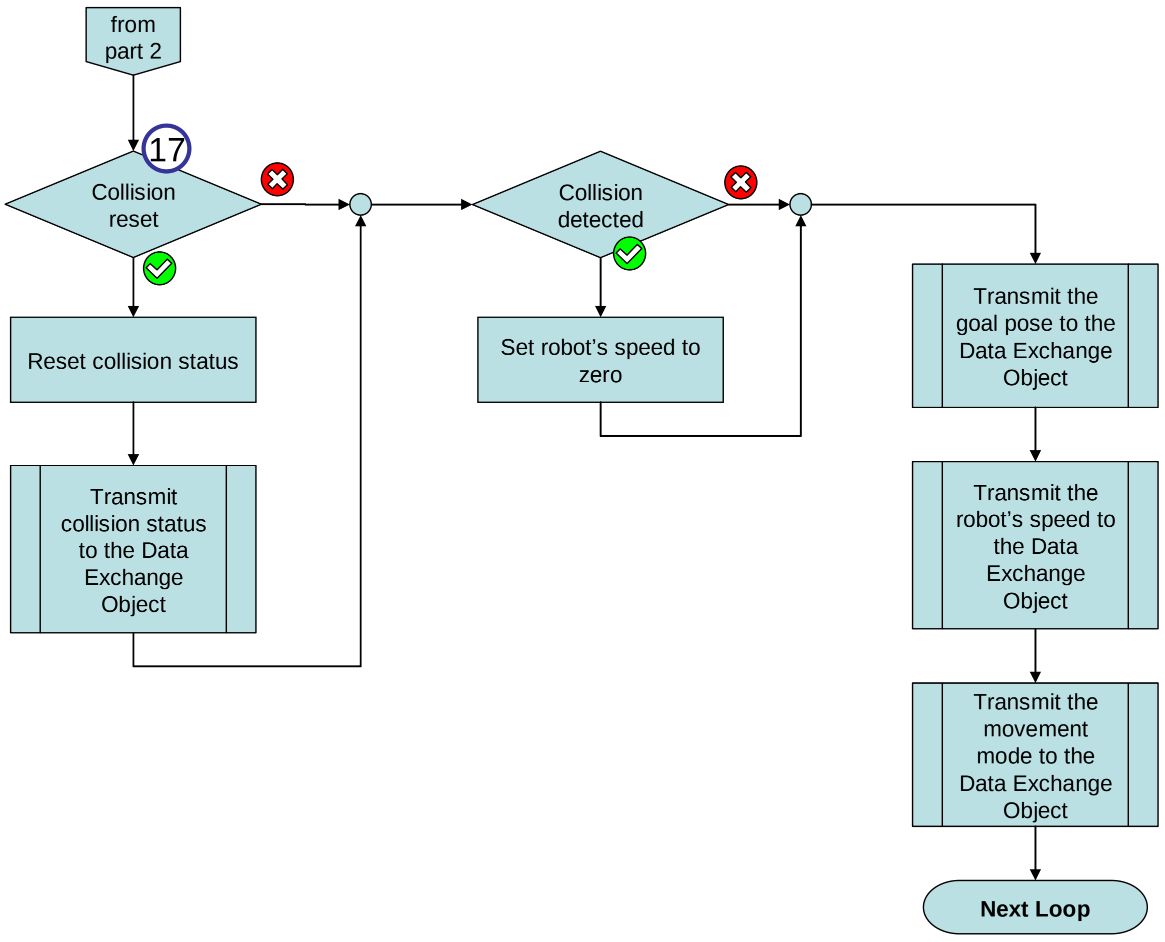

from part 2
17
Collision reset
Reset collision status
Transmit collision status to the Data Exchange Object
Collision detected
Set robot's speed to zero
Transmit the goal pose to the Data Exchange Object
Transmit the robot's speed to the Data Exchange Object
Transmit the movement mode to the Data Exchange Object
Next Loop

Figure 33: Flowchart path planning control thread – part 3
Handling collisions and transmitting the desired parameters

If a collision is still reported, the robot's speed has to be reduced to zero, which is handled in the next step. Since, due to technical reasons, this may not stop the robot at all times, a speed of zero is handled specially by the robot control thread (see chapter 4.5).

In the end of every path planning control thread loop, the goal pose, the desired speed and the movement mode are transmitted via the data exchange object to the robot control thread.

4.5 Robot control

To be able to control the robot effectively, the exact pose of its tool centre point needs to be known. This pose has to be acquired from the robot's internal sensors and made known to the developed program. On the other side, the pose data, computed in the path planning thread, have to be transmitted to the robot's PLC.

This data exchange between the PC-program and the robot is handled by the robot control task. Its task, as shown in chapter 4.1 is to pull all necessary sensor information from the robot, containing the current pose of all of the robot's axes, and relay them to the path planning thread, as well as to relay the movement commands from the path planning thread to the robot.

As a subtask, this thread also handles the manual movement commands, entered by the user in the GUI.

The robot is controlled, using the functions developed in previous work [78]. Those functions allow the robot to be controlled in its pose, its speed, its acceleration and its mode of movement via a C++ program running on a PC.

The connection between the robot's PLC and the developed program on the PC is handled via the TCP/IP protocol (see chapter 3.6).

The loop inside the robot control thread is depicted in Figure 34. This thread contains only the routines necessary for communication, in order to make the exchange of information between the robot and the PC-program as fast as possible. Although, it was necessary to delay between two loops for several milliseconds, because the robot's PLC stopped responding if movement commands were sent in a too rapid succession. A delay of 20 milliseconds proved to result in a stable communication.

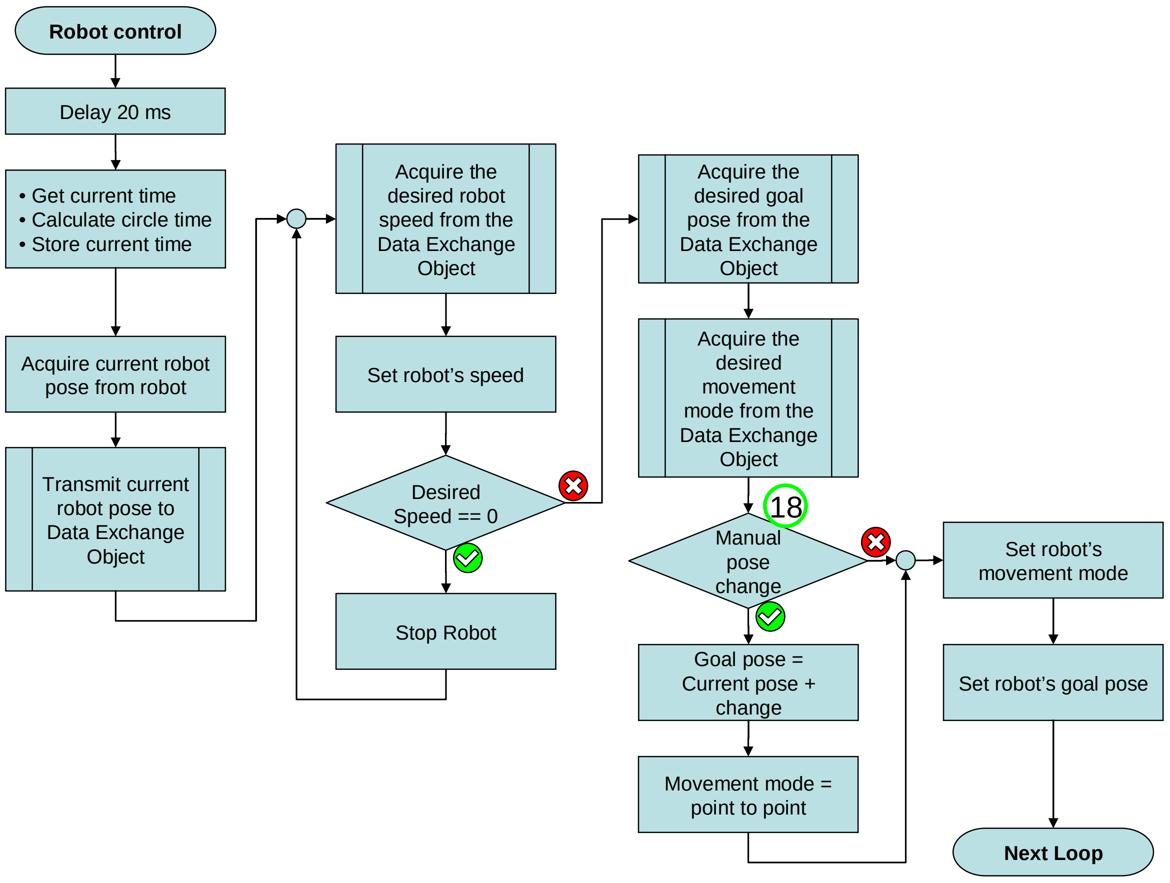

Robot control
Delay 20 ms
• Get current time
• Calculate circle time
• Store current time
Acquire current robot pose from robot
Transmit current robot pose to Data Exchange Object
Acquire the desired robot speed from the Data Exchange Object
Set robot's speed
Desired Speed == 0
Stop Robot
Acquire the desired goal pose from the Data Exchange Object
Acquire the desired movement mode from the Data Exchange Object
18
Manual pose change
Goal pose = Current pose + change
Movement mode = point to point
Set robot's movement mode
Set robot's goal pose
Next Loop

Figure 34: Flowchart robot control thread
Acquiring the desired settings and communicating them to the robot's PLC

After the initial delay, the thread measures its runtime, like the other threads. It then acquires the robot's current pose from the robot's PLC and transmits it to the data exchange object. Then it acquires the speed to be set and transmits it to the robot. As described in previous work [78], all commands concerning the mode of movement (speed, acceleration, movement mode) are collected in internal variables of the robot's PLC and only activated, if a new goal position is transmitted.

As described in previous chapters, the robot needs to be stopped in certain situations. Setting the desired speed to zero in the robot's PLC proved to result in the robot moving at the slowest speed possible that is not stand still. Thus, if the robot is to stand still, it proved necessary, to send the robot a stop command in stead of a speed of zero. After a stop command, the thread constantly checks the desired speed transmitted by the other threads and only continues in the thread’s loop if it deviates from zero.

If the desired speed is greater than zero, the task retrieves the goal pose and the desired mode of movement.

If a manual pose correction is desired by the user, by the use of button 18, the goal pose is set accordingly and overwrites the goal pose acquired from the path planning thread. The movement mode is set to point to point mode, allowing each axis to be driven separately.

In the end the movement mode is transmitted to the robot’s PLC. The movement of the robot towards its new goal, along with all commands concerning its mode of movement, is automatically activated in the robot’s PLC by the transmission of the new goal’s coordinates.

4.6 Exchange of information between tasks

Since reading and writing of data takes a certain amount of time and threads can overtake each other in their reading / writing process in the same variables, exchanging information between parallel running threads therefore can cause inconsistencies in the data. This is caused by threads reading in variables, or by threads writing to variables that are currently written in by other threads.

If a thread reads a variable that is currently written to by another thread, parts of the retrieved information might be old and parts might be new, which would corrupt the data once for the reading thread. If data is written in from several threads in parallel, this may result in the data being a mixture of both, which would corrupt the stored data, until it is overwritten again.

To be able to exchange data safely between the tasks and store this data to disk or retrieve it from disk, a data handling object was implemented. This object has internal status variables that contain information, whether a variable is currently read or written. If so, the access to that variable for all other threads is blocked.

For every variable that can be stored in or extracted from the data exchange object, there are two functions. The general layout of the functions is shown in Figure 35 and Figure 36.

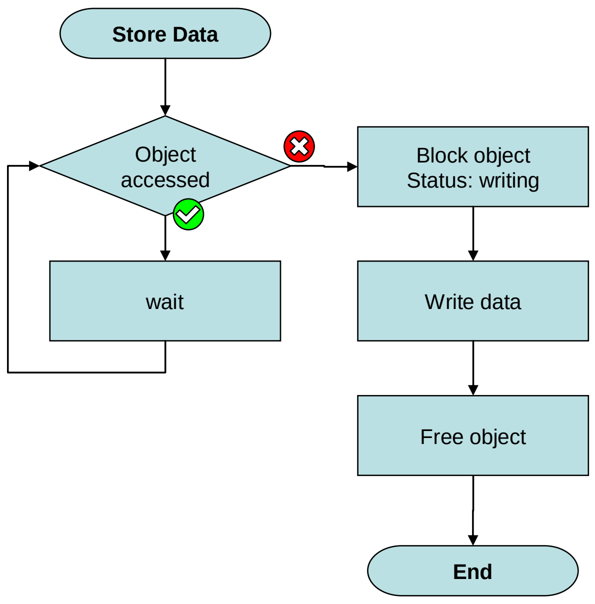


Figure 35: Storing data in the Data Exchange Object

Before data is stored in a variable, its status variable is checked whether the variable is already accessed. This ensures that a variable is not written to and read from, or written to by different threads simultaneously.

Since simultaneous writing of a variable is not important in this project, because objects are usually written to from only one thread, the security for this case is lower than in the reading procedure (see Figure 36).

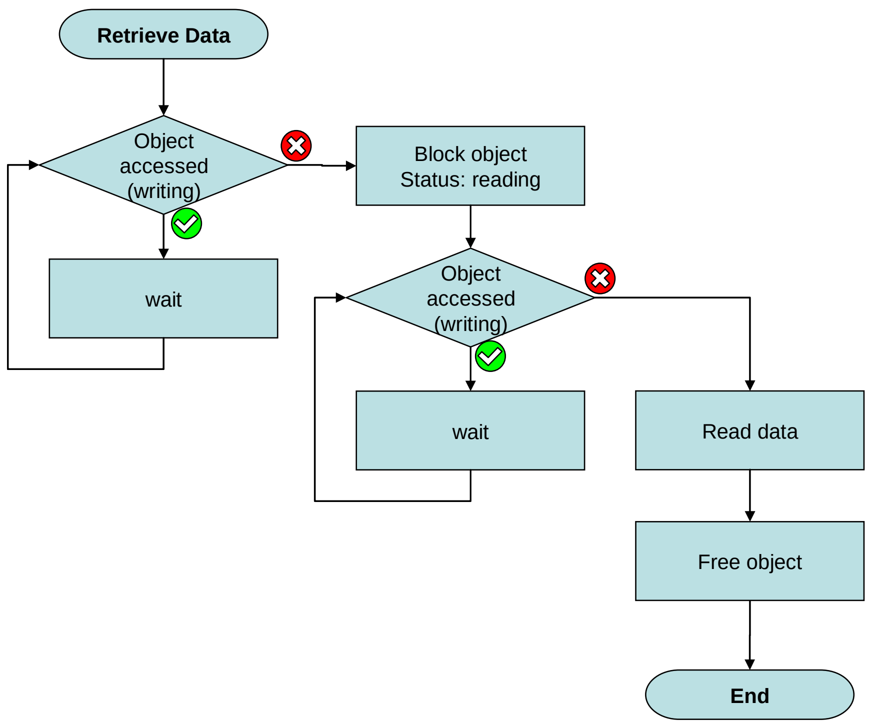

Retrieve Data
Object accessed (writing)
Block object Status: reading
wait
Object accessed (writing)
wait
Read data
Free object
End

Figure 36: Retrieving data from the Data Exchange Object

The reading procedure double checks against an accessed object, to avoid the occasion, where the reading and the writing thread access an object simultaneously. In this rare case, both procedures would check the object’s accessibility simultaneously, finding it free, and afterwards block it simultaneously and continue with their task. This is prevented in the retrieval routine by rechecking the access after the block.

The reading procedure does not need to concern itself with other tasks reading the variable simultaneously, since in that case the variable is not changed.

5 Algorithms for robotic evasion

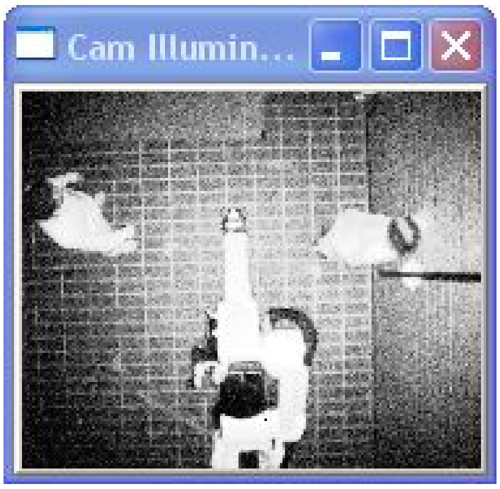
Chapter 4 gives an overview on the flow of the data and the internal connection of the subparts and algorithms, which control the robot by analysis of the 3D images of the camera and the position information of the robot. To understand this project in total, a deeper view on the algorithms is necessary. Therefore the used algorithms are explained in this chapter in detail. Some algorithms have been developed in previous work [78], but only the depiction of the acquired information (see chapter 5.1) could be reused without further improvement.

5.1 Depicting the acquired information

As described in chapter 4.3.1 it is necessary, to show the images and information acquired from the camera, in order to set up the workplace and explain the camera's function during demonstrations. These algorithms are not necessary in the object avoidance process of the robot. The algorithms controlling the robot are designed to work with the unmodified data, delivered by the camera.

Figure 37 shows two sample images, taken during operations.

a)



b)

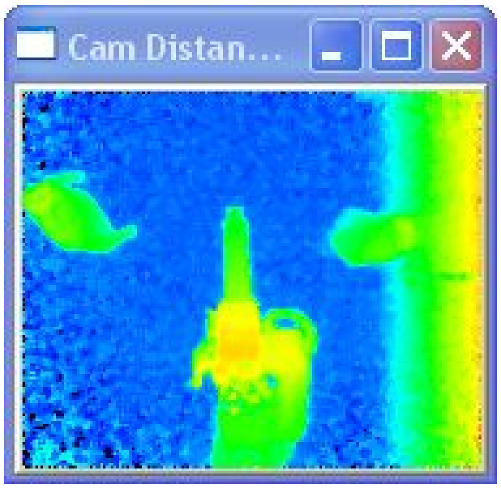


Figure 37: Sample images, taken in the workplace
a) reflection image and b) distance image (see also Figure 21)

The image data are acquired from the camera with the help of the C++ library supplied by the camera's manufacturer (see chapter 3.4). The library's function, responsible for the data acquisition, fills a prepared buffer variable in the developed program with an array of float numbers followed by an array of unsigned integer numbers. This process is described in detail in previous work [78].

The float numbers represent the spherical distance data in meter, the integer values indicate the quantity of reflected light per pixel.

To translate the spherical distance data into carthesian coordinates, the vectors of the respective pixels have to be used. This step is also achieved by a function supplied by the camera's manufacturer. For demonstrative purposes it is sufficient to only display the carthesian value of the z-axis, the one pointing away from the camera. The x and y value are not used in the depiction of the workspace but only in the later steps in path planning, to obtain a three dimensional representation of the environment.

There are several problems in the depiction of this data for a human observer. The problem in depicting the intensity data is that in many cases the reflectional properties of nearly all objects present in the image are similar. This leads to data in which many pixels have similar greyscale values. With the distance data, the task is to convert the distance data to colour values, to be able to create an image in which the distances are visualized.

Therefore for a human to be able to visually process the information acquired from the camera, the intensity image needs to be enhanced in its brightness and the distance image needs to be translated into colour. This master thesis applies the two algorithms that proved to be the best for this task in earlier parts of the project [78].

The earlier project [78] ascertained that the enhancement of the intensity image can be achieved by a scaling algorithm or histogram equalization. While the scaling algorithm is faster in its execution, the histogram equalization delivers better and more stable results, as shown in chapter 5.1.1. Since the enhancement of the intensity data is not necessary in the later process involving the robot, the duration of the algorithm does not need to be taken into account. Additionally the longer duration of the histogram equalization proved to be only a minor factor in the duration of the complete camera control thread loop (see chapter 4.3). Thus this algorithm was chosen for the project, because of its better output.

It was also ascertained in the earlier project [78] that to translate the acquired distance, a colour bar is to be used that is also used in the depiction of heat images and stress images. This depiction proved to be easy to understand since people are used to it.

5.1.1 Histogram equalization – improving the grey scale values

The problem in depicting the intensity data is that in many cases the reflectional properties of nearly all objects present in an image are similar. This leads to data in which many pixels have similar greyscale values. As long as all intensity values present in an image are within a short range, a simple scaling algorithm of those intensity values, described in Equation 3 and Figure 38a and b, can enhance the image sufficiently. The problem of this algorithm is the possibility of more than one cluster of intensity values. This is mainly created by the presence of highly reflective material, as shown in Figure 38c and d. In this case, the image’s intensity values are left nearly unaltered by the scaling algorithm.

$$used\_range = \max\_brightness - \min\_brightness$$

$$scaling\_factor = \frac{complete\_range}{used\_range}$$

$$new\_brightness = scaling\_factor * \left(brightness - \min\_brightness\right)$$

Equation 3: Scaling algorithm

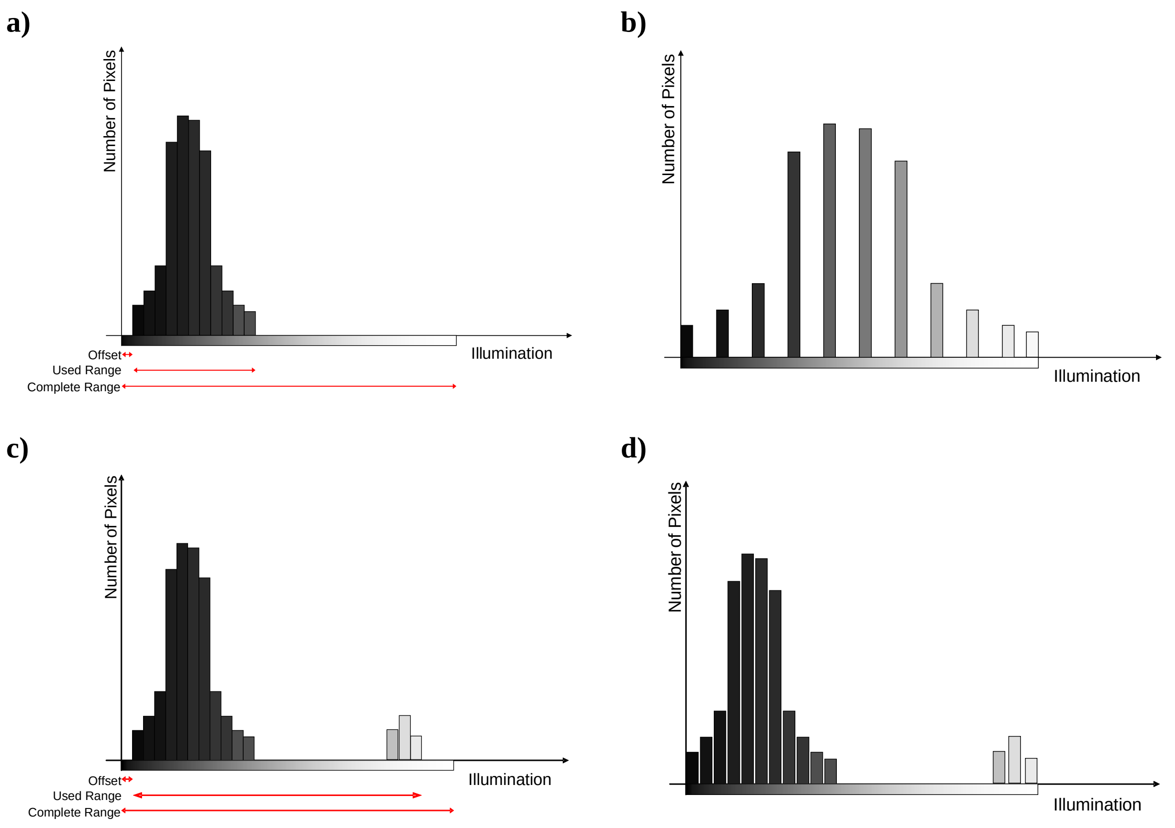


Figure 38: Scaling algorithm a) to b) with a small used range and c) to d) with a large used range

This pixel distribution of more than one information cluster is not a special, but a normal case, especially in industrial environments where lowly reflecting materials like cloth meet highly reflective materials like metals. In the case of the used workspace the mounting of the camera, visible in the camera's image, has a high reflectivity, whereas the floor has a very low one. Thus clusters at both ends of the intensity range are always present.

This problem can be avoided by using the histogram equalization, which is described in Figure 39, starting with the real distribution in Figure 39a and ending with the resulting distribution in Figure 39e. In the real distribution, there are two regions visible in which the majority of the pixels' values are located.

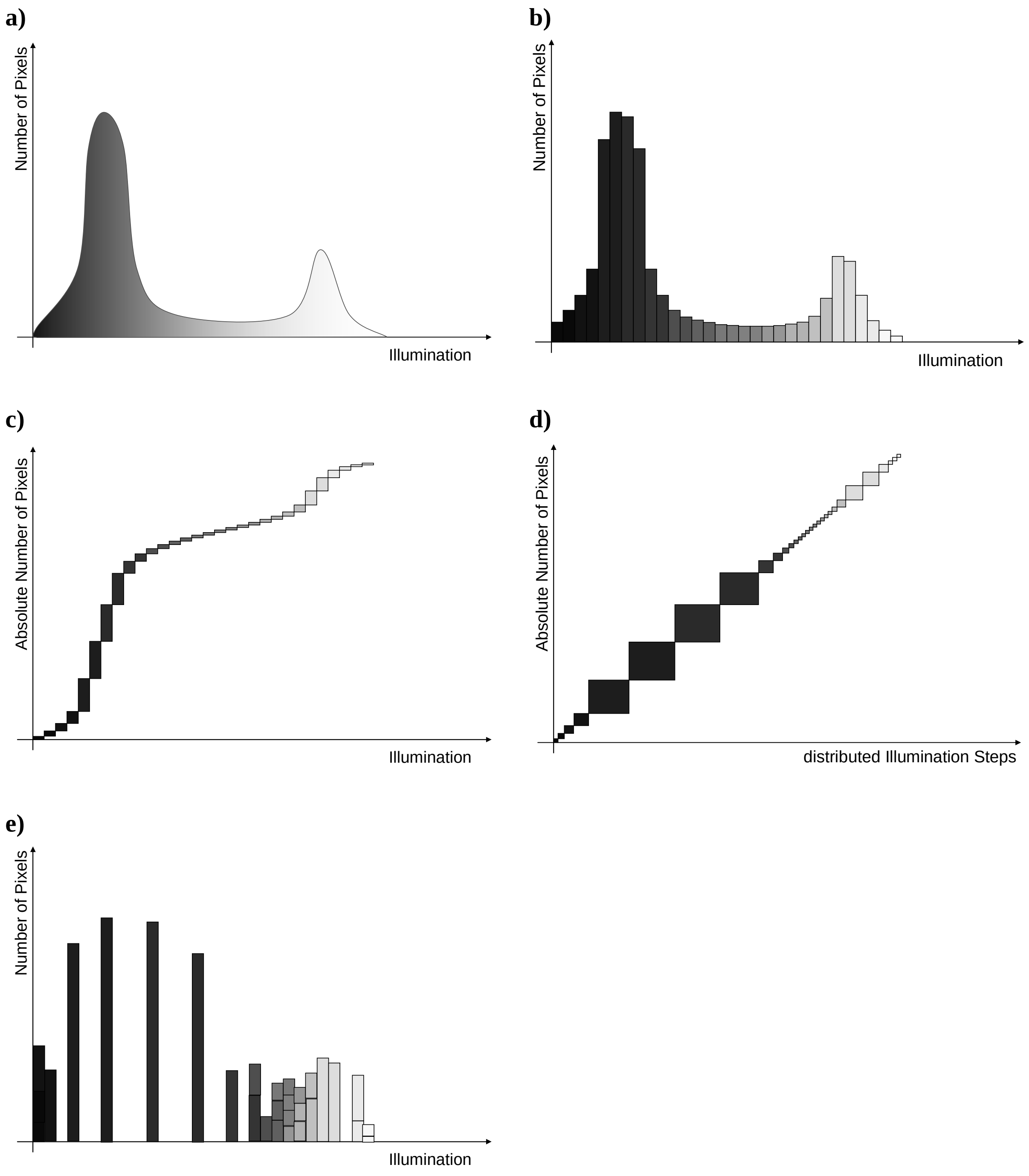


Figure 39: Histogram equalization
Redistribution of the greyscale values (a, b), based on their probability distribution function (c)

Figure 39a shows the distribution of the real values of an image throughout the spectrum. The location of the different pixels in the image is not considered, only their intensity value.

Those values are discretised by the observing camera, resulting in a spectrum of up to 65536 (in the given example 30) different greyscale values. The different amounts of pixels per value are counted, resulting in the histogram shown in Figure 39b.

To redistribute the values a probability distribution function (PDiF) is calculated. The PDiF, as shown in Figure 39c, is created by adding the amount of all pixels with lower intensity to each amounts of pixels (see formula in Equation 4).

$$PDiF_{pixels}(n) = \sum_{i=0}^{n} Number\_of\_pixels(i)$$

Equation 4: Formula to calculate the PDiF

$PDiF_{pixels}(n) =$ Amount of all pixels with an intensity value equal or lower than n

$Number\_of\_pixels(i) =$ Amount of pixels with the intensity value i

Based on this PDiF the intensity value of each pixel is changed by a mapping function, shown in Equation 5. This function changes the width of the intensity values, to match their respective heights. This results in the values being centred on a straight line (Figure 39d).

$$newIllumValue = IllumRange * \left(\frac{PDiF_{pixels}(IllumValue)}{Number_{Values}} \right)$$

Equation 5: Intensity value redistribution function

$newIllumValue =$ New intensity value of all pixels that have the old value

$IllumRange =$ Size of the complete range of intensity values

$PDiF_{pixels}(IllumValue) =$ see Equation 4

$IllumValue =$ Old intensity Value of a pixel

$Number_{Values} =$ Number of different intensity values

In the resulting intensity distribution, shown in Figure 39e, the intensity values are spaced evenly throughout the image. As can be seen from the result, due to the missing information behind the decimal point some previously separated values are compressed in a single value. This loss of information is not important to the normal observer, who is not trained to distinguish 65536 different greyscale values. In this project the reflection image was only used for depicting the information to the user.

The visual improvement of the resulting image can be observed in Figure 40, which shows a sample office environment with different materials present, including metallic objects with high reflectivity and dully lacquered materials with low reflectivity.

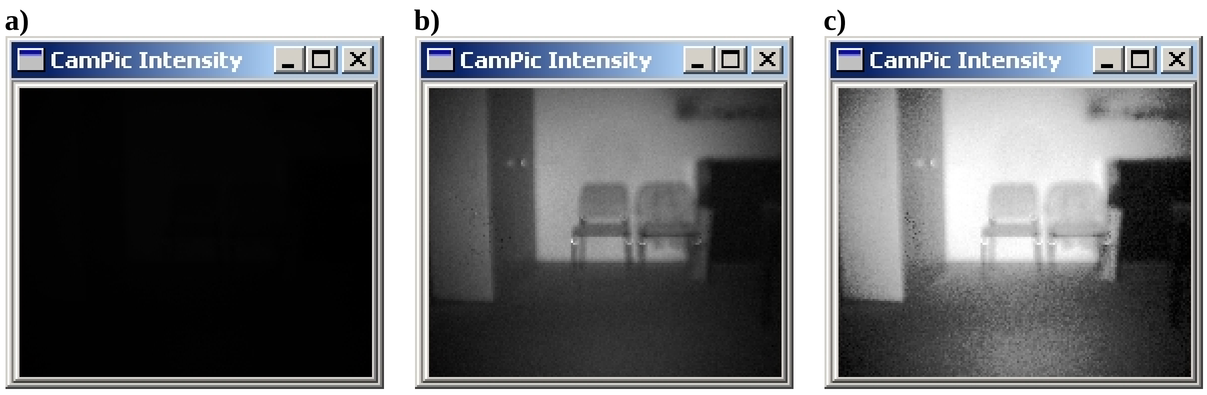


Figure 40: Reflection data, collected in a sample office environment a) unmodified, b) scaled greyscale values, c) histogram analysis

5.1.2 Converting distance to colour

As already mentioned, the range data is to be depicted using colours. In the depiction of heat images and stress images, it is common to use a colour-bar ranging from red to blue. This colour-bar, depicted in Figure 41, arises from the use of the RGB Colour space.

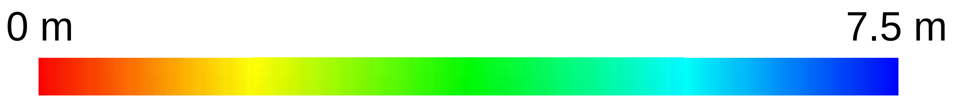


Figure 41: Converting 0 to 7.5m into colour

It assigns the given range values to colour values from red over green to blue. For the maximum distance of the used camera of 7.5 meters, the distribution would be as depicted in Figure 41. For lower ranges the colour-bar can be adjusted, similar to the stretching algorithm explained in chapter 5.1.2. The actual range has to be adjusted in the source code, no interface to the GUI was installed (see chapter 7.2).

The RGB Colour space consists of every colour a standard monitor can show by mixing red, green and blue. Figure 42 shows the two used sides of the cube and the chosen path that leads to the colour-bar. To make the differences in the used colour-bar as severe as possible, only the colours on the cube's edges are used.

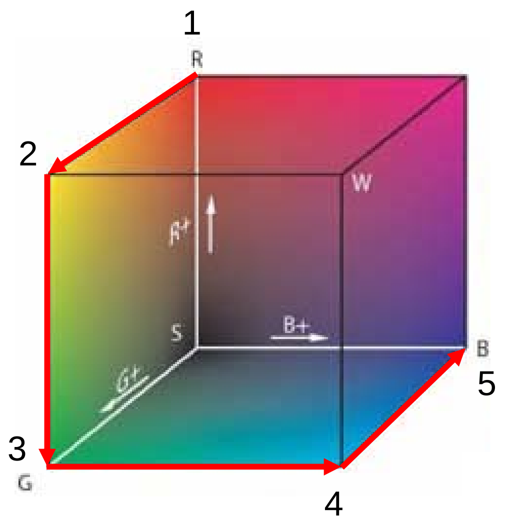


Figure 42: Modelling distance to colour in five steps along the edge of the RGB colour space RGB cube image taken from [http://en.wikipedia.org/wiki/RGB_color_space]

The algorithm starts at red (1) (RGB – 255, 0, 0), increases the value of green until it reaches yellow (2) (RGB – 255, 255, 0), reduces the red value until only green is left (3) (RGB – 0, 255, 0), increases the blue value till cyan (4) (RGB – 0, 255, 255) and finally decreases the green value until it stops at blue (5) (RGB – 0, 0, 255).

The function used in this project was programmed by Jörn von der Lippe for another project that is based on the same distance sensor [79].

5.2 Finding and identifying intrusions in the workspace

The path planning and proximity monitoring developed in this thesis require the knowledge of the position of the robot and all dynamic objects in the workspace. Those positions have to be acquired from the distance data, delivered by the camera. The steps that have to be performed are:

- Taking a distance measurement of the unoccupied workspace, the background data
- Filtering the currently acquired distance data for fluctuation noise, described in chapter 4.3.3.
- Correcting the filtered data by adding a virtual robot base
- Finding all intruding objects in the corrected data
- Identifying the robot in the objects found

While previous work [78] already showed a simple approach to separate known background data from data, generated by a dynamic object, it was also stated that the applied algorithms were too inaccurate for safety applications. Thus, in this work a new approach is described, which uses the median of a set of background images for the later comparison of background – foreground data. Since this causes a lot of fluctuation noise in the comparison, the result has to be filtered in further steps.

As described in chapter 4.3.4 the base of the robot is part of the background data and a virtual base needs to be added to the detected intrusions. Afterwards a region growing algorithm can create objects from connected pixels that show intrusions. From those regions the intrusions created by the robot can be identified with the help of the virtual robot base.

5.2.1 Acquiring the background data

As described in chapter 4.3.2 the data delivered by the camera have a certain error margin. The less the error in the background data, the better the detection of dynamic objects can be performed. The background data needs only to be taken once during the set-up phase. Thus to achieve a set of reliable data, the acquisition of the background is performed by taking several images of the unoccupied working space. Taking the median distance value in each pixel of those images, using the Kalman filter [36], reduces the error margin in the data. Tests showed that some surfaces like metal tend to produce larger errors, which would need a huge amount of data to compensate. Since most of the surfaces in the workplace of this project are flat, the remaining noise can be reduced by smoothing the data by applying a small Gaussian blur filter [62]. This filter reduces peaks in the background and also the amount of necessary sample data is reduced by this measure.

Each combination of pixel position, distance and surface leads to a specific error margin. To estimate the maximum error in each pixel, used in later algorithms, the smallest value of each pixel is tracked and recorded, in addition to the median background value.

Due to the similarities in the requirement on the background data, the acquisition of the background was developed in collaboration with another master thesis [81].

5.2.1.1 Kalman Filter

The Kalman filter [36] was chosen, because of its ability to work on consecutive values with taking their possible error into account. Its application, as described in this chapter, is commonly known in data processing. This property of the algorithm allows the processing of unlimited images without the need for a large amount of memory. In the course of this work it was sufficient to use a set of 500 images during background acquisition.

This filter works in iterations. In the first iteration the filter is set to the currently measured value and its estimated error. In each of the following iteration the filter uses its previous results and the currently measured value and estimated error to create a new result.

The estimated error results from the manufacturer information, which is specified to be 1% of the measured distance (see chapter 3.2).

To compute the currently estimated result x (see Equation 6) and its error σ_x (see Equation 7), using the Kalman filter, only the current sensor value z_2, its error σ_{z_2} and the previous estimation z_1 and its error σ_{z_1} are needed.

$$x = \left[\frac{\sigma_{z_2}{}^2}{\sigma_{z_1}{}^2 + \sigma_{z_2}{}^2}\right] * z_1 + \left[\frac{\sigma_{z_1}{}^2}{\sigma_{z_1}{}^2 + \sigma_{z_2}{}^2}\right] * z_2$$

$$\Leftrightarrow x = \left[\frac{\sigma_{z_1}{}^2}{\sigma_{z_1}{}^2 + \sigma_{z_2}{}^2}\right] * [z_2 - z_1]$$

Equation 6: Kalman filter: calculating the currently estimated value x according to [36]

$$\frac{1}{\sigma_x{}^2} = \frac{1}{\sigma_{z_1}{}^2} + \frac{1}{\sigma_{z_2}{}^2}$$

Equation 7: Kalman filter: calculating the currently estimated error σ_x according to [36]

In the following iteration the calculated x becomes z_1 and the error σ_x becomes σ_{z_1}.

Figure 43 shows a graphical explanation of the Kalman filter, starting with two values in a and adding an additional value to the result from a in b.

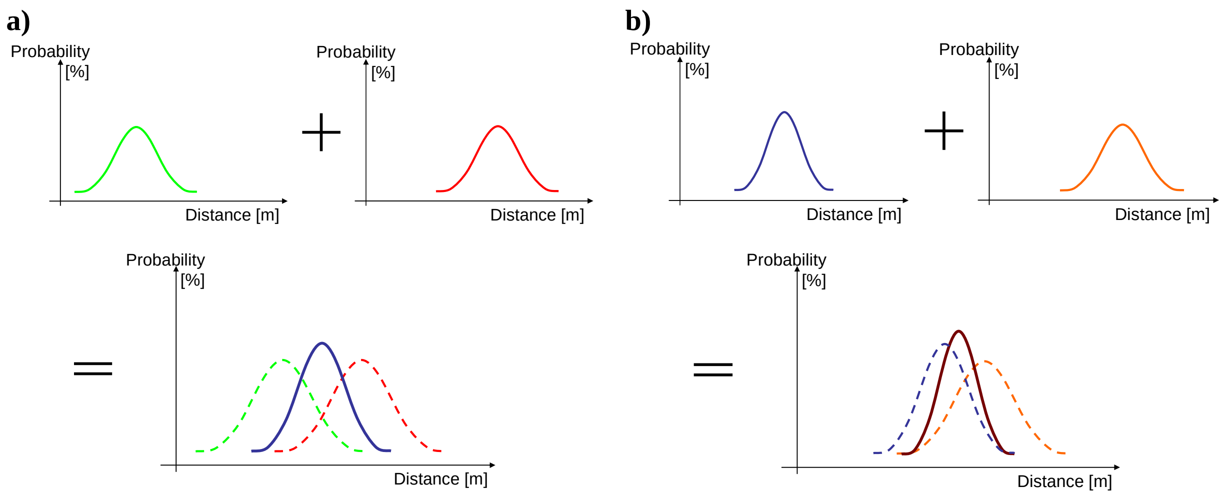


Figure 43: Application of a Kalman filter on consecutive sensor data – images created in collaboration with [81]

5.2.1.2 Gaussian blur

The Gaussian blur [62] is a filter, commonly used in image processing. Its application, as described in this chapter, is commonly known in image processing. It smoothes an image by combining the pixel information with the information in neighbouring pixels.

As described in chapter 5.2.1 some surfaces, especially metallic ones, cause a lot of noise in the camera’s sensor. Some of this noise still remains after combining the data of 500 images with the Kalman filter. Reducing this noise by increasing the number of images processed by the Kalman filter is possible, but the time it takes to set up the workplace would be lengthened. As an alternative solution a Gaussian blur filter was programmed in this project that works on a single image and therefore needs much less time.

In order to smoothen the background image sufficiently, a Gaussian blur using a large matrix would be necessary. The larger the matrix, the more detail in an image is lost. Thus, to save some time in the background acquisition and avoid too much information loss, a small Gaussian blur matrix was applied on the result of the Kalman filter.

To apply a Gaussian blur to an array of values, transforming it from its original state to its smoothened state (see Figure 44), a Gaussian matrix can be used (see Figure 45).

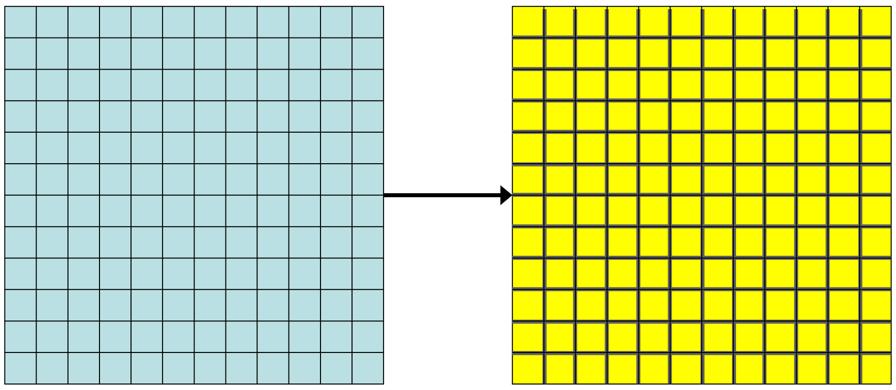


Figure 44: The value matrix has to be transformed from the original state (blue) to the smoothened state (yellow) – image created in collaboration with [81]

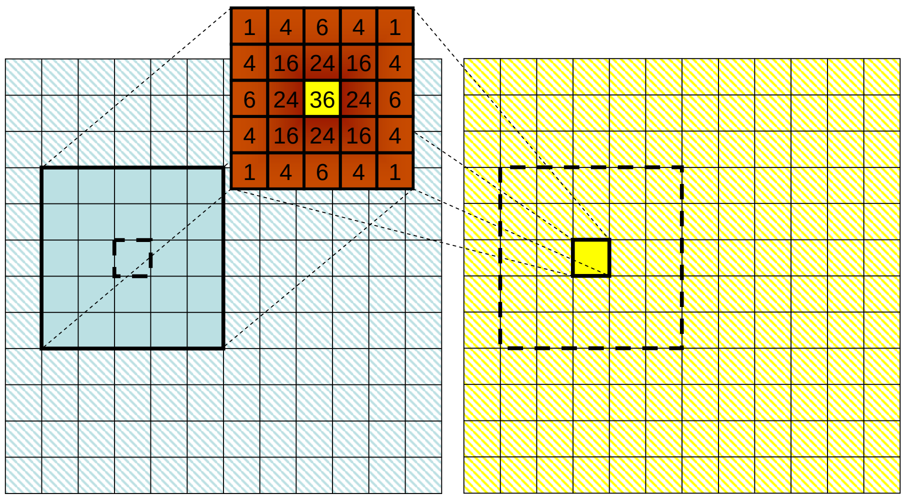


Figure 45: Applying a Gaussian matrix to get the result of one element – image created in collaboration with [81]

A Gaussian matrix is build by multiplying a Gaussian vector with its transpose (see Figure 46). Applying it to the original image means computing each pixel in the smoothened image from a section of the original image (blue area in Figure 45) that is centred on the desired pixel (yellow area in Figure 45) and has the same dimensions as the Gaussian matrix (Figure 46). Each element of this section is to be multiplied with its corresponding value in the Gaussian matrix, and the results are added up. The result of this addition has to be divided by the combined values of the Gaussian matrix. Therefore, in praxis, the values in a Gaussian matrix used in a computer program add up to one.

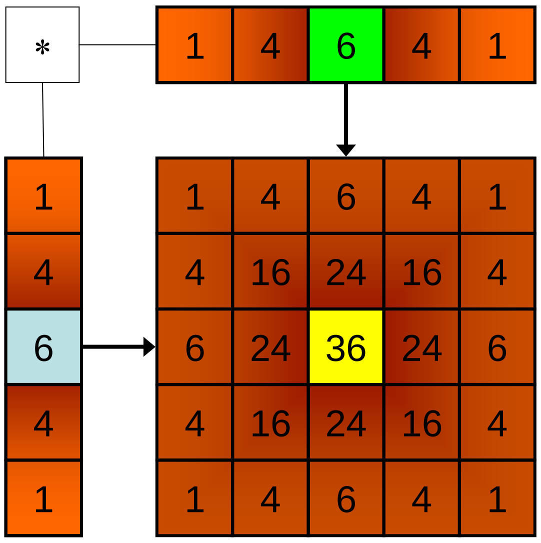


Figure 46: Gaussian matrix, build by multiplying a Gaussian vector with its transpose – image created in collaboration with [81]

The disadvantage of this approach is the high computational effort. A filter with the width and height of n uses n^n multiplications and $n^n - 1$ additions. With the 5 times 5 filter in the given example, each centre element needs 25 multiplications and 24 additions.

Due to the fact that a Gaussian matrix is build by multiplying a Gaussian vector with its transpose (see Figure 46), the computational effort can be reduced by applying each vector separately.

The optimized algorithm, applied in this project, uses the Gaussian vector, to convert the original matrix into a temporary matrix (see Figure 47a), which is then turned into the result matrix by applying the transpose of the Gaussian vector (see Figure 47b).

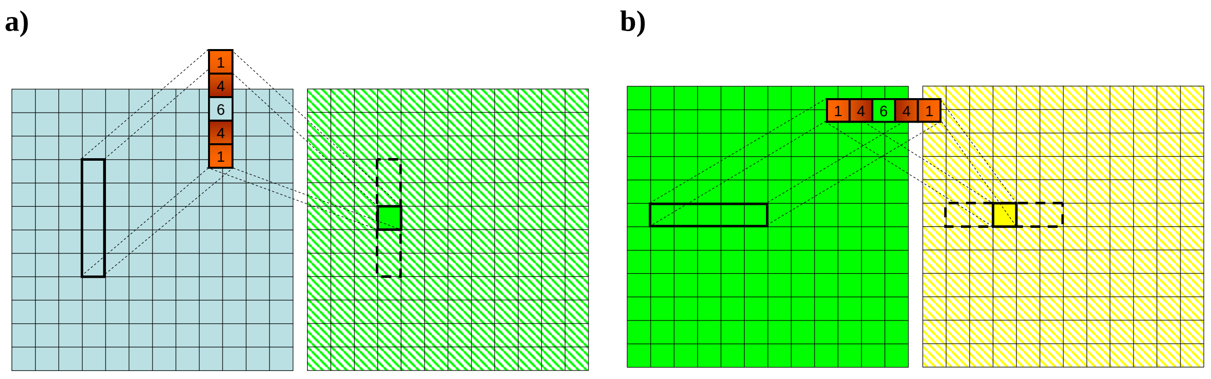


Figure 47: a) Converting the original matrix (blue) to a temporary matrix (green) b) and converting the temporary matrix into the result matrix (yellow) – images created in collaboration with [81]

Using this optimization, a filter with the width and height of n uses $n*2$ multiplications and $(n-1)*2$ additions. Thus the presented 5 times 5 filter needs only 10 multiplications and 8 additions per pixel.

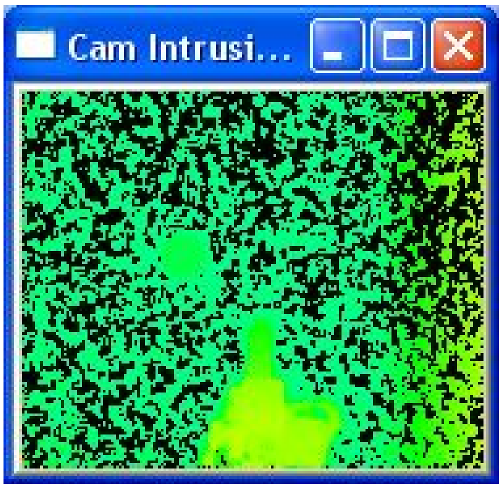
5.2.2 Finding intrusions and filtering

While the depiction of the images from chapter 5.1 is used in set-up and demonstration only, finding intrusions is the first step that has to be done in order to enable the robot to avoid dynamic objects.

To find intrusions in the workspace, the stored distance data of the background needs to be compared with the currently delivered distance data from the 3D camera. Every pixel whose distance is closer to the camera in the current image than its distance stored in the background might indicate an intruding object and is stored as a possible intrusion in a binary "intrusion map". Each pixel of the image is connected to one binary value of this intrusion map, indicating whether or not it is believed to contain an intrusion. As described in chapter 5.2.1 the distance measurements of a single frame of the camera contain fluctuation noise. While the fluctuation noise in the background image was sufficiently reduced, this reduction is not possible for the current distance measurement used in the search for intrusions. Thus the fluctuation noise is carried over into the intrusion map. This fluctuation noise, shown in the depicted intrusion map in Figure 48a, needs to be separated from the detected intrusions.

The intrusion images, used in this chapter to display the algorithms' progress, show the measured distance in each pixel that might be an intrusion. All other pixels are identified as background and remain black. Figure 48b shows the sample set-up, used in the intrusion images in this chapter. As an intrusion a round stool was chosen, since it remains stationary in consecutive measurements.

a)



b)

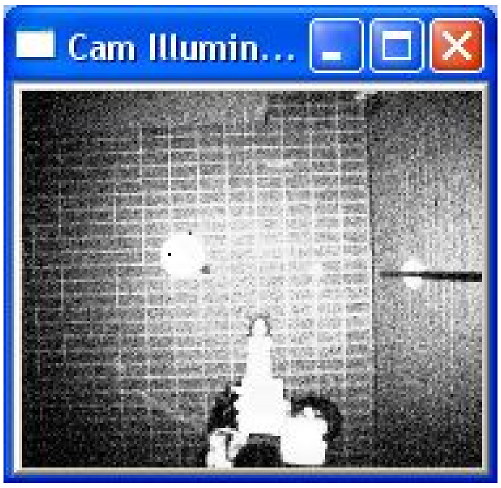


Figure 48: Fluctuation noise in background-foreground comparison as shown in Figure 24

For the necessary filtering assumptions have to be made:

- Every pixel that shows no intrusion is very likely to not contain an intrusion. There are few false negatives in the original intrusion map.
- No intruding object will occur below a certain offset. This offset is 30 cm for humans, according to EN 999 [69]. During this project the even lower offset of 12 cm, according to EN ISO 13857 [73] (see chapter 6.3) was applied to increase the achieved safety.
- Intruding objects generally cause intrusions above the stored minimal distance, measured in background acquisition (see chapter 5.2.1).

With those assumptions filter can be created. The second assumption can be used by an offset filter. The third assumption can either be dealt with by a spike filter, concentrating on all intrusions, or by a plausibility filter that only concentrates on intrusions next to free spaces. While both filter use the assumption that noise only needs to be found near free pixels, praxis showed that the plausibility filter causes much less false negatives. The placement of those filters in the program flow is described in chapter 4.3.3.

After the application of those filters, the remaining intrusions have to be dealt with by the region growing algorithm, as described in chapter 5.2.4.2.

It is also possible to use filters that are based on the history of the distances measured in the pixels. The Kalman filter, described in chapter 5.2.1.1, is an example for those filters. While those filters deliver good results in the reduction of fluctuation noise, they lower the reaction time of the system to such an extent that the system can no longer function properly. The reaction time of the system is the most critical part, as chapter 6.3 shows. Therefore this work refrained from the usage of such filters in its object avoidance part.

5.2.2.1 Offset filter

The first filter to be applied is the offset filter. It increases the distance measured in each pixel of the current image by a certain distance, causing values that are closely before the background to be moved behind it.

This filter was already implemented in previous work [78]. It contains the drawback, that it hides objects which are close to the background. Thus the value of the offset needs to be chosen carefully as described in chapter 6.3.

The false intrusions present in Figure 48 can be reduced by this filter to those shown in Figure 49.

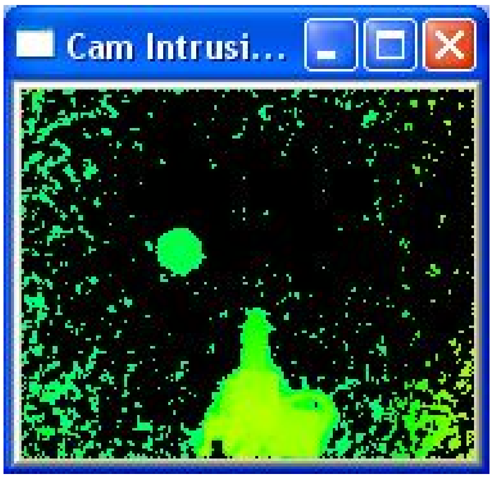


Figure 49: Fluctuation noise after the offset filter (12 cm) as shown in Figure 24

5.2.2.2 Spike filter

The remaining false positives have to be filtered out. One algorithm, which was developed in cooperation with another master thesis [81], the spike filter, is designed to deal with those false positive intrusions.

The spike filter checks every pixel for an intrusion. Once it finds an intrusion in a pixel, it looks at the neighbouring pixels and counts the intrusions in these pixels. The result of this count is compared with a threshold. If the counting of the neighbouring intrusions results in less or equal intrusions as the threshold value allows, the original intrusion is considered a “spike” and the original pixel’s intrusion is deleted.

If this filter’s resulting intrusion map would be stored in the same variable as the original intrusion map, this would cause a chain reaction. The deletion of one intrusion lowers the number of neighbouring intrusions for other pixels, which in turn would eventually be deleted. To prevent the algorithm from this chain reaction, the result of this algorithm has to be stored in a new variable.

Figure 50 shows the results of different thresholds (Figure 50b to d) of this filter on an intrusion map from Figure 50a. The dark green squares represent intrusions caused by dynamic objects, while the light green squares represent fluctuation noise. Since the intruding objects cause larger connected areas than the fluctuation noise, the noise is reduced more by the algorithm than the intruding objects. The higher the threshold, the less noise remains, but also the remaining information about intruding objects is reduced.

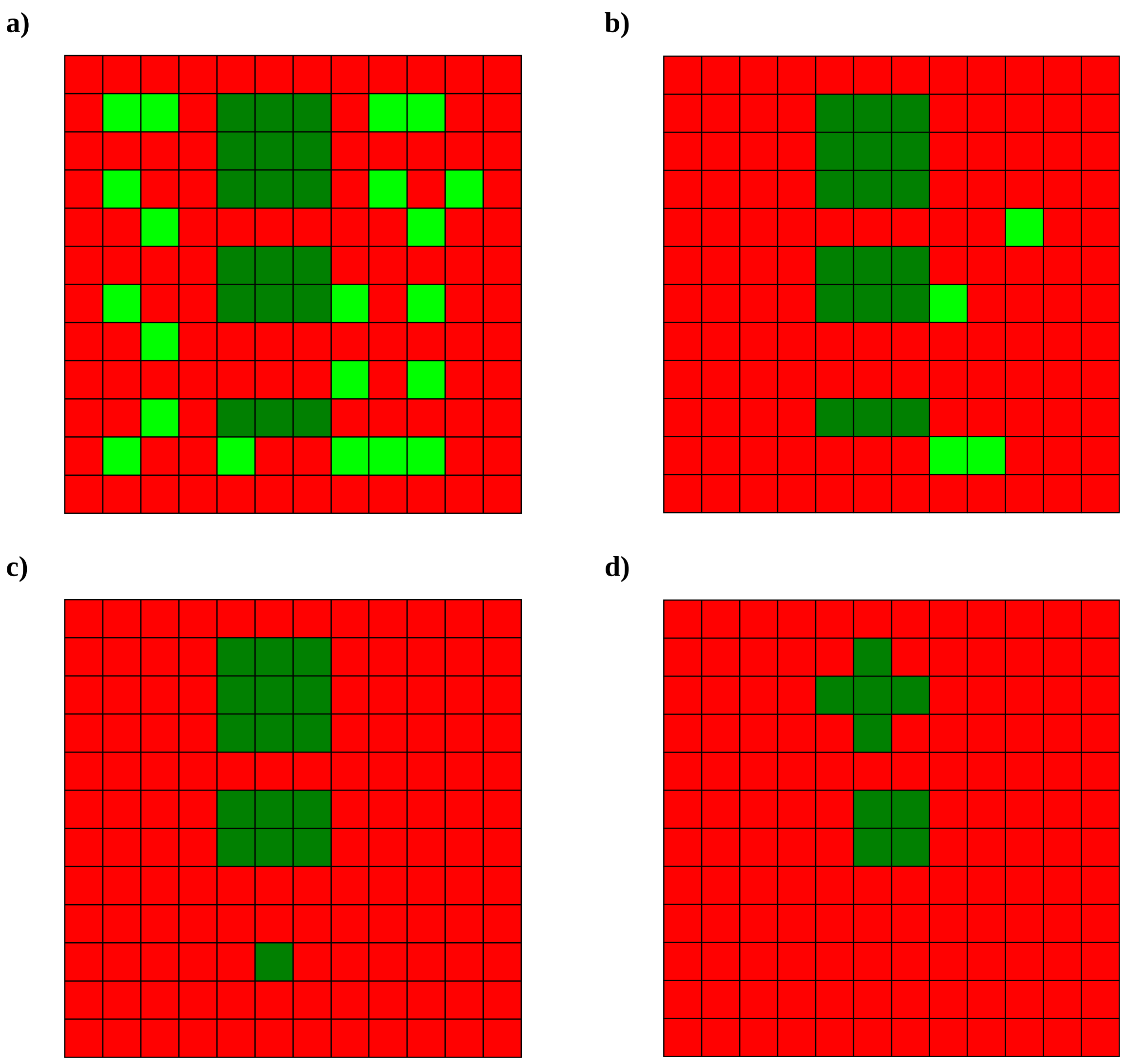


Figure 50: Intrusion map, red = no intrusion, dark green = intrusion, light green = noise
a) before filtering b) spike filter, threshold 1
c) spike filter, threshold 2 d) spike filter, threshold 3

Most intruding objects cover large connected areas, where the problem of this filter is the reduction of their outline. A severe problem remains with thin objects, such as fingers, which are easily deleted by this filter from the intrusion map.

In praxis the threshold needed to be set to four and higher, to remove the fluctuation noise sufficiently. This resulted in a loss in information that was too severe for this project, which is why it was not applied in the final project's program. Instead, a new filter (see chapter 5.2.2.3) was developed based on the insight of the implementation of this approach in this project.

5.2.2.3 Plausibility filter

The problems in information loss in the spike filter in chapter 5.2.2.2 needed to be avoided. A better solution, developed in this project, is the plausibility filter. It has a similar approach as the spike filter, but instead of looking at the intrusions, it looks at the free spaces for filtering.

The filter's algorithm searches for pixels that are considered free. It then looks at the neighbouring pixels, to find intrusions. If an intrusion is found, it is compared to the minimum value in that pixel that was measured during the background acquisition (see chapter 5.2.1). If the value is greater than that minimum value, the pixel is considered as fluctuation noise and set to free space, since that value is likely to be caused by the background.

The distance around free pixels, in which the algorithm searches for fluctuation noise can be configured.

As Figure 51 shows, the search-area around free pixels, in which occupied pixels can be found and eliminated, can be larger than the one of the spike filter, which only searches in neighbouring pixels. The larger the search-area, the more pixels are checked that belong to intruding objects. This effect is compensated by the fact that this filter keeps values that are definite positives and eliminates only those of which are behind the minimal background distance. Although this filter may cause reduction in the size of objects, this only occurs with objects that are close to the background. The reduction must still be considered in the safety approval of the system (see chapter 6.3). This leads to far better results than the spike filter.

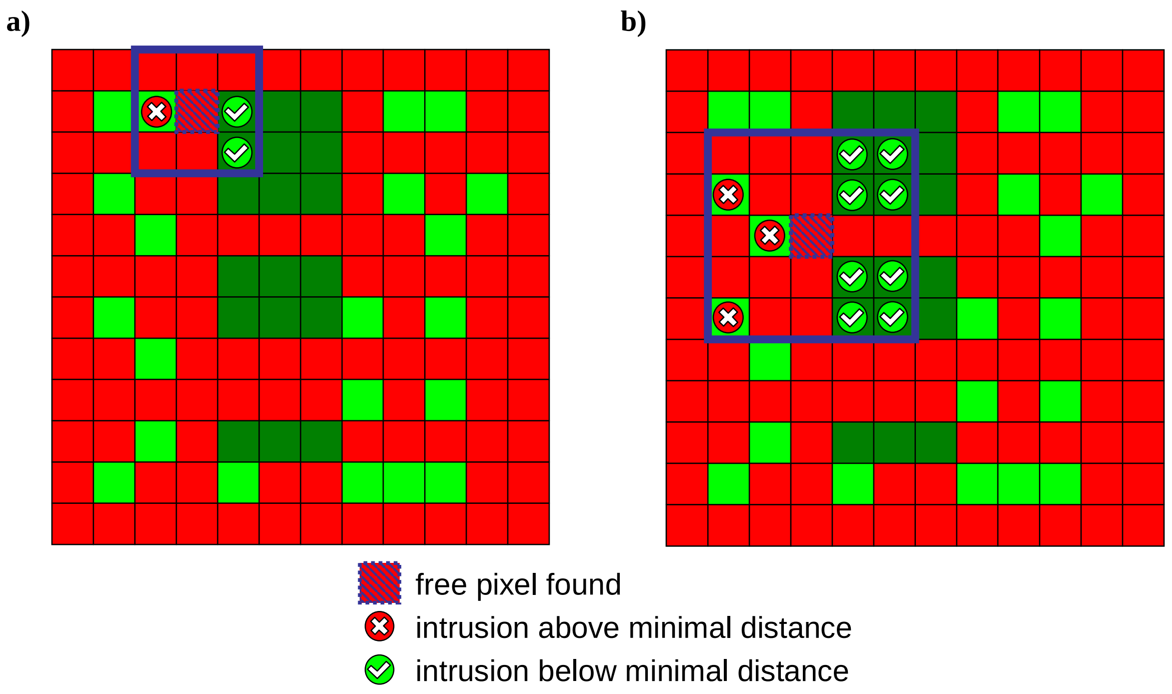

a)
b)
free pixel found
intrusion above minimal distance
intrusion below minimal distance

Figure 51: a) probability filter size 1, b) probability filter size 2
Intrusion map, red = no intrusion, dark green = intrusion, light green = noise

In this work only the smallest size of this filter, the search in neighbouring pixels, needed to be used, to attain sufficient results. The intrusion map, after application of this filter, is shown in Figure 52.

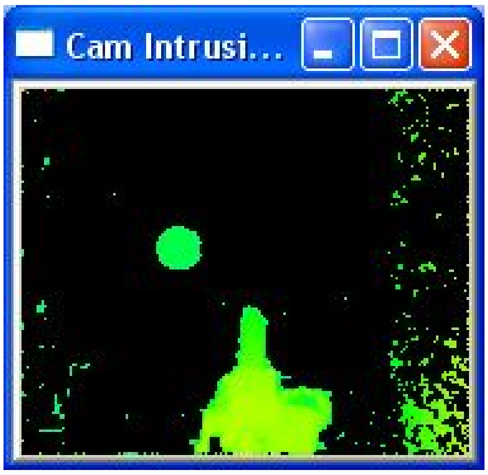


Figure 52: Fluctuation noise after the offset filter (12 cm) and plausibility filter as shown in Figure 24

5.2.3 Adding a virtual robot base

As described in chapter 4.3.4 the base of the robot is not detected as an intrusion, because it can not be removed during the background acquisition. Thus a virtual base has to be created. It is added to every intrusion map after filtering, because otherwise the filter would remove it as noise.

Since the robot base is a near perfect circle the virtual base can be simulated by a circle. It is entered by the user by its centre point and its radius (see chapter 4.3.4) in the GUI.

To be able to quickly add the base to the intrusion map in every loop, the virtual robot base is computed from the user's values after their entry and stored in a binary two dimensional array (see Figure 53a). Since the base is to be a circle and the array a rectangle, each value in the array has to indicate whether or not it is part of the base (inside or outside the circle).

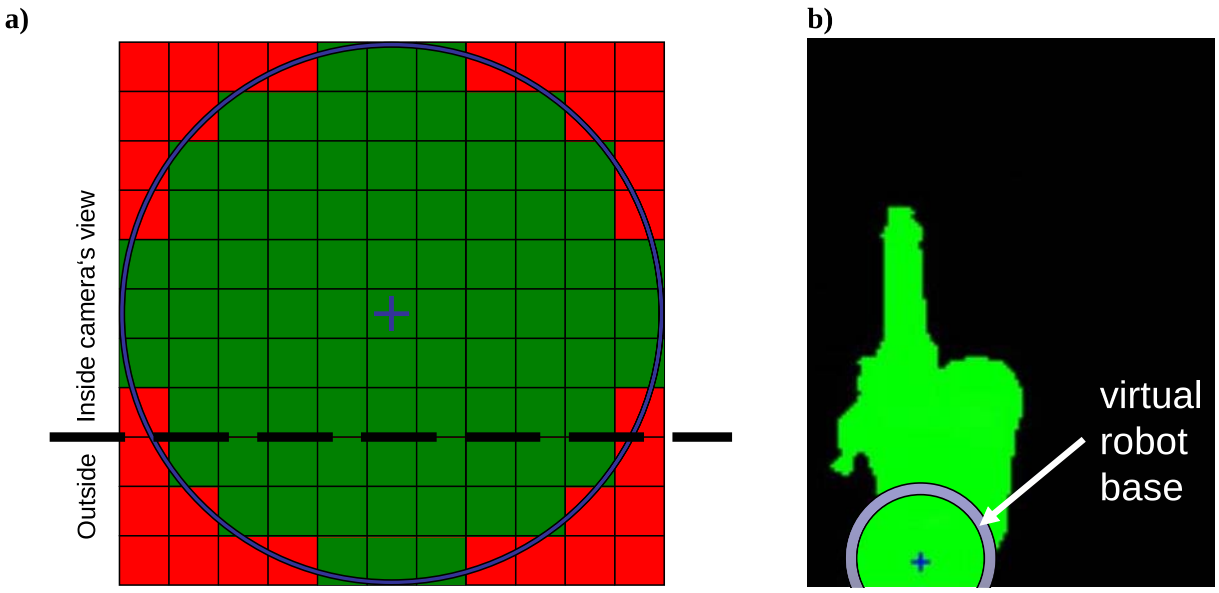


Figure 53: a) Creating the virtual robot base, blue circle: robot base radius, blue cross: centre of robot base, red square: not robot base, green square: robot base
b) Virtual base, added in the program (see Figure 26)

To compute the array, the distance of each pixel's centre to the centre of the robot's base is computed, using Pythagoras. If it is equal or less than the entered robot base radius it is set as part of the base.

If the virtual base's size expands partly beyond the camera's view, as given by the camera's position in this project (see Figure 53b and chapter 3.3), it is cut at the end of the camera's view. This programming step prevents problems when the virtual robot base and the intrusion map are merged, since every part of the virtual robot base has a corresponding part in the intrusion map.

5.2.4 Region growing

As described in chapter 4.3.4 it is necessary for the path planning and proximity monitoring to identify the robot and the intruding objects in the filtered intrusion map. In targeting the robot's base (see chapter 5.2.3), the first step towards identifying the robot has been done.

After filtering there are three kinds of intrusions.

- Intrusions caused by the robot
    - from its virtual base
    - from its arm
- Intrusions caused by other dynamic objects
- Intrusions caused by remaining fluctuation noise

In order to be able to differentiate between these different kinds of intrusion in the intrusion map, at first all single pixels showing intrusions are grouped into connected regions (region growing).

In the next step, the robot can be identified from these regions by the known position of its base.

As Figure 52 shows, all other regions are caused by other dynamic objects and fluctuation noise, which has to be filtered out. With the assumption that the remaining noise causes only small connected regions, it can thus be identified and filtered out by a region size filter (see chapter 5.2.4.2).

5.2.4.1 Region growing by outlining and filling

The standard approach on region growing is to take a random pixel in the map and to keep adding neighbouring pixels that fit a certain criteria, in this case being marked as intrusion, to the region, until no new neighbouring pixels remain [63].

Due to the filtering process, objects close to the ground may contain holes in their regions. To estimate the size of the regions – and therefore the size of the objects – as good as possible, these holes need to be filled. Since the robot is not allowed to work above or below intruding objects (see chapter 2.3) real holes that are filled within this region make no difference to the mobility of the robot. The mobility is not reduced by this assumption.

The need for filling the holes therefore allows two possible approaches.

- Growing the region and filling its holes

- Outlining the region and filling the complete region

Since filling the holes in a grown region takes as much time as filling a region that has only been outlined, whereas outlining a region tends to be faster than growing the region, this work deviates from the common approach on region growing by using an outlining and filling algorithm.

To explain the developed outlining and filling algorithm, the region shown in Figure 54 is used.

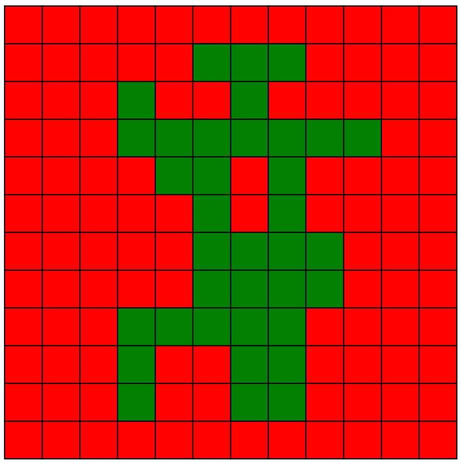


Figure 54: Sample region

Figure 55 shows the outlining algorithm step by step. In Figure 55a the first pixel of the region is found by searching through the pixels of each row. This pixel is remembered as the starting pixel.

To find the next outline-pixel in Figure 55b, the last direction, in which the current outline pixel was found, is reversed and turned 45° in clockwise direction. The pixel that is aimed at is searched for an intrusion. Two results are possible:

- If an intrusion is found in this pixel, the algorithm adds that pixel to the outline and starts at its position from the beginning.
- If there is no intrusion in that direction (the light blue arrow 1 to arrow 3), the direction is shifted clockwise, until an intrusion is found (dark blue arrow 4). The first intrusion found by this search is set as the next outline pixel. From this the search continues (Figure 55c to e). The shift in the search-direction continues, until the reversed last direction has been searched.

Searching the reversed last direction has two possibilities:

- An intrusion is found
- No intrusion is found

Since pixels already included in the outline are not excluded from the search, the reversed last direction leads to the pixel, where the outline came from.

- If an intrusion is found at that position, this allows the outlining algorithm to backtrack and thus walk along threads of pixels that only have a width of one pixel.

- If no intrusion is found at that position, the found pixel was a single pixel and the outlining is finished at this point.

The outlining algorithm is completed in Figure 55f, when the next pixel to be set as the outline is the starting pixel.

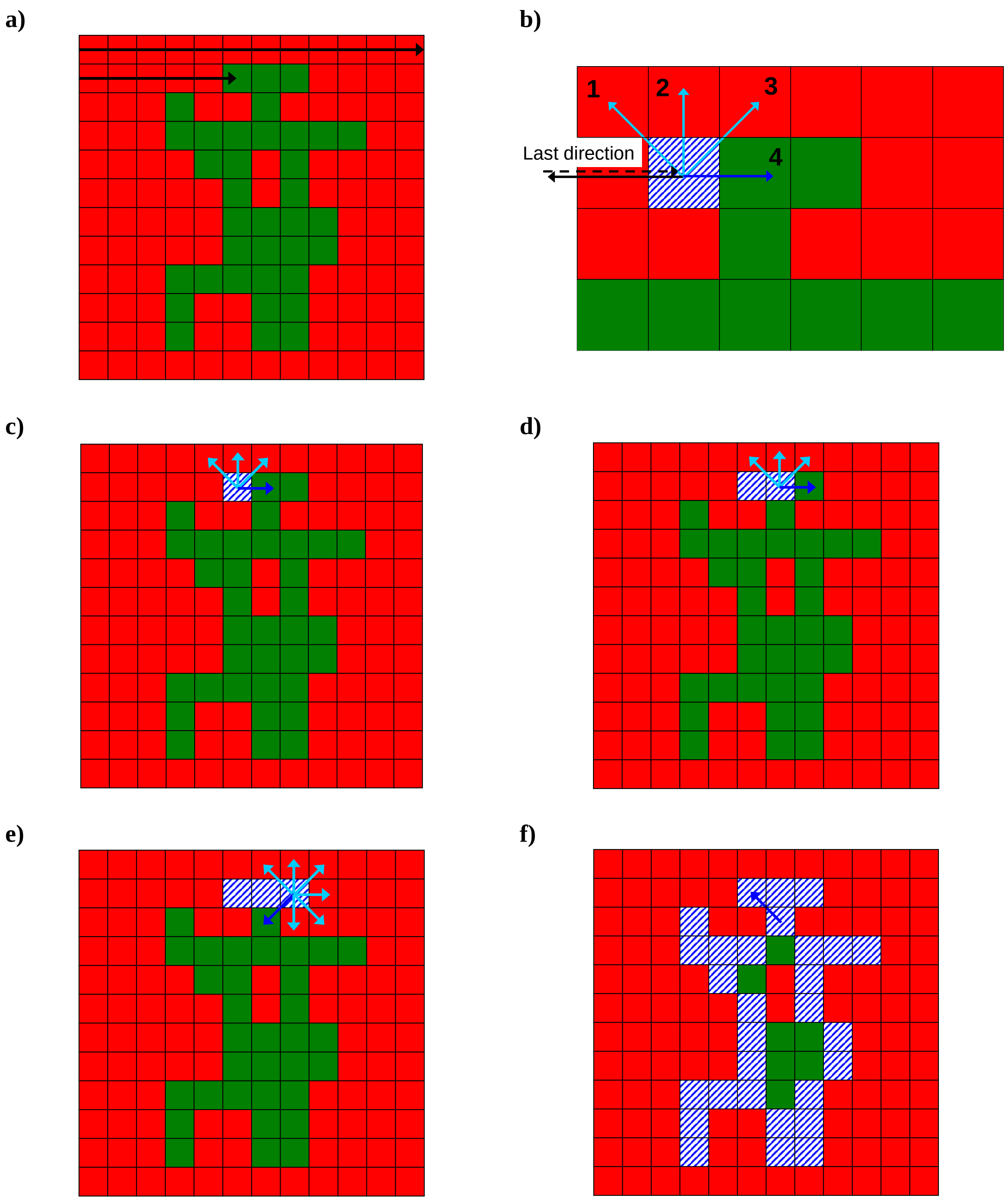


Figure 55: Region outlining algorithm
a) finding the first pixel
b) direction of search
c) to e) finding the next pixel
f) finished outline

After the outline algorithm is finished, the outlined region has to be filled. This is achieved by the algorithm depicted in Figure 56.

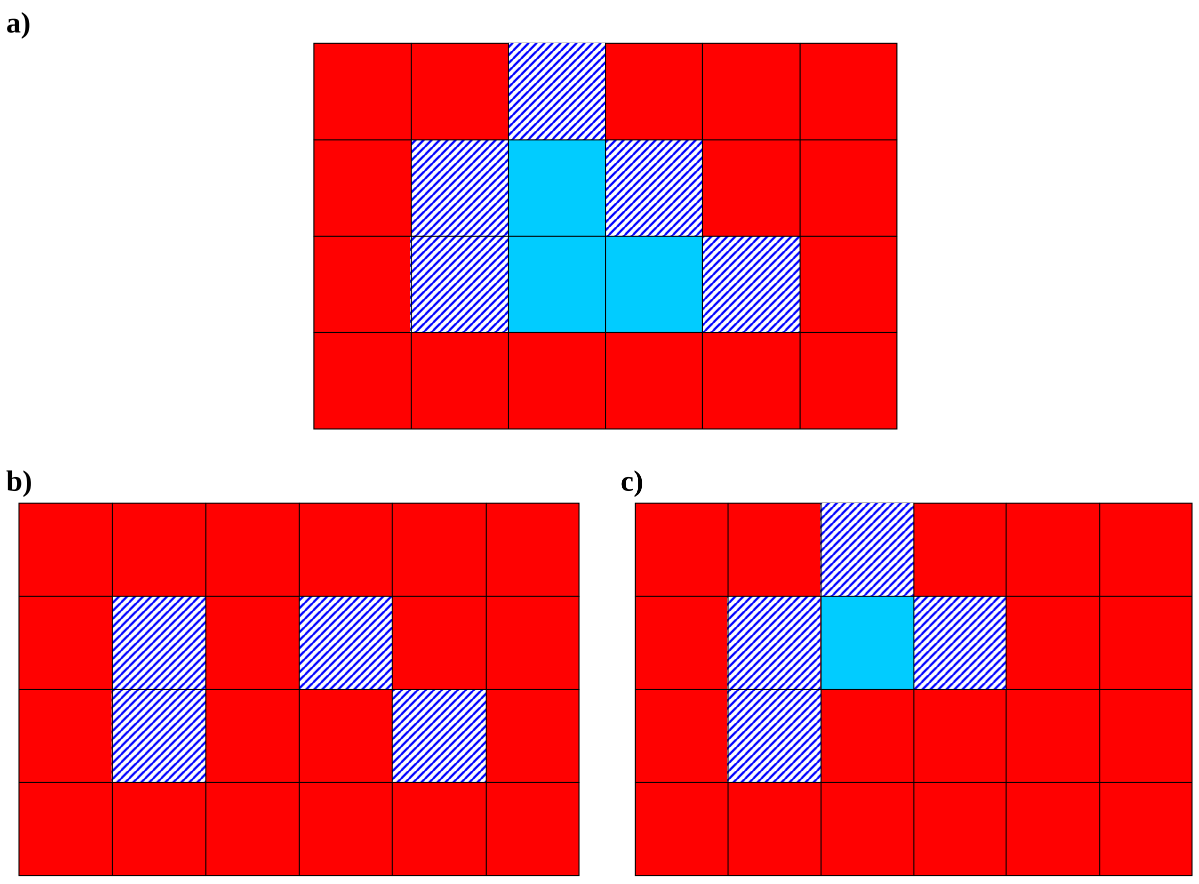


Figure 56: Filling principle

The algorithm starts, working on a temporary array, by filling every pixel that is between two outline pixels of one row and has outlined or filled pixels in the row above. In Figure 56a the light blue pixels would be filled, because they have a blue-white outline on the left and right and outlined or filled pixels above.

The pixels would not be filled, if not every pixel above was outlined or filled (see Figure 56b) or if there was no outline to the left and right (see Figure 56c).

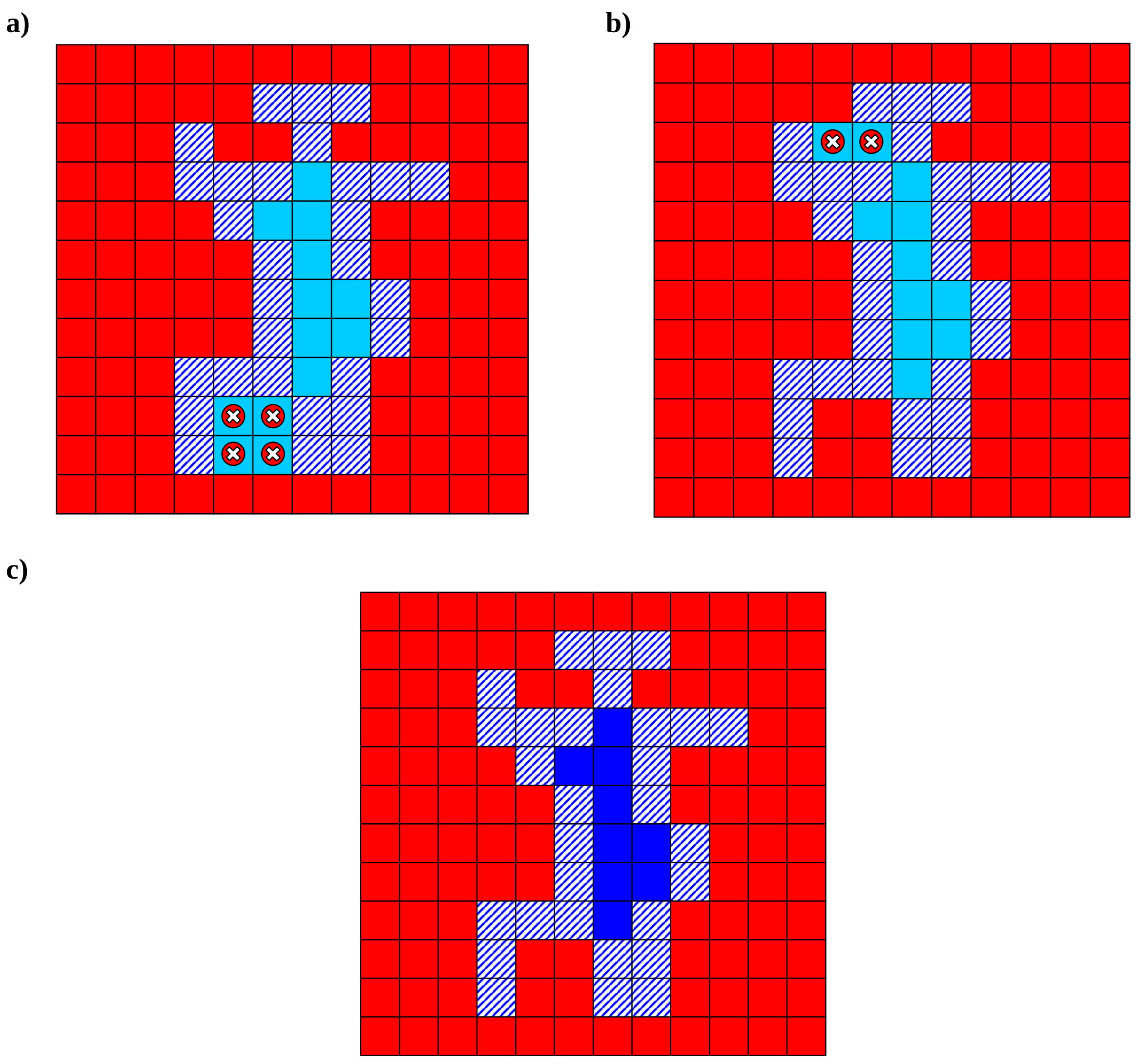


Figure 57: Filling from above and below

When this part of the algorithm is applied to the outlined region from Figure 55 the result is the region in Figure 57a. In this region, more pixels are filled than belong to the desired region (marked by a ⊗). To compensate this, the filling algorithm is reversed, looking for the connected line below the pixels and working bottom to top through the same outlined region as the first part of the algorithm. This results in the Figure 57b, again with more pixels than belong.

The combination of Figure 57a and b, in which only the light blue pixels are kept that are set in both regions is the desired region (see Figure 57e).

Once the desired region has been found and stored, its pixels are removed from the intrusion map and the next region can be searched, until all intrusions in the intrusion map have been allocated to a region.

5.2.4.2 Region size filter

As described in chapter 5.2.2.3, some fluctuation noise remains after filtering. The noise connects to regions that are smaller than intruding dynamic objects, especially humans. Thus, to eliminate this noise, the size of a region is counted while it is being filling (see chapter 5.2.4.1). If this size is below a certain threshold, the region is dropped from further processing. Figure 58 shows the objects of the sample workspace set-up that remain after size filtering.

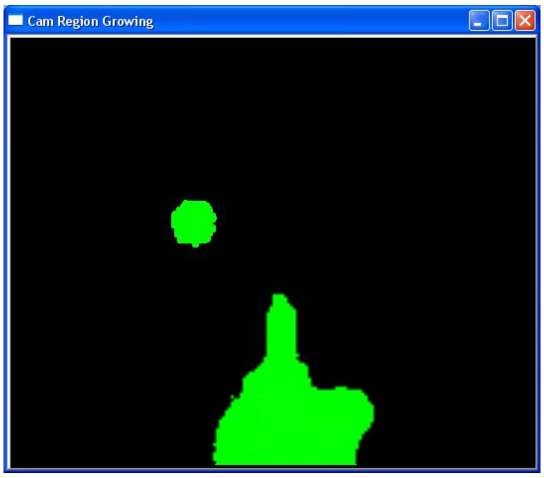


Figure 58: Remaining objects after region size filter

Objects that only partially enter the camera's view from the outside cause only small regions. For example this is the case, if only the hand of a human enters the visual field. If the number of pixels in those regions is smaller than the threshold, they are eliminated as well. Therefore the application of this filter is only possible in a workplace, where the camera's view on the robot is larger than the robot's possible reach (see Evaluation of safety in chapter 6.3).

5.2.4.3 Distinguishing the robot from objects

To distinguish the robot from other dynamic objects in the environment, the robot's base was targeted manually in the intrusion map (see chapter 3.3).

Every region that is extracted by the region growing algorithm (see chapter 5.2.4.1) is cross referenced with the base's centre position. If this position is in the given region, the region is set as the robot's region, as shown in Figure 59.

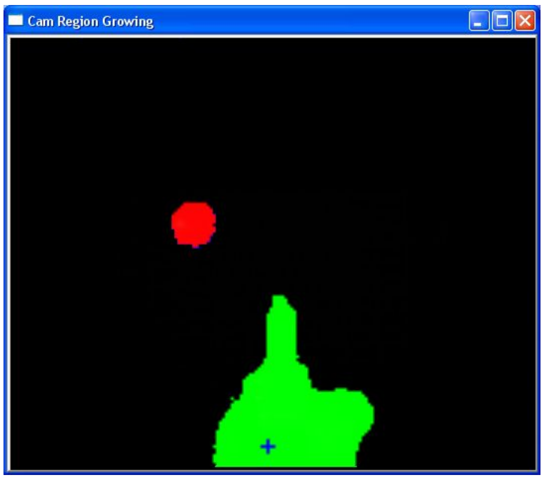


Figure 59: Robot region identified by its base's centre (blue cross)

Since at this point the fluctuation noise and the robot have been identified, all other intrusions have to be dynamic objects.

One main advantage of using the grown region rather than the position given by the robot's PLC itself, is that the grown regions also automatically include tools and workpieces the robot is wielding.

5.3 Detecting collisions

In the dynamic workplace, collisions between the robot and other dynamic objects are possible. This can happen due to several facts:

- The robot is approached by dynamic objects while standing still (stopped by the proximity monitoring).
- The dynamic object is approaching the robot faster than the robot can avoid it, due to its reduced speed.
- The robot has no more space for avoiding motions.

If a collision between the robot and a dynamic object occurs their corresponding regions would merge, and the algorithm distinguishing the robot from other objects would include the object into the robot. This happens, since the robot is detected as the one region that includes the robot's base centre (see chapter 5.2.4.3). The dynamic object would then be no longer considered an obstacle for the robot's movement and the program would plan the robot's movement accordingly, leading to possible injuries / damages to the intruding object.

Figure 60a and b depicts the described case, where one of the dynamic objects from Figure 60a merges due to its movement with the robot in Figure 60b.

To avoid this problem, the current robot region is checked against the intruding object regions, detected in the previous frame. If the robot region and the last frame's object regions overlap, a collision took place (Figure 60c).

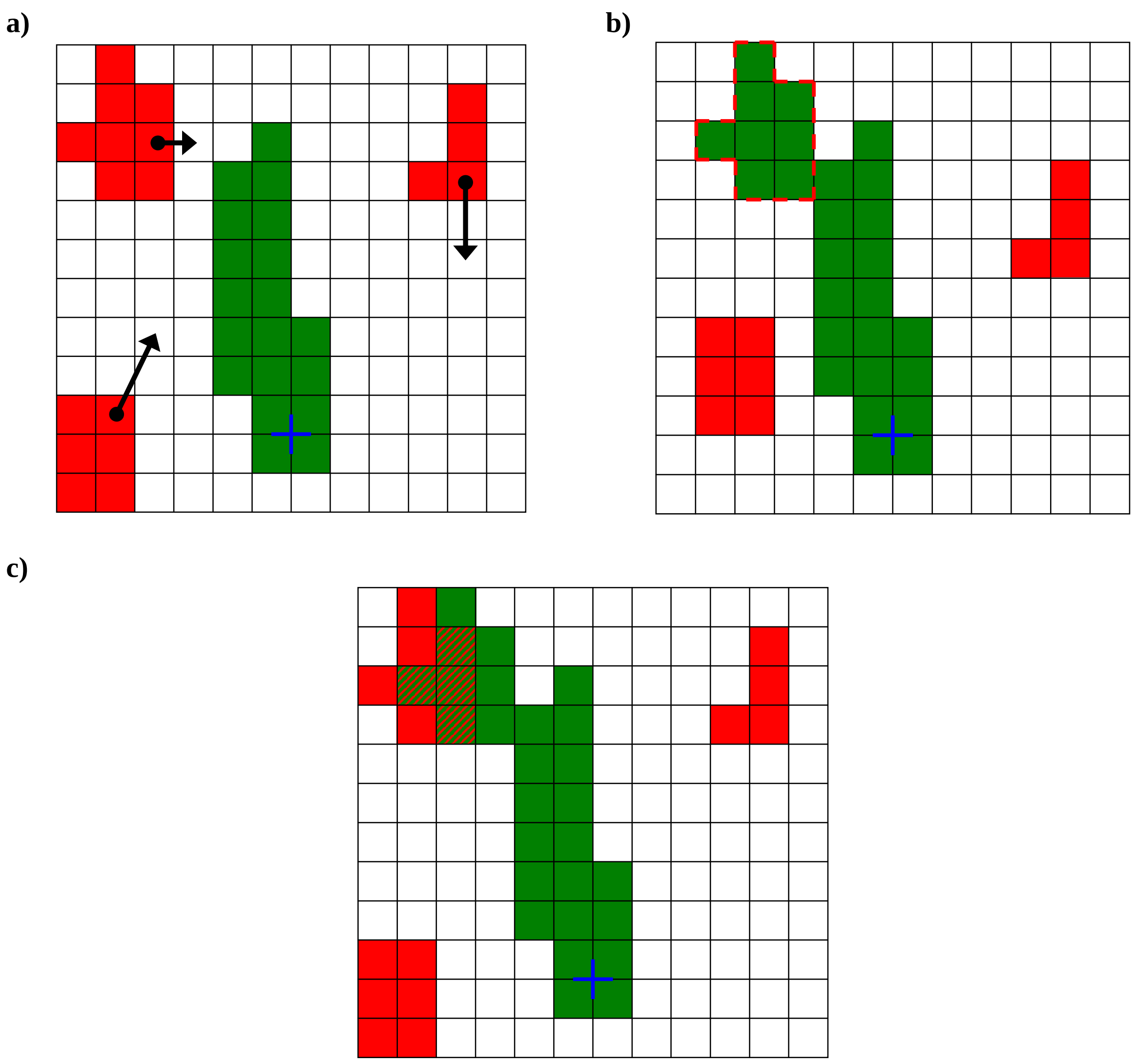


Figure 60: Detecting collisions by region history

If a collision is detected, the robot has to be stopped by the program. The clearance, to continue with the working process, can only be manually approved by the responsible operator, as described in chapter 4.4.3.

5.4 Robot speed control

As described in chapter 2.3.1, the monitoring of the approach from dynamic objects to the robot is vital for the safety of the whole system, especially for the human operator (see chapter 6.1). Since the time the program is allowed to react to dangerous situations shrinks with the distance of robot and objects, the speed of the robot also needs to be reduced, the closer objects get to the robot.

To control the robot's speed depending on its distance to objects, this distance has to be known. In this project it was calculated from the images delivered by the camera. Another approach would be to use the data from the robot's internal position sensors for the robot's position and the data from the camera for the position of the intruding objects only. This approach has the advantage that the robot's position can be updated more frequently, but the resulting algorithm is more different to verify. Since the algorithm containing the distance measurement is critical for the safety of the whole system, it should be as simple as possible, in order to be able to be verified. Another aspect is its stability, which would decrease with increasing complexity.

To calculate the distance between the robot and the other acquired regions, the size of the robot's region is enlarged, until it touches an intruding object in one point. Figure 61 shows a screenshot of this algorithm's result.

a) **b)**

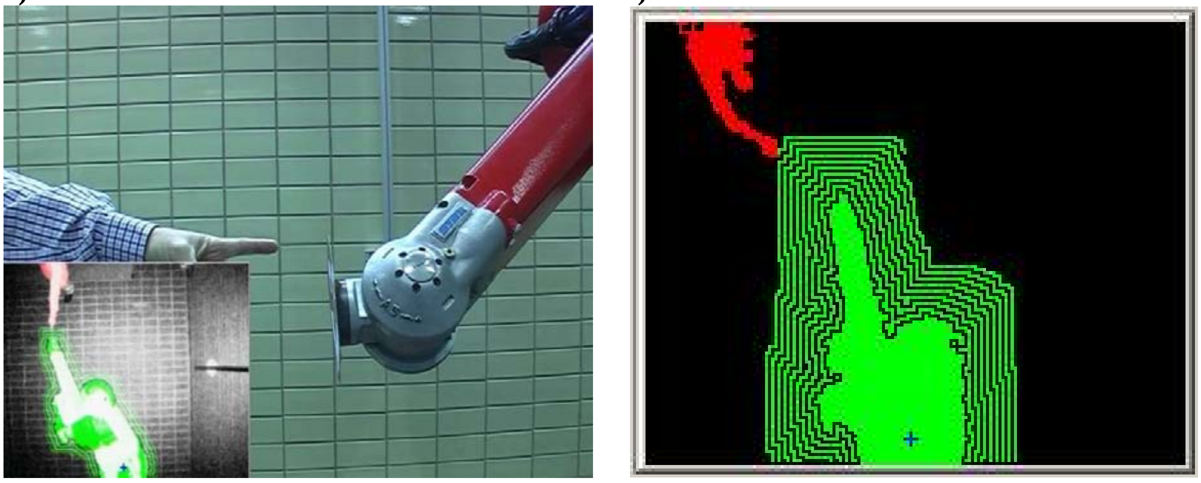


Figure 61: Robot region (green) increased until it touches the (red) intruding person
a) scene from the video on CD, picture shows live scene
picture in picture shows region enlargement overlapped with intensity values, shown on PC
b) enlarged robot and object only, as observed by the 3D camera and shown on the PC

Since the controlling program is to be used in a demonstrative workplace, the algorithm was also to be designed to show its functionality to the observer. Thus the algorithm was implemented in such a way that allows a depiction of the enlargement of the robot's region. By forming rings around the detected robot region the distance is easily visible to the observer (see Figure 61b).

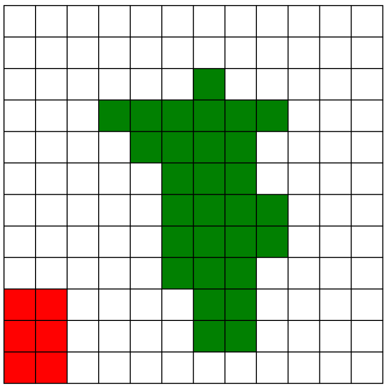
The single steps of the implemented algorithm are depicted in Figure 62. Figure 62a shows a region map, with a dark green robot region and a red intruding object region. The robot region is enlarged stepwise form Figure 62b to e, until it touches the nearest object in Figure 62e.

To enlarge the robot, each pixel in the array containing the robot region is checked for its intrusion setting. If a pixel indicates an intrusion, all surrounding pixels that contain no intrusion are marked as intruding regions as well. Since this would cause a chain reaction (similar to the spike filter in 5.2.2.2), in this case leading to the filling of the complete array in one step, an intermediary step is necessary.

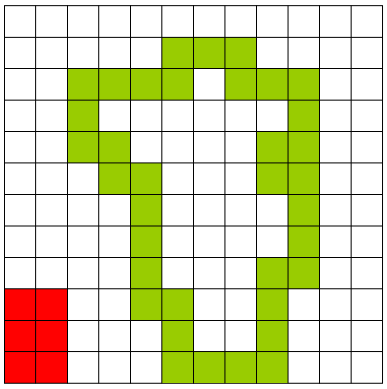
Thus, to enlarge the region stepwise, a secondary array is used, in which the results from the first array's search are stored (Figure 62b and d).

Every time a new intrusion is entered into an array, the array containing the dynamic objects is checked, if it contains an object at that position. In that case the algorithm is stopped. If no dynamic object was found during one enlargement step, the arrays are switched. That means in the second step, the intermediary step's array is increased and written into the original robot array. This continues, until a dynamic object is hit by one pixel of an enlarged region (Figure 62e), or all edges of the image are reached.

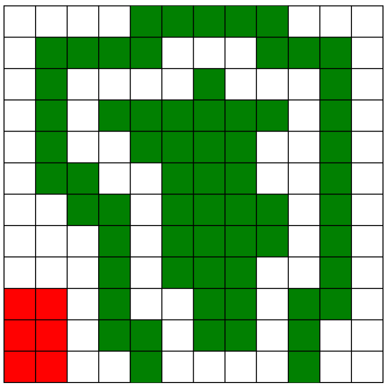
a) original



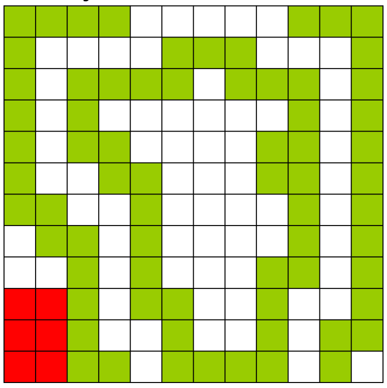
b) intermediary



c) original



d) intermediary



d) original

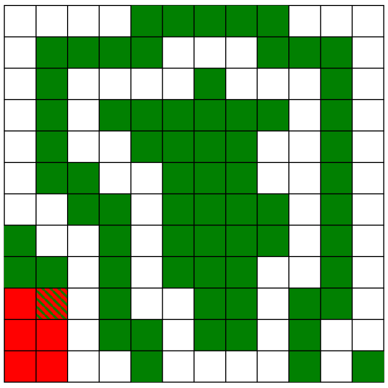


Figure 62: Region growing algorithm
dark green: robot's original region
light green: temporary region
red: intruding object

During the execution of the algorithm each loop is counted and the result is stored as the measured distance in pixels. This result is used to change the robot's current speed (see chapter 4.4). Since the size of one pixel varies only slightly at the working distances in the given demonstrator, no transformation from pixel to real distance was programmed. This feature can be added in future work (see chapter 7.2).

For the demonstration of this algorithm, impacting the robot’s current speed, the speed of the robot was reduced stepwise from 100% at 15 to 0% at 5 pixels distance. These values have to be adapted in future work, dependant on the respective safety analysis (see chapter 6.3).

5.5 Robot path planning

Building a collaborative workplace, a robot working together with humans, the basic solution is the described speed control from proximity measurement (see chapter 5.4). This basic solution implements the required safety, but it induces the problem that the robot often needs to wait for the humans to clear the working path before it can continue with its task. As described in chapter 2.3.1, planning a path for the robot around intruding objects in the workplace can reduce the time the robot is waiting. This optimization in collaboration is the main goal of this thesis.

The path planning algorithm needs to control the robot’s complete process. A robot’s process consists of moving towards a desired goal in a certain way (direct drive, point to point) – the control of the trajectory – and in controlling the robot’s tools. Since the given robot was not equipped with any tools, the control of any tool was neglected. Adding a tool to the robot’s arm would change the physical boundaries of the robot. In this case, not only the tool handling would have to be implemented in the controlling program, but the change of the robot’s physique would have to be considered, when calculating the necessary safety distances during path planning (see chapter 6.3).

To plan a path for the robot around intruding objects, the following points have to be considered:

- The robot’s current position and its relation to the next goal position have to be known, in order to allow path control (see chapter 5.5.1).
- The camera’s coordinate frame and the robot’s coordinate frame have to be determined and their relation has to be calculated. This is necessary to compare the robot’s position and its destination, which are both given in the robot’s coordinate frame, with those of the objects, which are given in the camera’s coordinate frame (see chapter 5.5.2).
- The reachable space, the space the robot is allowed to reach, has to be determined (see chapter 5.5.3) in order to allow the calculation of an evasion path (see chapter 5.5.5), if intruding objects are present in the path.

Contrary to the speed control in chapter 5.4, where the information about the object’s and the robot’s position was solely acquired from the camera, the algorithms in this part rely on the robot’s internal sensors, to acquire its current position. While the speed control requires the actual shape of the robot, which is only delivered by the camera, the path planning can work on assumptions of the shape, which are computed from the robot’s position. The position acquisition from the robot’s internal sensors, apart from being more precise than the camera’s data, has the advantage of being considerably faster.

5.5.1 Calculating the distance between the current pose and the goal pose

In order to be able to implement a simple path control, controlling the robot's movement from a start point through several sub-goal points to an end point, the program needs to be able to compare the robot's current position with that of its next goal. This allows the program to determine whether a goal has been sufficiently reached and the next goal is to be activated (see chapter 4.4.1).

This comparison was achieved by building a vector from both positions and calculating the length of the vector. Equation 8 shows the calculation used to calculate the distance from the robot's current position to the given goal position.

$$|\vec{v}| = \sqrt{\left(x_{current} - x_{goal}\right)^2 + \left(y_{current} - y_{goal}\right)^2 + \left(z_{current} - z_{goal}\right)^2}$$

Equation 8: Length of a vector;
the three dimensional distance of two points [61]

If the distance between the current position and the goal position is below a certain threshold, the next goal position is activated in the program.

5.5.2 Transformation of coordinate systems

Two coordinate systems exist in the given workplace, the robot's coordinate system and the camera's coordinate system. One common coordinate system is needed for the path planning algorithms. Therefore both systems have to be merged. The easiest way to accomplish this is the transformation of one system into the other. For this thesis the robot's coordinate system has been chosen as the basic system, because the robot has to be controlled, whereas the objects in the camera's view only have to be observed. Using the robot's coordinate system results in fewer transformations of coordinates and thus in less computational effort.

To transform the coordinates from the camera's coordinate system to the robot's coordinate system, the shift in the origin and the rotation of the systems towards each other have to be determined.

The transformation of the single coordinates from one system into the other takes place by multiplication of the position vectors with a homogenous transformation matrix (see Equation 9).

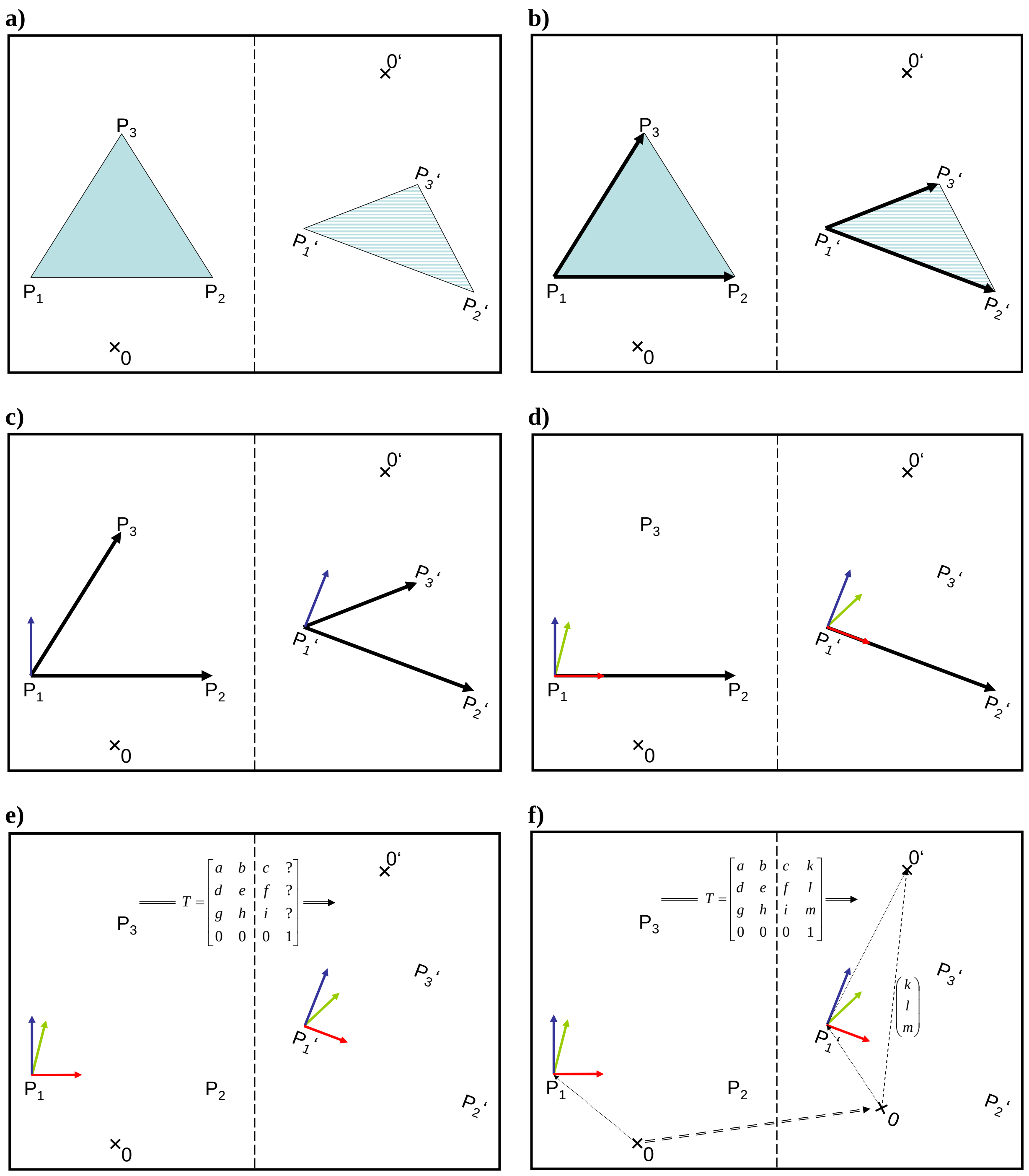
$$M_{4x4} = \begin{bmatrix} \begin{bmatrix} & & \\ & R_{3x3} & \\ & & \end{bmatrix} & \begin{bmatrix} \\ T_{3x1} \\ \\ \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} & \begin{bmatrix} k \\ l \\ m \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Equation 9: Transformation Matrix M [61]

This transformation matrix consists of a rotation matrix R_{3x3}, giving the rotation from one coordinate system into the other, and a translation vector T_{3x1}, giving the shift of the origin of the coordinate systems.

Both, the rotation and the translation between the coordinate systems, are defined by the setup of the workplace. They have to be calculated, if the setup of the workplace was changed. To automatically acquire the transformation matrix, the workplace was outfitted with three highly reflective balls, used as reference points, whose coordinates can be determined in the robot's coordinate frame, as well as in the camera's coordinate frame, as described in chapter 4.3.5.

Figure 63 graphically shows the determination of the shift in origin and rotation.



a)
b)
c)
d)
e)
f)
P1
P2
P3
P1'
P2'
P3'
0
0'
$T = \begin{bmatrix} a & b & c & ? \\ d & e & f & ? \\ g & h & i & ? \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$T = \begin{bmatrix} a & b & c & k \\ d & e & f & l \\ g & h & i & m \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$\begin{pmatrix} k \\ l \\ m \end{pmatrix}$

Figure 63: Gaining the transposition matrix from one coordinate frame to another
P_1 to P_3 mark the reference points measured by the camera
0 marks the origin of the camera's coordinate system
P_1' to P_3' and 0' mark the respective points in the robot's coordinate system

Figure 63a to f show the camera's coordinate system to the left and the robot's coordinate system to the right.

The first step in Figure 63a is the determination of the coordinates of the three reference points in both coordinate systems. In case of the robot's coordinate system these points have been measured by moving the robot's tool centre point above those points and noting the coordinates in the developed program. Since the robot as well as the reference points have a fixed location in this thesis, these measurements do not have to be repeated and can thus have been hardcoded in the program. An approach on the measurement in different workplaces is described in chapter 7.2. For measuring the points with the camera, a subprogram was created, that allows the user to visually target the reference points in the GUI. Since the balls are covered in highly reflective material, they show as white pixels in the reflection image (see Figure 29 in chapter 4.3.5).

To compare the rotations of both coordinate systems, the measurement of these systems in Figure 63a would be sufficient, if all coordinates were acquired without any error. Since the camera's measurement as well as the manual robot measurement both induce small errors, those errors have to be compensated. The chosen way to achieve this is to create coordinate systems from the measured points that are equal in their dimensions. This is achieved from Figure 63b to d.

The second step of the algorithm in Figure 63b calculates the vectors from point one to point two and three in both coordinate systems.

From those two vectors a third vector (Figure 63c) is created, that is orthogonally to both previous vectors. This is achieved by using the cross product (see Equation 10). Since the points are numbered, both coordinate systems' third vector creates a right handed coordinate system.

$$\begin{pmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \end{pmatrix} \times \begin{pmatrix} p_{2x} \\ p_{2y} \\ p_{2z} \end{pmatrix} = \begin{pmatrix} p_{1y} * p_{2z} - p_{1z} * p_{2y} \\ p_{1z} * p_{2x} - p_{1x} * p_{2z} \\ p_{1x} * p_{2y} - p_{1y} * p_{2x} \end{pmatrix}$$

Equation 10: Cross product of two three-dimensional vectors [61]

Those three vectors build a three dimensional coordinate system in the camera's respectively robot's system. By comparing the rotations of each new system to their own original system, the rotation of the original systems towards each other could be determined. To compensate for the described errors in measurement, an additional step was included (Figure 63d) that creates a second orthogonal vector from the vector between point one and two and the previously calculated orthogonal vector. This results in three vectors that are orthogonal towards each other. After reducing their length to one, both coordinate systems are identical in all but their rotation. Thus the numerical errors from measuring are eliminated in further calculation.

In Figure 63e the rotational part of the matrix is determined. The camera's coordinate system matrix is converted by using the unity matrix into the coordinate system of the robot. In parallel, all used conversions are applied to a unity matrix, which results in the rotation matrix (see Equation 11).

$$\begin{bmatrix} C_{1x} & C_{2x} & C_{3x} \\ C_{1y} & C_{2y} & C_{3y} \\ C_{1z} & C_{2z} & C_{3z} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} R_{1x} & R_{2x} & R_{3x} \\ R_{1y} & R_{2y} & R_{3y} \\ R_{1z} & R_{2z} & R_{3z} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} a' & b' & c' \\ d' & e' & f' \\ g' & h' & i' \end{bmatrix} \rightarrow \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Equation 11: Determining a rotation matrix [61]

The translation vector is determined, by rotating the position vector of the first point $\overrightarrow{0P_1}$ from the camera's coordinate frame to the robot's coordinate frame and by calculating the translation between this rotated first point with the original first point of the robot's coordinate frame (see Equation 12).

$$\begin{bmatrix} k \\ l \\ m \end{bmatrix} = \begin{bmatrix} P_{R1x} \\ P_{R1y} \\ P_{R1z} \end{bmatrix} - \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} P_{C1x} \\ P_{C1y} \\ P_{C1z} \end{bmatrix}$$

Equation 12: Determining the translation vector [61]

Graphically this can be described as rotating the camera's point of origin – 0 – around the first measurement point – P_1 – and then calculating the shift between the rotated camera origin and the original robot origin – $0'$ – (see Figure 63f).

5.5.3 Determining the reachable space

This algorithm is designed to divide the workspace into reachable and unreachable space (as defined in chapter 4.3.4). Thereby, as defined in chapter 2.3.1, generally all space above and behind a dynamic object, must not be accessible to the robot and thereby builds the unreachable space.

Figure 64 shows the result of the algorithm, as it is depicted on the computer screen, showing the green robot, a red intruding object and the blue border of the reachable space.

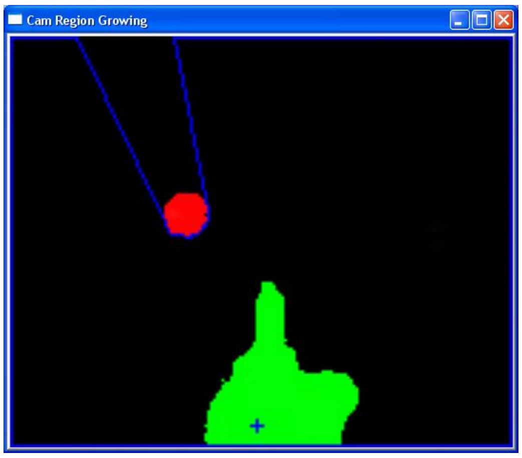


Figure 64: Image depicting the reachable space to the user

The algorithm, used to create this result, is depicted in Figure 65.

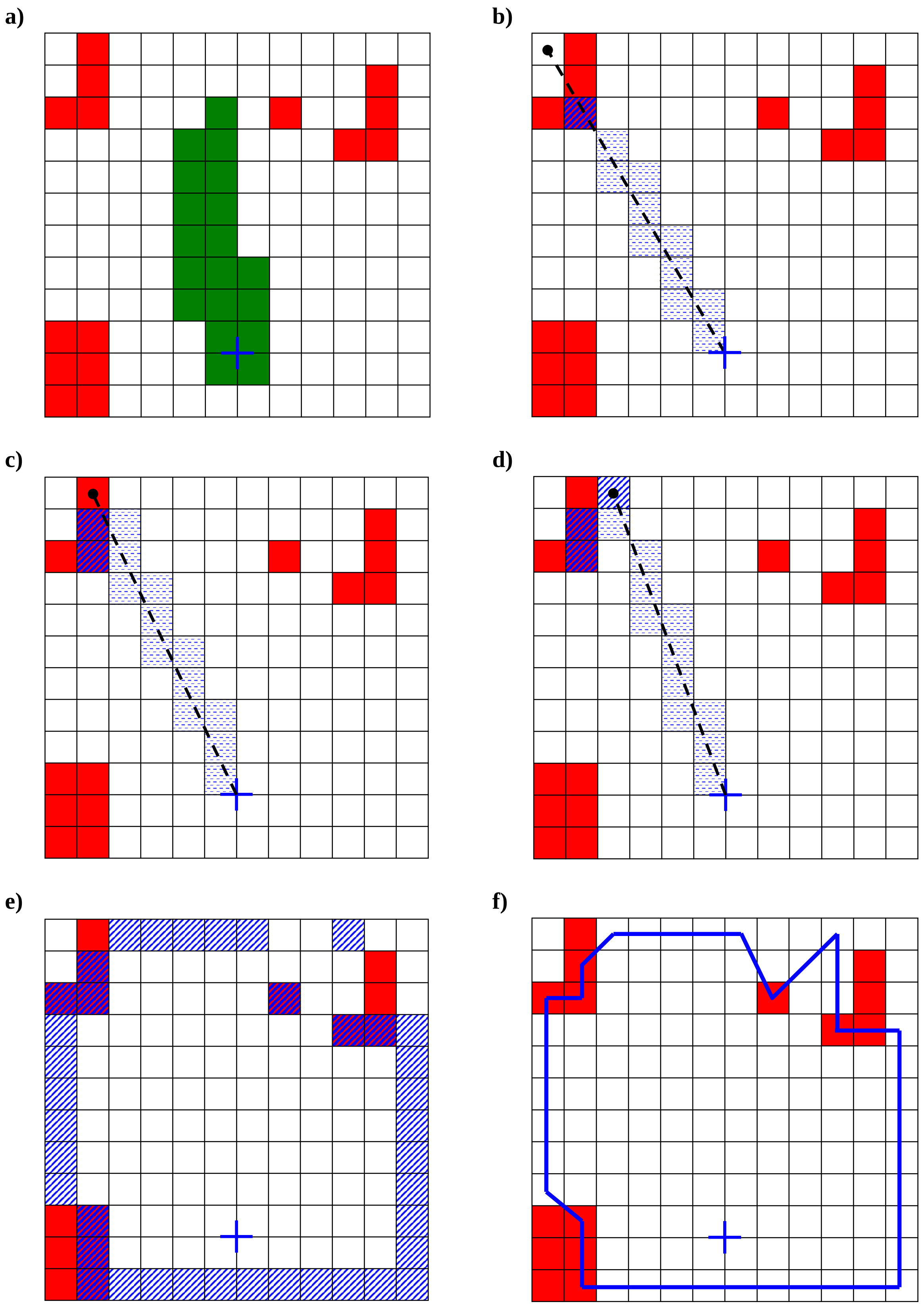


Figure 65: Determining the reachable space

From the intrusion map, shown in Figure 65a, the robot area is excluded.

In Figure 65b the algorithm calculates a line from the centre of the robot's base position to the first pixel on the border of the intrusion map. Along this line every pixel that is crossed by the line is checked for intrusion. If a pixel containing an intrusion is found, as well as when the border-pixel is reached, the algorithm stores that pixel as an unreachable pixel and the next border-pixel is tried to be reached (Figure 65c and d – stopped at the blue pixels).

The algorithm checks each border-pixel once, until all border-pixels are checked, as shown in Figure 65e.

From the remembered pixels in Figure 65e, the border of the reachable space is generated (Figure 65f) by drawing a closed line (borderline) through the remembered pixels. This borderline marks the first pixels that can not be reached.

5.5.4 Selection of significant part

When the robot is moving from one point to another, not all of the reachable space is necessary to compute the collision free space, but only the part that is between the current robot position and the position of the goal.

Therefore, in this algorithm:

- The borderline is transformed from the camera's coordinate system into the robot's coordinate system.
- The coordinates of the pixels of the borderline are transformed from Cartesian coordinate system to Polar coordinate system, the system of the robot's axes.
- The significant part of the borderline lying between the current robot's angle and the goal position's angle is selected.

The first step in the selection algorithm, transforming the values of the borderline into the robot coordinate system, is achieved by applying the transformation matrix determined in chapter 5.5.2.

There are problems that have to be solved before transforming the values:

- It was observed during this thesis that in the outer regions of the acquired distance images the camera's measurements sometimes wrongly show a very short distance.
- The safety distance has to be dependant on the tool centre point's height, since the robot can work directly below an intrusion, as long as the angle of the camera allows a clear separation (see Figure 66) of the robot and the intruding object.
- The reverse of the second point is true as well, where a high position of the tool centre point casts a blind spot on intruding objects that are farther away in the horizontal direction but on a lower position.

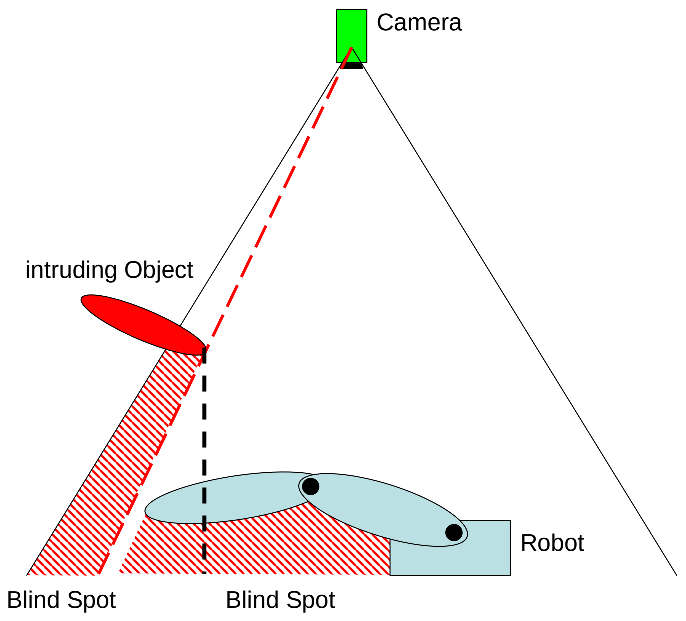


Figure 66: Robot working below intruding object

To overcome the camera errors and reduce later computational effort, dealing with the blind spots and their merging, the applied transformation does not use the actual distance values measured by the camera. Instead, it sets the height of the measured distances, before their transformation into the camera's Cartesian coordinate system, to the current height of the tool centre point of the robot (see Figure 67). This height can be reliably acquired from the robot's internal sensors.

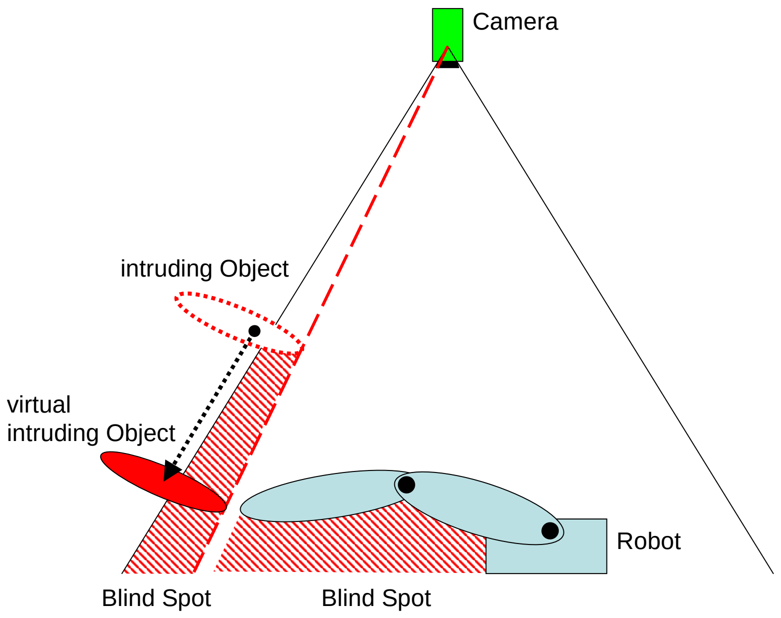


Figure 67: Resetting the height of intruding objects to match the robot's tool centre point

In this thesis, the height value of the robot's tool centre point does not change in the test program. Thus it was possible to set this value as a fixed value in the coordinate's transformation. This will be adapted in future work for dynamic heights, as described in chapter 7.2.

The transformed values are in the robot's Cartesian coordinate system. Further computation in this algorithm is easier, if the coordinates are transformed into the robot's Polar coordinate system. The representation of the borderline values of the reachable space is thus changed

from $\begin{pmatrix} x \\ y \end{pmatrix}$, x and y being the horizontal Cartesian coordinates, to $\begin{pmatrix} \alpha \\ l \end{pmatrix}$,

where α is the angle of the robot's first axis at the given Cartesian coordinate, and l is the distance between the coordinate and the robot base's centre. The height value z can be dropped at this point, since all values have the same height as the robot's tool centre point after the previous conversion.

The selection of the significant part is then every value where

$$\alpha_{current} \leq \alpha \leq \alpha_{goal} \text{ if } \alpha_{current} \leq \alpha_{goal},$$

or

$$\alpha_{goal} \leq \alpha \leq \alpha_{current} \text{ if } \alpha_{goal} < \alpha_{current}.$$

Figure 68 shows the definition of the significant part, Figure 68a showing the complete borderline, the current position and the goal position. In Figure 68b the borderline is reduced to the significant part.

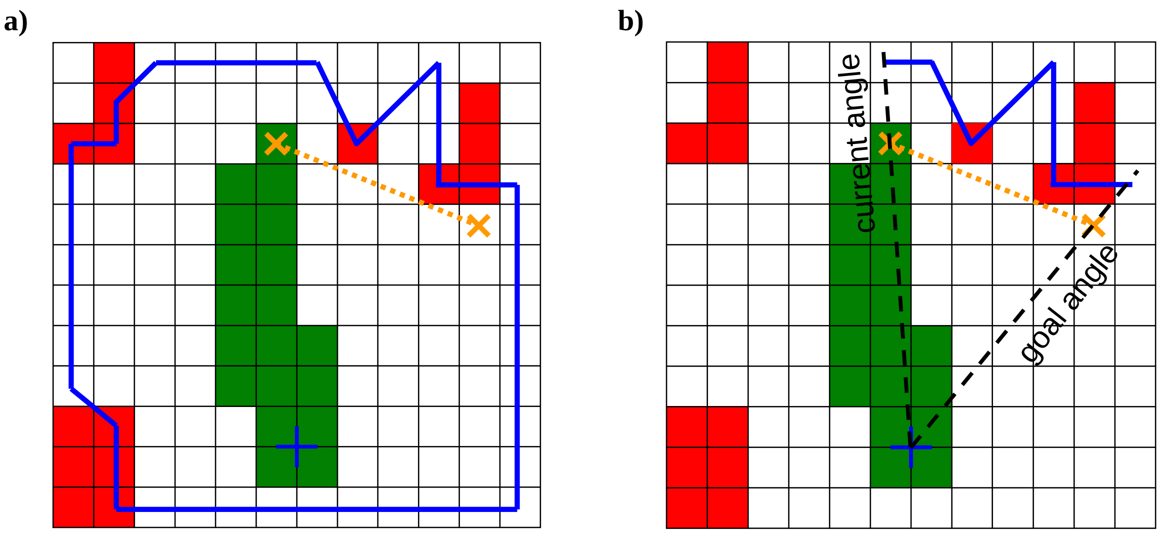


Figure 68: Selection of the significant part

The orange line in Figure 68 shows the shortest connection of the current position and the goal, a straight line. This line can only be chosen, if it is not violated by any intrusion.

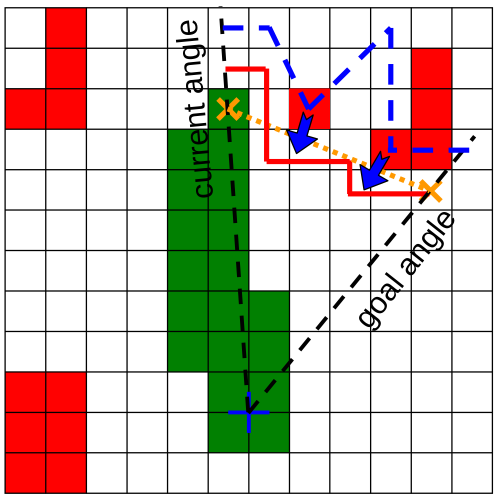
5.5.5 Calculating the shortest evasion path

For safety reasons, the length value in each point of the borderline from the previous chapter (5.5.4) is reduced by a certain value, effectively increasing the distance the robot has to keep from intruding objects by creating a safety borderline (see Figure 69a). As described in the outlook of this work (chapter 7.2), this increase in the safety distance will later be dynamical, dependant on several factors like the objects current speed and direction of movement.

If the current goal position of the robot is blocked by an intruding object, it is relocated at its given angle. Its new position is the same as the point of the safety borderline at that angle. This allows the robot to drive as close to its goal as possible.

As a next step towards the shortest evasion path, all parts of the safety border line that are further away than the direct path between the current angle and the goal angle are neglected (see Figure 69b). This safes computational effort in later steps.

a)



b)

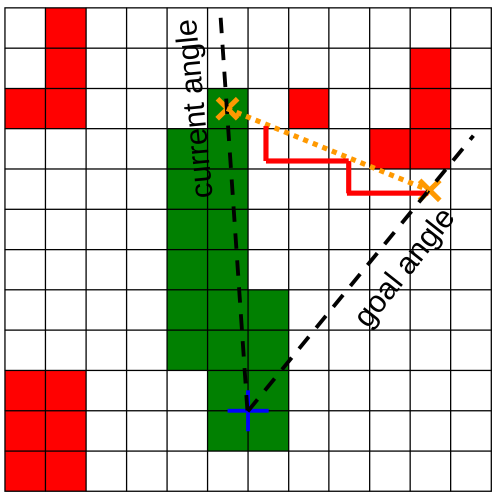


Figure 69: a) Red safety border line and b) its significant part

To find those parts of the safety borderline that are closer to the robot base's centre than the direct path to the goal, the intersections of the line connecting the current position and the goal position and the line from the centre of the robot to the individual safety borderline-points are computed (see Equation 13).

$$\begin{pmatrix} Current_x \\ Current_y \end{pmatrix} + \lambda \begin{pmatrix} Current_x - Goal_x \\ Current_y - Goal_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} Borderline_x \\ Borderline_y \end{pmatrix}$$

$$\Rightarrow \gamma = \frac{(Current_x - Goal_x) * Current_y - (Current_y - Goal_y) * Current_x}{(Current_y - Goal_y) * Borderline_x - (Current_x - Goal_x) * Borderline_y}$$

Equation 13: Intersecting two lines [61]

If the calculated γ, the length of the vector from the robot base's centre to the safety borderline-point, is less than 1, the point can be ignored in further processing, because it lies behind the direct connection of current position and goal.

Using the remaining points of the safety borderline, a path has to be found between the current position and the goal position that is avoiding this safety borderline and is as short as possible, in order to avoid the dynamic objects and optimize the execution time. The used algorithm assumes that the robot can always evade towards its base, which is true if only intruding objects have to be considered.

A sample safety borderline is shown in Figure 70. This line will serve as an example. The line includes five remaining points, each depicted as a cross.

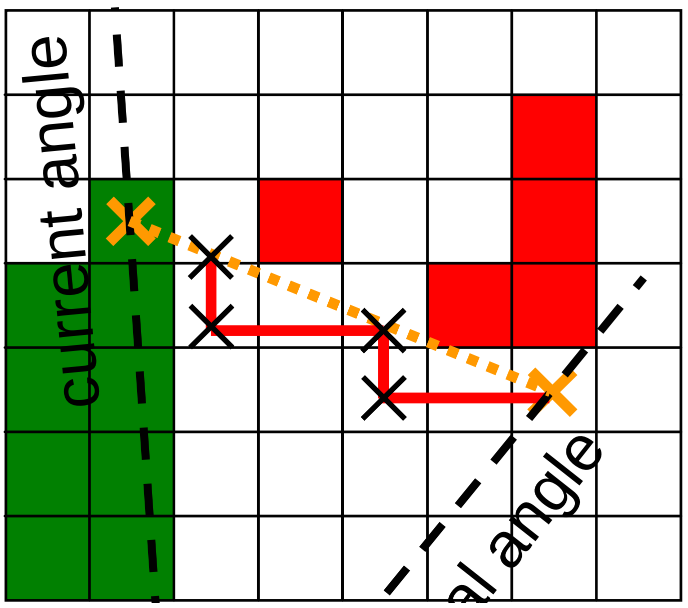


Figure 70: Sample intrusion map

The shortest path along a safety borderline is always defined through the fact that all of its angles turn in one direction.

The single steps of the devised algorithm to find such a path are depicted in Figure 71, starting with the first step in Figure 71a and ending with the final result in Figure 71f.

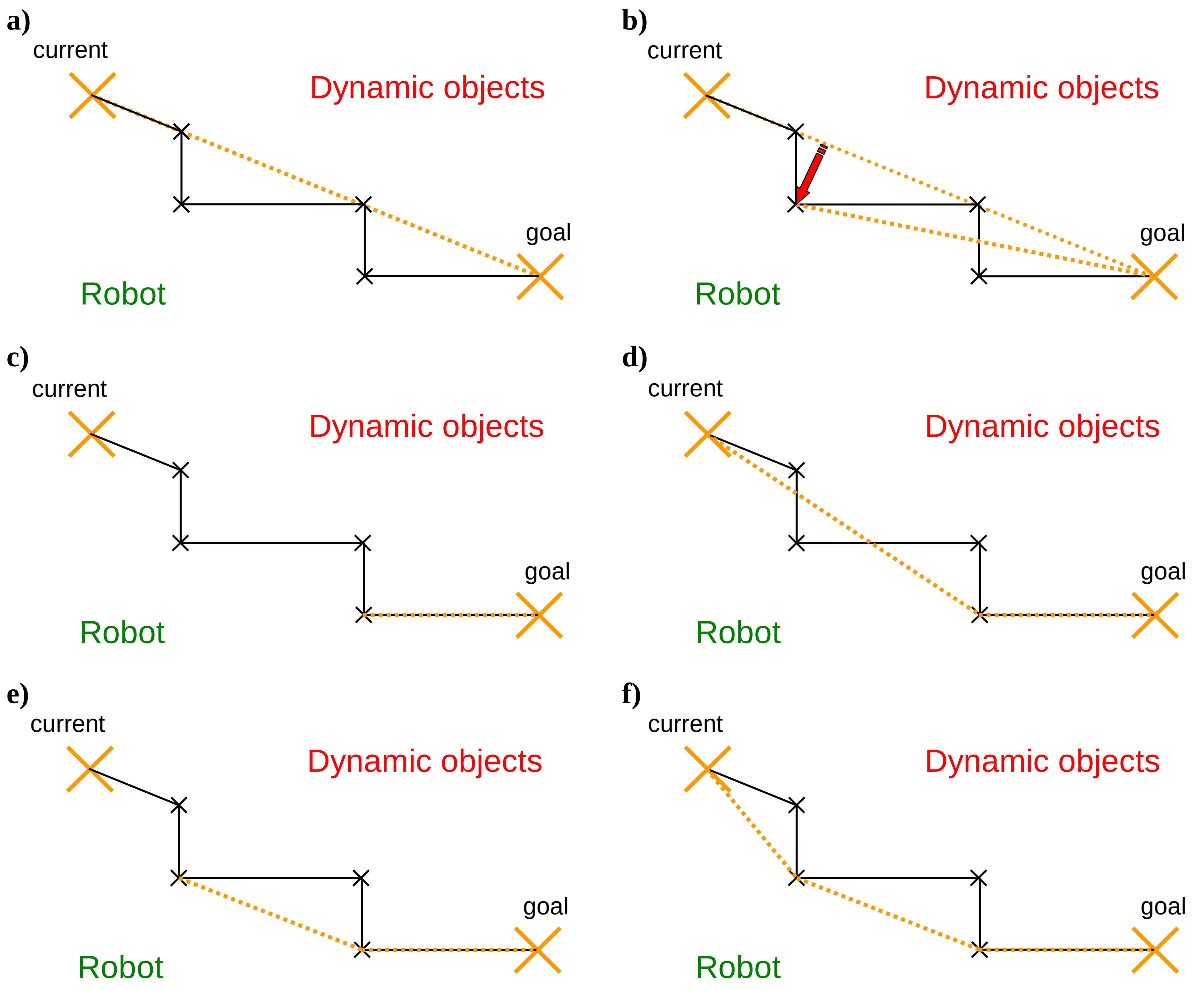


Figure 71: Graphical explanation of the calculation of the shortest evasion path

The calculations that lead to the results depicted by the images are:

- A straight line is drawn from the first to the last point (Figure 71a)
- The next point is checked, whether it is “below” the first line (closer to the robot) and if so it is set as the start-pixel for the current line. (Figure 71b)
- Step one and two are repeated, until no more pixels are below the current line (Figure 71c).
- The current line is stored and a new line is begun.
- The new line is drawn from the fist pixel to the start of the recently stored line (Figure 71d).
- Step one and two are repeated, until no more pixels are below the new line (Figure 71e).
- Again, a new line is drawn from the first pixel to the start of the recently stored line (Figure 71f).

The algorithm is finished, if the stored line begins in the first pixel.

A flowchart, showing the complete algorithm, is shown in Figure 72.

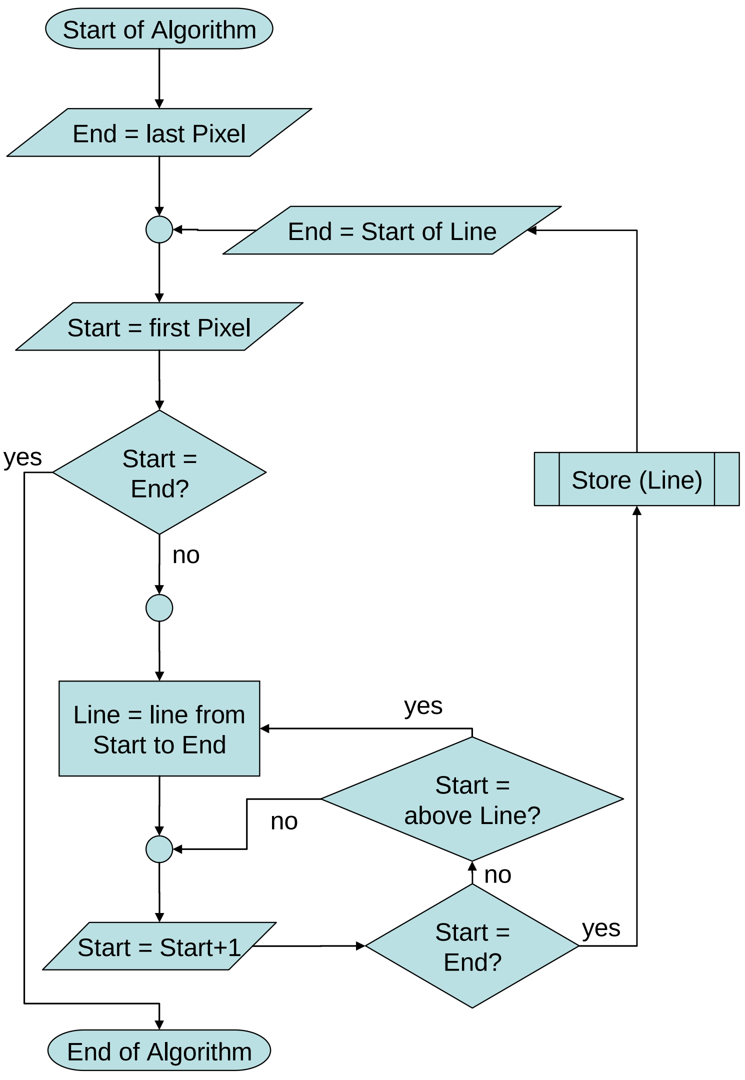


Figure 72: Flowchart for the “shortest Line” algorithm

The final path, starting at the robot’s current position, is the combination of this position and of all stored endpoints of the lines, beginning with the endpoint of the latest line that has been stored.

This first endpoint is transmitted as an intermediary goal point to the robot controlling task (see chapter 4.4). The remaining points need not be transmitted, since the path is continuously recalculated. If no changes of the intruding dynamic objects occur in the workplace while the robot is moving, the next point in line will be transmitted when the first intermediary point is reached. If changes occur the respective new intermediary goal point is transmitted.

As described in chapter 4.4.1 the robot can not approach every position as exactly as its sensors can measure the robot’s current position. Thus to avoid problems in the path execution, a goal point is not transmitted if it is within 3 cm of the current position of the robot’s tool centre point. In this case the first goal position is transmitted that is father away than 3 cm.

6 Evaluating the safety of the system

The primary goal of this master thesis was a feasibility study on realizing a working collaboration workplace, a workplace where human and robot work alongside each other. This goal has been reached, as described in previous chapters.

The safety of the testing environment was verified by the application of a hold to run button and the reduced speed of the robot. Realizing such a workplace in praxis would require an integration of safety with a reduced inclusion of humans into the safety concept.

For the integration of safety in robot workplaces there are several regulations to be considered. Apart from formal requirements those regulations also contain requirements on safety. Considering those safety requirements is made easier by applying appropriate standards.

The main point of the developed approach is to ensure the safety distance between robot and human. The safety distance required by the given workplace is evaluated, based on the performance of its components and the developed program. Afterwards the reverse is computed, taking a desired safety distance and showing the necessary performance in the components.

The results are compared, showing the state of development of the system and the necessary improvements.

6.1 Regulations

When placing machinery on the market and/or putting it into service several regulations have to be applied. In the European Common Market requirements are laid down in directives for most products, so called harmonized products. When concentrating on the requirements of this robot working place, especially on its safety, the main European directives that have to be considered are:

- Machinery Directive 98/37/EC – until 28.12.2009
- Machinery Directive 2006/42/EC – from 29.12.2009
- Pressure equipment Directive 97/23/EC
- Low voltage Directive 2006/95/EC
- EMC Directive 2004/108/EC

Those directives have to be transposed into national law by the member states without changes. Thus it is possible for this project to relay on the European law, making the results of the project applicable in the whole common market.

Considering the safety problems induced by the new approach on collaboration of human and robot, only the requirements of the machinery directive are important. The safety requirements of all other regulations are not relevant for this new concept. Their application causes no special problems that have not already been solved in today’s workplaces.

Since the realization in praxis of the presented concept will still need some time, the old machinery directive 98/37/EC can be disregarded in the considerations of this thesis.

6.1.1 Machinery Directive 2006/42/EC

Robots, as all machinery, are included in the scope of the Machinery Directive (2006/42/EC). The requirements on the placing on the market of machinery are laid down in article 5. The safety requirements are formulated in article 5 a.

“Article 5
Placing on the market and putting into service
1. Before placing machinery on the market and/or putting it into service, the manufacturer or his authorised representative shall:

(a) ensure that it satisfies the relevant essential health and safety requirements set out in Annex I;” [66]

The article 5 a refers to annex I, where the relevant essential health and safety requirements are formulated in detail.

The main requirements, important for the new hazards induced by the presented solution in this thesis are:

- 1.1.2. Principles of safety integration
- 1.2.1. Safety and reliability of control systems
- 1.2.2. Control devices
- 1.2.3. Starting
- 1.3.7. Risks related to moving parts

The most important aspects of those requirements are as follows:

- *1.1.2. Principles of safety integration*
  (a) Machinery must be designed and constructed so that it is fitted for its function, and can be operated, adjusted and maintained without putting persons at risk when these operations are carried out under the conditions foreseen but also taking into account any reasonably foreseeable misuse thereof. […]
  (b) In selecting the most appropriate methods, the manufacturer or his authorised representative must apply the following principles, in the order given:
  — eliminate or reduce risks as far as possible (inherently safe machinery design and construction),
  — take the necessary protective measures in relation to risks that cannot be eliminated,
  — inform users of the residual risks […]
- *1.2.1. Safety and reliability of control systems*
  Control systems must be designed and constructed in such a way as to prevent hazardous situations from arising. […]
  — a fault in the hardware or the software of the control system does not lead to hazardous situations,
  — errors in the control system logic do not lead to hazardous situations […]
- *1.2.2. Control devices*
  […]

Where a control device is designed and constructed to perform several different actions, namely where there is no one-to-one correspondence, the action to be performed must be clearly displayed and subject to confirmation, where necessary. […]

- *1.2.3. Starting*
  It must be possible to start machinery only by voluntary actuation of a control device provided for the purpose. […]
  For machinery functioning in automatic mode, the starting of the machinery, restarting after a stoppage, or a change in operating conditions may be possible without intervention, provided this does not lead to a hazardous situation. […]
- *1.3.7. Risks related to moving parts*
  The moving parts of machinery must be designed and constructed in such a way as to prevent risks of contact which could lead to accidents or must, where risks persist, be fitted with guards or protective devices. […] [66]

6.2 Standards

Generally directives all require applying the state of the art. When using so called harmonized standards, the manufacturer has the consumption of conformity. That means he can assume that he has fulfilled the essential health and safety requirements, as laid down in the technical annexes of the directives. In the machinery directive this is formulated in article 2 l.

"Article 2 (l):
'harmonised standard' means a non-binding technical specification adopted by a standardisation body, namely the European Committee for Standardisation (CEN), the European Committee for Electrotechnical Standardisation (CENELEC) or the European Telecommunications Standards Institute (ETSI), on the basis of a remit issued by the Commission in accordance with the procedures laid down in Directive 98/34/EC of the European Parliament and of the Council of 22 June 1998 laying down a procedure for the provision of information in the field of technical standards and regulations and of rules on Information Society services." [66]

Standards are divided into A, B and C norms, ranging from fundamental safety standards which are applicable to all machinery (type A) over standards dealing with aspects of safety that can be applied across a group of machinery (type B) to specific standards, applicable to a specific type of machinery (type C).

Since the approach given in this project is still under development no type C standard that matches the complete project exists. Thus, where possible, only type A and B standards can be used. Type C standards can only be used in parts.

The standards to be considered in a collaborative robot workplace are especially:

Type A:

- EN 12100-1 "Safety of machinery — Basic concepts, general principles for design — Part 1: Basic terminology, methodology"

  This standard covers the complete first part of annex I of the Machinery Directive. Therefore it covers also the points listed in chapter 6.1.1.

- EN ISO 14121 "Safety of machinery — Risk assessment — Part 1: Principles"

This standard contains a concept that can be used to reach the goals for the risk assessment which is required according to the general principles of annex I of the Machinery Directive respectively by EN 12100-1.

Type B:

- EN 999 “Safety of machinery — The positioning of protective equipment in respect of approach speeds of parts of the human body”

  This standard “*provides parameters based on values for hand/arm and approach speeds and the methodology to determine the minimum distances from specific sensing or actuating devices of protective equipment to a danger zone*” [69].

- EN ISO 13857 “Safety of machinery — Safety distances to prevent hazard zones being reached by upper and lower limbs”

  This standard gives distances that have to be kept between hazardous areas and openings in protective devices. Those values are designed to fit 95% of the population.

- EN ISO 13849-1 “Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design”

  This standard “*provides safety requirements and guidance on the principles for the design and integration of safety-related parts of control systems*” [72]. It covers the determination of the required performance level that a control has to achieve for a specific task and the evaluation of the performance level the given control is achieving.

Type C:

- EN ISO 10218-1 “Robots for Industrial Environment — Safety requirements — Part 1:Robot”

  This standard “*specifies requirements and guidelines for the inherent safe design, protective measures and information for use of industrial robots […]. It describes basic hazards associated with robots, and provides requirements to eliminate or adequately reduce the risks associated with these hazards.*” [70]

The type A standards are too general to be applied in the course of this thesis.

The type B and C standards are explained further in the following chapters.

6.2.1 Safety of machinery – Safety distances to prevent hazard zones being reached by upper and lower limbs EN ISO 13857:2008 (EN 294:1992)

This standard [73] is on safety distances to prevent humans from reaching harmful objects.

In this project an offset had to be chosen for the measured distance values, in order to decrease the amount of fluctuation noise (see chapter 4.3.3). This offset needed to be as large as possible, while still allowing intruding objects to be recognized.

Among other values, the EN 13857 [73] gives a maximum height of 120 mm above ground that can be left free when installing a fence. Thus this value was chosen as the desired value for the programmed “flexible fence”.

If those 120 mm are applied, the safety distance to the hazard zone has to be 850 mm for the safety of upper appendages, respectively 1100 mm for the safety of lower appendages.

6.2.2 Safety of machinery – The positioning of protective equipment in respect of approach speeds of parts of the human body EN 999:1998

The goal of the project is to safely avoid the collision between robot and humans. When using protective equipment in this task, in the case of this project the 3D camera combined with the data evaluating algorithms, safety distances have to be kept to compensate the systems stopping performance.

The standard EN 999 gives formulas, regarding the positioning of protective equipment in respect of approach speeds of parts of the human body. Applying those formulas to the present system results in the necessary safety distance that has to be kept between robot and human at all times.

Critical for the positioning of the protective devices is the overall system stopping performance T (see Equation 14).

$$T = t_1 + t_2$$

Equation 14: Overall system stopping performance, according to [69]

t_1 is the maximum time between the actuation of the sensing function and the output signal switching devices being in the full state, [69]

t_2 is the maximum response time of the machine, i.e. the time required to stop the machine or remove the risk after receiving the output signal from the protective equipment. [69]

Since the used 3D camera in this project has a maximum detection capability of below 40 mm in diameter (20 mm resolution at 4 m distance and 25 mm resolution at 5 m distance, according to [55]) the formula given in chapter 6.1.1 of the standard EN 999 has to be applied to the reaction time (Equation 15):

$$S = (K * T) + C$$

Equation 15: Minimum distance S from the danger zone to the detection zone [69]

S is the desired distance between the danger zone and the detection zone,

K is a fixed factor:

- 2000 mm/s if S is below or equal to 500 mm

- 1600 mm/s if S is equal to or greater than 500 mm

T is the calculated time in Equation 14

C is a constant for the protective equipment, derived from its detection capability:

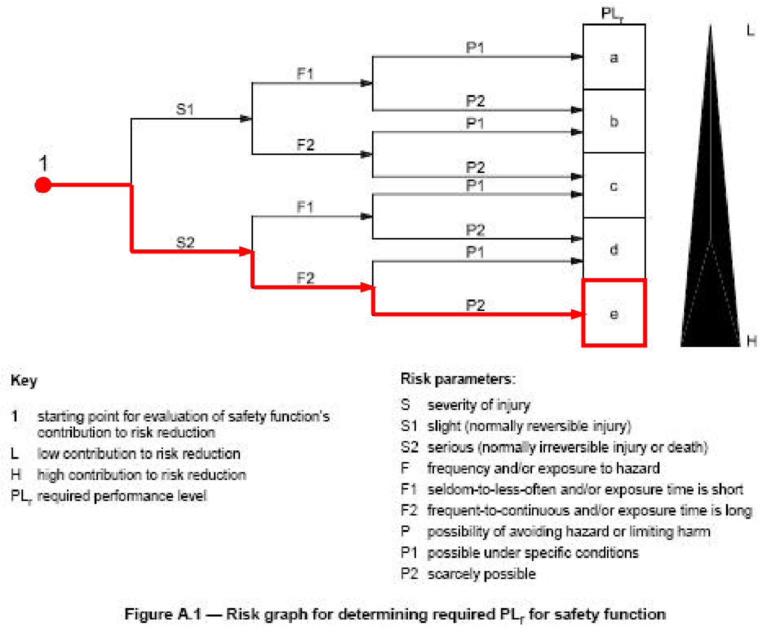
- $C = 8 * (d - 14mm)$, where d is the resolution of the protective equipment
- $C = 48mm$ at 4 m distance
- $C = 72mm$ at 5 m distance

The constant C represents the ability of the protective equipment to recognize hands, fingers, etc.

6.2.3 Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design EN ISO 13849-1

This standard can be applied to design the safety of the used equipment. Since in this thesis the developed equipment was left unsafe and the system was secured by other means only the necessary safety is evaluated at this point.

Figure A.1 of the standard gives an overview at how to choose the performance level the combined components need to achieve. Applying the graph shown in Figure 73 results in a required performance level of e, since the possible injuries are serious (S2), the exposure in a collaborative workplace is continuous (F2) and the possibility of avoiding a 4 m/sec fast robot are scarcely possible (P2).



Key

1 starting point for evaluation of safety function's contribution to risk reduction
L low contribution to risk reduction
H high contribution to risk reduction
PL$_r$ required performance level

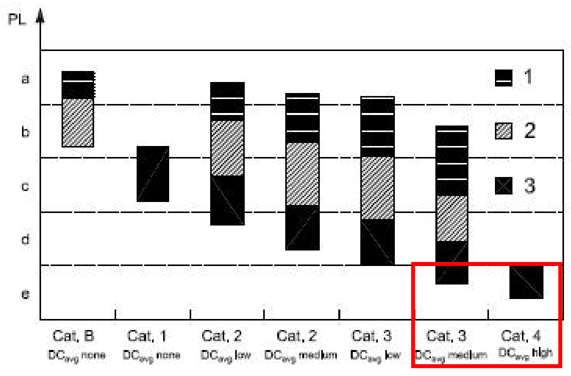
Risk parameters:

S severity of injury
S1 slight (normally reversible injury)
S2 serious (normally irreversible injury or death)
F frequency and/or exposure to hazard
F1 seldom-to-less-often and/or exposure time is short
F2 frequent-to-continuous and/or exposure time is long
P possibility of avoiding hazard or limiting harm
P1 possible under specific conditions
P2 scarcely possible

Figure A.1 — Risk graph for determining required PL$_r$ for safety function

Figure 73: Risk graph of the EN 13849-1 [72]

The performance level e also requires the control to achieve category 3 to 4, according to figure 5 of the standard (see Figure 74).



Key

PL performance level
1 $MTTF_d$ of each channel = low
2 $MTTF_d$ of each channel = medium
3 $MTTF_d$ of each channel = high

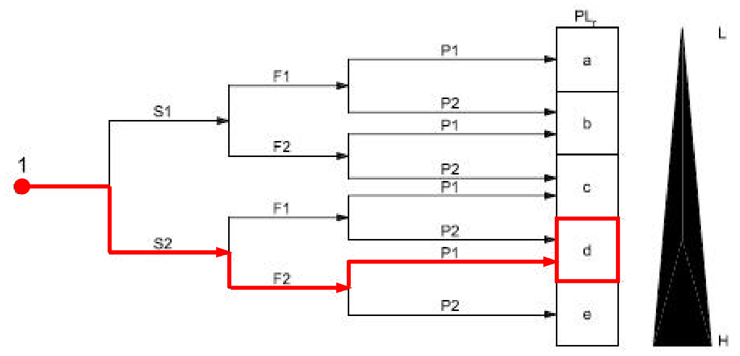
Figure 5 — Relationship between categories, DC_{avg}, $MTTF_d$ of each channel and PL

Figure 74: Relationship between categories, DC_{avg}, $MTTF_d$ of each channel and PL of the EN 13849-1 [72]

The standard has special requirements on systems with those categories:

In categories [..] 3 and 4, improved performance in respect of a specified safety function is achieved predominantly by improving the structure of the SRP/CS. [..] In categories 3 and 4 this is provided by ensuring that the single fault will not lead to the loss of the safety function. In category 4, and whenever reasonably practicable in category 3, such faults will be detected. In category 4 the resistance to the accumulation of faults will be specified. [72]

The necessity for the application of the highest performance level can be avoided, if the speed of the robot is sufficiently reduced. If this reduction in speed allows the possible avoidance of harm by the operator, the required performance level can be reduced to performance level d, as shown in Figure 75.



Key

1 starting point for evaluation of safety function's contribution to risk reduction
L low contribution to risk reduction
H high contribution to risk reduction
PL_r required performance level

Risk parameters:

S severity of injury
S1 slight (normally reversible injury)
S2 serious (normally irreversible injury or death)
F frequency and/or exposure to hazard
F1 seldom-to-less-often and/or exposure time is short
F2 frequent-to-continuous and/or exposure time is long
P possibility of avoiding hazard or limiting harm
P1 possible under specific conditions
P2 scarcely possible

Figure A.1 — Risk graph for determining required PL_r for safety function

Figure 75: Risk graph of the EN 13849-1 [72], reduced robot speed

This reduction in the required performance level also lowers the requirements on the category of the used system, as shown in Figure 76.

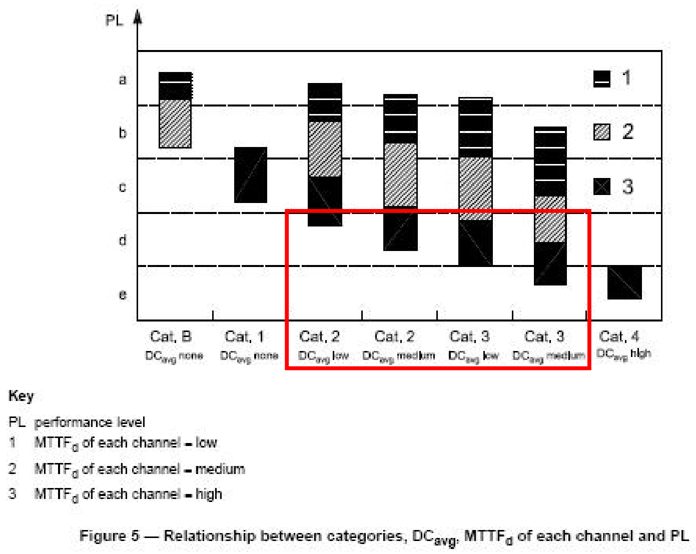


Figure 76: Relationship between categories, DC_{avg}, $MTTF_d$ of each channel and PL of the EN 13849-1 [72]

This would allow the application of controls with category 2 or higher.

The reduced robot speed, which is necessary if a lower performance level is to be used, can be achieved mechanically, by constructing a robot that can not move faster, or electronically, e. g. by observing the robot’s speed with a safety controller (see chapter 3.1).

Reducing the speed mechanically has the disadvantage that the robot can not move faster if no operator is present in the robot’s vicinity and also that existing robots can not be easily updated to collaborative workspaces.

Reducing the speed electronically has the disadvantage that this reduction in speed has to be safe as well. Therefore the risk graph has to be applied, based on the robot moving with a speed of 4 m/sec. Thus the control, which enforces the electronic speed reduction, has to comply with performance level e (see Figure 73). If sensors are used that observe the working place, distinguishing between the presence and absence of operators in the robot’s vicinity, they have to comply with the requirements of performance level e as well.

6.2.4 Robots for Industrial Environment – Safety requirements – Part 1: Robot EN 10218-1

To achieve conformity with the machinery directive (see chapter 6.1.1) to the point of “presumption of conformity” harmonized standards have to be applied that cover all hazards related to the machinery. The applicant can assure the coverage of all relevant hazards for his special machinery by applying type C standards. The standard EN 10218-1 [70] is such a standard for robots.

This type C standard contains general remarks on the developed approach, the “flexible fence”:

- *5.12.3 Safety-rated soft axis and space limiting*

“Soft limits are software-defined spaces that define the restricted space where the robot is inhibited from exiting or entering while in automatic mode or any mode using speeds above reduced speed. […] Motion may be restricted to keep the robot inside the defined space or to prevent the robot from entering or accessing the defined excluded space.
[…]
A safety-rated soft limit shall be set as a stationary zone that cannot be changed without a system power up condition and shall not be changed dynamically. […]” [70]

- *5.12.4 Dynamic limiting devices*
  “Dynamic limiting is the automatically controlled change in a robot’s restricted space during a portion of the robot system’s cycle. Control devices such as, but not limited to, cam operated limit switches, light curtains [...] may be utilized to further limit robot movement within the restricted space while the robot performs its task programme provided the device and associated controls are capable of stopping the robot motion under rated load and speed conditions and the associated safety controls comply with category 3 of ISO 13849-1, unless a risk assessment is performed and determines that another category is required.” [70]

While the standard allows “dynamic limiting devices”, such as the flexible fence, it forbids the inclusion of this fence in the safety concept of the used robots safety trips (see chapter 3.1), since it demands for the robot’s control to be rebooted for every change.

The connection of safety trips and developed approach is not yet realized, but in order to achieve the category 3, as requested by this standard and the EN 13849-1 as well (see chapter 6.2.3), this would be an advantageous step in the future.

The standard therefore can be applied only in parts. Future developments in the field of collaborative workplaces, “flexible fences” becoming state of the art, will possibly lead to changes in this standard.

6.3 Achieved safety

To estimate the achieved safety of the workplace, the system was tested on its performance. From this performance the necessary safety distance of the system is calculated in this chapter. During the course of this chapter the used equipment is treated as if it were safe, knowing well that this is not the case.

All tests were performed on an Intel Core 2 Duo E8200 (2.66 GHz), equipped with 2 GB of RAM, running WinXP SP2. The developed program was executed in the debug mode of Visual Studio 2008 Version 9.0.30729.1 SP, in order to acquire the different runtimes.

The additional computational overhead of the debug mode has only minor influences in the complete runtime, since the largest amount of time is used for communication between camera, PC and robot’s PLC.

The error in measurement is too low to have a visible influence and was thus omitted in this chapter. If the presented calculations would not be done to show a measurement of the achieved results of this thesis and the constructed demonstrator, but to calculate the performance of such a system in industrial praxis, an additional error calculation would be necessary.

Due to the necessary accuracy of the camera, used in the workplace, only 10 frames per second (FPS) could be delivered by the camera. The camera is able to deliver more frames per second, but when it does not deliver the maximum amount, its internal chip is already using filters to combine the internally taken images, reducing the fluctuation noise. The increase in fluctuation noise would have made the application of additional filters necessary that work on multiple images. Thus the FPS would have been reduced in the program and additional computational effort would have been necessary.

The performance of the camera control thread is thus limited in its execution time to a maximum of 10 FPS, a reaction time of 0.1 seconds.

The first performance test used the reachable space computation (see chapter 4.3.4) with different offsets. The worst case for the reachable space computation is an environment free of objects, which was chosen in the test. During the test the offset value was increased, reducing the amount of fluctuation noise that had to be filtered. Figure 77 shows the measured results, including the median value of all measurements, the maximum (best case) value and the minimum (worst case) value observed.

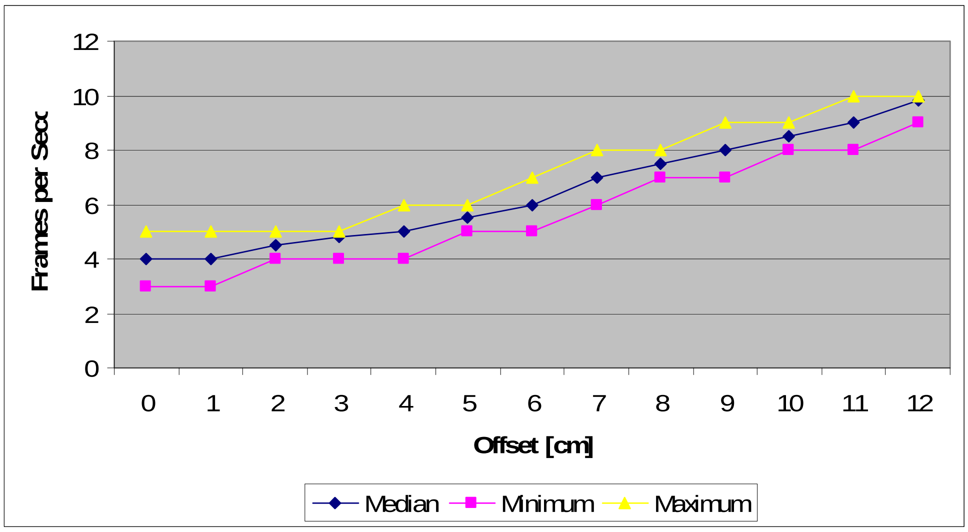


Figure 77: Reachable space computation, no intrusion present

The test showed that the developed filter algorithms are dependant in their performance on the present noise. The filters are able to perform sufficiently at an offset of 12 cm, the maximum value that is allowed according to EN 13857 (see chapter 6.2.1).

Since the calculation of the distance between the robot and intruding objects is more complex than the reachable space computation, only the offset of 12 cm was tested with the distance calculation. This algorithm takes longer, the farther away an intruding object is, because it has to execute more cycles to find the object. Thus it was tested with different distances of an intruding object to the robot. The maximum distance tested, 14 pixels (see chapter 5.4), is the distance at which the first decrease in the robot’s speed is induced by the program. Figure 78 shows the measured results, including the median value of all measurements, the maximum (best case) value and the minimum (worst case) value observed.

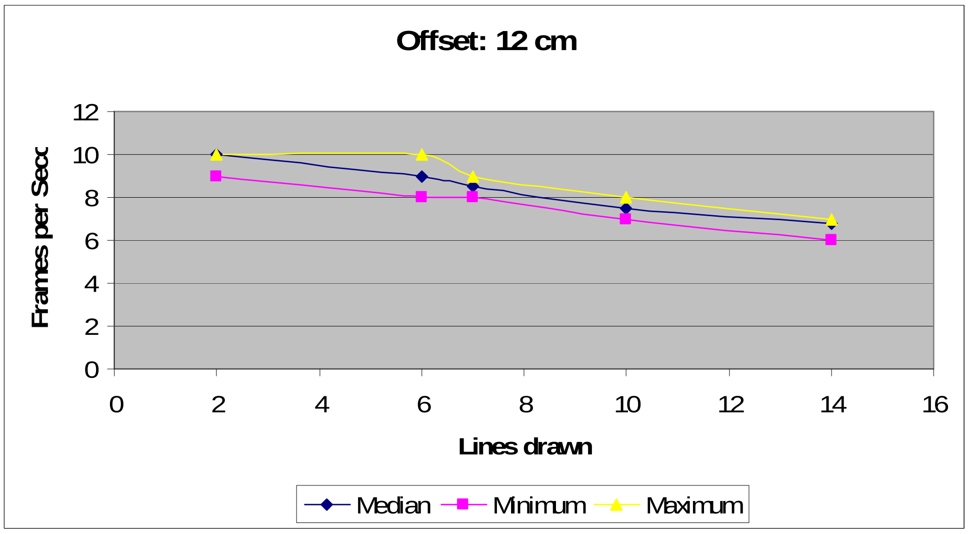


Figure 78: Distance calculation, intrusion at different distances to the robot, offset at 12cm

From those measurements it can be seen that the camera control thread, using the desired offset of 12 cm, works on at least 9 FPS with reachable space computation and decreases to at least 6 FPS with the calculation of the distance.

Thus the reaction time of the camera control thread is:

- reachable space: 111 ms
- distance calculation: 167 ms

Within those reaction times, the acquisition of a single frame of the camera is already included. This means that 100 ms of every camera control thread cycle is not used for calculation but is idle time, waiting on the next image.

The path planning thread proved to be too fast to allow the measuring of a single cycle. Thus it is assumed at 1 ms.

The robot control thread, communicating with the robot's PLC, used around 63 ms to complete one cycle most of the time. At 5 % of the measured cycles it used as much as 78 ms and the worst result measured was 100 ms. The responses of the PLC to the separate commands were already included in this time. The thread has to be assumed with its worst result, 100 ms.

When combining all those reaction times, it has to be taken into account that the path planning and the robot control thread need the results of the previous threads to achieve their task. Because the threads are not timed, threads can start their computation with old results, just before the previous thread is finished. Thus, when computing the worst reaction time, the secondary threads have to be counted twice.

Applying the doubling of thread-time to the results of the single threads leads to an overall reaction time of the program of 313 ms when executing the evasion path via the reachable space (see Figure 79) and 369 ms when executing the speed control via distance computation (see Figure 80).

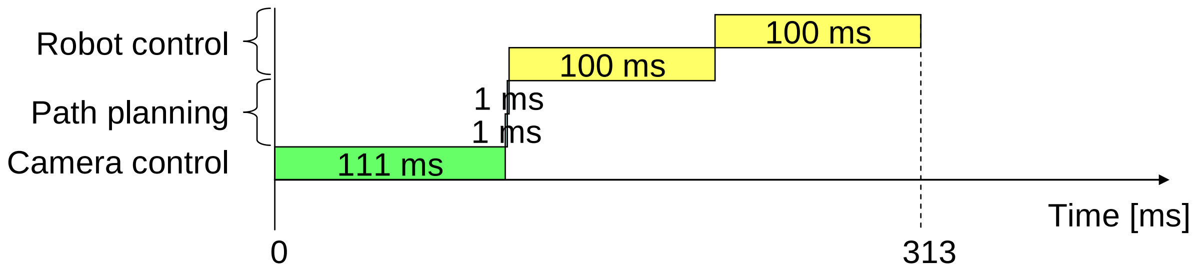


Figure 79: Overall reaction time of the program when executing the evasion path

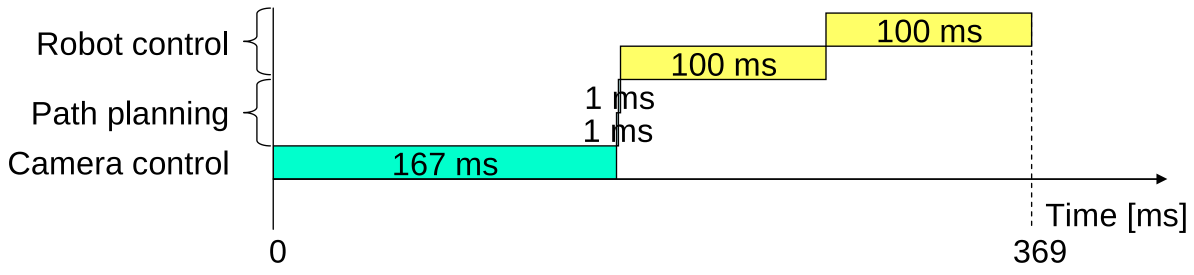


Figure 80: Overall reaction time of the program when executing the speed control

The implementation of synchronisation was omitted in this project, because other advantages would be lost. For example, if the robot control is completely integrated into the robot, the position of the robot is much faster than the acquisition of the camera data. Thus the robot's collision status could be rechecked several times for single images. Additionally the doubling of the runtime of a single process is a major problem of the robot control thread only, a thread whose execution time will be largely improved if this project's results are directly implemented into the robot's PLC and safety controller.

To get the performance of the complete system, consisting of the developed program, the 3D camera and the robot's PLC, the other components have to be added to the evaluation as well.

- The camera, acquiring 10 frames per second, can be assumed with a reaction time of 100 ms.
- The time to transfer the data via USB can be neglected because it is already included in the calculated program time of the camera control thread.
- The time to transfer the data via TCP/IP can also be neglected because it is already included in the calculated program time of the robot control thread.
- The reaction time of the robot's PLC, to stop the robot after a received command can be assumed at 20 ms after a command has been received.

Those points combine into an overall reaction time of 333 ms for the evasion path (see Figure 81) and 389 ms for the execution of the speed control (see Figure 82).

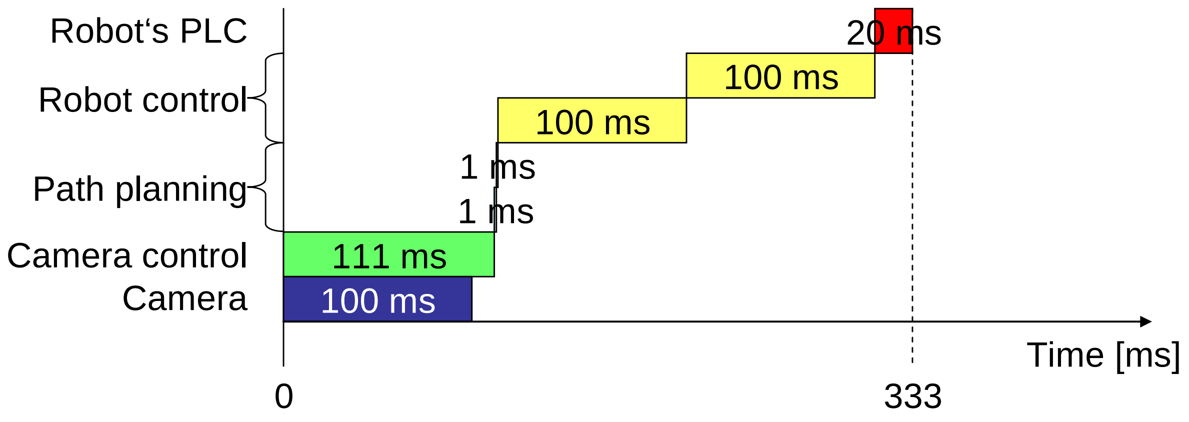


Figure 81: Overall system stopping performance when executing the evasion path

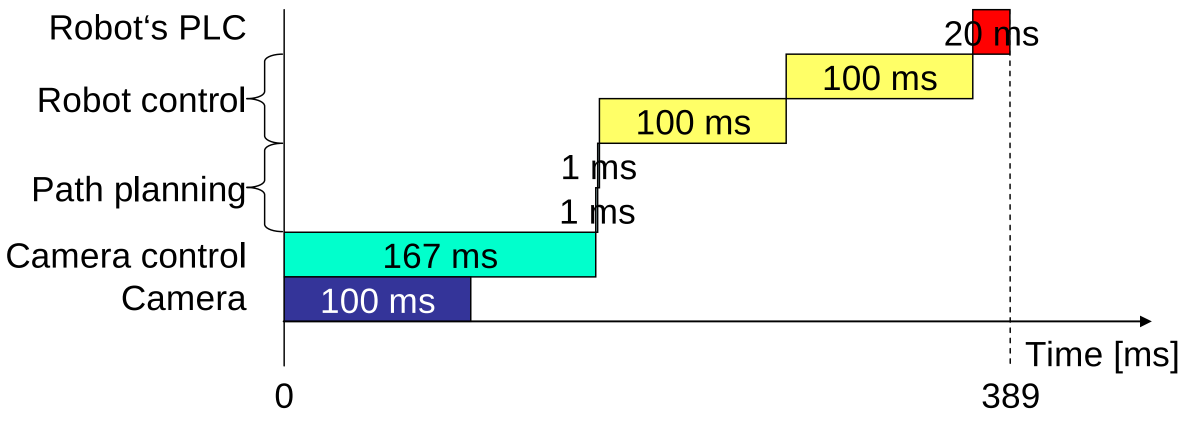


Figure 82: Overall system stopping performance when executing the speed control

From the acquired overall system stopping performance, the necessary safety distance can be computed with the help of EN 999 (see Equation 15, chapter 6.2.2).

For these computations, the position of the camera is assumed at 4.12 m above the floor. This value consists of a 4 m maximum detection range for the camera and an offset of 12 cm.

$$S = (1600mm/s * 333ms) + 48mm = 580.8mm$$

Equation 16: Minimum Distance from the detection zone [69], evasion path, camera mounted at 4.12 m

$$S = (1600mm/s * 389ms) + 48mm = 670.4mm$$

Equation 17: Minimum Distance from the detection zone [69], speed control, camera mounted at 4.12 m

These safety distances have to be kept between robot and intruding object. Since the robot is also moving, the distances have to be increased by the distance the robot can move in that time. The robot is restricted to a speed of 250 mm/s.

Thus another 83.25 mm have to be added to Equation 16, resulting in a safety distance of 664.05 mm.

The result of Equation 17 is increased by another 97.25 mm to a required distance of 767.65 mm.

The safety distance of 1100 mm, according to EN 13857 in chapter 6.2.1, has not to be added, according to EN 999. The EN 999 already takes into account the crawling below the lowest beam of the protective device. For the used device setup, it gives the recommendation for the height of the "lowest beam" of protective equipment to be at 300 mm. Thus the 120 mm, which were applied in accordance with EN 13857 are just an enhancement of the safety measures.

The result for the determination of the necessary safety distance is:

- 664.05 mm for the evasion path
- 767.65 mm for the speed control

The distance of 767.65 mm, determined in the speed control is farther than the distance covered by the assumed 14 lines at which the algorithm starts to react. Increasing the lines would reduce the reaction time further, thus demanding another increase and so on. It has to be kept in mind that this particular algorithm was designed with respect to its presentability and not its speed in execution. Further projects have to implement alternative algorithms, that focus more on their execution time.

Since those distances are not yet practical for demonstration, especially when the robot is only moving slowly, additional or alternative protective equipment was used, as proposed by the methodology of the standard EN 999. Thus, to be able to demonstrate the algorithms with shorter distances between robot and human, the hold to run button of the control panel was used in the course of this project as alternative protective equipment.

6.4 Desired safety

To be able to use the system in praxis, a safety distance of 500 mm, according to EN 349 [68] the safety distance against crushing of the human body, would be applicable. In this chapter it is shown, how short the reaction time of a system needs to be, in order to allow such a safety distance, while still operating the robot at full speed.

Equation 16 is expanded by the distance the robot can move during the reaction time of the system, resulting in Equation 18.

$$S_R = (K * T) + C + (V_R * T)$$

Equation 18: Safety distance of the robot, including the robot's velocity

In Equation 19 the values are inserted, including the maximum speed of the robot, 4 m/s. This speed is far greater than the safety speed (250 mm/s), used in this project.

$$\begin{aligned}
&500mm = (1600mm/s * T) + 48mm + (4000mm/s * T) \\
&\Leftrightarrow T = \frac{452mm}{5600mm/s} \\
&\Leftrightarrow T = 81ms
\end{aligned}$$

Equation 19: Desired reaction time of the complete system

The result is a required reaction time of 81 ms, which is even faster than the used camera.

6.5 Comparison of achieved and desired safety

Chapter 6.3 and 6.4 showed the difference of the safety distance that can already be achieved and the safety distance that would be practicable in industrial applications.

It has to be kept in mind that the hardware used in this project is not designed to be fast but to be flexible. It is reasonable to assume that hardware designed for the task can execute the desired functionality within a few milliseconds. The implementation of this hardware into the robot's control would allow the robot to respond within a few milliseconds as well.

Running the camera at 20 frames per second could then achieve the desired result from chapter 6.4.

If a closer collaboration of human and robot is desired, the speed of the robot could be reduced, as it was done in this project. Reducing it to a speed of 2 m/sec would result in a necessary reaction time of 125 ms (see Equation 20).

$$
\begin{aligned}
&500mm = \left(1600mm/s * T\right) + 48mm + \left(2000mm/s * T\right) \\
&\Leftrightarrow T = \frac{452mm}{3600mm/s} \\
&\Leftrightarrow T = 125ms
\end{aligned}
$$

Equation 20: Desired reaction time of the complete system, robot standing still

This reaction time of 125 ms is already within the possible boundaries of such a system.

7 Conclusion and Outlook

7.1 Conclusion

This work shows new ways towards a collaborative workplace, using a 3D camera and a PC program, to allow an industrial robot and humans to work alongside each other.

Two approaches towards this goal have been implemented in the course of the project. The first is a speed control based on proximity monitoring. The second is a path planning algorithm that plans a collision free trajectory of the robot arm, based on the location of dynamic objects, the position of the robot and its goal.

The resulting demonstrator, programmed during the course of this work, uses both approaches separately, demonstrating their functionality. The developed workplace has been successfully demonstrated during presentations at the BGIA and has proven to function reliably. While still tasks remain for further studies (see chapter 7.2), the stability of the system was seldom disturbed by outside influences. Only two specific pieces of clothing (a grey jeans and a black shirt) proved to be problematic to the developed algorithm because of their unexpected values from the 3D camera's measurement.

Due to a problem in the external control mode of the robot, which is caused by the robot's PLC, the robot tends to judder if goal changes occur in the direct drive mode. This has no influences on the functionality but only in the smoothness of the path execution in this mode.

The algorithms, the PC and the sensor equipment used in the demonstrative workplace have not been designed towards the necessary safety, but have only been implemented for this case study. The safety of the system at this stage is ensured by the application of a hold to run button.

If the complete project is implemented for industrial purpose and both approaches are combined, only the control of the robot's speed needs to be safe. The trajectory will only allow the robot to use its full potential. The advantage of this approach, other than the reduced demand on the safety of the planning of the trajectory, is that the algorithm of the trajectory planning can be changed afterwards without reevaluation of the safety of the whole system.

As chapter 6.3 showed, the system is not ready for industrial application at this point. In chapter 6.5 a conclusion was formulated from the observed performances, resulting in the observation that the designed system, implemented in special hardware, could be used to control a robot at 2 m/sec with a safety distance of 500 mm.

7.2 Outlook

During the course of this work, several points have been found that would improve the project and thus need further evaluation:

- The depiction of the running algorithms need to be reprogrammed in OpenGL. This would allow a three dimensional depiction of the computed results to the operator, without additional computational effort.
- The effort in the set-up of the system and the controlling of the developed program needs to be automated.
- The problems induced by badly reflecting material (see chapter 3.2) need to be dealt with. One way to do this would be to detect “holes” in the workplace and treat them as intrusions as well.
- The set-up of the transformation matrix (see chapter 5.5.2) could be automated, by giving the robot a reference block as a tool and using computer vision algorithms to find the position of the tool in the camera’s coordinate system. This would also allow the computation of a calibration matrix for the camera, because the positions delivered by the internal sensors of the robot are much more precise.
- If the transformation matrix is automatically set up, the position of the robot base’s centre can be calculated as well (compare chapter 4.3.5). Its radius could be extracted from known physical parameters of the robot.
- The distance between the robot and dynamic objects is currently calculated in the value “pixel” (see chapter 5.4). It needs to be advanced to show the real distance.
- An aspect important to the feeling of safety is the speed and velocity of the robot, when it is approaching a human. This aspect, shown in previous work [77], was thoroughly investigated by Thiemermann [12]. The speed control in this work is only based on proximity (see chapter 5.4) and not on the direction of the robot’s movement. Further projects should include this direction into the speed calculation.
- The height value of the robot’s tool centre used in the transformation of camera coordinates for path planning is fixed in the program (see chapter 5.5.2). It needs to be programmed as a flexible value.

- The increase in the safety distance, described in chapter 5.5.5, can be made dynamic, dependant on several factors:
    - The size of the robot’s current tool
    - The speed of the robot and of dynamic objects
    - The direction of the movement of robot and dynamic objects
    - The robot’s maximal physical dimension at that distance
- The problem of the robot’s juddering in direct drive mode (see chapter 7.1) needs either to be fixed by the robot’s manufacturer or a workaround using the point to point mode needs to be programmed.
- For the application in praxis, the safety controller of the robot would have to be included in the process of object avoidance.

  The basic function, necessary for the safety of the desired approach of a flexible fence, is already implemented by the safety trips of REIS [48]. They act as a virtual wall, through which the robot is not allowed to move. This could be used to secure a safe execution of the commands of the PC. Today those safety trips, in accordance with EN 10218-1 [70], can not be changed during the runtime of the control but only during its start-up phase, thus building a rigid fence.
- The application of human models to the detected dynamic intrusions would allow the algorithms to predict occlusion of the workspace in the future. This would enable the path planning algorithm to compute more optimal trajectories. The proximity monitoring, which needs to be safe, can not profit from predictions.
- Previous work discussed the usage of splines to smoothen the robot’s trajectory. Field tests in this work showed that those splines were not helpful, due to the slow movement of the robot and the problems with a smooth driving (juddering – see above). Later versions of this project should study this subject further, after the removal of the juddering.

8 Bibliography

[1] "First human killed by robot",
United Press International Dispatch, July, 1982

[2] Fault free analysis of hazards created by robots,
Proceedings 13th International Symposium on Industrial Robotics and Robot (1983)
N. Sugimoto and K. Kawaguchi,

8.1 Paper on human robot interaction

[3] A sensory-based robotic safety system,
Control Theory and Applications, IEE Proceedings (1985),
J.H. Graham, J.F. Meagher; Rensselaer Polytechnic Institute, NY, USA

[4] A Safety and Collision Avoidance System for Industrial Robots,
IEEE Transactions on Industry Applications (1986),
J.H. Graham, J.F. Meagher, Stephen J. Derby; Rensselaer Polytechnic Institute, NY, USA

[5] Controller Design for Human-Robot Interaction,
http://www.cs.rpi.edu/research/pdf/07-10.pdf (2007)
Eric Meisner, Volkan Isler, Jeff Trinkle; Rensselaer Polytechnic Institute, NY, USA

[6] An approach to collision detection and recovery motion in industrial robot,
Industrial Electronics Society, (IECON '89) Shinji Takakura, Toshiyuki Murakami and Kouhei Ohnishi, 1989

[7] A failure-to-safety robot system for human-robot coexistence,
Robotics and Autonomous Systems, Volume 18, Yoji Yamadaa, Kazutsugu Suitaa, Koji Imaia, Hiroyasu Ikedab and Noboru Sugimoto, 1999

[8] Collision Avoidance and Handling for a Mobile Manipulator,
MORPHA, Thomas Wösch and Werner Neubauer, 2002

[9] rob@work: Robot Assistant in Industrial Environments
MORPHA, E. Helms, R. D. Schraft and M. Hägele, 2002

[10] Safe Human-Robot-Cooperation: Image-based collision detection for Industrial Robots,
IEEE/RSJ International Conference on Intelligent Robots and System, Dirk M. Ebert and Dominik D. Henrich, 2002

[11] Toward Safe Human-Robot Co-Operation in Manufacturing,
Advances in Human Robot Interaction p. 255ff. (MORPHA), ISBN 978-3-540-23211-7, Springer, Berlin, Andreas Stopp, Thomas Baldauf, Sven Horstmann and Steen Kristensen, 2004

[12] Direkte Mensch-Roboter-Kooperation in der Kleinmontage mit einem SCARA-Roboter,
Dipl.-Ing. Stefan Thiemermann, 2004

[13] Multi-camera Collision Detection between Known and Unknown Objects,
2nd ACM/IEEE International Conference on Distributed Smart Cameras – ICDSC 2008, Sept 7–11, Stanford/USA, D. Henrich, T. Gecks, 2008

[14] Markerless human tracking for industrial environments,
2008 Canadian Conference on Electrical and Computer Engineering,
M. Elshafie, G. M. Bone; Dept of Mech Eng, McMaster Univ, Hamilton, ON, Canada

[15] Image-based 3D-surveillance in man-robot-cooperation,
2004 2nd IEEE International Conference on Industrial Informatics, Berlin, Germany,
J. Kruger, B. Nickolay, O. Schulz; Fraunhofer Inst for Production Syst & Design Technol, Berlin, Germany

[16] The DLR lightweight robot: Design and control concepts for robots in human environments,
The Industrial Robot Band 34 (2007) Heft 5, Seite 376-385,

A. Albu-Schäffer, S. Haddadin, C Ott, A Stemmer, T Wimböck, G Hirzinger; Deutsches Zentrum für Luft- und Raumfahrt (DLR), Germany

[17] Weighted path planning based on collision detection,
The Industrial Robot Band 32 (2005) Heft 6,
Lu-Shujun, H. Chung-Jae; Stevens Institute of Technology, Hoboken, US

[18] The model of the video safety system for industrial robots,
ISMCR, International Symposium on Measurement and Control in Robotics, 4 (1995),
A. Zycki; Central Institute for Labour Protection, Warsaw, PL

[19] Safe human-robot cooperation for robots in the low payload range,
ISR, International Symposium on Robotics, 35 (2004),
P. Heiligensetzer; KUKA Roboter, Augsburg, Germany

[20] Safe human-robot cooperation in industrial production,
VDI-Berichte Band 1679 (2002),
P. Heiligensetzer; H. Wörn; KUKA Roboter, Augsburg, Germany; Institute for Process Control & Robotics (IPR), Universität Karlsruhe (TH), Germany

[21] Safeguarded human-robot interaction in production,
ISR, International Symposium on Robotics, 33 (2002),
P. Heiligensetzer; H. Wörn; KUKA Roboter, Augsburg, Germany; Institute for Process Control & Robotics (IPR), Universität Karlsruhe (TH), Germany

[22] Applying reflexes to enhance safe human-robot-co-operation with a humanlike robot arm,
ISR, International Symposium on Robotics, 35 (2004),
Sadi Yigit, Catherine Burgart, Heinz Wörn; Universität Karlsruhe (TH), Germany

[23] Visual surveillance system for detection of moving objects by scene modelization in uncontrolled robotic environments,
Acquisition, Tracking, and Pointing XI, Orlando, FL, USA (1997)
A. Abdallah; Dept Productique, Ecole des Mines de Douai, France

[24] Merging of range images for inspection or safety applications,
Two- and Three-Dimensional Methods for Inspection and Metrology VI, San Diego, CA, USA, (2008),
J. Mure-Dubois, H. Hugli; Inst of Microtechnol, Univ of Neuchatel, Neuchatel, Switzerland

[25] Joint-action for humans and industrial robots for assembly tasks,
2008 RO-MAN: The 17th IEEE International Symposium on Robot and Human Interactive Communication, Munich, Germany,
C. Lenz, S. Nair, M. Rickert, A. Knoll, W. Rosel, J. Gast, A. Bannat, F. Wallhoff; Dept of Comput Sci, Robot & Embedded Syst Lab, Munich, Germany

[26] Virtual force/tactile sensors for interactive machines using the user's biological signals,
Advanced Robotics Band 22 (2008) Heft 8,
Tamei-Tomoya, Ishii-Shin, Shibata-Tomohiro; Nara Institute of Science and Technology (NAIST), Ikoma, JP; Kyoto University, Uji, JP

[27] A safety concept for robot assistants in manufacturing,
Automatisierungstechnische Praxis - atp Band 47 (2005) Heft 2,
Andreas Stopp, Thomas Baldauf, Sven Horstmann, Steen Kristensen; DaimlerChrysler, Berlin, Germany

[28] Development of a stereo vision system for industrial robots,
International Conference on Control, Automation and Systems, 2007. ICCAS '07,
Oh Jong-Kyu, Lee Chan-Ho; Hyundai Heavy Ind. Co. Ltd., Ulsan

[29] Robot introduction in Human work environment. Developments, Challenges and Solutions,
IEEE International Conference on Computational Cybernetics, 2007. ICCC 2007.
O. Ogorodnikova; Budapest Univ. of Technol. & Econ., Budapest

8.2 Paper on path planning algorithms with industrial joint arm robots

[30] A fast path-planning algorithm for industrial robots
IEEE International Conference on Control and Applications, B. Fink, H.-D. Wend, 1989

[31] Path Planning for Industrial Robot arms - A Parallel Randomized Approach
Caigong Qin, Dominik Henrich, 1998

[32] Automated Generated Collision-Free Time Optimized Robot Movements in Industrial Environments Based on Rounding
Proceedings of the IEEE International Symposium on Assembly and Task Planning, Björn Hein, Marcos Salonia, Heinz Wörn, 2001

[33] Dynamical Systems: A Framework for Man Machine Interaction
MORPHA, Ioannis Iossifidis and Axel Steinhage, 2001

[34] Control of an 8 dof manipulator by means of neural fields
MORPHA, Ioannis Iossifidis and Axel Steinhage, 2002

[35] Implementation of a path-planning algorithm for a robot arm
MORPHA, Rohrmoser, B. and Parlitz, C., 2002

8.3 Paper on filter algorithms

[36] A New Approach to Linear Filtering and Prediction Problems
Transactions of the ASME--Journal of Basic Engineering, Kalman, Rudolph Emil, 1960

8.4 Paper on XML commands used to control robots

[37] Programmieren durch Vormachen für Assistenzsysteme – Schweiß- und Klebebahnen intuitive programmieren
it Information Technology 04/2007, C. Meyer, R. Hollmann, C. Parlitz, M. Hägele, 2007

[38] Entwicklungstendenzen in der Industrierobotik
VDI/VDE-GMA Fachausschuss, H. Wörn, 2006

[39] ARIKT:Adaptive Robot Based Visual Inspection
Künstliche Inteligenz 02/03, H. Wörn, T. Längle, M. Gauß, 2003

[40] The need for an intuitive teaching method for small and medium enterprises
SMErobot™ - The European Robot Initiative, R. D. Schraft, C. Meyer, 2006

[41] Intermediate Language for Mobile Robots
VTT Publications 510, Ilkka Kauppi, 2003

[42] An XML-based Scripting Language for Chain-type Modular Robotic Systems
cs.princeton.edu, Ying Zhang, Alex Golovinsky, Mark Yim, Craig Eldershaw, 2008

[43] Human-Robot Interface Using Agents Communicating in an XML-Based Markup Language
CityU niversity of Hong Kong, M. Makatchev, S. K. Tso, 2000

[44] Adaptive Robot Based Visual Inspection of Complex Parts
ISR 2003, M. Gauss, A. Buerkle, T. Laengle, H. Woern, J. Stelter, S. Ruhmkorf, R. Middelmann, 2003

8.5 Internet

[45] Phriends — Physical Human-Robot Interaction: Dependability and Safety --
http://www.phriends.eu/publications.htm
Project time: October 2006 — September 2009

[46] OSHA Instruction PUB 8-1.3 SEP 21, 1987 –
http://www.osha.gov/pls/oshaweb/owadisp.show_document?p_table=DIRECTIVES&p_id=1703

[47] Pilz SafetyEye – http://www.pilz.de/company/press/messages/sub/products/articles/00951/index.en.jsp;jsessionid=23A0CF01D72922E37E3A55A78F8A36D9

[48] Robots without separating safety equipment – http://www.mav-online.de/O/121/Y/82154/VI/10014106/default.aspx

[49] “Safe” & Cooperative Robots – http://www.autofieldguide.com/articles/060604.html

[50] MORPHA – http://www.morpha.de/php_e/index.php3

[51] SIMERO – http://www.ai3.uni-bayreuth.de/projects/simero/index.php

[52] Reis RV30-16 -- http://www.reisrobotics.de/ROBOTER/ROBOTER/Vertikalknickarm+%28RV%29/RV30_16.html

[53] The KUKA lightweight robot -- http://www.kuka.com/en/pressevents/news/NN_060515_Automatica_02.htm

[54] Reis Safety Trips, tested by the BGIA
http://www.dguv.de/bgia/de/pub/ada/pdf/abia0101.pdf

[55] Mesa Imaging AG -- http://www.mesa-imaging.ch/

[56] Robotersteuerung für sichere Mensch-Roboter-Kooperation, F. Som
http://www.bg-metall.de/fileadmin/downloads/FA_MFS/Symposien/Praesentation_Reis.pdf

[57] Technische Vorraussetzungen zur Mensch-Roboter Kooperation, P. Heiligensetzer
http://www.bg-metall.de/fileadmin/downloads/FA_MFS/Symposien/Praesentation__KUKA.pdf

[58] IFR Statistical Department, World Robotics 2008
http://www.worldrobotics.org/downloads/2008_executive_summary.pdf

[59] VDMA – Verband deutscher Maschinen- und Anlagenhersteller
http://www.vdma.org/wps/portal/Home/de

[60] Datenbank BG-PRÜFZERT – tested products
http://www.hvbg-service.de/pruefz.tpl/produkt.htm

8.6 Books

[61] Höhere Mathematik 1, 5. Auflage
K. Meyberg, P. Vachenhauer, Springer-Verlag Berlin Heidelberg 1999
ISBN 3-540-66148-4

[62] Machine Vision Theory Algorithms Practicalities Third Edition
E. R. Davis, Morgan Kaufman Publishers, 2005

[63] Parallele Bildverarbeitung
T. Braunl, S Feyer, W. Rapf, M. Reinhardt, Addison-Wesley, Bonn, 1995

8.7 Lecture Slides

[64] Prof. Dr. Erwin Prassler, FH Bonn-Rhein-Sieg, Robot Manipulation SS07 -- https://www5.inf.fh-bonn-rhein-sieg.de/intern/Prassler/RobotManipulation/Slides/RM_2007_Chap1.pdf

8.8 Laws and Regulations

[65] EC Machinery Directive 98/37/EC
DIRECTIVE 98/37/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 22 June 1998 on the approximation of the laws of the Member States relating to machinery

[66] EC Machinery Directive 2006/42/EC
DIRECTIVE 2006/42/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 17 May 2006 on machinery, and amending Directive 95/16/EC

[67] The new EC Machinery Directive 2006
ISBN 978-3-410-16310-7, Beuth, Berlin, Thomas Klindt, Thomas Kraus, Dirk von Locquenghien, Hans-J. Ostermann, 2007

[68] EN 349:1993 + A1:2008
Safety of machinery — Minimum gaps to avoid crushing of parts of the human body

[69] EN 999:1998
Safety of machinery — The positioning of protective equipment in respect of approach speeds of parts of the human body

[70] EN ISO 10218-1:2006
Robots for Industrial Environment — Safety requirements — Part 1: Robot

[71] EN ISO 12100-1:2003
Safety of machinery — Basic concepts, general principles for design — Part 1: Basic terminology, methodology

[72] EN ISO 13849-1:2008
Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design

[73] EN ISO 13857:2008
Safety of machinery — Safety distances to prevent hazard zones being reached by upper and lower limbs

[74] EN ISO 14121-1:2007
Safety of machinery — Risk assessment — Part 1: Principles

8.9 Internal Documents

[75] RSV-Command- XML-Kommunikationsschnittstelle
Dokumentation REIS GmbH & Co Maschinenfabrik Obernburg, A. Brunn, 2006

[76] Externes Verfahren
Dokumentation REIS GmbH & Co Maschinenfabrik Obernburg, ESS, 2006

[77] Development of a prototype of an autonomous motion control for assisting industrial robots
Research and Development Project 1, Master Autonomous Systems at the FH Bonn-Rhein-Sieg, B. Ostermann, 2007

[78] Infrastructure between autonomous motion controls and robot simulating software
Research and Development Project 2, Master Autonomous Systems at the FH Bonn-Rhein-Sieg, B. Ostermann, 2008

[79] Development of a Software tool to automatically evaluate 3D Time of Flight Distance measurement cameras
Research and Development Project 1, Master Autonomous Systems at the FH Bonn-Rhein-Sieg, J. v. d. Lippe, 2007

[80] Reliability Tests of 3D Camera for safety applications
Research and Development Project 2, Master Autonomous Systems at the FH Bonn-Rhein-Sieg, J. v. d. Lippe, 2008

[81] Person Detection and Tracking based on Range Data for Safety Applications
Master Thesis, Master Autonomous Systems at the FH Bonn-Rhein-Sieg, J. v. d. Lippe, 2008

9 List of Figures

10 List of Equations