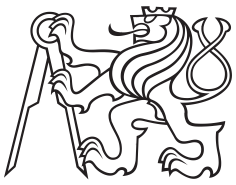
Master Thesis



Czech Technical University in Prague
in cooperation with
Aschaffenburg University of Applied Sciences

F3

Faculty of Electrical Engineering
Department of Control Engineering

Vision-guided handling operations for assistive robots in un-structured environments

Bc. Zuzana Kožnarová

Supervisors: Prof. Dr.-Ing. Hartmut Bruhm,
doc. Ing. Tomáš Svoboda, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: Cybernetics and Robotics
May 2019

Acknowledgements

I would like to express my gratitude to my supervisors Prof. Dr.-Ing. Hartmut Bruhm and doc. Ing. Tomáš Svoboda Ph.D. for relevant feedback to my master thesis. I would like to thank Prof. RNDr. Olga Štěpánková, Ph.D. and Ing. Petr Novák, Ph.D. for the course assistive technology (A6M33AST). Many thanks belong to my family that supports me not only during the last five years at the university, but also in the whole of my life. I would like to thank my partner Jakub Havlíček and my friends for their support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Abstract

The theoretical part of the thesis concentrates on the topic of assistive robots in health and social care and illustrates possible applications for mobile manipulators in the given area. In the practical part, it processes the proposal of an assistive robot in the form of a vision-guided mobile manipulator. The implementation is simplified on the pick and place task of a detected object by the camera using the Katana 400 manipulator written in ROS.

Keywords: assistive technology, Katana 400, MPS-450, mobile manipulator, robot-camera calibration, Vision-guided pick and place task

Supervisors:
Prof. Dr.-Ing. Hartmut Bruhm,
doc. Ing. Tomáš Svoboda, Ph.D.

Abstrakt

Teoretická část práce se zabývá tématikou assistivních robotů ve zdravotnické a sociální péči a ilustruje možné aplikace pro mobilní manipulátory v dané oblasti. V praktické části zpracovává návrh asistivního robotu ve formě mobilního manipulátoru řízeného pomocí kamerou detekované sceny. Realizace je zjednodušená na pick and place task kamerou rozpoznaneho objektu pomocí manipulátoru Katana 400 vypracovaná v ROSu.

Klíčová slova: asistivní technologie, Katana 400, MPS-450, mobilní manipulátor, robot-kamera kalibrace, Vision-guided pick and place úloha

Překlad názvu: Vision-guided manipulační operace pro assistivní roboty v ne-strukturovaných prostředích

Contents

Figures

Tables

Chapter 1

Introduction

Our motivation for interest in the topic of assistive robots is that they can help to mitigate the problem related to shortage of healthcare and social workers(2.1). Specifically, we will deal with mobile manipulators[1]. During development of an assistive robot, technically interesting questions such as object detection, calibration, trajectory and grasping planing are offered. In the same time, there are also social questions such as designing of the system and choosing of the technologies to meet the needs of different users, with different technical skills.

The goal of this work is to design a hardware and software proposal of the assistive mobile manipulator (section: 3.7, 2.3.3 and 2.2.3) and demonstrate the solution on the pick place task (as general as possible) of a subject applicable in healthcare using the Katana manipulator (section: 5.2).

We describe used hardware (sections B.5, 3.4) and we will explain our choice of the sensing system (3.2) for the mobile manipulator.

The results of studying the literature for the bin-picking problem (3.1) and related vision and robotics works and considering of all the above-mentioned topics are presented as a comprehensive proposal of the overall EtaKatana project with the hardware part (section 2.2) and the software part with a flowchart of daily EtaKatana functions (section 3.7). The proposal for the communication between the user and EtaKatana is presented, because it is a

[1]The term assistive robot should here-in-after be understood as assistive mobile manipulator.

significant part in assistive technologies to create a user-centred design for the people with almost no experience with computers (section 2.2.3).

A 3D model of EtaKatana hardware robot was created for more natural image and also for more comfortable fitting of the parameters (see 2.5). Which parameters can be easily changed and how they can be changed is available in B.5.

After the printing the calibration ball with holding system (section B.4) and manufacturing of the gallows for the Kinect (section B.6), it was started with development of the recognition system, calibration and pick and place task. Each of these tasks covers many complications, such as different size and shape and placement of various objects. Therefore this thesis focuses on implementing the most critical parts of the concept to be able to bring an example of a solution that can be used in assistive technologies.

It was decided to use three different methods for the object detection (4.1). Hough circle transformation (4.1.1) for detection of the calibration ball and a can. Then Deep neural network (DNN) YOLOv2 (4.1.2) was used to detect general objects. The general objects can be already predefined and the DNN can be trained on datasets containing general objects classes. The possibility of recognizing general objects without adding data by user makes this application usable from the first day. An assistive robot must allow easy personalising. This personalising is possible through user-defined objects that are detected by a method based on feature detection (4.1.3).

The pick and place task is a combination of using detected vision data (see 4.1), calculated transfer matrix (see 4.2) and calculation of the robot trajectory and inverse kinematics concerning the robot joints and environmental limits. Algorithms used for implementation of these tasks are described in chapter 5. The task work as the grasping position is determined by vision, and the user chooses placement position by writing the coordinates in the camera coordinate system in pixels to the console. The name of the object that will be grasped is written as the first argument. The name of the object is then published by the robot part and subscribed by the recognition part.

The practical information related to connecting of the hardware, to implementation and launching of the programs is available in chapter B.

Chapter 2

Requirements analysis and definition of exemplary application scenarios

2.1 Motivation

Our motivation was year-by-year longer life expectancy [4] that brings higher demands on social and healthcare systems of developed countries, where life expectancy is the highest. However the trend of increasing life expectancy is strongest in Africa, where the life expectancy increased by 10.3 years between 2000-2016 [4], which means, this is a global problem.

Moreover, civilisation diseases like Parkinson and Alzheimer disease come with higher life expectancy. The patients with these illnesses need daily care. Furthermore, according to the demographic development, a higher percentage of people will be in post-productive age.

The second important motivation for us was the problem of lacking medical staff. According to the World Health Organization (WHO)), 12.9 million healthcare workers will be missed in 2035. In 2013, 7.2 million health workers were missed[5]. Moreover, there is also a problem of uneven geographical distribution of healthcare workers.

The third problem is highly infectious illnesses. If robots can at least reduce the medical staff´s contact with a highly infectious person, it will be a big step further to make work in hospitals safer[6].

The problems mentioned above can be improved by increasing assistive technologies in healthcare. One of the main goals of assistive technology is to increase independence and related quality of life, as it is expressed in "*Assistive Technology is any tool, device, software or system that uses usually modern technologies (especially sensors, actuators, information and communication technology) to strengthen, maintain or improve the functional abilities of individuals with special needs, thereby facilitating their daily lives and improving their quality of life, independence and self-sufficiency. Persons with special needs are the elderly, the disabled and chronically ill people.*"[7].

However, there are also some complications related to robotic assistive technologies in healthcare system, mostly there is a problem of estrangement and loss of communication with people, which can be a risk especially for elderly people. It is crucial to decrease the repetitive or physically demanding work assignments by using robots and to spend saved time on more creative activities with the ill people. It is also essential to make the design to be user-friendly not only for healthcare workers but also for patients. Patients need to be able to set a task to the robot themselves. Another topic to consider is the possibility of building a friendly relationship between the user and the robot, which would help to alleviate loneliness of patients, users and senior citizens.

2.2 Vision of the comprehensive solution

2.2.1 Use cases

In this section possible scenarios for the assistive robots will be discussed.

1. There will be a need for robots to be able to find and bring drinks, medicaments or food. In this way, robots will help ill people in hospitals and elderly people with in-home care or in homes with nursing care.

2. Another appropriate task is to do exercise with the user. It means that the robot will show the patient an exercise. Then the system with camera recognition will check if the patient does the exercise properly. Otherwise, it will advise him and show the exercise again. A good recognition system will be needed to put this idea in practice. Tracking of the movement is possible, an example of a possible solution is ROS library for Kinect openni_tracker that perfectly fit our concept.

3. Support when standing up, or transferring the patient (an example is ROBEAR[8], a successor of RIBA-II that is now further developed in Meijo University). This task is also important for hospital personnel. However Katana is not strong enough (Katana can only raise to 0.4 kg heavy objects), moreover the shape of Katana is not ideal for this application. For that reason, a decision was made not to advance with this task.

A decision was made to concentrate on manual tasks with light objects that were described in the first point 1.

2.2.2 Movement of the robot

The criteria for movement of the robot are: easy orientation in space, easy use at homes and in hospitals, which includes robot size requirements. The possible solutions can be as follows:

1. It can be used as a vehicle that will be allowed to go anywhere. Moreover, it will need to solve simultaneous planning, localisation, and mapping.
2. Otherwise the robot can move in a prescribed area defined by the black line.
3. Last but not least- the robot can only move along a route defined by a black line.

The solutions consisting in moving anywhere and within a prescribed area make sense for in-home care or at homes with nursing care. On the other hand, the predefined routes make more sense in hospitals because there are tighter conditions for permissions of entry. The rooms are also designed similarly, and it is necessary that the robot does not crash into anything and that it does not interfere with hospital staff or patients.

2.2.3 User control

At this point, it is needed to first think about the user for whom the interface of the robotic system is designed. It is not only for the hospital personnel

but also for the patient and mostly for senior citizens so our goal is to design it according to User centered design (UCD) described in the standard ISO 9241-210:2010. The actual problem is how the robot can communicate with the users. There are three possible basic solutions available.

1. First possibility is communication using a spoken language.
2. Second possibility is to use a text.
3. Third possibility is to use pictures.

Spoken language can appear at first sight to be the most useful option. However, sometimes it can bring difficulties because, for example, if a doctor needs to bring a particular object it is hard to describe it to the robot well enough. Moreover, strong Artificial intelligence (AI) is needed to understand the context. Therefore it is planned to use spoken words only for some basic instructions or for navigation to an object that has already been stored in the database. The preferred solution is a combination of pictograms of daily used things with a database of pictures where each user can add a new picture 2.2) of an object that can be then found. The picture will be added to the database for the user´s account, and the user will type or say the name of the object. Then he or she will be able to send the robot for the object using a click on the chosen picture(2.1) or by confirming a particular object selection with spoken word.

The picture can be found in the database by typing the picture name or moving in the database by clicking on left or right arrows (see figure 2.1). To stand by UCD, the linear layout was used for pictograms as well as for pictures, both of them will be shown on screen. . It is recommended to work on the project on the multiplatform basis of the application, so users do not need to adapt to a new device. To avoid the obstacle of poor eyesight, it is recommended to use tablets with at least 8.5" display.

2.3 Mobile manipulators

The mobile manipulator has a basic characteristic that it is an industrial manipulator which is located on a vehicle where there is also a vision system. The advantage of EtaKatana is that it is based on Robot Operating System (ROS)which means that any part of the system can be easily replaced with a new one. First, let us take a short look back at the history of this project.

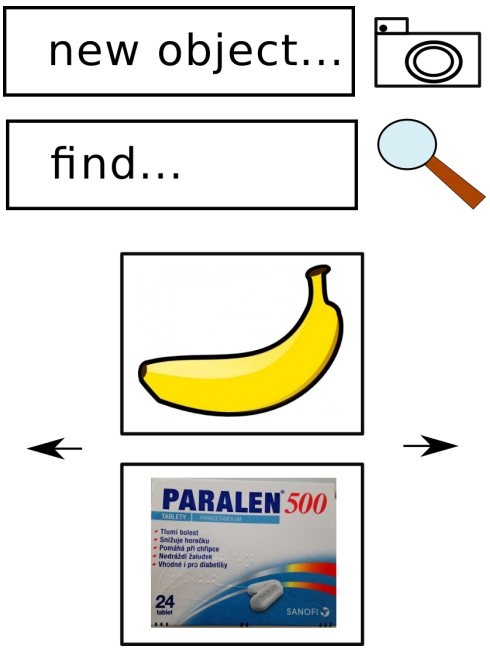


Figure 2.1: Proposal for the main screen layout of the application

Stefan Neumann discovered the first concept (see figure 2.3) and the Katana robot was bought on the basis of it. In the concept it can be seen that it has one big disadvantage, higher positions such as tables.

2.3.1 Calvin robot

Another example of a mobile manipulator is robot Calvin[9] that can reach the table but not the ground because it has a fixed frame that sets the robot in the height of the table (see figure 2.4a). This solution is not suitable for us either.

2.3.2 Robotnik solution

Mobile manipulator RB-1[10] (see figure 2.4b) has 13 degrees of freedom and is focused on developing indoor solutions. This robot has an advantage because it does not need to have a big size as Robot Assistant for Nurses (RAN)[11]], which can be highly practical indoor to be able to fulfil a task in a different height above the surface. The hight of the robot can be changed between 1,029 m and 1.379 m.

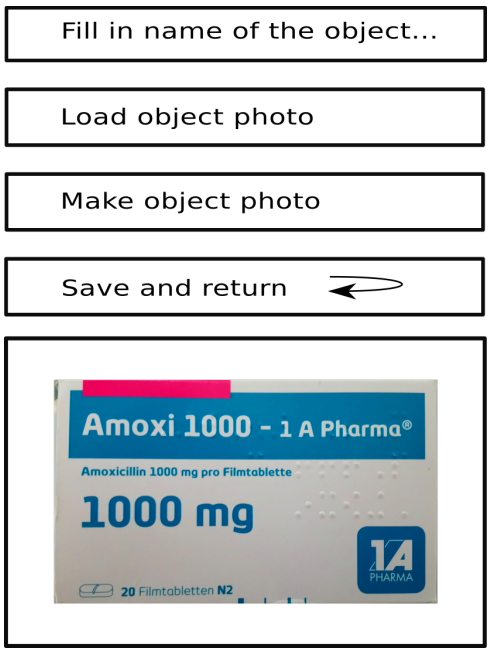


Figure 2.2: Proposal of the screen add new photo

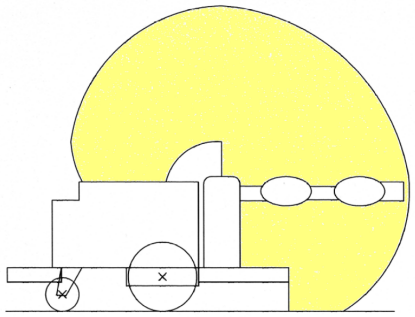


Figure 2.3: The first concept of mounting Katana on Roboter platform MP-400[1]

(a) : Calvin solution (source: [9]) from Osnabrück university

(b) : Robotnik solution RB-1 (source: [10])

Figure 2.4: Example of similar solutions

2.3.3 3D model of hardware proposal for the project

It was decided to use an alternative approach in comparison with Stefan Neumann's solution [1]. The robot will be put on additional construction that can move in vertical direction (see figure: 2.5), which means that one vertical translation joint will be added. The relatively small size of the robot will thus be compensated, and the robot will get a considerably bigger work envelope, which includes possibilities to take things for example from a shelf, tables of different heights, kitchen unit and bedside table.

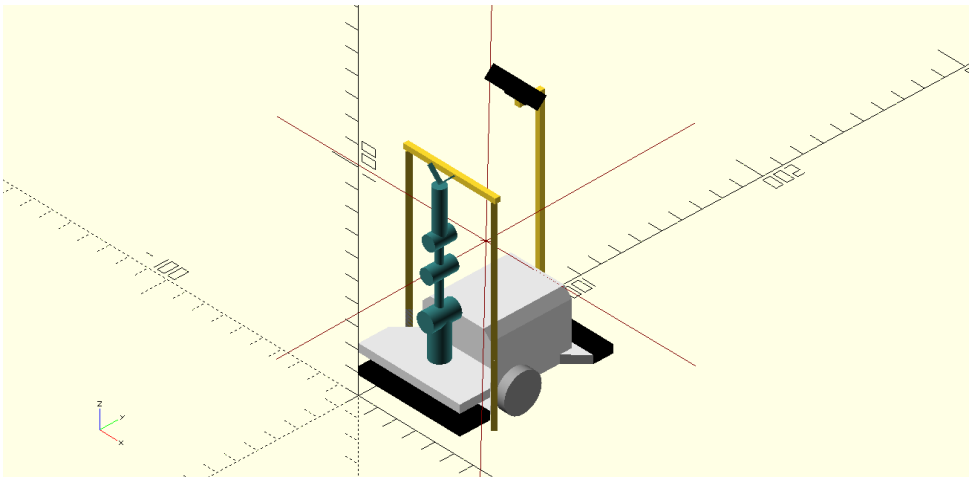


Figure 2.5: EtaKatana solution with additional vertical joint

The following 3D model (see figure 2.5) has the primary function to comfortable fit the parameters of the gallows for the camera and to find a useful range of heights for the additional joint.

There is a possibility to easily fit the length of vertical and horizontal bar of the gallows by changing two parameters in the script (more in B.5). This is good for visualising how the Field of view (FoV) changes (2.6).

The simplified working envelope can estimate which places the robot can reach. For a more exact solution of the robot moving dificulty in a given position, it would be possible to calculate the dexterity ($\frac{1}{\mathrm{cond}(J)}$ where cond(J) is a conditionality of robots Jacobian). The simplified working envelope helps to fit the range of the additional joint. The approximate working envelope is marked by a transparent blue area (2.7).

Simplification of the work envelope is done by neglecting the stops of the

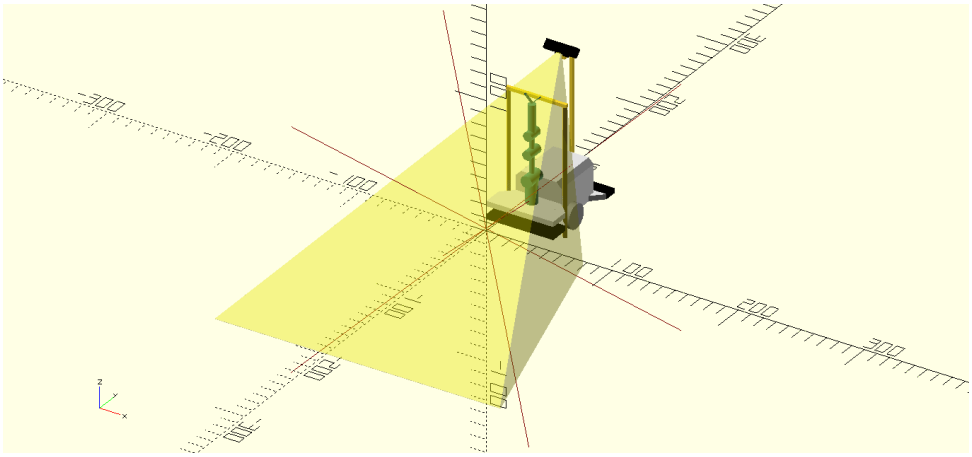


Figure 2.6: EtaKatana solution with FoV marked by transparent yellow colour

joints and by fixing the third, fourth and fifth joint.

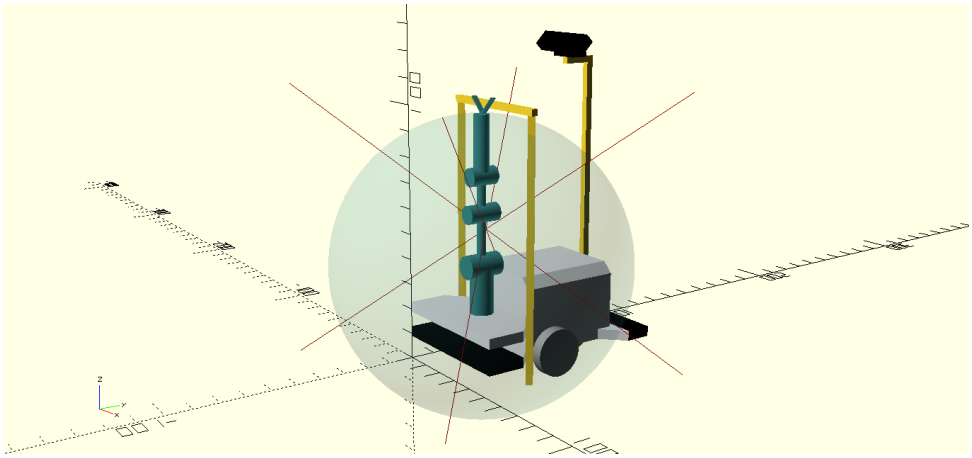


Figure 2.7: EtaKatana solution with simplified working envelope marked by transparent green colour

Chapter 3

Literature research and concept development

3.1 Bin-picking problem

The Bin-picking problem is for decades a huge topic at the edge between robotics and machine vision. It can be shortly described as grasping an object of unknown shape from a box with more objects inside. That can sound pretty easy for people. However, it is much harder for a machine. An universal and easily applicable solution has not yet been found for general objects. Nowadays, the bin picking problem is most commonly simplified in the industry using a known 3D shape of the searched object with an off-line predefined set of grasping positions. Grasping position defines how the object will be grasped by the gripper. Moreover, each gripper can be more useful for different applications. That brings up the question of how to choose a gripper suitable for the bin picking problem and at the same time a gripper that is not as complex as a human hand.

The bin-picking problem is defined as a problem on the border of the vision and robotics. It can seem a little surprising, but the vision is fundamental in the bin-picking task because the robot needs to find the object between the other objects. Moreover, the robot needs to have an overview of the surroundings. An environmental overview is important for collision avoidance and is useful also for choosing the grasping position and planing the trajectory to get to the searched object and get back with the object.

There are also some easier, however not simple, problems which are the control of the robot and preparing a usable user interface. The last but not least problem consists in cooperation between the camera and the robot (camera calibration is one of the subproblems for example).

The bin picking task is usually solved in a simplified version, because of all of the above-mentioned complex problems. Alternatively, the researchers focus more closely on one of the subproblems, which they deal with in more detail[12].

3.2 Sensory system

Sensory systems and work with them are a significant challenge within finding a solution to the bin picking problem because the sensory system provides information for the object recognition and localisation in its environment.

3.2.1 Requirements

In order to choose a suitable sensory system for our task, it is needed to set the main criteria for daily use of EtaKatana. These are FoV, to get a wide view in a short time and type of 3D solution, which is not too time-consuming. The 3D solution is highly time-dependent either because of calculating the 3D scene or getting and finding the correspondences in camera images. Medium important requirements are the developer´s base of software for used devices and presumed stability of the manufacturer and product in the market. Other parameters important for choosing the camera were resolution, weight and price.

3.2.2 Possibilities

The possibilities are to use Xbox Kinect v2 [13],[14], LifeCam HD-5000[15] or Axis 214 network camera[16] (see table 3.1 and figure 3.1).

	Xbox kinect v2	LifeCam HD-5000	Axis 214
Resolution	1920 x 1080 (512 x 424 depth)	704x576	1280 X 720
3D solution	internal Infrared (IR)	external (gripper)	external (none suitable)
Proportion [mm] (L x W x D)	249 x 66 x 67	37.8 x 40.8 x 109	165 x 151 x 157
Weight [g]	1400	96.5	1110
Price [$][1]	99.95	48.99	399.00
Horizontal FoV	70.0°	35.3°	48.0°
Vertical FoV	60.0°	18.4°	26.5°

Table 3.1: Comparison of camera features for choosing the visual system

(a) : Axis 214[a]

[a]source: [16]

(b) : LifeCam HD-5000

(c) : Xbox kinect v2

Figure 3.1: Available cameras

3.2.3 Camera type decision

Working with a 3D scene is highly needed for our task because the gripper needs to take things that have an unknown shape (In the industry, the bin-picking problem can also be solved with 2D vision, when 3D models of an object that needs to be grasped are available[12]). The 3D scene will also

help in further development, when the robot moves in the environment using the robotic platform.

Due to the above-mentioned reasons, it is essential to get a 3D picture. When a webcam is used, it will be necessary to mount it on the gripper. That brings the advantage that the robot can quickly look everywhere around itself. On the other hand, the camera on gripper is not practical, because in the bin-picking problem one of the goals is to avoid collisions and take an object from a tangle of other objects. The camera can obstruct the picking, and more complex trajectories might be required, or it will be needed to relocate other objects to get to the searched object. Fortunately, the camera is of relatively small dimensions, as can be read in table 3.1. To get a good quality picture with good location during the moving of the gripper will be at least complicated. Speed of motion needs to be regulated on a low level not to get a blurry photo. That was the reason for not choosing the LifeCam HD-5000, despite the low weight and price. Moreover, with the camera mounted near the gripper, the weight which the object taken by Katana can have reduces.

Axis 214 has an advantage in having twenty preset positions and eighteen times optical zoom[16]. However, to get a 3D scene it will be necessary to move with the vehicle because the camera Axis 214 cannot be mounted on the robot because of its weight that is more than two times higher than Katana can take. Moreover, the size of the camera is too big to be mounted on the gripper.

It was decided to use Kinect v2 because it has a big developer base and long support from Microsoft and ROS community can be expected. Furthermore, Kinect has a big advantage because it uses Time of Flight (ToF) technology to get the 3D image. Therefore no mounting on the gripper and only one device is needed. Another reason is that it is already used for EtaBot project[17] that is found in the same lab as the Katana project, the team will be familiar with this sensor. Moreover, the plan for the future is to use also ultrasonic sensors in a similar way as it was used in case of the EtaBot project.

3.2.4 The principle of Kinetic v2 functioning

Kinect v2 [13] includes a Red, green, blue - depth (RGB-D) sensor. The depth image system is based on ToF technology. It measures the time between sending and receiving back the IR rays signal. Kinect v2 includes IR camera and IR sensor. This sensor uses the method of quantum efficiency modulation for high-frequency signals. There are two signals with different frequency.

The time where both signals have a maximum determines the distance of that object and the sensor[18]. This method decreases depth uncertainty. Time of flight technology has the advantage of high resolution in depth that is not dependent on shadows and colours as it is the case with two RGB cameras in stereo vision.

However, even if this Kinect v2 is more resistant to sunlight in comparison with Kinect v1, it cannot be used in direct sunlight although it is possible outside in shadow now. This is more than suitable for EtaKatana indoor application.

3.3 Katana 400

Katana 400 is 5DOF robot. Specifically, it is a model Katana 400 6m180 which is used in the thesis. Katana rotates according to modified DH-notation[2], drawn in figure 3.2, two times around x axis (angles α_1 and α_4)[20].

It is possible to connect to the Katana robotic arm through an ethernet cable or using a USB cable (description of LAN configuration is available in B.2). One significant disadvantage of Katana is that it can only hold an object weighting up to 400 g.

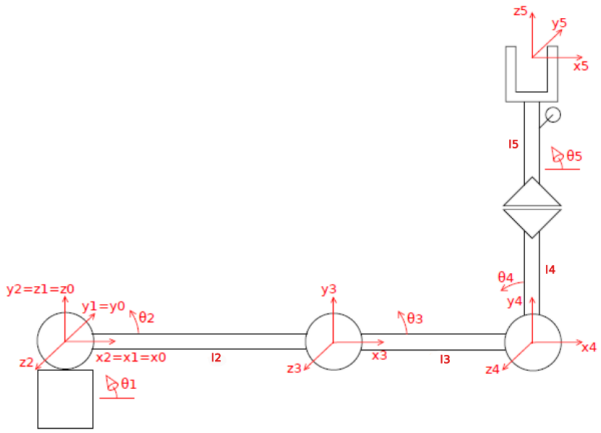


Figure 3.2: Katana 400 described in modified DH notation (source: [2]) values of parameters 3.2

Meaning of the variables (also described in figure 3.3):

[2]For more information about modified Denavit-Hartenberg (DH)-notation see part 7.5.3 in [19].

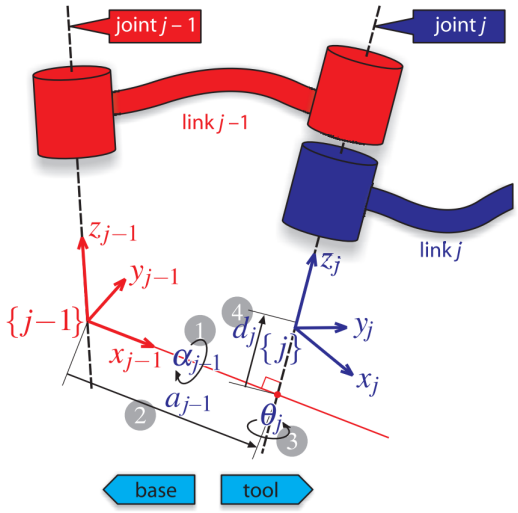


Figure 3.3: Modified DH notation (source: [3])

j-th joint	$\boldsymbol{\theta_j}$**[°] (range)**	$\boldsymbol{d_j}$**[mm] (range)**	$\boldsymbol{a_{j-1}}$**[mm]**	$\boldsymbol{\alpha_{j-1}}$**[°]**
1	0 (-169.5/+169.5)	0 (0/0)	0	0
2	0 (-30/+102)	0 (0/0)	0	90
3	0 (-122.5/+122.5)	0 (0/0)	190	0
4	0 (-168/+168)	0 (0/0)	139	0
5	0 (-29.5/229.5)	313.3 (313.3/313.3)	0	-90

Table 3.2: Table with modified DH notation in the figure 3.2 with added ranges

a_{j-1} is distance from z_{j-1} to z_j along x_{j-1},
α_{j-1} means angle between z_{j-1} and z_j around x_{j-1},
d_j represents distance from x_{j-1} to x_j along z_j,
θ_j determines angle between x_{j-1} and x_j around z_j
l_j is length of the i-th link.

The origin of the Katana coordinate system lies at the intersection of the axis of the first and second joints. The base coordinate system corresponds to the x_0, y_0, z_0 (Fig. 3.2). The gripper coordinate system is located in the middle of the tool with axis z in the direction of the usage of the tool. It is marked in the figure as x_5, y_5, z_5. Euler angles around Z-X-Z were used for the rotation of the tool in the KNI library [20].

3.4 Neobotix MPS-400

Neobotix MPS-400 (see figure 3.4) is a robotic platform. This model is no longer on the market and its successors support ROS (models MP 400, MPO 700, MP 500 and MPO 500 [21]). There is also available a basic operation with MPS-400 using ROS (see [22]), it is possible to control the platform using keyboard or program the trajectory using C++.

Figure 3.4: Robotic platform MPS-400

3.5 Robot Operating System

ROS is defined according to the documentation[23] as "ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.[3]" The advantage of ROS is the possibility to easily connect parts of a robotic system and communicate between the parts of the system. Another advantage of ROS is that a part of the system can be easily replaced with a new one.

[3]Basic ROS tutorial [24]

3.5.1 Architecture of ROS and their communication

Each element in ROS either with real hardware (for example katana) or only virtual (katana_arm_kinematics) is called node. The first collection of nodes that are normally launched is roscore. This node is not only a basic library with core functions of ROS, but it also makes it possible to communicate between nodes.[25] Whenever a node is launched, it needs to register its session with roscore. The nodes communicate with each other using messages or services.

Messages

The easiest example 3.5 is for two nodes. First it is needed to set the topic for which the messages will be published. In example case it can be *gripper_grasp_posture_controller\goal* that sets the Katana's gripper position. A subscriber is a node that waits until other node publishes a message on the topic and then subscribe it. In this case *katana* is the subscriber. That means that *katana* waits and listens if somebody published something on some of the topics that katana subscribes to. The node that publishes the message is a publisher. The publisher needs to send a message that strictly corresponds to the format expected by the topic. An important note is that publishing message is here a communication between n publishers and n subscribers.

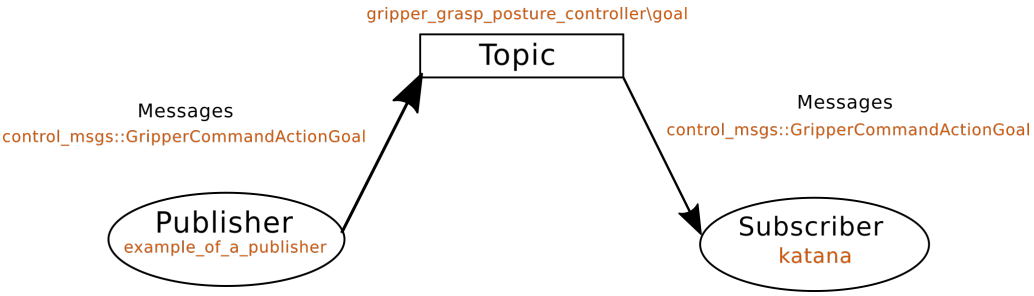


Figure 3.5: Architecture of publishing messages using ROS

Services

The most important difference from messages is that the client sends a service request to a server and also expects an answer and waits for the answer 3.6. The client sends service request → the server processes the request → the

server sends the response to the client. An easy example of service for server *katana* is *switch_motors_off*.

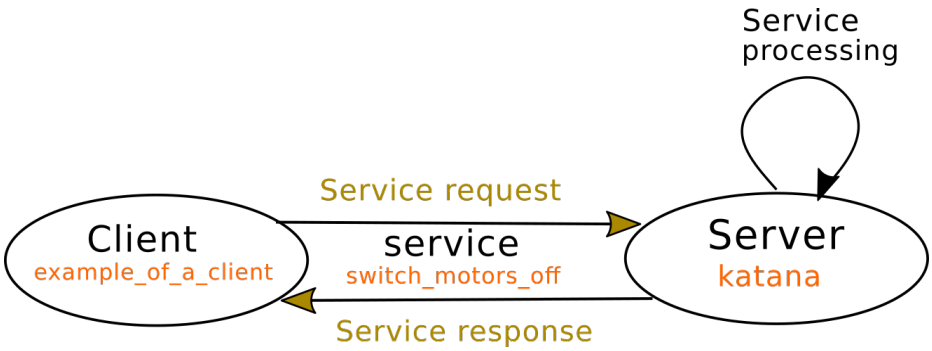


Figure 3.6: Architecture of service using ROS

3.6 General kinematics solution for a 5 DOF robot

The forward kinematic for the 5Degree of freedom (DOF) robot has a manageable set of solutions. However it is harder with inverse kinematic, there are the solutions dependent on a parameter. To elegantly solve this problem it was decided to plan to place Katana on a vehicle (Neobotix MPS-400). Neobotics has two Controllable degree of freedom (CDOF). It means that with Katana the whole mechanism has 7CDOF and 6DOF. If the additional vertical translation joint (see section 3.6) is also counted, the EtaKatana mechanism has 8CDOF. For solving the inverse kinematic only for Katana, that is a 5DOF robot; there are three possible approaches to find the solutions. An analytic solution and a geometrical solution are together called a closed-form of solutions. Then there exists a possibility to solve the inverse kinematic using iterative methods. The iterative method has the main problem that the accuracy of the solution is highly dependent on the time devoted for calculating the solution[19], and because of that, the solution cannot be found in a sufficiently short time even if it exists.

There are three different solutions available for Katana for inverse kinematics. First possibility is KNI based solution that is calculated using geometrical solution (possible to find the calculations in KNI library file *kinematics6M180.cpp*) that is not available in working variant for ROS. The second possibility is to add a virtual joint to get a 6DOF robot in MoveIt!, then calculate an analytical solution of Inverse kinematic (IK) using plugin

IKFast MoveIt! IKFast[4] used in package ros-kinetic-katana-driver[5]. Either the numerical package in MoveIt! called The Kinematics and Dynamics Library (KDL) (available at [26]) or other iterative method can be used.

MoveIt! cannot be used because of incompatibility with Linux Mint[27] that is generally used operating system in the lab where the project was implemented. MoveIt! guarantees compatibility for ROS Kinetic only for the Ubuntu 15.10, 16.04. Another reason is the plan to add the translation joint and also the robotic platform because then it will be nessesary to use another algorithm, not only use the plugin IKFast. That was one of the main reasons why we decided not to change the operating system. We write the kinematics (5.1) on our own instead.

3.7 System design

The scheme of the system (see figure 3.7) shows the proposal of all hardware base components needed for the EtaKatana project. Each component except the central computer symbolises a package in the project. The arrows symbolise the communication between hardware components in packages and the control and computing unit.

The Flowchart (the attached A2 paper C.1) shows how the robot will proceed during looking for an object and then handing the object over to the user. The green inscriptions represent the planned tasks for the master project, and the blue one represents possible extensions of goals if there is enough time left.

To briefly outline the situation, the camera solution includes more sub-steps, first the steps that are made once in a while as is the camera calibration. Then a pattern needs to be prepared for the detection of the object to be grasped. Detection of the object is followed by finding coordinates in the camera coordinate system which are then recalculated into the robot coordinate system. Moreover, it is needed to calculate the relative orientation of the object in the database and the recognised object. That will be used for grasping the object by the gripper.

After that it is necessary to solve the problem of grasping the object by

[4]Desctiption of using `http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/ikfast/ikfast_tutorial.html`.

[5]More information about katana driver `wiki.ros.org/katana_driver`

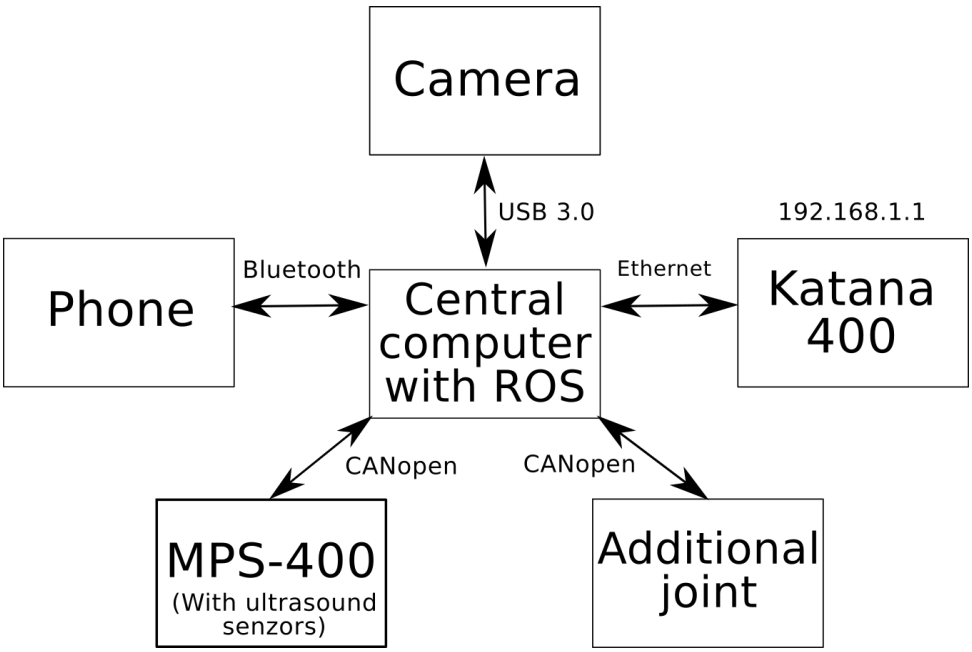


Figure 3.7: Architecture of the project

the Katana arm. First it is needed to decide how the object will be grasped. The grasping position of the object can be saved in a database, or it can be generated by an algorithm which is a highly complex task. Then it is necessary to recalculate the grasping position from the world coordinates into the robot coordinates. When the grasping position of the object in the robot coordinate system is already known, the next step is to solve inverse kinematic to get all joints rotations (see section 3.3 and 3.6). Then the trajectory needs to be calculated to reach the grasping position. The object can be grasped after the grasping position has been established. The collision avoidance can be considered additionally.

Chapter 4

Object detection and robot-camera calibration

The main topic of this chapter is recognition of an object and also localisation of the object in the picture captured by the Kinect. The intended use needs to be considered. Thus, general objects and user-defined objects will be detected.

4.1 Object detection

Combination of three different algorithms is used for the object detection task: Hough circle transformation, YOLO deep neural network and features detection. Hough circle transformation was used to detect the calibration ball and to implement the first pick and place task. In the project proposal, the idea to be able to detect objects during the robot movements was considered and chosen as well as the idea that objects will be divided into two categories, i.e. general objects and user-defined objects. The predefined general objects make Etakatana application useful for the user from the first days. In comparison with them, the user-defined objects allow personalising of the program. They are recognised using feature object detection. The user will have a possibility to add a new user-defined objects[1], e.g. a box with medicaments. The combination of DNN for general objects and detection based on feature detection makes the program useful in different stages of user experience. Above that, knowledge of object detection in robots environment

[1]Now without a graphic user interface, how to add new user-defined object see B.7

can be used for more complex decision making, collision avoidance and trajectory planning.

4.1.1 Hough circle transform

The hough transformations[28] are based on finding a lot of potential candidates of one type of geometrical shape, that it is looked for in the picture. In each edge, the algorithm looks for example for lines. It takes the descriptors of the geometrical shape and create a histogram. That is, for line example, a 2D histogram with the slope on one axis and the intercept on the second one. That means that each line is represented by one point in the transformed space. The lines that are over a threshold in the transformed space are the detected lines.

The recognition using hough circle transformation works like hough line transformation, with the difference, that transformed space is three dimensions because circles are represented in 2D by two coordinates of circle centre and its radius.

To shortly explain the algorithm – first, the image is blurred. Then the edges are detected. When the hough gradient method is used, three-dimensional data (circle center coordinates and circle radius) are found and accumulated in the histogram, the circles are the maximums of the 3D histogram. The weakness of this algorithm is primarily that the algorithm can be numerically unstable and if the accumulator threshold is set low, the computational time can grow fast[29].

This algorithm is used for the calibration (see 4.2) and for pick and place task of a can (see 5.2).

4.1.2 YOLO

The YOLO Deep neural network was chosen because it is nowadays one of the state-of-the-art object detectors, which is being constantly developed, too. YOLOv2 that had been used for EtaKatana implementation was trained on Common Objects in Context (COCO) dataset[30] that has 80 classes and also contains different view on the object for labelling; typical for daily life scenes. As the name suggests, there are common objects in COCO, around 61

% of them are suitable for EtaKatana application. Information about objects recognised by the YOLO can be useful for the pick and place tasks and also planning of robot behaviour in the environment. The dataset contains for example objects needed for eating as fork, furniture as chair that can be useful for orientation in the environment, then objects for entertainment as a book, for communication as a phone, to help with cooking as a microwave, for exercises as a sports ball, for better collision avoidance as a person. An example of future application can be: put the cake in the fridge, bring me the laptop, put the dishes to the sink.

A big advantage of this neural network is that it is not computationally demanding during classification. Another advantage is also the hierarchical view of object classification. Hierarchical view means that the detected object can be classified in more non-exclusive classes at the same time. For example, an apple can be classified as fruit, food and also as an apple. This hierarchical point of view allows a deeper context of understanding. The idea to train the object detector not only on the datasets intended for an object detection but also on the datasets intended for the classification task expands the volume of training datasets significantly. The disadvantage of YOLOv2 is that in comparison to slower nets it has in some cases lower mean Average Percision (mAP).It has, for example, 1.5% lower mAP then SSD512 on PASCAL VOC2012 test dataset (source [31]). The YOLOv3 was not used because of the compatibility with the currently used OpenCV version for the project (OpenCV 3.3.1).

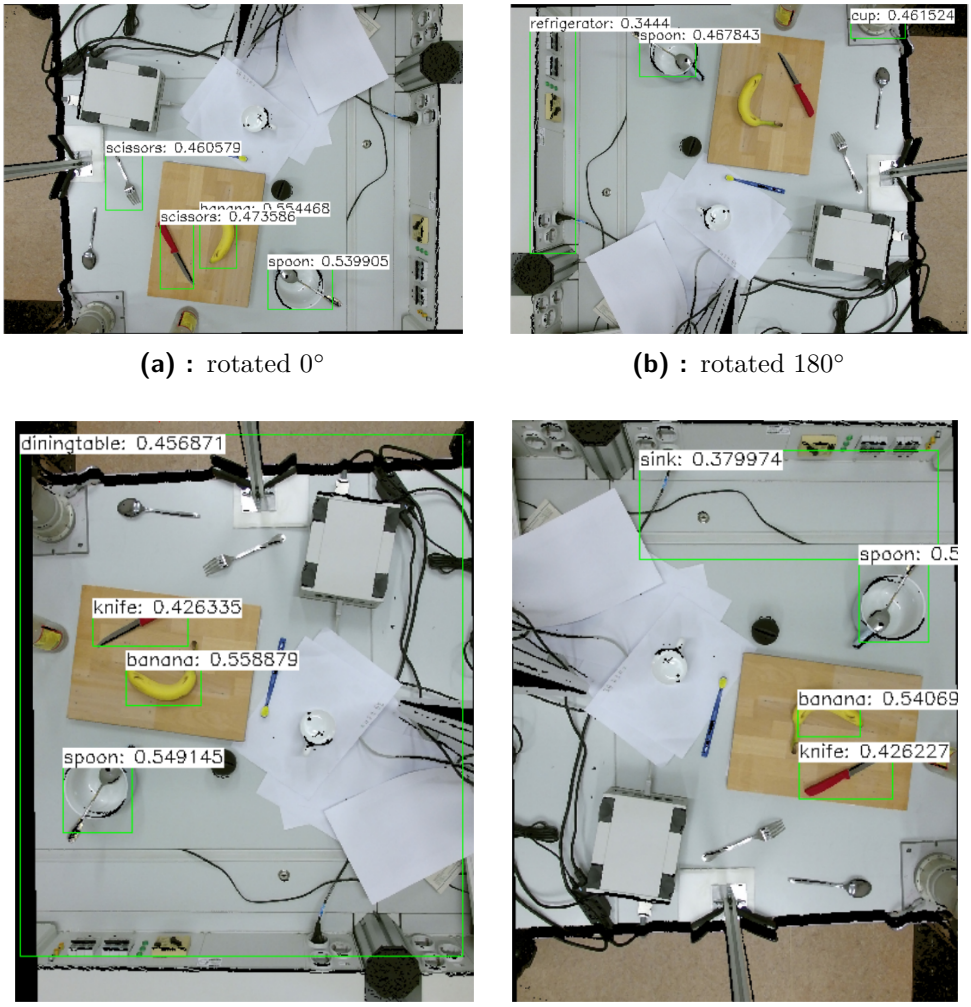
The implementation using YOLO[32] detects various objects at the same time (figure 4.1) and also more objects from the same class (figure 4.2). The detected object and its confident are different for different points of view (4.1). In EtaKatana implementation, the user can ask for an object class; all the found represents are sorted by confidence. Then objects positions and the sizes of the recognised boxes in the camera coordinate system (CCS) are published.

4.1.3 Features based detection

A user-defined object can be almost everything, so we decided to concentrate during the testing mostly on boxes with medicaments which are an integral part of the medical and assistive environment (see figure 4.4).

To shortly describe the algorithm[2], first the Speeded-Up Robust Features

[2]Source of the algorithm [33].



(a) : rotated 0° **(b) :** rotated 180°

(c) : rotated 90° **(d) :** rotated 270°

Figure 4.1: Example of detecting more different object classes at the same time and its dependency on rotation of the captured image.

(SURF) algorithm is used to detect the key features and get their descriptors. The descriptors are rotation and scale robust and they are then used to find correspondences between the pattern image and the actual captured camera image. The correspondences between the found features are identified using the Fast Library for Approximate Nearest Neighbors (FLANN) search and only the enough suitable matches are filtered. In the end, localisation of the recognised object is done by finding homography using the Random sample consensus (RANSAC). For more information about algorithms mentioned in this section see [34]. To reduce the number of misdetections, requirements for the localised quadrilateral were added (table 4.1).

A weakness of this method is that it is necessary to make more photos of the same object from different viewpoints. It is recommended to capture

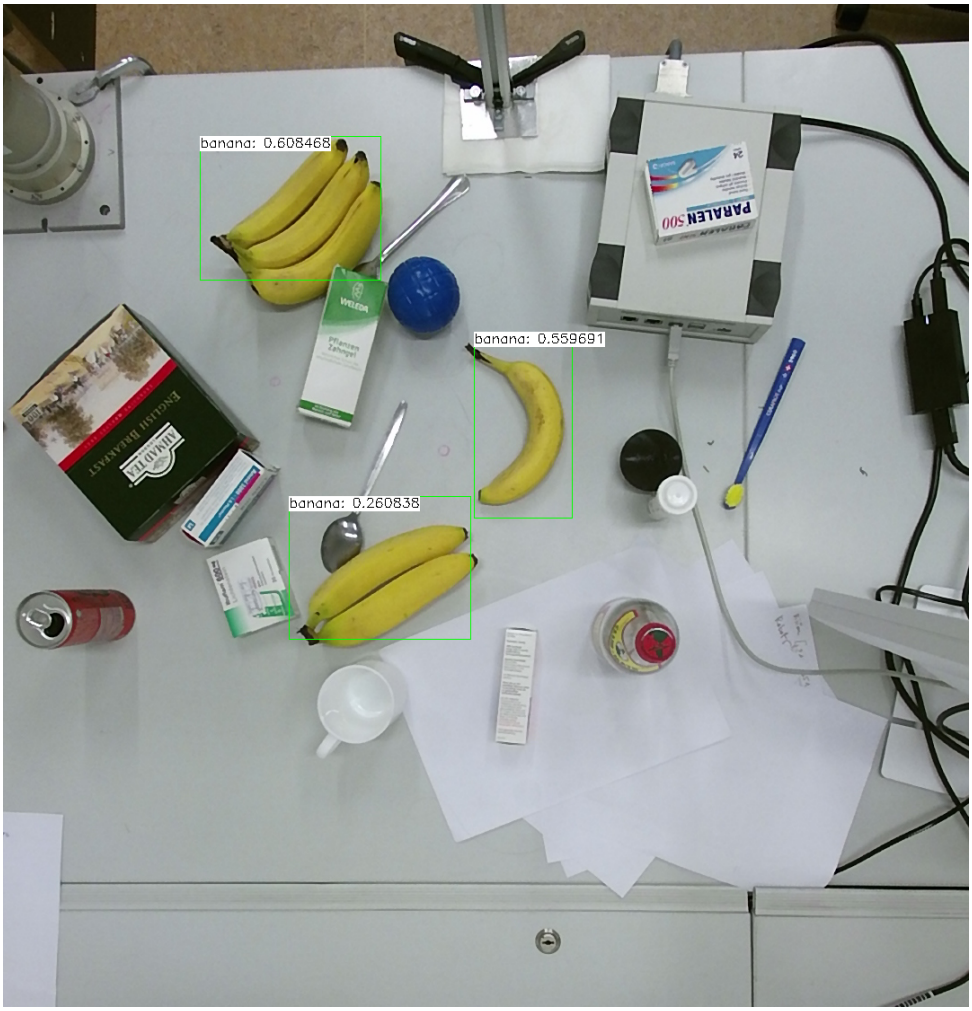


Figure 4.2: Example of detecting one type of object using YOLO

the object on some neutral background. Even if algorithms for finding the features correspondences are presented as scale invariant, when the scale between pattern figure and the object size on the captured figure is too huge, then it is needed to downscale the pattern image. As it was necessary in case of pattern images photographed by phone camera[3]. A phone camera was chosen to simulate the expected conditions in final usage where tablets and phones are expected as devices for user interface 2.2.3.

[3]23MP, 1/2.3" Exmor RS™ for mobile image sensor

$[\alpha,\, \beta,\, \gamma,\, \delta] \in \left(\frac{\pi}{4},\, \frac{3\pi}{4}\right)$	$\left[\frac{s_1}{s_3},\, \frac{s_2}{s_4}\right] \in [0.5, 1.5]$	$\begin{bmatrix} \frac{l_1}{l_2} & \frac{l_1}{l_2} \\ \frac{s_1}{s_2} & \frac{s_3}{s_4} \end{bmatrix} \in [0.5, 1.5]$	$[s_1,\, s_2,\, s_3,\, s_4] > 10\,\text{pxl}$

Table 4.1: requirements on the localization of the object, variables description in 4.3

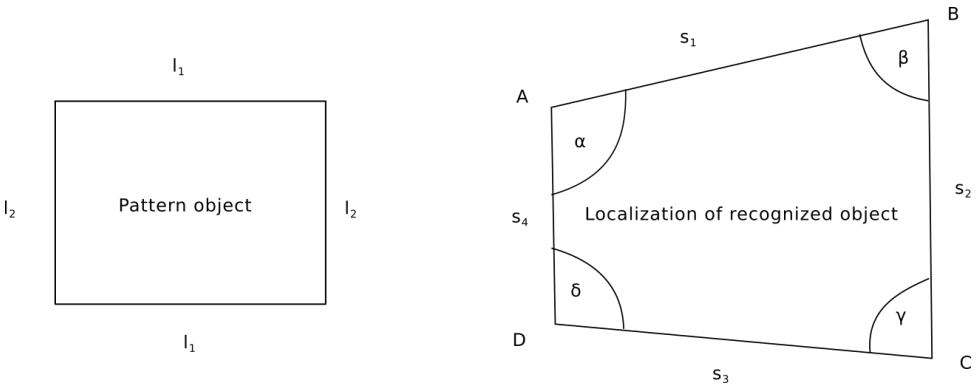


Figure 4.3: Variables description for the table 4.1

4.1.4 Outlook

A beneficial next step can be recalculation of the depth map data for the occupancy grid. That step enables calculating a trajectory with the collisions avoidance, for example by using rapidly-exploring random tree (RRT) algorithm[35].

4.2 Robot-camera calibration

The calibration[4] between coordinate systems of the sensors is an essential task in each robotic application. Two coordinate systems need to be calibrated with each other, Camera and Robot coordinate systems (see figure 4.5). Because of the need to repeat the calibration from time to time we decided to compute the calibration automatically.

[4] For information about launching the implementation see B.9.1.

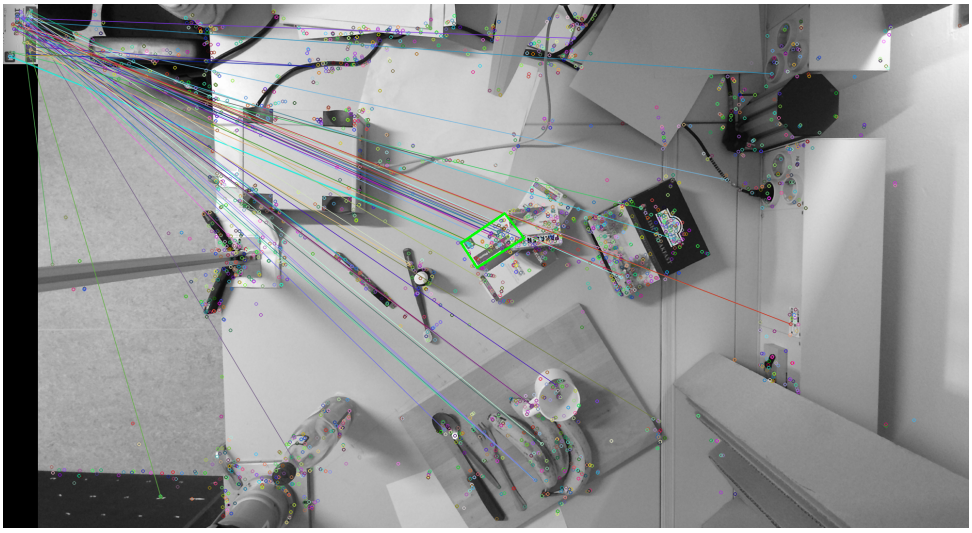


Figure 4.4: Example of detecting user defined object (in this case medicaments).

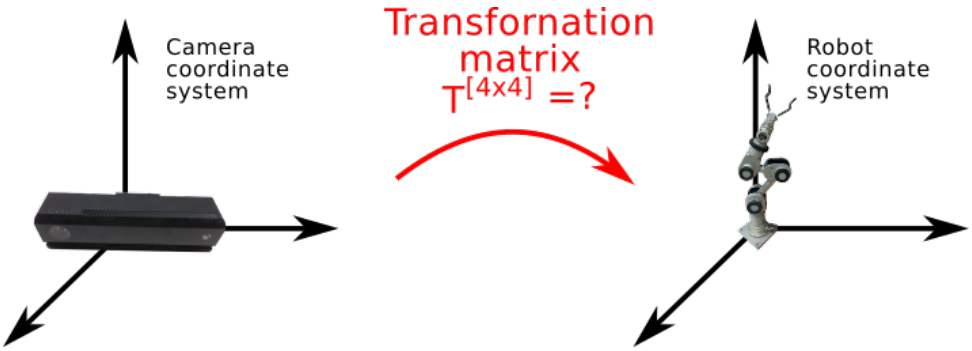


Figure 4.5: Calibration between two coordinate systems.

4.2.1 Algorithm

The calibration is carried out using a 3D printed calibration ball (see figure B.1). The calibration ball has the advantage of symmetric that makes its recognition and also the calculation of the end effector position in z-axis easier. First it is necessary to define eight robot positions with orientations. It is recommended to use end effector roll and pitch angle equal to zero, to ensure a good view on the calibration ball. The robot moves between the defined steps by sub-steps[5]. In each step, the person is asked to confirm moving of the robot to the next position and then also to confirm the recognised position of the calibration ball. The robot goes to different positions, and in each position, the camera recognises the position of the calibration ball. The robot position is calculated from actual published data \joint_states to reduce errors of the joint positions. The position detected from the camera is

[5]That can be redefined by changing the constant POINTS_IN_GRID in file calibration.cpp

recalculated using information about focal length, depth and move in z-axis.

$$x = (u - u_0) \cdot z \cdot s. \tag{4.1}$$

Where x is x coordinate in the camera coordinate system in meters,
u means the detected x coordinates value in pixels,
u_0 represents x coordinate of the principal point(middle of the picture) in pixels,
z is the detected depth value in meters,
s is a scale for one-meter distance from the camera in px^-1,
u_0 and s are dependent of the image resolution. The calculation for the y coordinate is similar. When the positions of the robot and the detected position of the calibration ball are concerned as point clouds, then algorithm[36] can be applied to find the transformation between the two coordinate systems. First are calculated the centroids of both point clouds as,

$$t_1 = \frac{\sum_{i=0}^{n} p_i}{n}. \tag{4.2}$$

Where n is the number of points in the point cloud, p_i is the i-th point in the point cloud.

The translation of first point cloud is t_1 from the origin of the coordinate system. The point clouds are transformed to the origin and then a covariance matrix between camera and robot point clouds is calculated.

$$\mathrm{Cov}[X, Y] = \sum_{i=0}^{n} \mathrm{Cov}[x_i, y_i] \tag{4.3}$$

where X is the point cloud for robot and Y means the point cloud for the camera. Then the Singular Value Decomposition (SVD) of the covariance matrix is calculated. Rotation is then

$$R = U \cdot V^T \tag{4.4}$$

Where U is matrix of left singular vectors and V^T means transposed matrix of the right singular vectors. We get the Transformation matrix T between the camera and the robot point cloud as follows:

$$T = T_2 \cdot R \cdot S \cdot T_1 \tag{4.5}$$

Where T_2 is the translation matrix between the centroid of robot positions point cloud and origin. T_1 is the translation between the centroid of detected ball positions point cloud and origin and S means scale matrix.

After the recalculation of the dynamic part of the calibration, it is needed to recompute the static distance between the recognised ball and the tool position. The fixed position was used because if the robot grasps the object

	average error(cm)	maximal error(cm)
train data	0.35	0.56
test data	0.99	1.83

Table 4.2: Results of testing the the calibration matrix

then the object is grasped each time a little bit differently, and translation from the coordinate tool system to calibration ball is not always the same. The translation in our case is 3.5 cm in z-axis.If the roll and pitch angle angle is equal to zero then the calibration ball with a circle on the top of it can be used. The circle was added to reduce the problem of zero values of the depth on the edge between two heights, which starts to be more significant with using a higher resolution of the camera (The change from HD to SD was a necessary step for enough high-quality picture for the object detection). Moreover, adding the circle reduces the average error in testing data by 42.8 % over the original value and by 45.1 % in maximal error over the original value, which brings us to the next section, calibration testing.

4.2.2 Testing of the algorithm

The calibration matrix was tested for the training data (data which were used to calculate calibration) and also for testing points (set of new independent points). The results show that it is possible to grasp an object like a can or a box. The biggest source of measurement error can be caused by the inaccurately recognised depth of the ball. Negative influence on error can also be caused by small movements of the calibration ball in the fixing system, compared to the fact that the fixing system has big advantages: it is easily removable and easily reproducible. The error caused by the joints movements was reduced, by calculating the end effector position directly from the published joint states. The average error values are in table 4.2 and the errors are also marked in figures 4.6 and 4.7 between the real value of the end effector position and the value calculated from vision.

4.2.3 Outlook

Other variables that can be tested and can bring further accuracy improvement are definition of calibration points in a way that they covered the expected robot working area as much as possible (distribute more calibration points near the surface). Next possibility for further testing can be to gradually increase the number of uniformly distributed calibration points and to monitor

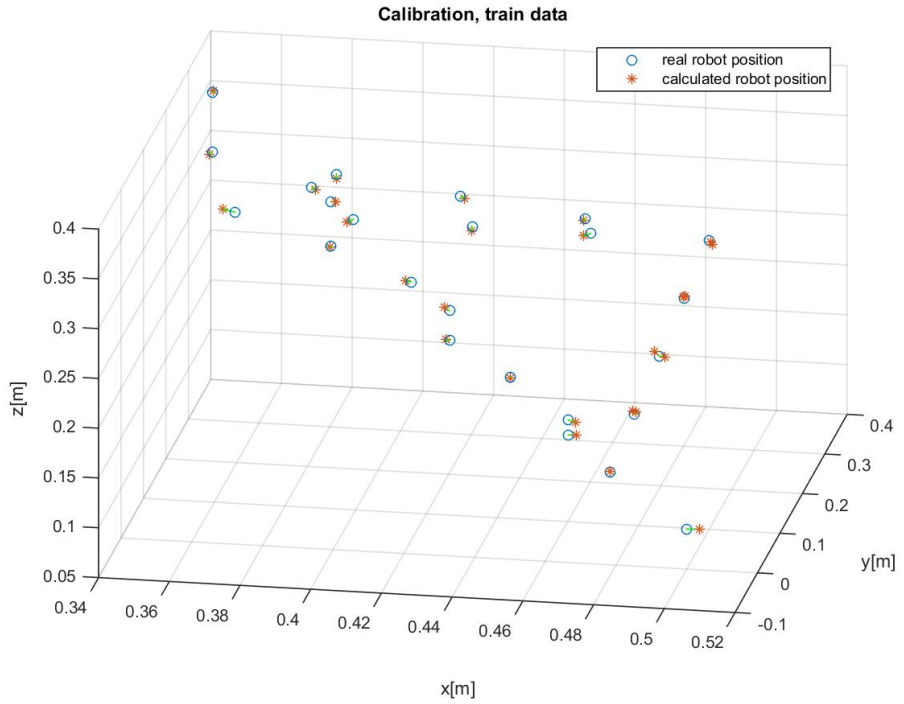


Figure 4.6: The error between calculated and real value of calibration ball positions on training data

when the number of points means improvement which is significant enough in comparison to the time necessary to perform the calibration task. The third possible improvement can be calculation of the internal camera calibration.

4.3 Camera parameters

The camera parameters for SD and HD camera were calculated:

$$f = \frac{s}{\left(\frac{y}{y'} + 1\right)} \tag{4.6}$$

where y is the size of the object
y' is the size of the image
s is the distance from camera
f is the focal length

The depth camera and the RGB camera were translated to each other for the HD resolution (see figure 4.8), because of that it was needed to translate

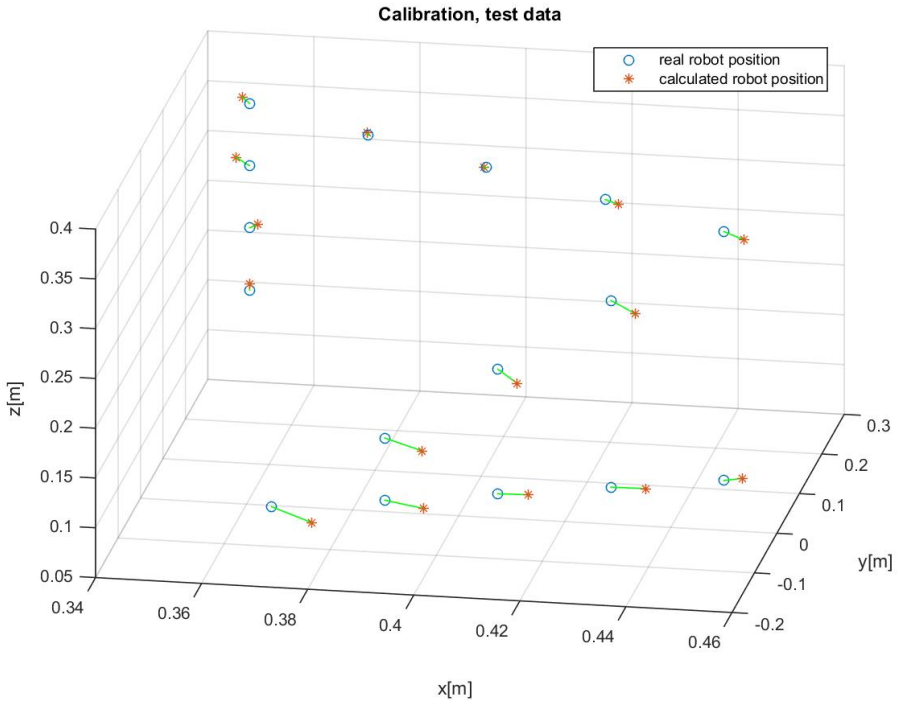
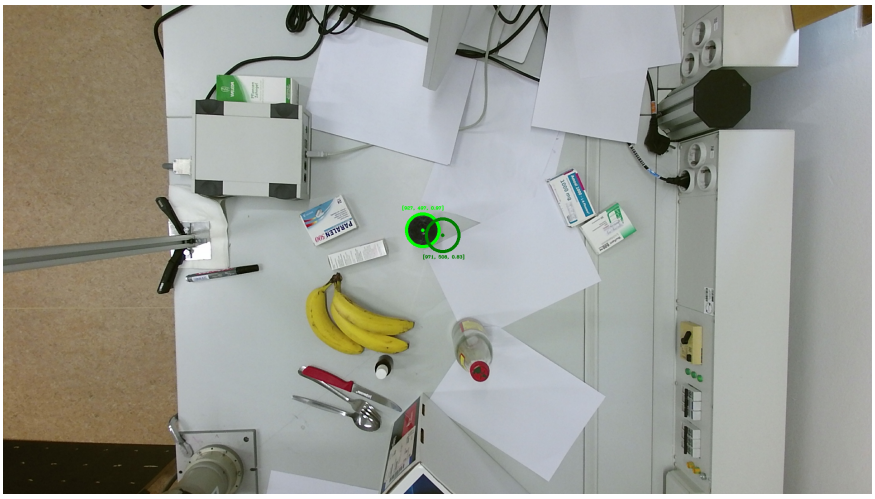


Figure 4.7: The error between the calculated and real value of calibration ball positions on testing data

Resolution	**Focal length[px]**	t_x**[px]**	t_y**[px]**
HD	483.8	44	11
SD	158.8	0	0

Table 4.3: Camera parameters

the place where the depth for the recognised object position in the RGB image is detected.



(a) : Light green recognized, dark green expected position in depth map

(b) : Recognized position from RGB camera and the recalculated position for depth map

Figure 4.8: RGB versus depth picture

Chapter 5

Movement planning for gripping and depositing the detected workpieces

Before movement planning begins it is necessary to carry out forward and inverse kinematics for Katana 400_6m180. MoveIt! cannot be used because of incompatibility with the operation system[27]. After the kinematics, the pick and place task is described and in the end, there is a short outlook of the project.

5.1 Forward and inverse kinematics for Katana robotic arm

Forward and inverse kinematics are essential for every robot because the forward kinematic allows to recalculate the rotation of the joints to the end effector position with the orientation. Inverse kinematic allows computing of the joints rotations from the position and orientation of the end effector. The inverse kinematic can be calculated using geometric methods, algebraic methods or numerical methods[37]. The geometrical approach was chosen, because it is computationally faster then iterative methods and in comparison to algebraic methods it is an illustrative solution.

5.1.1 Forward kinematic

The position of end effector was calculated using modified DH notation, which was changed in comparison to the section 3.3, because in the practical case the DH notation was calculated depending on the topic /joint_states published by the node /katana. The published joint rotations have a direction description which is different from the one described in the Katana manufacturer documentation[20]. The used parameters are summarised in the table 5.1 and the axis orientation in each joint is illustrated using RviZ in the figure 5.1. The main difference is in the placement of the origin of the first joint in the base of the first joint, marked in figure 5.1 by longer axes and in the change of the direction of the fourth joint rotation.

j-th joint	$\boldsymbol{\theta_j}$**[°] (range)**	$\boldsymbol{d_j[mm]}$	$\boldsymbol{a_{j-1}[mm]}$	$\boldsymbol{\alpha_{j-1}}$**[°]**
1	0 (-169.5/+169.5)	274.5	0	0
2	0 (-30/+102)	0	0	90
3	0 (-122.5/+122.5)	0	190	0
4	0 (-168/+168)	0	139	180
5	0 (-29.5/229.5)-90	0	152.3	90

Table 5.1: Table with modified DH notation in the figure 3.2 from the topic /joint_states point of view

Meaning of variables (also described in figure 3.3):
a_{j-1} is distance from z_{j-1} to z_j along x_{j-1},
α_{j-1} means angle between z_{j-1} and z_j around x_{j-1},
d_j represents distance from x_{j-1} to x_j along z_j,
θ_j determines angle between x_{j-1} and x_j around z_j

The gripper changes its positions in the range $[-0.44,\ 0.3]$, which corresponds to the range $[2.41,\ 11.58]$cm.

Calculation of modified DH notation:

$$^{j-1}A_j = R_x(\alpha_{j-1}) \cdot T_x(a_{j-1}) \cdot R_z(\theta_j) \cdot T_z(d_j) \tag{5.1}$$

Where T_i is the transformation from (i-1)-th joint to the i-th joint.
$R_x(\alpha_{j-1})$ represents rotation along (i-1)-th x-axis and $R_z(\theta_j)$ i-th rotation along z-axis.
$T_x(a_{j-1})$ means translation along (i-1)-th x-axis and $T_z(d_j)$ translations along i-th z-axis.

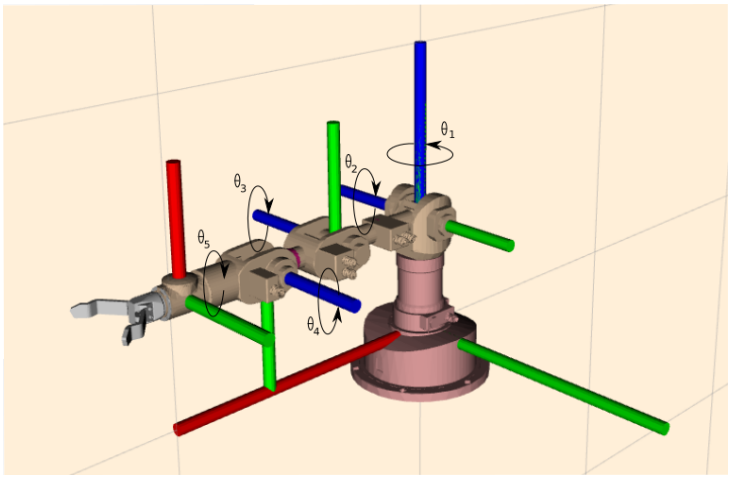


Figure 5.1: Robot in configuration with all zero angles with grid in X-Z plane

$$^{0}A_5 = ^{0}A_1 \cdot ^{1}A_2 \cdot ^{2}A_3 \cdot ^{3}A_4 \cdot ^{4}A_5 \tag{5.2}$$

$$E = ^{0}A_5 \cdot \vec{i} \tag{5.3}$$

Where T is the transformation from the robot base to the robot end effector,
E represents the end effector position,
$\vec{i}$ means the vector $[0\ 0\ 0\ 1]^T$.

For the calculation of the end effector orientation (figure 5.2), it can be deduced from the figure 5.1, that

$$\alpha_{roll} = -\theta_5 \tag{5.4}$$

$$\beta_{pitch} = -\theta_2 - \theta_3 + \theta_4 \tag{5.5}$$

The yaw angle is already determined by x and y position of the end effector, because Katana is a 5DOF robot.

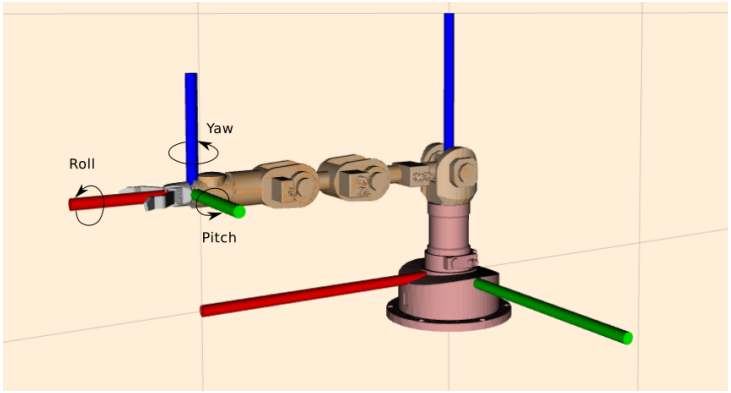


Figure 5.2: Coordinate system of end effector

5.1.2 Inverse kinematic

It was decided to solve the inverse kinematic geometrically. First, the joints will be numbered from base to the gripper ($j_1 - j_5$). To simplify the robot in a planar robot, we assume a viewpoint orthogonal to the arm plane, i.e. we rotate with j1. The j_5 represents only the roll angle of the gripper, see figure 5.2. The j_1 means for us rotation around z-axis in base, thanks to which it is possible to recalculate the manipulator to planar X-Z plane only using the Pythagorean theorem applied on the end effector coordinates (E_x, E_y), see figure 5.3.

$$x_{4m} = \sqrt{E_x^2 + E_y^2} \tag{5.6}$$

The angle of the j_1 can be calculated with respect to the sign for the quadrants as

$$\theta_1 = \arctan\left(\frac{E_y}{E_x}\right) \tag{5.7}$$

Then the situation can be solved for j_2, j_3, j_4 alone in the X-Z plane.

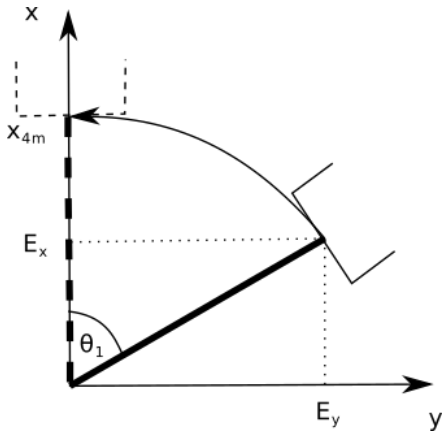


Figure 5.3: Recalculation of the robot from the 3D space to the X-Z

Let´s move one joint in the front from the j_1 by translation in z-axis.

$$z_1 = l_1 = 274.5 \text{ mm} \tag{5.8}$$

$$x_1 = 0 \tag{5.9}$$

Where l_1 represents the length of the first joint. Then the position of the fourth joint (x_3,y_3) can be calculated.

$$x_3 = \cos(\theta) \cdot l_4 + x_{4m} \tag{5.10}$$

$$z_3 = \sin(\theta) \cdot l_4 + z_{4m} \tag{5.11}$$

Where θ is the end effector pitch angle. As a next step the rotations θ_2, θ_3 and θ_4 of joints $j_2 - j_4$ will be calculated. The number of solutions for the

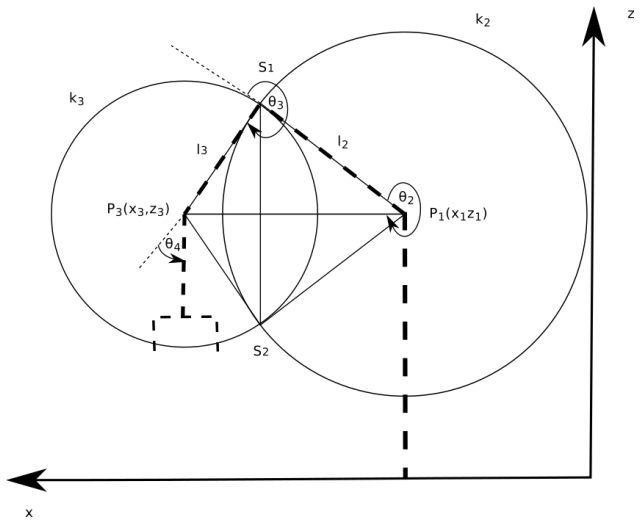


Figure 5.4: Calculation of a 2D manipulator

planar manipulator is calculated. There are three possibilities; there may exist two solutions, one solution or no solution, it exactly corresponds to the number of the intersections of the circles drawn by rotating with the links l_2 and l_3 (figure 5.4). The intersections of circles are marked as S_1 and S_2. The angles for the first and second intersection look similar, here is the example for the second intersection

$$\theta_2 = -\left(\arctan\left(\frac{-z_1}{-x_1}\right) - \arctan\left(\frac{z_1 - S_{2z}}{x_1 - S_{2x}}\right) - \frac{\pi}{2}\right), \tag{5.12}$$

$$\theta_3 = -\left(\arctan\left(\frac{z_1 - S_{2z}}{x_1 - S_{2x}}\right) - \arctan\left(\frac{S_{2z} - z_3}{S_{2x} - x_3}\right)\right), \tag{5.13}$$

$$\theta_4 = \theta + \theta_2 + \theta_3. \tag{5.14}$$

The published rotations were chosen from the valid solutions[1]. The solution with the intersection with a higher value in z-axis was preferred, because then we can expect less collisions with the surface and other objects that lay on the table.

All the calculations were done concerning the quadrants of goniometric functions and concerning the orientation of joint rotations published node /katana.

[1] solutions in the range of the robots joints rotations

5.2 Pick and place task

After finishing the kinematics, it is possible to start to solve the pick and place task[2]. This task is a combination of using detected vision data (see 4.1), calculated transfer matrix (see 4.2) and calculation of the robot trajectory with the robot joints and environment limits controls. For the pick and place task, the grasping position is determined by vision, and the user chooses the placement position by writing the coordinates in the camera coordinate system in pixels to the console. The position of the cursor on the picture can be read in the lower left corner of the window for RGB image[3]. Last but not least, the name of the object that will be grasped is written as the first argument. This information is then published into the recognition part.

5.2.1 Possibilities

The user has two possibilities now, grasping cylindrical objects by detection of the circles (see section 4.1.1) or cuboid objects by detection of the user-defined objects (see section 4.1.3). It is predefined to grasp cylindrical objects parallel to the surface and the cuboid objects perpendicular to the surface[4]. However, the pre-grasping position can be calculated in pick and place algorithm for any approaching angle, which can be used later to grasp objects with a more complex shape and to expand the possibilities of planning more complex trajectories with collision avoidance.

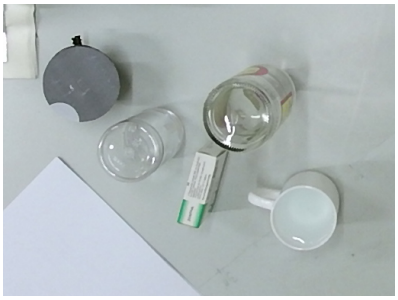
The box was chosen because it represents the second huge category, medicaments and food. Cylindrical objects are essential because it represents another important object in medical applications, drinks. A can was chosen to demonstrate the solution because there is no access to the liquid in comparison to a cup and it reduces the danger of spilling the liquid on the robot. Moreover, with a cup, there is the problem of the wrong detection of the height of the object, because the cup appears to be almost zero in height.Even if there is water inside the cup it does not help with the right detection because water is not reflexive enough for the water level to be recognised. The mug wall is hard to be detected because of the thickness of the wall and the ambiguous determination of height that is marked by black colour in the depth map. Now the depth is taken from the middle of the

[2]For the information, how to run the implementation go to B.9.2.

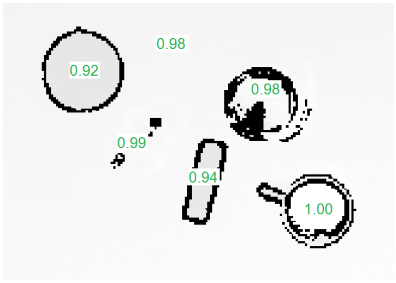
[3]Im case of the user-defined object; it is needed to subtract the x size of the pattern image from the x coordinate read from the shown picture.

[4] During grasping we encounter deficiencies of a gripper with two fingers of inelastic material.

recognised object because near the border of the object a problem can appear with the precise calibration of RGB picture on depth picture. The different angle of the camera can solve the problem with determination of the height of the cups, this solution is planned to be used for the robot (see EtaKatana model with camera FoV 2.6) or also by the other additional ultrasonic sensors that are planned to be added in the future (see section 3.2.3). There are other materials for which it is hard or even impossible to determine the depth using the Kinect sensor, for example, glass objects or objects from some types of plastic (figure 5.5).



(a) : RGB image of reflecting problematic materials.



(b) : Depth image of reflecting problematic materials.

Figure 5.5: Depth of an object made of glass, water or some types of plastics cannot be determined by a Kinect depth camera.

Videos that can be found in folder \Vision_guided_handling_operations\attachment\videos have been created for both of mentioned categories.

5.2.2 Procedure

The robot pick and place trajectory consists of six positions with their orientations. The pre-position and after-position are calculated for each pick and also for each place position to ensure smooth fulfilling of the robot task.

First we need to get the pick point from the camera picture. Information about the detected object is published by vision part of the program (for the detection methods see 4.1). The information is the pick position, and in case of user-defined objects, the orientation of the box where the object was recognised, corners of the recognised box and also sizes of the box are available.

The next step is to recalculate the pick position from the camera to the

robot coordinate systems.

$$x = (u - u_0) \cdot z \cdot s. \tag{5.15}$$

Where x is x coordinate in camera coordinate system in meters,
u means the detected x coordinates value in pixels,
u_0 represents x coordinate of the principal point(middle of the picture) in pixels,
z is the detected depth value in meters,
s is a scale for one-meter distance from the camera in px^-1,
u_0 and s are dependent of the image resolution. The calculation for the y coordinate is similar. In this way we get the vector $p_{\text{cam}} = [x, y, z, 1]^T$.

$$p_{\text{rob}} = D \cdot p_{\text{cam}} \tag{5.16}$$

Where p_{cam} represents modified camera position vector and D means transformation matrix. The same recalculation was needed to be done for place position and in case of user defined objects also for the corners of localized area.

The solution is divided into two branches in this step. The solution for the recognised circles - for that it is needed to calculate only the diameter of the circle in the robot coordinate system. For user-defined objects, it is needed to calculate the object angle additionally, while for cylindrical objects, the object angle is predefined to be zero, because of the horizontal grasping of the cylinders. The diameter in meters is calculated as

$$d_m = \frac{\left(d_{pxl} \cdot \left(\frac{z \cdot 1000}{f-1}\right)\right)}{1000}. \tag{5.17}$$

Where the d_{pxl} is diameter in pixels,
z is the z coordinate of the object and
f is focal length of the camera.
The object angle δ for user defined objects follows the cosine rule on detection rectangle.

$$\delta = \arccos\left(\frac{|(S - \frac{R}{2})S|^2 + |SA|^2 - |A(S - \frac{R}{2})|^2}{2 \cdot |(S - \frac{R}{2})S| \cdot |SA|}\right) \tag{5.18}$$

Where S represents the middle of the transformed rectangle in robot coordinate system (RCS),
R is length of rectangle sizes in RCS,
A means the left upper corner of the rectangle in RCS. The idea is to calculate the rotation around the middle of the object using the calculated left upper corner when the object is without rotation, and the real measured left upper corner.

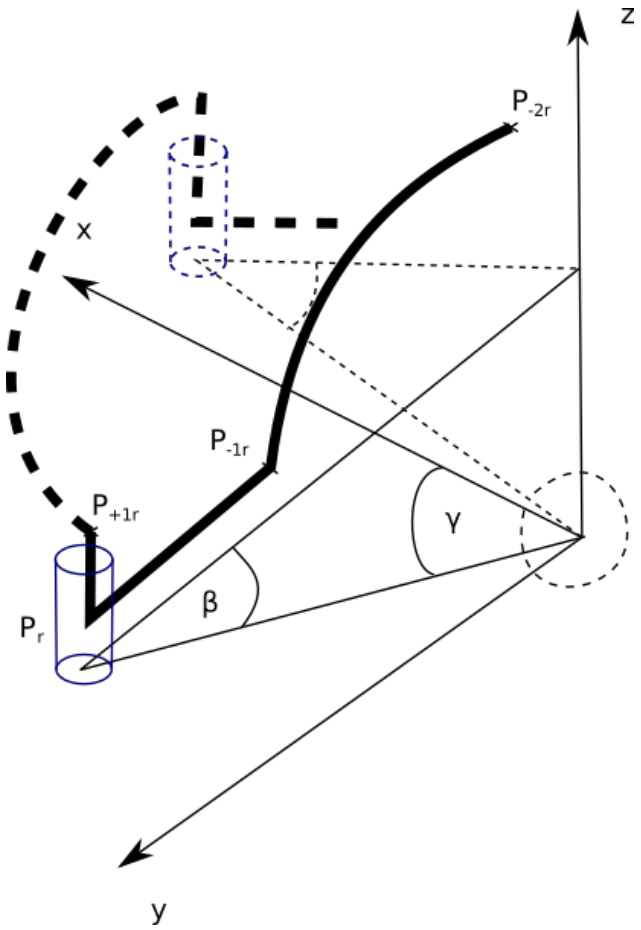


Figure 5.6: Pick and place trajectory with approaching angle β

When pick and also place position and all other needed variables are in RCS, it is time to calculate the pre-grasping/pre-placing and after-placing/after-grasping positions (figure 5.6[5]). So let us calculate the pre-grasping and after-grasping positions first. The pre-grasping-position is the position that is distanced by the d_m from the grasping position concerning the pitch angle of the gripper because the pitch angle indicates the direction in which the robot is approaching the object. The pre-grasping position $P_{(-1r)}$ can be calculated as:

$$\gamma_{yaw} = \arctan\left(\frac{P_{ry}}{P_{rx}}\right) \tag{5.19}$$

$$P_{(-1rx)} = \left(\sqrt{P_{rx}^2 + P_{ry}^2} - \cos(\beta_{pitch}) \cdot d_m\right) \cdot \cos(\gamma_{yaw}), \tag{5.20}$$

$$P_{(-1ry)} = \left(\sqrt{P_{rx}^2 + P_{ry}^2} - \cos(\beta_{pitch}) \cdot d_m\right) \cdot \sin(\gamma_{yaw}), \tag{5.21}$$

$$P_{-1rz} = P_{rz} + (\sin(\beta_{pitch}) \cdot d_m) + d_s. \tag{5.22}$$

Where $\left(\sqrt{P_{rx}^2 + P_{ry}^2} - \cos(\beta_{pitch}) \cdot d_m\right)$ represents distance to the $P_{(-1r)}$ in the X-Y plane from the origin of robot coordinate system.

[5] P_{-2r} is robot position before start of the pick and place task

The P_{rx} and P_{ry} are x and y coordinates of grasping position,
d_m is the already mentioned size of the object from side that will be grasped. In case of top grasping there is nonzero save distance d_s and
β_{pitch} and γ_{yaw} are gripper angles that were modified with respect to the quadrants signs of goniometric functions. The α_{roll} is changing only for top-grasping. For all other cases it remains on zero.

$$\alpha_{roll} = \begin{cases} \gamma_{yaw} + \delta, & \text{if } \beta_{pitch} = \frac{pi}{2} \\ 0, & \text{otherwise} \end{cases} \tag{5.23}$$

Where δ means object angle, β_{pitch} angle is the approaching trajectory angle.

The after-grasping position changes only in z-axis in comparison to the grasping position, it is moved up by a constant.

The calculations of pre-placing/after-placing position are the same as for the pre-grasping/after-grasping position. The only three differences are that we are using the place position as a source position and not the pick position. The second difference is that, for the calculation of pre-placing position, the same procedure is used as for after-grasping position and for the after-placing position the same calculation is used as for pre-grasping. The last but most important difference is that, in case of placing the object, it is needed to consider the z-size of the object and add the z-size to the z coordinate of pre-placing, placing and also after-placing positions. Otherwise, the robot would collide with the surroundings.

The recalculation from the d_m to the gripper value y will be calculated using gripper characteristic 5.24.

Before the robot goes on the calculated positions the trajectory needs to be validated. Validation of the trajectory means that all the points in the trajectory need to be controlled if there exists a solution of the inverse kinematics in the robot angles range and also if the gripper is not out of his range. Otherwise, there will be a collision danger, because when a value out of joint rotation range is published, the joint goes to the closest possible joint values and then stops. Moreover, all the points from the trajectory need to be validated to perform the trajectory, because when the robot skips the not validated points, there will be again a high probability of collision.

Performing the pick and place task consists in publishing the following actions concerning the completion of each action.

1. Go (using point to point movement) to the pre-grasping position and

open the gripper to the maximum value.

2. Move linearly to the grasping position.
3. Change the gripper value to y.
4. Move linearly to the after-grasping position.
5. Go (using point to point movement) to the pre-placing position.
6. Move linearly to the placing position.
7. Open the gripper to the maximum value.
8. Move linearly to the after-placing position.

5.3 Grasping

For the gripper, there is a problem with conductive objects. If both fingers of the gripper connect by a conductive object, the robot disconnects from the computer, because current insulation is not sufficient. So it is recommended to avoid conductive objects or to better isolate the gripper from the environment.

The second order polynomial simulates the dependency of the gripper and distance of the fingers. However the actual values can change, if the robot crashes with some object or environment because the fingers are made of a material that can be bent by hand.

$$y = 0.002x^2 + 0.052x - 0.573 \tag{5.24}$$

Where y is the value for the gripper and x means the real distance of the gripper fingers in cm (figure 5.8). The area occupied by the gripper changes depending on the gripper value. It can affect grasping, in case of possible collisions with the environment. This phenomenon is especially noticeable by vertical grasping of small objects from a hard surface like a table.

5.4 Outlook

The first point that can be solved is the automatic finding of the object size in z-axis. At first sight, it seems to be enough to read the depth from the depth map in the surroundings of the object. However, it can be enough only

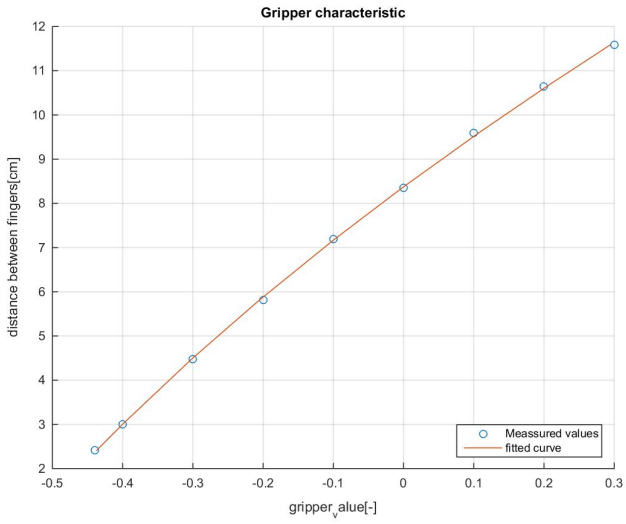


Figure 5.7: Gripper characteristic

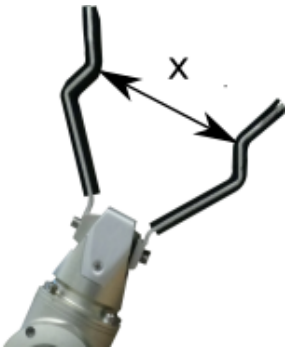


Figure 5.8: Where the distance was measured

if the object is the only one on a flat surface. If we also consider other objects lying next to the detected object, it is probable that the object z size will be detected incorrectly (see figure 5.9a). The next idea can be the saved value of the distance from the basic surface (table, ground) or taking of the most common non-zero value from the captured image and setting it as the basic surface value. That can be highly problematic when there are objects lying on other objects (see figure 5.9b). The situation is even more complicated for the convex and concave objects.

The first mentioned situation would result in collision with the surface during the placing, which can have harmful consequences for the robot, surface and also for the object. In the second case, the object will be dropped from height, which can have adverse consequences especially for the object[6]. For both of the above mentioned reasons, a fixed set value was used for each

[6]The values are saved in file Vision_guided_handling_operations/code/EtaKatana2/data/user_defined_objects/measurements.ext

(a) : Right value 2 cm detected high probably negative value

(b) : Right value 2 cm detected value 7.7 cm

Figure 5.9: Problematic placement of objects.

object. Two possible solutions are offered as an outlook for further developers.

1. Katana can grasp the object and lift it in safe distance from the surface and other objects and rotate the object and then reconstruct a 3D model from measured data. Moreover, the 3D model information can also be saved and used to learn effective object grasping. This solution can be dangerous for liquids.
2. Another possible implementation is to use tactile and force sensors and with them to detect the change of force that acts on the object and the gripper.

The next and colossal step can be to create an automatic prediction of the grasping positions for any shape of the object.

Chapter 6

Conclusion

Our master thesis was motivated by the fact that the WHO assumes that in 2035 it will be missing 12.9 million of healthcare workers[5] and assistive robots can help to solve this expected big gap between the number of patients and healthcare workers.

Having been inspired by already developed mobile manipulators 2.3 and previous projects in direction of mobile manipulators (2.3) at the university, we designed a software (section 3.7) and a hardware (section B.5 and 2.3.3) solution EtaKatana. Our concept offers a UCD (section 2.2.3) with expectation of visual and also voice control. The hardware solution is based on a mobile manipulator with additional vertical translation joint to ensure reaching of an object in a high range, which corresponds to a range that an ordinary human being can reach.

The practical part of the development of the project included a combination of the vision and robotic tasks. The final robot task was to detect a user-defined object by the camera, then to recalculate object parameters from the camera into the robot coordinate system, to calculate additional information on the object such as its size and rotation. The user determines the place position for the object. A trajectory is calculated from the pick and place position. The trajectory is then recalculated using the inverse kinematics to the joints rotations, and the object size is recalculated to the gripper value. A video example is in the attached file pick_and_place_medicaments[1]. To calibrate between the robot and the camera coordinate systems, a dynamic method was used 4.2 for which we get an average distance between the

[1] Vision_guided_handling_operations/attachment/videos/pick_and_place_medicaments.mp4

real and calculated value less then one centimetre, which allows the above mentioned tasks.

The current situation is that Katana can grasp cuboid and cylindrical objects from a horizontal surface, which covers essential objects in the healthcare sector such as drinks and medicaments that lay on the table.

Further tasks of this project which could develop the already implemented part of the project can concentrate on development of automatic finding of the grasping positions for general objects. Trajectory planning with collision avoidance using occupancy grid and RRT will also be really beneficial. Besides other tasks from the expected use of EtaKatana C.1, there are three parts of the EtaKatana project which can be processed as separate projects. The first task is to program the user interface using communication based on ROS and using Bluetooth. The second task is to implement a sound output with possible extension to a chatbot and the last but not least task is the searching in the environment using information from the camera and the ultrasound sensors. For searching in the environment it can be possible to use some of the methods already developed within the project EtaBot[2]. The last and one of the hardest tasks will be to add an additional translation joint and to connect all the subsystems to work synchronously and reliably.

If the reader wants to work further on the project or only to verify the presented solution, practical information on installing of the libraries, connecting of the hardware parts of the project and also on launching of the subprograms are available in B. The program is available on the attached DVD and on local gitlab[3].

[2] Another project running at the University of Applied Sciences in Aschaffenburg
[3] `https://gitlab.h-ab.de/ETARA/etakatana/tree/master`

Bibliography

[1] S. Neumann, "Konzeptstudie zur ausrüstung einer mobilen roboterplattform mit einer handhabungseinrichtung." Aschaffenburg, 2009.

[2] *Katana 450 Benutzerhandbuch, 2.0.4*, Neuronics AG, Zürich, 2008, abbildung 5.24: mDH Koordinatensysteme und Winkel beim Katana 6M180.

[3] "Fig. 7.15. definition of modified denavit and hartenberg link parameters," in *Robotics, vision and control*, 2nd ed. Berlin: Springer, 2011, p. 219.

[4] WHO. (2016) Global health observatory data repository. World Health Organization. [Online]. Available: http://apps.who.int/gho/data/?theme=main&vid=60430

[5] ——. (2013) Global health workforce shortage to reach 12.9 million in coming decades. World Health Organization. Recife, Brazil. Media centre. [Online]. Available: https://www.who.int/mediacentre/news/releases/2013/health-workforce-shortage/en/

[6] L. D. Riek, "Healthcare robotics," *Communications of the ACM*, vol. 60, no. 11, pp. 68–78, 2017-10-24. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3154816.3127874

[7] L. Lhotská, J. Kužílek, and O. Štěpánková, "Asistivní technologie," Praha, 2013.

[8] Robear. RIKEN and Sumitomo Riko. Nagoja, Japan. [Online]. Available: http://rtc.nagoya.riken.jp/ROBEAR/

[9] M. Günther. (2014) Calvin robot. University of Osnabrück. [Online]. Available: http://wiki.ros.org/calvin_robot

[10] (2018) Mobile manipulator rb-1. Robotnik. Barcelona. [Online]. Available: https://www.robotnik.eu/manipulators/rb-one/

[11] T. Zimmermann. European robot network helps nurses and home builders. Stuttgart. [Online]. Available: https://robohub.org/european-robot-network-helps-nurses-and-home-builders/

[12] D. Buchholz, *Bin-Picking New Approaches for a Classical Problem*, 1st ed. Springer, 2016.

[13] E. Lachat, H. Macher, T. Landes, and P. Grussenmeyer, "Assessment and calibration of a rgb-d camera (kinect v2 sensor) towards a potential use for close-range 3d modeling," *Remote Sensing*, vol. 7, no. 10, pp. 13 070–13 097, 2015. [Online]. Available: http://www.mdpi.com/2072-4292/7/10/13070

[14] J. Seßner. (2018) Microsoft kinect v2 3d-camera. Institute for Factory Automation and Production Systems. Nürnberg. [Online]. Available: https://www.faps.fau.eu/ausbio/microsoft-kinect-v2-3d-camera/

[15] *LifeCam HD-5000*, Microsoft Corporation, 2012, technical Data Sheet.

[16] *AXIS 214 PTZ Network Camera*, 2008, datasheed, Day and night camera with pan/tilt/zoom and audio.

[17] V. Kreß, J. Staab, M. Rieder, and P. Fischer, *EtaBot: Dokumentation des EtaBot-Projektes*, University of applied sciences Aschaffenburg, Aschaffenburg, 2019. [Online]. Available: https://gitlab.h-ab.de/ETARA/ETARA/tree/master/Dokumentation

[18] C. S. B. et al., "A 0.13 μm cmos system-on-chip for a 512 $\times$ 424 time-of-flight image sensor with multi-frequency photo-demodulation up to 130 mhz and 2 gs/s adc," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 303–319, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6964815/

[19] P. I. Corke, *Robotics, vision and control*, 2nd ed. Berlin: Springer, 2011.

[20] *Katana 450 Benutzerhandbuch*, 2nd ed. Zürich: Neuronics AG, 2008.

[21] H. Nagaraja. (2017) Neobotix - robotics and automation. Open Source Robotics Foundation. [Online]. Available: http://wiki.ros.org/Robots/neobotix

[22] S. Lackas, "Ros-basierte antriessteuerung für eine mobile roboterplattform," university of applied sciences Aschaffenburg, Aschaffenburg, 2015.

[23] A. Dattalo. (2018) Ros/introduction. Open Source Robotics Foundation. [Online]. Available: http://wiki.ros.org/ROS/Introduction

[24] (2019) Ros tutorials. [Online]. Available: http://wiki.ros.org/ROS/Tutorials

[25] RioS. (2016) Roscore. Open Source Robotics Foundation. [Online]. Available: http://wiki.ros.org/roscore

[26] Kdl wiki. Orocos Kinematics and Dynamics. [Online]. Available: http://www.orocos.org/kdl

[27] I. A. Sucan and S. Chitta. Install moveit! [Online]. Available: https://moveit.ros.org/install/

[28] A. Kaehler and G. Bradski, *Learning OpenCV 3*. Sebastopol, CA: O'Reilly Media, [2017], pp. 349–358.

[29] (2014) Hough circle transform. [Online]. Available: https://docs.opencv.org/2.4.13.7/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html

[30] T.-Y. L. et al., "Microsoft coco: Common objects in context," *ArXiv:1405.0312v3*, 2015.

[31] J. R. Redmon and F. Ali, "Yolo9000: Better, faster, stronger," *ArXiv preprint arXiv:1612.08242*, 2016.

[32] A. de Oliveira Faria. (2017) Yolo dnns. [Online]. Available: https://docs.opencv.org/3.4.0/da/d9d/tutorial_dnn_yolo.html

[33] claudiu. (2016) Object detection with surf, knn, flann, opencv 3.x and cuda. [Online]. Available: http://www.coldvision.io/2016/06/27/object-detection-surf-knn-flann-opencv-3-x-cuda/

[34] A. Kaehler and G. Bradski, "Keypoints and descriptors," in *Learning OpenCV 3*. Sebastopol, CA: O'Reilly Media, [2017], pp. 493–583.

[35] B. Siciliano and O. Khatib, "7.3.2 single-query planners: Incremental search," in *Springer handbook of robotics*, 2nd ed. Berlin: Springer, [2016], pp. 143–144.

[36] Vagran. (2018) Opencv: Rigid transformation between two 3d point clouds. [Online]. Available: https://stackoverflow.com/questions/21206870/opencv-rigid-transformation-between-two-3d-point-clouds

[37] B. Siciliano and O. Khatib, "Kinematics," in *Springer handbook of robotics*, 2nd ed. Berlin: Springer, 2016, pp. 29–33.

[38] Iai kinect2. ETARA. Aschaffenburg. [Online]. Available: 10.205.202.188/Dokumentation/html/a00132.html

[39] (2017) Opencv: Installation in linux. Open Source Computer Vision. [Online]. Available: https://docs.opencv.org/3.3.1/d7/d9f/tutorial_linux_install.html

[40] J. Redmon and A. Farhadi. Installing darknet. Darknet. [Online]. Available: https://pjreddie.com/darknet/install/

[41] M. Kintel. Openscad - downloads. [Online]. Available: http://www.openscad.org/downloads.html

Appendix A

Acronyms

AI Artificial intelligence. 6

CCS camera coordinate system. 25

CDOF Controllable degree of freedom. 19

COCO Common Objects in Context. 24

DH Denavit-Hartenberg. 15, 36

DNN Deep neural network. 2, 23

DOF Degree of freedom. 19, 37

FLANN Fast Library for Approximate Nearest Neighbors. 26

FoV Field of view. 9, 12, 13, 41, 62, 65

IK Inverse kinematic. 19

IR Infrared. 13, 14

KDL The Kinematics and Dynamics Library. 20

mAP mean Average Percision. 25

RAN Robot Assistant for Nurses. 7

RANSAC Random sample consensus. 26

RCS robot coordinate system. 42, 43

RGB-D Red, green, blue - depth. 14

ROS Robot Operating System. 6, 14, 17, 18, 20, 57, 60

RRT rapidly-exploring random tree. 28, 50

SURF Speeded-Up Robust Features. 25

SVD Singular Value Decomposition. 30

ToF Time of Flight. 14

UCD User centered design. 6, 49

WHO World Health Organization. 3, 49

Appendix B

Practical procedures

B.1 Installation of needed ROS packages

The necessary commands to install needed ROS packages are in the following listing.

B.1.1 ROS and its basic packages

Listing B.1: Commands for instaling ROS with additionally needed packages.

```
{

sudo apt-get update
sudo apt-get install ros-kinetic-desktop-full
sudo rosdep init
rosdep update
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc

sudo apt-get install \
ros-kinetic-audio-common \
ros-kinetic-cmake-modules \
ros-kinetic-convex-decomposition \
ros-kinetic-desktop-full \
ros-kinetic-gazebo-ros-control \
```

```
ros-kinetic-ivcon \
ros-kinetic-joystick-drivers \
ros-kinetic-libuvc-ros \
ros-kinetic-object-recognition-capture \
ros-kinetic-object-recognition-ros \
ros-kinetic-openni-launch \
ros-kinetic-openni2-launch \
ros-kinetic-pr2-controllers \
ros-kinetic-python-orocos-kdl \
ros-kinetic-ros-control \
ros-kinetic-ros-controllers \
ros-kinetic-slam-gmapping \
ros-kinetic-sound-play \
ros-kinetic-urdfdom-py \
libarmadillo-dev \
sudo apt-get install \
ros-kinetic-roscpp \
ros-kinetic-pluginlib \
ros-kinetic-urdf \
ros-kinetic-tf-conversions \
ros-kinetic-joint-state-publisher \
ros-kinetic-robot-state-publisher \
ros-kinetic-xacro \
ros-kinetic-rosdoc-lite

killall -9 roscore
killall -9 rosmaster
roscore
}
```

B.1.2 Katana packages

Installation of Katana in ROS. Please copy the next lines to the command line. The KATANA_TYPE is dependent on the used robot type (the type of robot can be found on the label on the robot base, the last number indicates the configuration of the robot in the last joint. The configuration for this thesis robot is katana_400_6m180.

Listing B.2: Commands for instaling Katana packages.

```
sudo apt-install ros-kinetic-katana-driver
export KATANA_TYPE="katana_450_6m180"
```

B.1.3 Kinect packages

Please use the following listing (source [38]) to install the Kinect driver and to test if everything works properly also with ROS.

Listing B.3: Commands to install Kinect packages.

```
git clone https://github.com/OpenKinect/libfreenect2.git
sudo apt-get install build-essential cmake pkg-config
    libturbojpeg
libjpeg-turbo8-dev
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libglfw3-dev
sudo apt-get install beignet-dev
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/usr/local -DENABLE_CXX11=ON
make
sudo make install
# test if it is possible to connect to Kinect
libfreenect2/build/bin/Protonect
# test if you can connect with Kinect using ROS if you have
# a graphics card in the computer you do not need to type
# _depth_method:=cpu _reg_method:=cpu
#please run each of following commands in a new command line
    window
roscore
rosrun kinect2_bridge kinect2_bridge _depth_method:=cpu
    _reg_method:=cpu
rosrun kinect2_viewer kinect2_viewer ir cloud
```

B.1.4 Other third side libraries

The additional two third side libraries that are required is Darknet and OpenCV (for the compatibility use version 3.3.1). Installing and compiling the OpenCV depends on your system and needs many dependencies to be installed. The installation of OpenCV 3.3.1 for Linux is described in [39]. The Darknet can be downloaded and installed by typing commands from following listing[40].

Listing B.4: Commands to install Darknet.

```
git clone https://github.com/pjreddie/darknet
# Then change in the makefile OpenCV flag to OPENCV=1
cd darknet
```

```
make
./darknet
wget
# download pretrained weights https://pjreddie.com/media/files/
    yolov2.weights
# test if the detectoris running
./darknet detect cfg/yolov2.cfg yolov2.weights data/dog.jpg
```

B.2 Connect Katana using local network

For connecting Katana robot using a USB cable with a computer is needed to set the computer IP address to 192.168.0.1. Katana will be found on IP address 192.168.1.1 or 192.168.168.232. In order to connect Katana robot using an Ethernet cable it is needed to set the computer IP address to 192.168.168.200. Also, Katana will be found on address 192.168.168.232.

B.3 Launch Katana

In order to start ROS, open the command line and write the command from the following listing. It calls the collection of most important nodes from ROS

Listing B.5: Launch ROS.

```
{
roscore
}
```

After that, we need to start node Katana 450. We need to type

Listing B.6: Launch Katana.

```
{
roslaunch katana katana.launch
}
```

Try if Katana is connected properly and the drivers are installed correctly. After typing the following command in the command line, it is possible to move with Katana using keyboard keys.

Listing B.7: Launch Katana control using the keyboard.

```
{
roslaunch katana_teleop katana_teleop_key.launch
}
```

B.4 Preparation of the camera calibration

For the camera calibration it is needed to use the calibration ball installed in the gripper of the robot (see figure B.1). If there is no calibration ball available,

Figure B.1: Kalibration ball

it can be printed using a 3D printer (In this thesis, material PLA with a layer height of 0.4 mm was used.). The files that need to be printed can be generated using Vision_guided_handling_operations/code/models/kalibrationModul.scad. The calibration model is divided into three parts because then there is no need to use supports during printing. The printed parts can be stuck together using a heat gun. Both parts to be glued must be heated in the area where they will be glued together.

The file kalibrationModul.scad can be found in the attachment files, dimensions of the calibration module can be changed in the file kalibrationModul.scad and a new *.stl files need to be generated. In order to draw subparts of the calibration model, please follow this description.

Please choose which part will be drawn. The model was divided into three

parts for easier 3D printing:
upper hemisphere (command choosePart("upper",1));
bottom hemisphere (command choosePart("bottom",1.2));
plug to the katana (command choosePart("bridge",1));
whole model (command choosePart("all",1));
Where the first argument means the part of the model that will be drawn and the second describes the scale of the plug to the Katana. The second argument is as default one. However, for the bottom hemisphere, it needs to be more than one, depending on used material and settings for 3D printing.

Opensource software OpenSCAD can be used to open the *.scad files and regenerating the *.stl files (available installation: [41]). OpenSCAD has an advantage that the 3D model is generated from the code.

B.5 3D model of EtaKatana for fitting the parameters

A 3D model was created for a better selection of the height and length of the gallows concerning the FoV of the Kinect (see figure 2.6). Moreover, it is also possible to fit the range of additional vertical translate joint for the outer working envelope (see figure 2.7). The 3D model can be found in file Vision_guided_handling_operations/code/models/EtaKatana.scad.[1] To fit the parameters mentioned above, please follow the description in next paragraph.

The Camera parameters that can be changed:
highOfAdditionalJoint simulates the additional vertical translate join in different height,
gallowHigh height of the gallows where there is the camera and
gallowLength symbolises the distance from the vertical part of the gallows to the camera in y-axis,
visibleFeature can be 'beams' for camera beams or 'envelope' robot outer working envelope.

The parameter highOfAdditionalJoint can help us with visualisation to choose fitting parameters for the upper limit of the additional joint. The lower limit is on the upper surface of the vehicle.

[1] Used OpenSCAD (available installation: [41]).

B.6 Implementation of the gallows

The gallows are made of aluminium profiles. The gallows are designed for Kinect looking directly down on the working area (see picture B.2). The Kinect is mounted on a metal sheet because of the need to change the screw thread type and then the metal sheet was mounted to an aluminium profile using two screws. The Kinect is mounted to the metal sheet using an additional nut. This mounting can bring us a safer solution for Kinect.

Figure B.2: Katana with gallows for the Kinect

B.7 Adding a new user-defined object

To add a new picture it is needed to make a photo of the object, downscale it to about ten per cent of the camera resolution and insert it in folder relative to the project EtaKatana2 /data/user_defined_objects.

After that it is necessary to write the name of the object to /data/user_defined_objects/user_defined_objects.names and then write the object size in z-axis in centimetres into the file measurements.ext in the form name: double_number. Then it is possible to recognise the object and also grasp the object if it will lay up with the photographed side.

B.8 Compilation

For compilation of the program EtaKatana go to the folder /EtaKatana2 and type

Listing B.8: Compilation of the EtaKatana project.

```
{
catkin_make
}
```

If the compilation is unsuccessful because of missing header files, comment compilation of all nodes in CMakeLists.txt in package vision_part and also in package robot_part and compile the project again using the following command.

Listing B.9: Compilation of the EtaKatana project with instalation of services and messages.

```
{
catkin_make install
}
```

Then uncomment compilation of all nodes and compile again using

Listing B.10: Compilation of the EtaKatana project.

```
{
catkin_make
}
```

For compilation of the robot_part package documentation go to the folder /EtaKatana2/src/robot_part and type the following command.

Listing B.11: Compilation of the documentation.

```
{
rosdoc_lite .
}
```

Repeat the same procedure for compilation of the vision_part package documentation.

B.9 Launch programs

B.9.1 Calibration launch

Do the calibration again every time the position of the robot or the position of the camera are changed, or when it seems that the robot is less precise then before.In case of minimum changes in the position of the robot and camera no code changes will be necessary. In case of a different camera robot position it will be necessary to change the calibration points in the file calibration.cpp (variable cv::Mat_<double> end_effector_position_matrix) to be in the camera FoV. When you run calibration after pick and place or backwards, please run the go_home before to avoid collision.

Listing B.12: Commands for starting the calibration.

```
{
roscore
rosrun kinect2_bridge kinect2_bridge _depth_method:=cpu
    _reg_method:=cpu
roslaunch katana katana.launch
rosrun robot_part my_inverse_kinematic
rosrun robot_part my_forward_kinematic
rosrun vision_part recognize_and_show_object circles
rosrun vision_part calibration 1 #testing
rosrun vision_part calibration #training
}
```

B.9.2 Pick and place launch

The pick and place task using the vision is the main task of the master thesis. It includes multiple subparts of the vision and also robot part. In order to run this task it is necessary to launch all of the folowing comands, if the files do not run already. Moreover, make sure that roscore was run as the first one. When you run calibration after pick and place or backwards, please run the go_home before to avoid collision.

Listing B.13: Commands to pick and place an object.

```
{
roscore
rosrun kinect2_bridge kinect2_bridge _depth_method:=cpu
    _reg_method:=cpu
roslaunch katana katana.launch
rosrun robot_part my_inverse_kinematic
rosrun robot_part my_forward_kinematic
rosrun vision_part depth_on_RGB_coordinates
rosrun vision_part recognize_and_show_object
rosrun robot_part pick_and_place circles 1050 867 # run for
    grasping can
rosrun robot_part pick_and_place ibuflam 715 960 #run for
    grasping box with medicaments
}
```

B.9.3 Additional files

Listing B.14: Commands to send the robot to the home position, for example to have a good point of view.

```
{
roscore
roslaunch katana katana.launch
rosrun robot_part my_inverse_kinematic
rosrun robot_part go_home
}
```

Listing B.15: Commands to detect an object.

```
{
roscore
rosrun kinect2_bridge kinect2_bridge _depth_method:=cpu
    _reg_method:=cpu
rosrun vision_part recognize_and_show_object ibuflam
#when without argument then circles otherwise it will choose
    the detection method dependent on the argument
}
```

B.10 Content of the attached DVD

Three main folders can be found on the enclosed DVD.

1. folder /thesis: There is the text of the thesis.
2. folder /code: There are two subfolders EtaKatana2 where there is the program of the whole project described in Doxygen documentation and README.md. The ROS based implementation is written in C++ and consists of vision and robot package for better modularity. The folder models includes the proposal for the mobile manipulator, in a form of the 3D model and calibration the ball.
3. folder /attachment/videos: There are the example videos of the pick and place task for a box with medicaments and a can.

Appendix C

Flowchart

The Flowchart (the attached A2 paper) shows how the robot will proceed during looking for an object and then handing the object over to the user. The green inscriptions represent the planned tasks for the master project, and the blue one represents possible extensions of goals if there is enough time left.

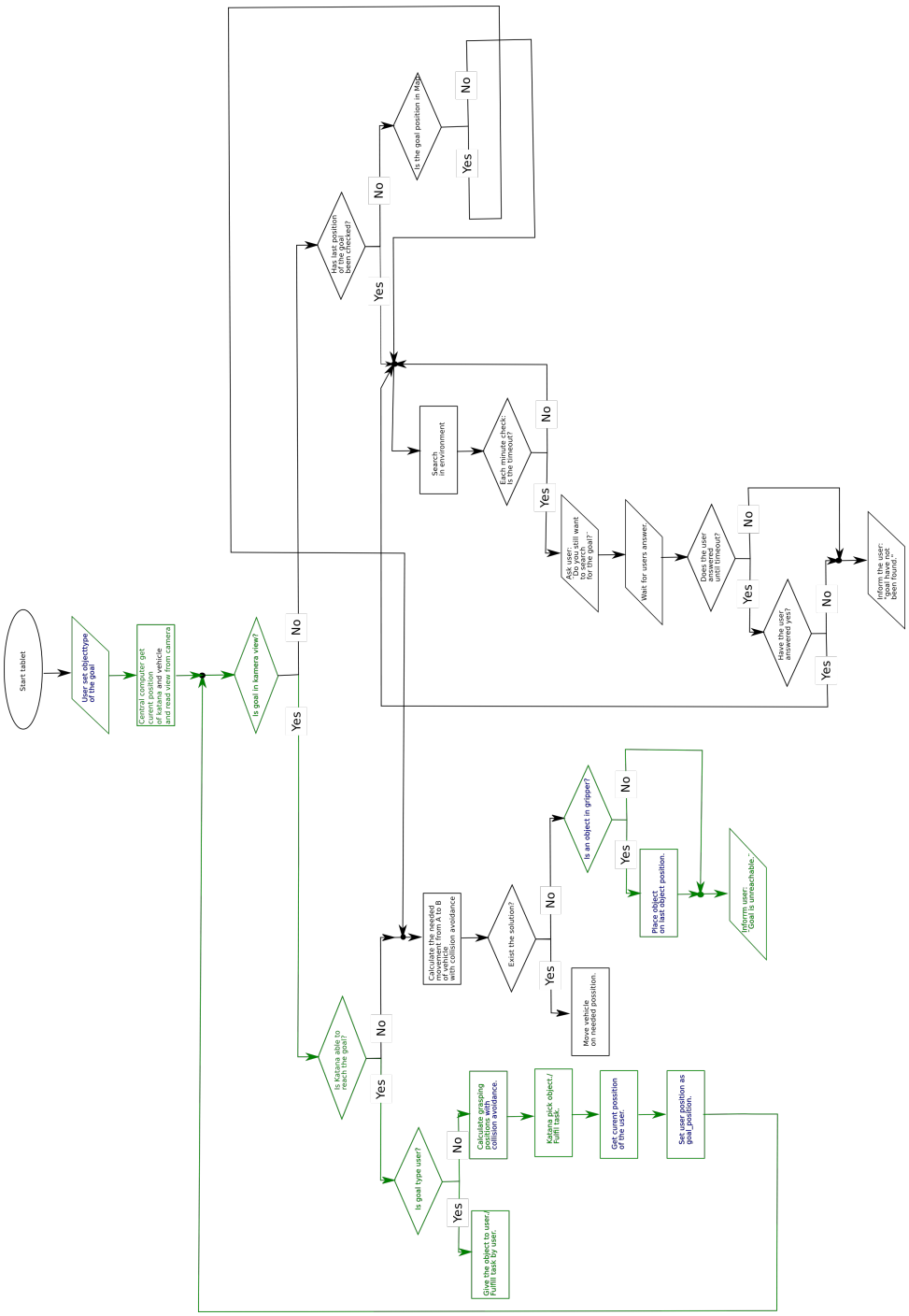


Figure C.1: Flowchart of vision of the project

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kožnarová Zuzana** Personal ID number: **434934**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Vision-guided handling operations for assistive robots in un-structured environments

Master's thesis title in Czech:

Viděním-řízené manipulační operace pro asistivní roboty v ne-strukturovaných prostředích

Guidelines:

In the course of the master project, vision-controlled handling operations in the field of assistive robots will be prototypically developed and tested on the basis of an existing laboratory facility. The problem setting is characterized by an unstructured environment, as given in typical application scenarios of assistance robots in hospitals and care facilities. The problem of bin-picking should also be addressed in this context.
Goal / planned procedure:
The laboratory system is based on a Katana 450 robot arm from the company Neuronics. In the past, gripping tasks have already been solved with this robot, however, this was still done with the native control software KNI. Meanwhile, the arm is controllable via ROS and MoveIt. This new software platform will be used for the master project. In addition, at the beginning of the work a suitable vision sensor (e.g., Kinect) shall be selected and implemented.
Method:
· Requirements analysis and definition of exemplary application scenarios
· Literature research and concept development
· Selection and implementation of suitable algorithms for object recognition and grip planning
· Automatic movement planning for gripping and depositing the detected workpieces
· Testing of the demonstrator and identification of possible applications in the health and care sector.
Scientific claim:
The detection and collision-free and secure gripping of handling objects in unstructured environments is a challenging task in the field of robotics, which requires the application of the latest scientific findings.

Bibliography / sources:

[1] Koubaa, Anis (ed.) (2016): Robot Operating System (ROS). The Complete Reference (Volume 1). 1st ed. 2016. Cham s.l .: Springer International Publishing (Studies in Computational Intelligence, 625).
[2] · Buchholz, Dirk (2016): Bin Picking. New Approaches for a Classical Problem. 1st ed. 2016. Cham, s.l .: Springer International Publishing (Studies in Systems, Decision and Control, 44).
[3] Corke, Peter (2017): Robotics, vision and control. Fundamental Algorithms In MATLAB Second, Completely Revised, Extended And Updated Edition. Cham: Springer (Springer tracts in advanced robotics, 118).
[4] Fernando; Frese, Christian (2016): Machine vision. Automated Visual Inspection: Theory, Practice and Applications. 1st edition 2016. Berlin: Springer.

Name and workplace of master's thesis supervisor:

Prof. Dr.-Ing. Hartmut Bruhm, University of applied sciences Aschaffenburg

Name and workplace of second master's thesis supervisor or consultant:

doc. Ing. Tomáš Svoboda, Ph.D., Vision for Robotics and Autonomous Systems, FEE

Date of master's thesis assignment: **04.12.2018** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until:
by the end of summer semester 2019/2020

Prof. Dr.-Ing. Hartmut Bruhm Supervisor's signature	prof. Ing. Michael Šebek, DrSc. Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature