AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

FACULTY OF ELECTRICAL ENGINEERING, AUTOMATICS, COMPUTER SCIENCE AND BIOMEDICAL ENGINEERING

DEPARTMENT OF AUTOMATICS AND BIOENGINEERING

Bachelor of Engineering Thesis

Design and development of the system for data acquisition and analysis for the mobile platform with a set of two manipulators.

Projekt i wykonanie systemu akwizycji i analizy danych dla mobilnej platformy z zespołem dwóch manipulatorów

Author:	*Jakub Olesiński*
Degree programme:	*Automatics and Robotics*
Supervisor:	*dr inż. Paweł Rotter*

Kraków, 2014

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Contents

Introduction

The aim of this work is to design and implement a mobile manipulation robotic platform with a vision system for obstacle detection and object recognition with a three dimensional camera. This work is a part of a project named "Mobile set of manipulators on a wheeled chassis". The project was realized in a three person team, as part of the Second Edition of ABB Students Scientific Association programme organized by ABB Corporate Research Center in Cracow.

In the first chapter, a design of the robotic system is presented. General concept of mobile manipulation is described and possible application areas are mentioned. Subsequently, design requirements, selected hardware components, and implemented software architecture is described. The second chapter provides information about modern depth map acquisition techniques, including stereo, time-of-flight and structured light cameras. For each method, its operation principle basics and general advantages and disadvantages are provided. The third section is focused on modern methods of depth image analysis. Concepts of point clouds and point descriptors are introduced. Moreover, general object-recognition pipeline is provided.

The final section presents an implementation of selected object recognition method. Test results in the real environment are provided. Possible future improvements to the algorithm are also mentioned.

1. Mobile manipulation system design

1.1. Project background

A mobile manipulation system (MMS) is a robotic system that is capable of both manipulating objects and locomotion. Typically, the system is composed of a robotic arm based on a robotic mobile platform. Such configuration enables the manipulator to operate in an unlimited workspace, thus providing great application opportunities.

Typically, mobile manipulators are designed to relieve people in hostile situations. They are, for example, widely used in the field of chemical, biological, radioactive or nuclear (CBRN) defense. Explosive materials and other hazardous substances can be disposed without exposing operators to any danger. Another example is space exploration, where manipulation systems are used in planetary rovers in unmanned exploratory missions on other planets. Substitution of human operators in such expeditions significantly reduce their expenditure and risk.

Augmenting the MMS design with a certain degree of autonomy, brings the possibility for the robot to be used as a human-assistant in the household. Most of current applications in this field refer to pick up and delivery services, that have the potential to improve lives of the elderly, injured and disabled people []. Furthermore, typical service robots in human environments are dedicated to accomplish only a single task, such as vacuum cleaning, lawn mowing, pool cleaning or window washing. Their operation is achieved by adjusting existing domestic appliances with a degree of autonomy. With a multi purpose robotic system such as a mobile manipulator, it is potentially possible not to replace existing devices but to substitute the human operator.

Autonomous gripping and transportation with a MMS could also be used in the industry for designing flexible factory plants and intelligent warehouses []. A manufacturing process that utilizes the MMS robots is simpler and more cost effective in adapting to the market changes. The production could be more customized by applying MMS robots in transportation, preparatory and post-processing tasks. Furthermore, even the most advanced distribution centres, such as the Amazon Kiva Systems [kiva?], require repetitive human work to manipulate goods in the unstructured environment. Therefore, the warehousing process could benefit by applying autonomous MMS robots and there are ongoing efforts in this area [amazonchallenge].

One of the most promising applications of MMS is the PR2 development platform from the Willow Garage company. PR2 is a human-sized robot, equipped with two 7 degree-of-freedom manipulators,

omnidirectional mobile base, a processing power of two Xeon servers with 8 cores and 24GB of RAM each, and a multitude of sensors, including stereo, time-of flight and structured light cameras, LIDAR scanners, and an inertial measurement unit [prspecs]. The PR2 robot has already proven successful in such dexterous experiments as opening doors, playing pool, folding towels and serving beverages to people[prapps].

Wide functionality of a MMS robot makes its design a challenging task. In MMS robots, the unstructured environment and additional degrees of freedom complicate the control task. Furthermore, the workspace of a manipulator is often shared with people and other vehicles. Such work environment renders many potentially hazardous situations. Therefore, it requires a highly sophisticated control system, with vision based feedback.

1.2. Design requirements

In order to clarify the design process, the requirements that should be met by the final construction were defined. The main goals and features of the project are summarized in the following list:

1. The robotic system should consist of two cooperating serial manipulators, based on a common mobile robotic platform.
2. The whole construction should be assembled from easily accessible components, available on the consumer market.
3. The serial manipulators should be equipped with grippers for general object manipulation.
4. The manipulators reachable workspace should allow to reach objects located on the ground.
5. The reachable workspace of each manipulator should also intersect, to allow collaboration on manipulation tasks.
6. The mobile platform should be wheeled and is expected to move only on flat surfaces at indoor areas.
7. The steering mechanism of the robot chassis should be kept simple and convenient to control.
8. The robotic system is expected to provide an interface for remote control, developed in a commonly used and well supported standard.
9. The control interface of the robot is required to provide methods for manipulator joints positioning and for setting velocity and direction of the platform movement.
10. The whole system should be powered by a source that could withstand at least an hour of continuous work.

Furthermore, as the robot is intended for autonomous operation, it should possess a vision based data acquisition system and a computing unit with enough processing power to analyse the acquired data in an online fashion.

1.3. Hardware components

The first stage of the project was to create the mechanical construction of the robot, which would satisfy the established requirements. The obtained model is shown in the Figure 1.1. The robot consists of two serial manipulators with 5 degrees of freedom (DoF). Both of the robotic arms are in the articulated structural configuration and possess only rotational joints. The link lengths for each manipulator are presented in the Table 1.1. Both of the manipulators possess impactive grippers, which open to about 50 mm wide.

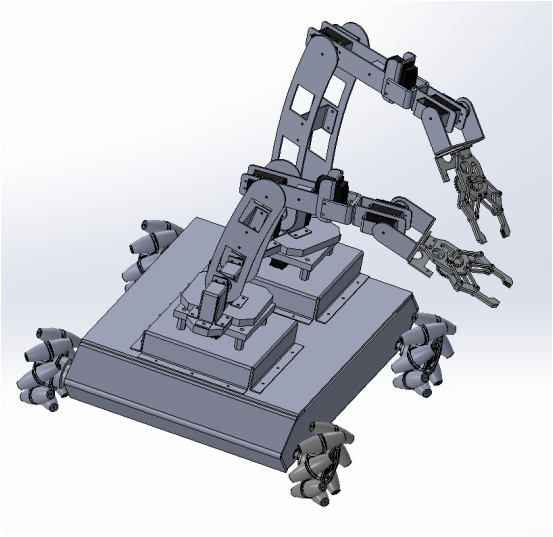

Figure 1.1: Model

The manipulators are based on a four wheeled mobile platform. To avoid complicating the mechanical design of the chassis with a steering mechanism, a special kind of wheels were used. Mecanum wheels [wikipedia], as they are called, are equipped with a set of rollers attached to their circumference at a $45°$ angle, which allow a vehicle to move in any direction without turning the wheels. To achieve the desired steering, the wheels are independently actuated, in a way that the resultant force vectors move or rotate the entire platform in a specified direction. In addition, such mechanism increases the stability of the entire platform, which is particularly important during the manipulation tasks. The physical dimensions of the mobile platform are also provided in the Table 1.1. The entire mechanical structure has been laser cut from aluminium sheets of 1.5, 2 and 4 mm thickness.

Manipulator links (smaller)	20x20
Manipulator links (bigger)	58 °H, 45 °V, 70 °D
Mobile platform	VGA (640x480)

Table 1.1: The MMS robot dimensions

The next step in the construction of the MMS robot required a selection of hardware components, such as the actuators, power supply, embedded computing units and control electronics. The Figure 1.2 presents a categorized diagram of the selected elements.

The MMS robot utilises two types of actuators: four DC motors for each of the mobile platform wheels and the servomechanisms for manipulator joints. The selected DC motors type is the Pololu 1444 [datasheet] with 50:1 metal gearboxes. They achieve 200 RPM of no-load speed and 12 kg cm of stall current. Additionally, they possess 64 CPR encoders, which multiplied by the gear ratio provide a total of 3200 counts per revolution. The type of servomechanism used for each joint, is the Turnigy 1269HV [datasheet], with titanium gears and a total stall torque of 21 kg cm. Both types of actuators are driven by the Pulse Width Modulation signal (PWM). In case of the DC motors, the PWM signal is provided by a dedicated motor driver board, named Rover 5 [datasheet]. This controller provides four low-resistance FET H-bridges, together with current monitoring on each motor channel. To generate the PWM signal for servomechanisms positioning, a 16-channel PWM driver chip, the PCA9685 [datasheet] was used. This driver has an I2C interface for setting the PWM frequency and duty cycle on a selected channel with a 12-bit resolution.

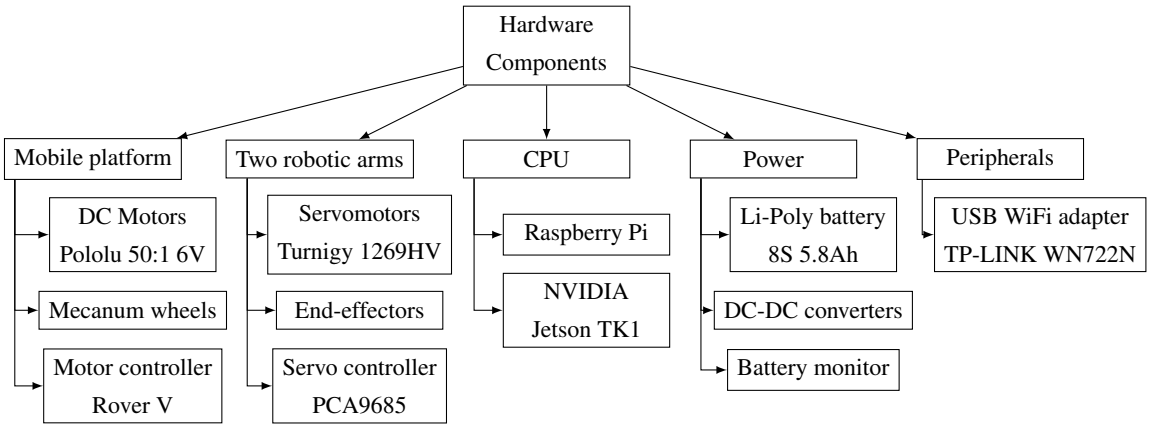

Figure 1.2: Hardware components

The main computing unit, used for hosting the control server and directly interfacing with the actuator drivers, is the model B of Raspberry Pi development platform [datasheet]. This board is equipped with 700Mhz ARM11 family CPU, 512 MB of RAM and a set of GPIO ports with hardware UART, I2C and SPI interfaces. The MMS robot utilises an additional processing unit, with a much larger computing power for the vision system implementation. The NVIDIA Jetson TK1 [datasheet] development board

is based on the Tegra K1 System-on-Chip, that includes a quad-core 2.3 GHz ARM Cortex-A15 CPU, 2 GB of RAM, and a NVIDIA Kepler GPU with 192 CUDA cores, capable of performing over 300 GFLOPS of 32-bit floating point computations. This board provides a similar performance to a modern desktop GPU, while maintaining relatively low power consumption, which makes it a perfect match for image processing applications in mobile systems, such as the MMS robot. **Network setup.**

The whole system is powered by a lithium-polymer, 8 cell battery with a nominal voltage of 29.6 V and capacity of a 5.8 A h. Due to the specific nature of a lithium-polymer battery, each of its cells has to be protected against excessive discharge. For this purpose, a cell voltage monitor is connected to the battery and generates a sound alarm if the voltage falls below a selected threshold. To integrate different operating voltages for the individual components, three DC-DC voltage converters are applied in the power system: one with 6 V output and maximal current of 20 A for supplying all of the actuators, one with the output of 12 V and 4 A max. for the NVIDIA Jetson board, and a 5 V and 4 A max. for feeding the Raspberry Pi and the actuator drivers.

The final construction result in the form of the whole, assembled MMS robot is is presented in the Figure 1.3.

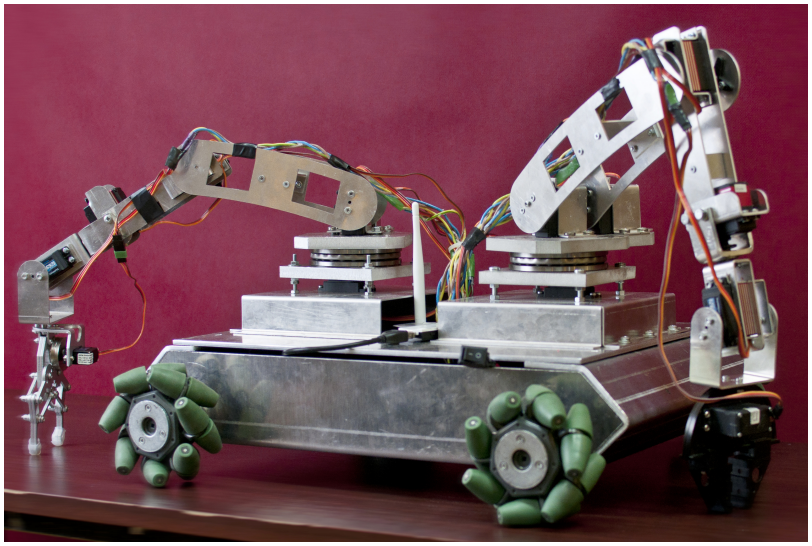

Figure 1.3: The final effect of the MMS robot construction

1.4. Software architecture

The final stage of the MMS robot construction, after the selection and assembly of the hardware layer, was to implement software that would allow for remote control over the system actuators, including manipulator joint positioning and setting the velocity and direction for each motor. To fulfil all of the preset requirements, a HTTP control server providing a web services interface has been created on the Raspberry Pi platform. The flow of information in such system is presented in the Figure 1.4.

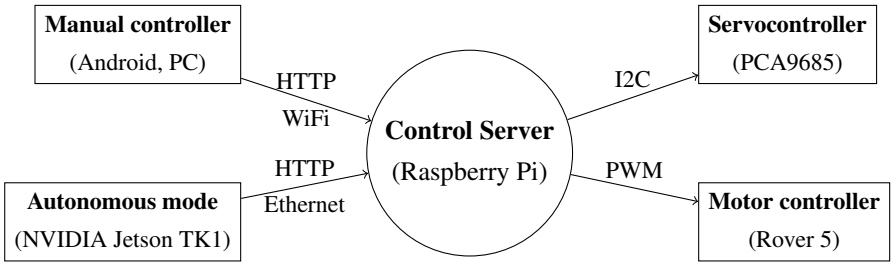

Figure 1.4: Data flow between the MMS robot components

A Representational State Transfer (REST) software architecture design pattern was used during the implementation of the control server. In this approach, the HTTP methods like GET or POST, which are typically used by the web browsers, serve as the communication interface between different devices. The web service API, that conforms to the REST pattern is composed of a base uniform resource identifier (URI) followed by a path to a given resource. In case of the MMS robot, the base URI has the form of: *http://ip:port/Robot/*. A list of example request supported by the implemented web service is presented in the Table 1.2. The control server was developed in the Python programming language, with the use of the Webiopi [??] Internet of Things framework. The Webiopi provides a convenient set of tools to share the peripherals of the Raspberry Pi development platform via the HTTP interface, compatible with the REST architecture.

Position of the manipulator *id*	
Position of the manipulator *id*	58 °H, 45 °V, 70 °D
Position of the manipulator *id*	VGA (640x480)

Table 1.2: Example HTTP requests

2. Depth data acquisition techniques

The manipulation and movement tasks of a MMS robot involves continuous interaction with unstructured and changing environment. In order to develop an autonomous system, which will be able to operate in such conditions, a data acquisition system is required, that would provide the necessary information about the nearby objects surface, size and relative distances. The measurement of this type can be performed by using depth acquisition techniques, also known as three dimensional (3D) imaging.

The first depth acquisition techniques emerged as a replacement for a contact-based coordinate measuring machines (CMM). CMMs were used in the industrial quality control applications and worked by recording the displacement of a probe tip sliding across a solid surface. Such method was time consuming and inadequate for fragile objects. Modern, contact-less 3D scanning apparatus overcome those limitations by using light to interact with the environment. The new technology had also extended the application area of 3D scanning to the field of robotic perception.

Modern methods of 3D data acquisition are classified by the light source they utilize to measure depth. *Passive* techniques rely only on the ambient light, whereas the *active* ones operate by projecting illumination onto an object and recording the reflected beam. In the following sections, examples of both categories are presented, with a brief description of their principles of operation and a discussion of advantages and disadvantages, finalized by hardware selection for the designed MMS robot.

2.1. Stereo vision

Stereo vision is a passive depth acquisition technique, widely used in research and industry. Its principle of operation is based on human vision system and the biological process of stereopsis, where the disparity between two different projections of the world on the human retinas leads to the depth sensation. Analogically, in computer stereo vision technique, two (or more) displaced in space cameras concurrently acquire the scene view. From the captured images disparity, a scene depth information is then inferred. A typical scheme for stereo vision depth calculation after image acquisition is divided into two steps: the correspondence problem and triangulation.

The correspondence problem is the most difficult part of the stereo vision. It can be stated as follows: given two displaced 2D images of the same scene, find points representing the same space location in those images. There are many point matching algorithms discussed in the literature. Some of them rely on computing point features, based on their neighbourhood, while others compute statistical descriptors

of characteristic areas in the image, and then minimise the measured difference between analysed regions to find the best matches. The correspondence problem in computer vision is a wide and open research area, that is beyond the scope of this work. A thorough description of the problem and its solutions, is provided, for example, by [1].

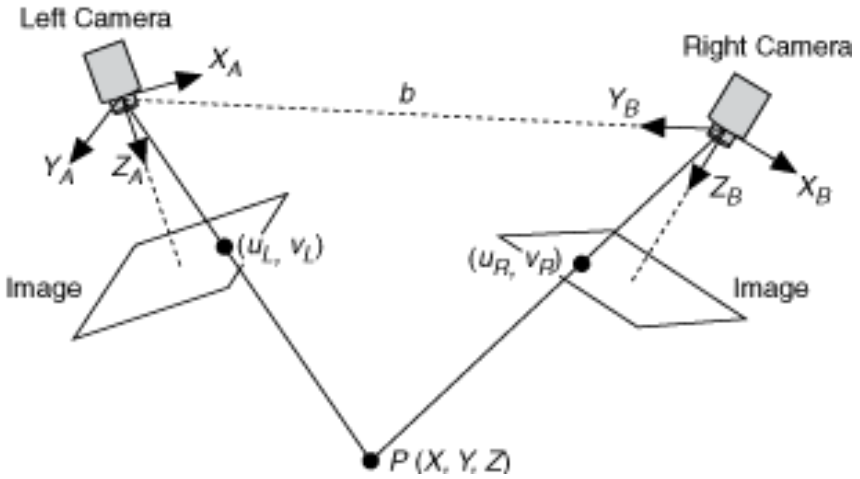

Figure 2.1: Typical Stereo Vision System

After obtaining the corresponding points, the point depth information can be derived by the means of triangulation (Figure 2.1). if the imaging properties of the camera are known, two three-dimensional lines, from the camera's projection centres to the examined point can be drawn. The intersection of those lines is then used to infer the depth of the point.

The main advantage of the stereo vision system is that it can be built with easily accessible, standard 2D cameras. Such cameras allow to reduce the expenditure and provide relatively high resolution. On the other hand, the stereo vision systems have many drawbacks. Firstly, their performance is reduced in environments with low ambient light intensity, which makes the system impractical in some settings. Secondly, the depth accuracy decreases quadratically with the distance. Finally, the algorithms used for feature extraction and matching, involved in solving the correspondence problem are computationally expensive and often limit the depth acquisition frame rate.

There also exists a variant of active stereo vision, utilized i.e. in the Ensenso N10 cameras [2]. In this case, besides the camera pair, a pattern projector is applied to add artificial texture to the scene, which helps to determine the disparity on untextured surfaces.

2.2. Time of flight

Time of flight (ToF) cameras work on a completely different principle than stereoscopic systems. ToF systems are characterised with active illumination and are composed of an near infra-red (IR) emmiter and IR camera. They work by illuminating the scene with a modulated IR beam and recording the

received light, as presented in the figure 5. The distance from the recorded object is then calculated by measuring the phase shift between the illuminated and reflected signals.

In the simplest form, the ToF cameras use single light pulses [3]. After a short period of illumination, the reflected energy is then integrated at every pixel with two out of phase windows over the same Δt time, resulting in two electrical charges Q_1 and Q_2. The depth distance d is then computed as $d = \frac{1}{2}c\Delta t\frac{Q_2}{Q_1+Q_2}$, where c is the speed of light constant. A more advanced method is to project illumination modulated by a continuous wave signal, commonly a square wave. In this case, four sampling windows, shifted by 90° angle, resulting in Q_1,Q_2, Q_3 and Q_4 charges. The phase angle ϕ between the projected and the reflected signals is given by $\phi = arctan\frac{Q_3-Q_4}{Q_1-Q_2}$ and the final depth $d = \frac{c}{4\pi f}\phi$. The principle of the continuous wave method is illustrated by the Figure 2.2.

ToF cameras, due to their specific architecture, are prone to errors caused by radiometric, geometric and illumination variations. The power of the emitted IR signal limits their measurement accuracy. The light entering to the sensor has an ambient and reflected components, thus high ambient light intensity reduces the signal to noise ratio. Moreover, the material and color of the object surface cause variations in the amplitude of the reflected IR signal.

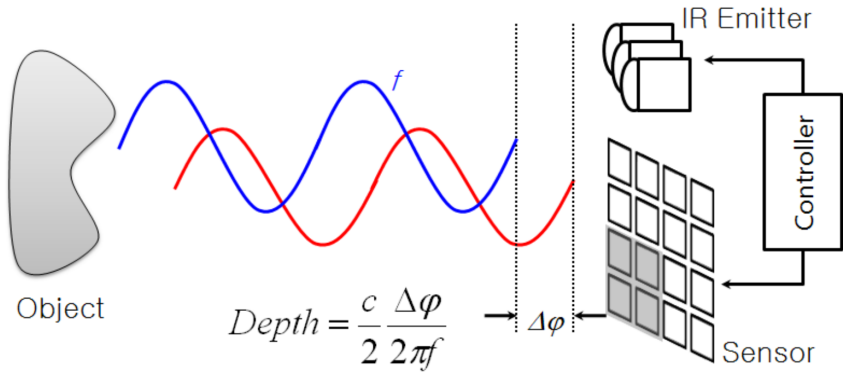

Figure 2.2: Continuous-Wave Time of Flight principle [4]

Despite problems arising from ToF principle of operation, modern ToF cameras are distinguished by relatively low latency and fast scanning speed. The depth measurements are acquired directly from the hardware, so the speed is not limited by software processing. Unfortunately, most of the products available on the end-user market provide relatively low depth image resolution, typically QQVGA (160x120) and the price increases significantly with the resolution.

2.3. Structured light

The structured light (SL) depth measurement technique combines some of the features of both time of flight and stereo vision principles. Similarly to ToF cameras, SL relies on active illumination, work in the near infra-red and is composed of an IR emitter and IR camera. As in the case of stereo vision, depth information is inferred from the disparity between two images by the means of triangulation. In this

case, the projected IR pattern is compared with the image captured by the IR camera. The IR projector emits patterns of non-coherent light, which then appears distorted from the perspective of the camera. In such settings, the projection of defined patterns makes explicit correspondences on the reflected image. A typical setting of the SL system is presented in figure 5.

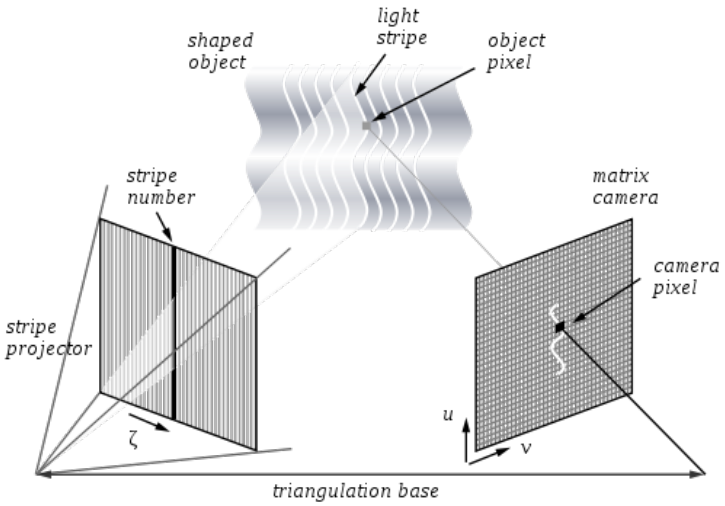

Figure 2.3: Structured light fringe patterns [5]

There are many pattern strategies that allow for correspondence identification, including projections of grids, dots, vertical slits and multi-color patterns. In the literature, particular attention is paid to fringe patterns (Figure 2.3), which are suitable to maximize the measurement resolution [6]. Moreover, to reduce the reconstruction artifacts, the measurement process is often extended into a sequence of different pattern projections. A comprehensive assessment of such pattern codes can be found in [7].

Depth measuring devices that employ the SL technique suffer from many drawbacks. Utilizing multiple pattern frames introduces high latency and makes the measurement ill conditioned for dynamic scenes. Moreover, the system performance is degraded in bright ambient light. On the other hand, popular SL devices are characterized with relatively high, VGA (640x480) depth map resolution and are easily accessible on the consumer market.

A particularly noteworthy application of SL based depth data acquisition device is the Kinect by Microsoft. In addition to an RGB camera and an array of microphones, the device includes an IR projector-camera pair from the PrimeSense Ltd. company, used for depth measurements. The SL light pattern used in Kinect is a non-periodic speckle pattern produced by the interference of partially coherent beams (Figure X). In this device, every pixel is identified in the IR image using a correlation window, after which the depth information is calculated, using triangulation. The Kinect device is produced as a gaming controller for the Xbox 360 console and it is widely available at the consumer market with a relatively low price. Moreover, there are many open source drivers for the Kinect device, which make it a perfect match for robotic and research applications.

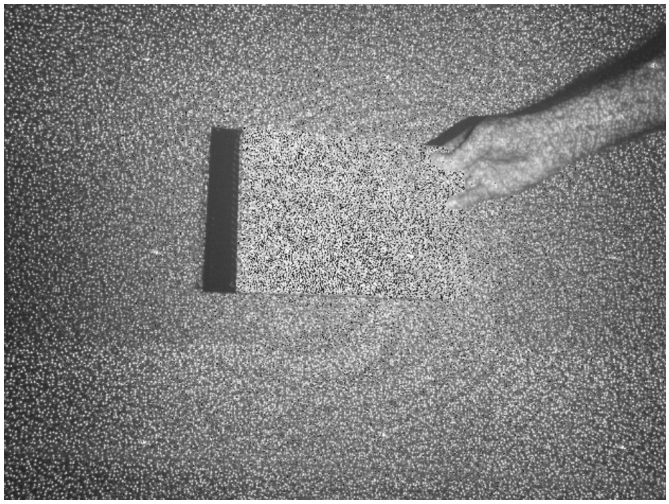

Figure 2.4: Kinect speckle pattern [8]

2.4. Summary and hardware selection

Three of the most popular depth map acquisition techniques were presented in this chapter, including stereo vision, time of flight and structured light. Each method has its advantages and disadvantages and they all have already been successfully applied in robotics. A comparative summary of each method characteristics is presented in Table 2.1.

Feature/Technique	Stereo Vision	Time of Flight	Structured Light
Depth data generation	Directly out of chipset	High software processing	Medium software processing
Latency	Medium	Low	Medium
Low light performance	Weak	Good	Good
Bright light performance	Good	Medium	Medium
Power consumption	Low	Medium - scales with distance	Medium - scales with distance
Resolution	Camera dependent	QQVGA, QVGA	VGA, 1080p
Accurracy	mm, cm	mm, cm	µm, cm
Scanning speed	Medium - limited by software complexity	Fast - limited by sensor speed	Fast - limited by camera speed

Table 2.1: Qualitative comparison of depth acquisition techniques [3]

In order to fulfil project assumptions, one solution providing depth map measurement had to be

chosen. In the stereo vision technology, ready-made devices unfortunately did not fit within the project budget. Utilizing this technique would therefore require an own design of the stereo vision system. Such solution would probably be the least expensive, but require an extensive amount of work and time. Therefore, the author focused on finding a ready-made solution in the remaining technologies. The DepthSense 311 [3] device was found as a representative of the ToF technique, and the Asus Xtion Pro Live [4] was considered as an option from the SL method. Both devices are available in similiar prices and provide the same frame rate of 30 frames per second. The Asus device, however, offers higher depth map resolution (VGA instead of QQVGA) and has better support from the open source community. The Asus Xtion is based on the PrimeSense Ltd. depth sensing hardware, similarly to the popular Kinect device and is even compatible with the same open source drivers. Nonetheless, it was chosen over the Kinect device, because it has smaller size and is powered directly from the USB port (the Kinect requires external power source). Main specifications of the Asus Xtion Pro device are provided in the Table 2.2.

Operating range	Between 0.8m and 3.5m
Field of view	58 °H, 45 °V, 70 °D
Depth map resolution	VGA (640x480) : 30 fps QVGA (320x240): 60 fps
Interface	USB2.0/ 3.0
Dimensions	18 x 3.5 x 5 cm

Table 2.2: Asus Xtion Pro Live specification [9]

3. Analysis of the depth data

The acquired depth information can be stored in computer memory in two ways. The first is the depth map, which takes the form of a two dimensional array, similarly to a plain gray-scale image. In this case, however, the depth measurement is stored in place of the color intensity. Depth map is the simplest way to represent and store the acquired depth of the scene and it is usually obtained directly from the sensor driver. The main disadvantage of a depth map is its inflexibility. This representation is strictly bound to the camera point of view and, thus, it is inconvenient in further processing.

The second representation of a scene's depth information is a point cloud. Generally speaking, it is a set of data points in some coordinate system. Typically, the Cartesian system is used and the points are defined by their x, y and z coordinates. Point clouds are derived from depth maps and offer new capabilities, such as viewpoint transformation or cloud concatenation. The point cloud representation can be rendered in the three dimensional space, which is particularly useful tool during the implementation of the system for depth data analysis.

In order to achieve the autonomous operation of the robot, a set of tools for efficient processing and analysis of the collected data about the environment structure must be established. The following sections present the main tools used further in the implementation of the autonomous control mode for the MMS robot.

3.1. Basic point cloud processing

In a given point cloud, a single point by itself does not provide much information about the surface to analyse. Therefore, an essential concept in depth data analysis is the local neighbourhood of a point. For a query point p_q in the point cloud P its neighbourhood is given by:

$$d(p_q, p_i) \leq r \tag{3.1}$$

where $p_i \in P$ is the neighbouring point, d is the selected metric, typically Euclidean, and $r > 0$ is the neighbourhood radius. In practice, approximate methods are used, as direct application of the definition would require calculation of distance from the query point p_q to all other points in P. The algorithms used for neighbourhood search require a parameter k specifying the maximum number of points in the neighbourhood or parameter r, denoting the maximum search radius. Proper determination of those

parameters is crucial in further analysis. Too small values will not provide enough information about the surface. Too large, on the other hand, will average the surface and skip small details.

A frequently used operation during the processing of a point cloud is the affine transformation. Point clouds can be translated, rotated and scaled by multiplying a transformation matrix $A \in \mathbb{R}^{4x4}$ with data points in homogeneous coordinates $[x, y, z, 1]^T$. Basic transformation matrices are demonstrated in the Figure 3.1. Presented transformations can be further combined by multiplication to produce more complex operations.

$$A_t = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation by a vector $t = [t_x, t_y, t_z]^T$

$$A_s = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling along x, y, z by factors S_x, S_y, S_z

$$A_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) & 0 \\ 0 & sin(\theta_x) & cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around x with θ_x angle

$$A_y = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around y with θ_y angle

$$A_z = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around z with θ_z angle

Figure 3.1: Basic affine transformations

One of the most important stages of data preprocessing is the filtration. Probably the most basic point cloud filter is the pass-through filter, which reject all points outside a given range along a specified dimension. This procedure allows to focus the analysis process on the region of interest, i.e. the reachable workspace of a manipulator. Apart from limiting the cloud dimensions, the number of data points can be also reduced by downsampling. The voxel grid filter is typically used for this purpose. This filter creates a three dimensional regular grid over the input point cloud data and then, in each voxel, approximates all the present data points with their centroid. Such reduction is particularly useful when large point cloud datasets have to be processed online with limited computing resources. Finally, the filtration process has to cope with numerous measurement errors present in the raw data acquired from the 3D camera. Such measurement noise manifests itself in the form of sparse outliers, which corrupt the results of further processing, i.e. the surface normals estimation. The impact of those irregularities can by reduced by applying an outlier removal filter. The simplest form of such filter rejects all the data points which does not have enough neighbours within a specified radius. A more refined version is based on the neighbouring points distance distribution. Firstly, the mean distance from each point to all its neighbours is calculated. Next, based on the assumption that the resulting distribution is Gaussian, all the points

whose mean distances lay outside of an interval defined by the global distances mean and their standard deviation are rejected from the dataset. The effects of statistical outlier removal are presented in the Figure 3.2.

The point cloud segmentation or cluster extraction is the process of grouping together data points that meet some common condition of similarity. Ideally, each of such segmented clusters represents a different object in the scene. A popular segmentation method is based on the euclidean distance. In this approach, two points are considered to belong to the same cluster, if their relative distance is less than a selected threshold. This process is similar to the flood-fill algorithm, popular in the 2D image processing. There are also other varieties of segmentation, i.e. based on the smoothness constraint or colour similarity.

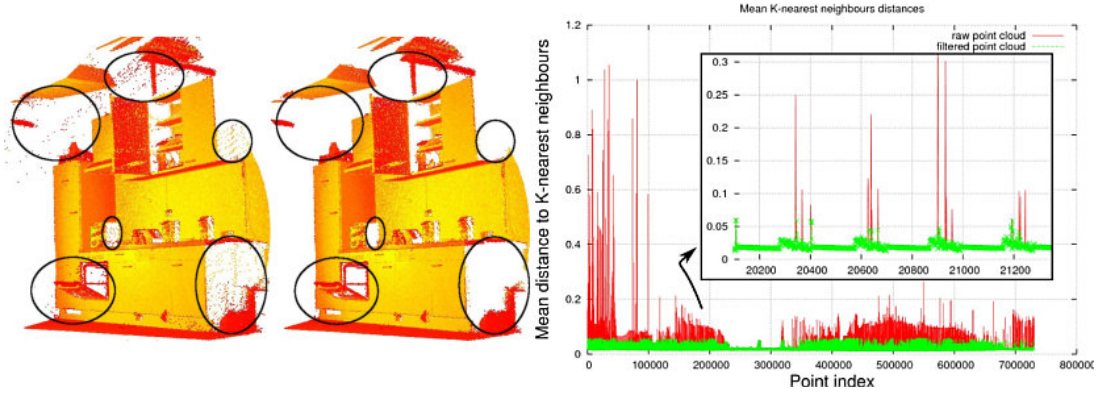

Figure 3.2: Statistical outlier removal [tutorials]

Many of the algorithms used further during the analysis of a point cloud base on the notion of a surface normal, known from the 3D geometry. These vectors are also widely used for shading in 3D computer graphics and a variety of methods have already been developed to solve the surface normal estimation problem. One of the simplest uses the least-squares plane fitting to estimate the normal to a plane tangent to the surface, which approximates the desired vector [10]. More specifically, for a given point on the surface, it's k-neighbourhood centroid $\bar{p}$ is calculated as:

$$\bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \tag{3.2}$$

The tangent plane is thereafter defined by the centroid $\bar{p}$ and the sought normal vector $\vec{n}$. The latter is computed by minimizing the total (squared?) distance from every k-neighbour p_i to the tangent plane, given by $\sum_{i=1}^{k}(p_i - \bar{p}) \cdot \vec{n}$. The minimization problem can be solved by utilizing the covariance matrix $C \in \mathbb{R}^{3x3}$, given by:

$$C = \frac{1}{k} \sum_{i=1}^{k} (p_i - \bar{p}) \cdot (p_i - \bar{p})^{\intercal} \tag{3.3}$$

The covariance matrix C is symmetric, positive semi-definite and possess three real eigenvalues $\lambda_j \geq 0, i = 1, 2, 3$. The eigenvector corresponding to the smallest eigenvalue is an approximation of the desired

normal vector $\vec{n}$, disregarding the sign. Furthermore, if the viewpoint v_p is known, the normal $\vec{n}$ has to be oriented towards v_p, which means that it has to satisfy the equation:

$$\vec{n} \cdot (v_p - p_i) > 0 \tag{3.4}$$

3.2. The Random Sample Consensus algorithm

The indoor human environment is abundant of regularly shaped objects that could be described with basic geometrical models, such as planes, spheres or cylinders. The plane model:

$$A \cdot x + B \cdot y + C \cdot z + D = 0 \tag{3.5}$$

is particularly useful, as the floor, walls or the furniture is typically composed of flat surfaces. In example, by knowing which points of the scene belong to the surface of a floor, the robot can autonomously plan a collision-free path of movement. For this reason, a robust model fitting algorithm is a strongly desired tool in the analysis of the depth data. The point dataset received from the depth camera, however, consists of both points that belong to the model, the inliers, and a lot of other points in the scene, the outliers. Therefore direct usage of classic model fitting algorithms, such as the least squares method, would not provide the desired effect, as they try to fit the model into all the input data points, including outliers. As an alternative, the Random Sample Consensus (RANSAC) algorithm, can effectively cope with such problems. In its basic form, the RANSAC algorithm is essentially composed of two, iteratively repeated steps [11, 12]:

1. Firstly, a minimal sample subset is randomly selected from the input dataset. The model parameters are computed using only the selected subset. The cardinality of the subset is the smallest sufficient to determine the model parameters.

2. Secondly, the remaining dataset points are tested to be consistent with the model computed in the sampling step. A data point will be considered as an inlier if it fits the computed model within a defined error threshold. The set of such elements is called a consensus set. If the consensus set contains enough points, the model is reestimated from all selected inliers and evaluated with the error of the inliers relative to the model.

This procedure is then repeated until a termination condition is met, which usually is a fixed number of iterations. The main advantage of the RANSAC algorithm is the robust estimation. This method is able to fit a model with high accuracy even if the data set contains a significant amount of outliers. On the other hand, the algorithm in its basic form has several disadvantages. There is no upper bound on the time needed to estimate model parameters and by limiting the number of iterations, the obtained solution is may not be optimal. Furthermore, the RANSAC algorithm can only estimate one model per dataset and if multiple models exist, it may fail to estimate any of them. Since the original RANSAC was first published in 1981, it has been widely adapted by the image processing community and many

modifications that address the RANSAC limitations has been proposed. A comparative summary of the recent extensions to the RANSAC algorithm can be found for example in [13].

3.3. Descriptors for object recognition

The object recognition, which involves the identification and localization of a given 3D object in the environment, is a major problem and limitation during the execution of autonomous manipulation tasks. Most of the 3D recognition methods base on some surface characteristics indicators, called features or descriptors. Surface normals are the most basic representation of the geometry around a certain point. Even if coupled with surface curvature, they usually does not provide enough descriptive information for object recognition and pose estimation. To achieve better performance in such tasks, more complex and higher dimensional descriptors have been proposed in the literature [14]. A descriptor is considered to be reliable if it is able to capture the same surface characteristics, regardless of rigid transformations, varying sampling density and noise. In general, 3D shape descriptors are divided into local and global. The former describe only the local geometry around a query point, while the latter represent the geometry of the whole object. A few selected descriptors are presented further in this section.

The Point Feature Histogram (PFH) [10] is a generalization of both surface normals and curvature estimates. It represents the relative orientation of normals between every point pair (p_i, p_j) in the neighbourhood of the query point p_q. For each point pair, using the surface normal n_i at p_i, a new coordinate frame u, v, w is constructed as follows:

$$u = n_i,\ v = u \times \frac{p_i - p_j}{d},\ w = u \times v \tag{3.6}$$

where $d = \|p_i - p_j\|_2$ is the Euclidean distance between p_i and p_j. Using this reference frame, the difference between normals at p_i and p_j is expressed by the angular features α, ϕ, θ, given by:

$$\alpha = v \cdot n_i,\ \phi = u \cdot \frac{(p_i - p_j)}{d},\ \theta = arctan(w \cdot n_i, u \cdot n_i) \tag{3.7}$$

Finally, to create the PFH descriptor, the angular features are binned into a histogram. The angular ranges are typically divided into 5 subdivisions, thus receiving a $3^5 = 125$ binned histogram, that counts occurrences of any value combination for every point pair (p_i, p_j).

The main disadvantage of the PFH descriptor is its $O(nk^2)$ complexity, where n is the number of points in the point cloud and k is the number of each points neighbours. For large datasets, this is one of the major bottlenecks during online processing. To overcome this problem a simplification to the PFH formulation, called Fast Point Feature Histogram has been proposed (FPFH) [15]. In the first step of the FPFH, the angular features α, ϕ, θ are computed only between the query point p_q and its k-nearest neighbours, as described in Equation 3.7. Those features produce a histogram, called Simplified Point Feature Histogram (SPFH). The SPFH is computed for every point in the cloud, and then, the FPFH descriptor is formed as follows:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_i}SPFH(i) \tag{3.8}$$

where w_i is a distance between p_q and p_k in some metric space. The FPFH descriptor reduces the computational complexity of the PFH to $O(nk)$, while maintaining similar descriptive performance.

Figure: Influence diagrams

A further extension of the FPFH descriptor is the Viewpoint Feature Histogram (VFH). It is a global type of descriptor, resulting in only one histogram for a given cloud. The VFH utilises the viewpoint p_v vector direction to add viewpoint variance to the FPFH. In this case, the angles α, ϕ, θ from 3.7 are computed between the entire point cloud centroid p_c and each of the points and binned into a histogram. This signature is further extended by a viewpoint component - a histogram of the angles between each of the points and the normalized viewpoint direction vector $\frac{p_c - p_v}{\|p_c - p_v\|}$. The main advantage of the VFH is its computational efficiency, with complexity of $O(n)$. Furthermore, this descriptor encodes the objects surface within a single histogram, which makes it convenient to test for similarity between the given point clusters.

3.4. Object recognition pipeline

After computing the descriptors of the input cloud, the task of object recognition can be accomplished by matching them with some classified training set. This training set is obtained on the basis of a object model, acquired i.e. by 3D scanning in a controlled environment. The local and global descriptors require two distinct approaches to the training model fitting procedure. The Figure 3.3 presents complete processing pipelines for both types of descriptors, as proposed in [16].

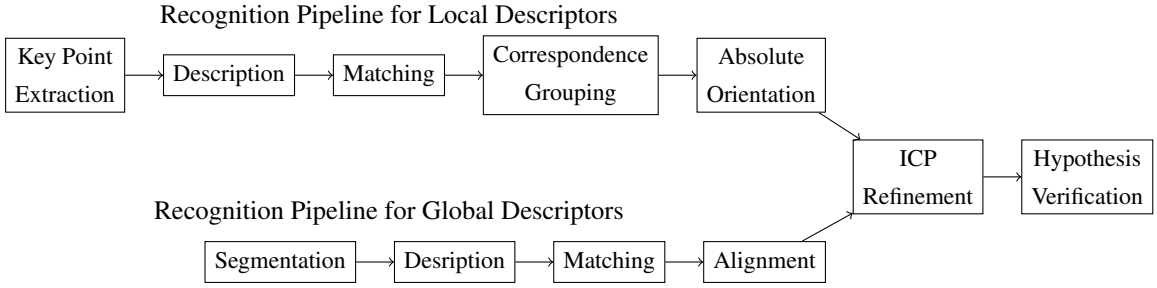

Figure 3.3: Object recognition pipeline [16]

In case of the local descriptors, the first stage of processing, is the extraction of key-points, that are characteristic points in which the features are further computed. The simplest approach to this step is to just downsample the input data, i.e. by a voxel grid filter. For the selected points, the local descriptors are then computed and matched with their closest correspondences in the training set, which results in a set of correspondence pairs. In the next phase, those pairs are grouped according to the geometric constrains of a rigid transformation and too small subsets are discarded. Two point pairs (p_i, q_i) and (p_j, q_j) are considered to be geometrically consistent, if, i.e., they meet the following relation:

$$\left| \|p_i - p_j\|_2 - \|q_i - q_j\|_2 \right| < \epsilon$$

where p and q represent the model and the scene points, and ϵ is a given threshold. By applying the grouping step, a large number of the false correspondences is discarded. To finalize the recognition process, the transformation matrix parameters are estimated, i.e. using the RANSAC algorithm, to further reduce the number of inconsistent correspondences and return the absolute orientation of the object.

As it comes to global descriptors, the whole process is preceded by a segmentation phase, in order to extract individual objects in the scene. Next, for each cluster, a global descriptor is computed and compared to the training set elements, from which its closest neighbours are selected. The position of the object is then obtained by aligning the clusters centroids and the viewpoint information encoded in the descriptor, allows to retrieve the objects rotation.

The final two post-processing steps of both pipelines are intended to refine the recognition outcome. By applying the Iterative Closest Point (ICP) algorithm, which minimises the difference between two point clouds [17], the accuracy of the object position and orientation is improved. The hypothesis verification step aims to reduce the number of false positives of recognition. An example strategy for this step is presented in [18] and is based on the number of inliers in the recognised cluster.

4. Implementation and testing

A MMS robot equipped with a depth acquisition system is a particularly well suited tool to perform tasks with a high degree of autonomy. The high volume of environmental information provided by a depth sensor allows the robot to successfully operate in an unstructured and changing environment without human guidance. In this work, the task which the robot has to solve fully autonomously is the problem of finding a predefined object in its working environment. This problem required implementing two mechanisms for depth data processing. The first was the detection of obstacles, in order to ensure save movement of the mobile platform and the second one was the recognition of the object to accomplish the final task. The Point Cloud Library (PCL) [19] was used to implement the depth data analysis on the robot. PCL is an open source project for 3D point cloud processing. The library contains a vast array of state-of-the-art algorithms for filtering, feature estimation, segmentation, registration and model fitting. OpenMP and FLANN. A detailed description of the developed autonomous control mode is provided in the following sections.

4.1. Hardware setting

The first issue encountered during the development of the autonomous mode was adequate positioning of the depth camera. As mentioned in section 2.4, the selected RGB-D sensor, Asus Xtion Pro operating range starts from $0.8m$, which is a relatively high distance compared to the dimensions of the whole robot. For this reason, the camera position and field of view determines the size of objects that can be used as targets. Moreover, the RGB-D camera location is also crucial during obstacle detection. The system should be able to determine whether it is possible to displace an entire width of the platform forward. While moving ahead, the analysed area is desired to be large enough that the robot could react to obstacle on time and avoid collision. On the other hand, the region close to the front of the platform should also be examined, as obstacles are likely to occur there during turning.

The mounting point of the RGB-D camera has been selected in place of an end-effector of the larger manipulator. This setting has allowed for reproducible and convenient adjustments of the camera pose. The Figure 4.1 presents a simplified view of camera setting relative to the mobile platform. The manipulator is positioned in the plane compliant with the drawing and only the joints relevant to this plane are specified. The joint angles $\theta_1, \theta_2, \theta_3$ were experimentally selected to meet the previously mentioned requirements. The obtained values $\theta_1 = 22^o, \theta_2 = 22^o, \theta_3 = 22^o$, together with physical dimensions

$L_0 = 22^o, L_1 = 22^o, L_2 = 22^o, L_3 = 22^o$ allow to calculate the camera height H as follows:

$$H = L_0 + L_1 sin(180^o - \theta_1) + L_2 sin(\theta_1 - \theta_2) + L_3 sin(\theta_1 - \theta_2 - \theta_3) = x$$

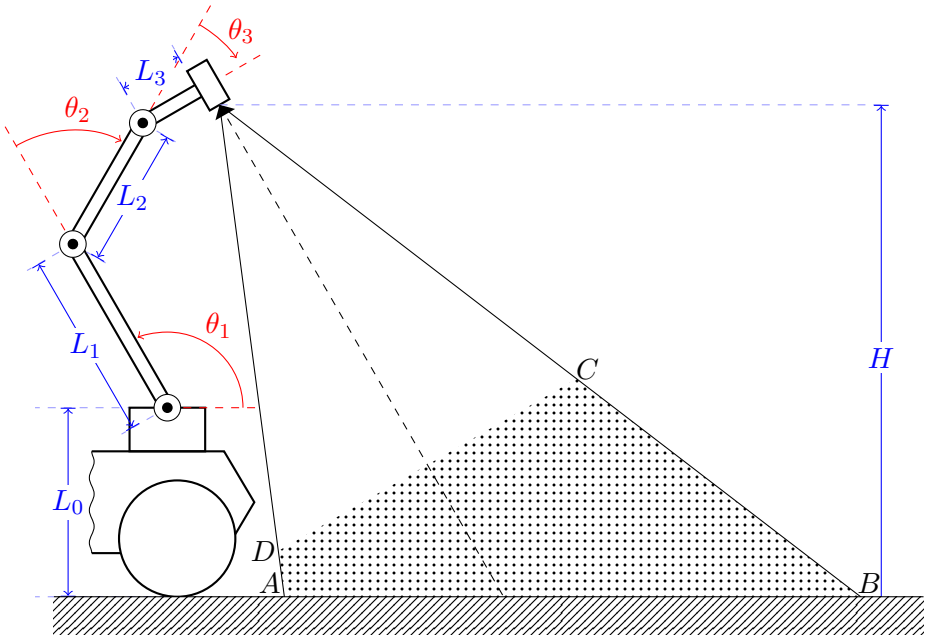

Figure 4.1: Field of view

Problem of high distance. Why such view: on a manipulator - easily adjusted, for future developments, manipulator angles => how to calc camera angle, angle - to see max closely to the front of the platform, and to be able to see objects of about 20x20 at 0.5 m.

Image processing on the NVidia board? here or in the firs chap

4.2. Obstacle detection implementation

Knowledge of the camera field of view facilitates the first stage of depth data processing, the obstacle detection. According to it, the view acquired from the sensor can be further adjusted to detect objects only in the designated area. The full, implemented processing pipeline, including this view adjustments, is presented in the Figure 4.2.

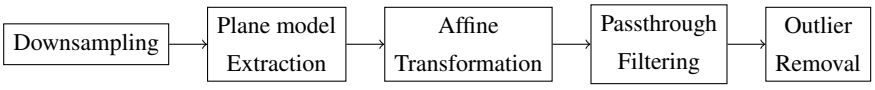

Figure 4.2: Implemented obstacle detection pipeline

Depth sensors provide a high volume of information, which has to be analysed online with a limited computing resources. Therefore, the first processing step was to reduce the amount of data by downsampling. For this purpose, a voxel filter with a leaf size of $1cm \times 1cm \times 1cm$ has been applied. Figure x

illustrates an exemplary point cloud before and after filtration. The number of points has been reduced from x to y. This step has significantly accelerated further processing stages, thus allowing a sufficiently fast response to obstacle emergence.

FIGURE

By design, the robot is intended to work in the indoor environment. A significant feature of such environment is the planar surface of the floor. Therefore, by fitting a plane model to the acquired point cloud, it is possible to determine the area belonging to the ground, and thus free of obstacles. The RANSAC algorithm implementation, available in the PCL library, was used to estimate the plane model parameters A, B, C, D, given by Equation 3.5. Dobrane parametry? After the fitting, all of the model inliers are removed from the point cloud. This step thus leaves only the points that potentially belong to an obstacle. On the other hand, if the RANSAC algorithm fails to fit the model, it is assumed that the entire view is occluded by obstacles and the procedure returns positive detection. The Figure x illustrates the point cloud from Figure x after plane model extraction.

FIGURE

The next processing step, the volume of interest (VoI) extraction, involves separation of a cuboidal space from the remaining point cloud. The dimensions of the extracted cuboid directly correspond to the total height and width of the robot, and the maximum distance at which the system is supposed to react to obstacles. As shown in Figure 4.1, the camera is pointing to the floor at a certain angle. This direction is represented by a Z-axis of the coordinate system in the acquired point cloud. To enable the VoI extraction with the specified dimensions, the coordinate system of the cloud is rotated by an angle θ around the X-axis, such that the Z-axis of the resultant coordinate frame is parallel to the floor. This transformation can be based on the angle $jakistam$, calculated in Section 4.1. However, because of the vibrations caused by the platform movement, a more accurate solution is to use the previously calculated plane model. A vector $\vec{n}$ normal to a plane 3.5 is given by:

$$\vec{n} = \frac{1}{\sqrt{A^2 + B^2 + C^2}} \cdot \begin{bmatrix} A \\ B \\ C \end{bmatrix} \tag{4.1}$$

It can be shown, that the third coordinate n_z of the normal vector $\vec{n}$ satisfies $n_z = sin(\theta)$ and the final rotation matrix is given by:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{1 - n_z^2} & -n_z \\ 0 & n_z & \sqrt{1 - n_z^2} \end{bmatrix} \tag{4.2}$$

After viewpoint transformation, the VoI is extracted by applying a Passthrough filter along X, Y and Z axes of the received cloud. The filtration ranges have been experimentally selected...

The last stage of the obstacle detection processing is the outlier removal.

4.3. Object recognition implementation

In addition to ensuring safe movement of the robot, previously described obstacle detection pipeline also performs most of the necessary preprocessing steps for object recognition. Upon obstacle detection, the resulting point cloud contains points that belong only to the objects in the scene, thus further processing is directly applied to that cloud.

Two different approaches of object recognition were implemented during the project. The first one uses an analytical model of the objects shape and allows to recognize objects with simple geometric structures, such as spheres or cylinders. The second, more general method is based on the objects detailed model, acquired during 3D scanning. Despite the differences, both of these approaches share some common preprocessing steps, including the normal estimation and cluster segmentation. To determine the normal vectors for a given cloud, the NormalEstimationOMP class, available in PCL library, is used. This class exploits the OpenMP [x] library to parallelize computations, which significantly accelerates the process of normal estimation.

It is assumed that during the exploration of the environment, the robot can encounter a scene with multiple objects. In such situation, it is convenient to isolate individual items and analyse them separately. For this purpose, a greedy algorithm of Euclidean Cluster Extraction is applied to the point cloud. The parameters of the algorithm were experimentally chosen, to match the size of individual clusters with objects used as targets. VALUES. The last parameter, due to the nature of the euclidean segmentation, imposes a minimal distance of $10cm$ between object surfaces. As a result of the segmentation process, a vector of point clouds is obtained and further analysis is applied to each of its elements.

In case of the first, geometric primitives based approach, the model fitting is performed directly on the point clouds obtained after the segmentation. The SACSegmentationFromNormals class from PCL library is used during the fitting process. In this approach, the recognition of two geometric shapes is implemented, the sphere and the cylinder. As a result of fitting a sphere, the algorithm returns four optimized parameters: three coordinates of the sphere center point and its radius. An example of spherical object fitting is presented in the Figure X.

FIGURE

For the cylinder model, seven parameters are obtained, including the radius, three coordinates of its axis direction and three coordinates of a point lying on that axis. Furthermore, in order to distinguish cylindrical objects with different heights a few additional steps are required after the model is found. Firstly, the point cloud coordinate system is transformed so as to cover the cylinder axis with the Z-axis. Next, two points with minimum and maximum z-coordinates are found with the getMinMax3D function from PCL. Finally, the resulting height h of the cylinder is given by the difference of those coordinates, that is $h = z_{max} - z_{min}$. Figure X illustrates an example input cloud with a fitted cylinder model.

FIGURE

The second, general method of object recognition utilises the global recognition pipeline described in Section 3.4. This selection is justified by the fact, that the global approach requires a set of partial

object scans as a training set, which can be easily acquired with available hardware. The VFH descriptor, estimated by the PCL VFHEstimation class, is used to compare the point clusters. To obtain the training data set, a series of scans by the RGB-D camera are taken from different, characteristic views of the object. For each of these scans, the VFH descriptor is calculated and added to a KD-Tree structure, which is then saved to the hard drive. During the recognition process, for a given VFH descriptor of a cluster, a nearest-neighbour search is performed in the KD-Tree structure, by the *knnSearch* method from the FLANN library. If the resultant L_1 distance between the neighbour and the segment VFH is less than a given threshold, the procedure returns positive object recognition. The further post-processing steps, listed in Section 3.4, which estimate the 6DOF pose of an object were omitted, as the task of the robot is to only determine the existence of a given object.

4.4. Autonomous mode implementation

Procedures of obstacle detection and object recognition provide the necessary tools to achieve the stated task of autonomous finding of predefined objects. The autonomous mode was implemented in the form of a C++ application, executed on the NVIDIA Jetson TK1 platform. This application communicates with the control server via HTTP requests, as described in Section 1.4. For this purpose, the HTTP POST and GET methods from cpp-netlib [??] library are utilised. All of the required communication request are encapsulated in the form of a Robot class object. The methods of this class enable the realization of such actions as driving forward, mobile platform rotation, stopping and setting the manipulator joints in given positions.

The depth data processing procedures described in preceding sections posses a significant amount of configurable parameters. To avoid having to recompile the code with every parameter customization, all of the relevant constants are included in a configuration file in a XML format. The path to this XML file is specified as a command-line argument of the application, together with the name of the target to search for. The configuration file contains also information on the recognition method type and the path to the training set for the specified target name.

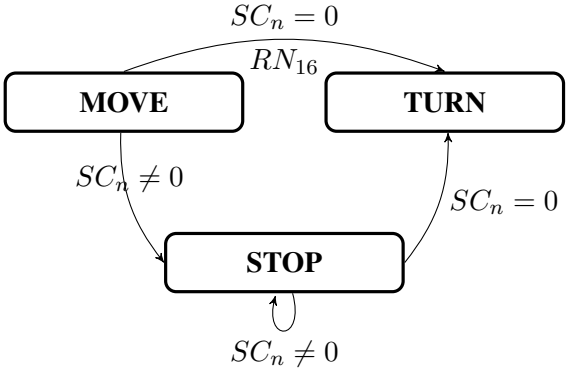

Figure 4.3: State diagram of the autonomous mode

The system operation outline is presented in the Figure 4.3, in the form of a state diagram. Three different states of motion are distinguished: forward movement (**F**), turning (**T**) and stopping (**S**). At the time of entrance to each of the **F**, **T**, **S** states, a corresponding command is sent to the control server. Transitions between these states are determined by the logical values returned from the procedures of obstacle detection (*OD*) and object recognition (*OR*). The robot moves forward if there are no obstacles in front of it, stops if it encounters the target object and turns if it discovers an obstacle. The direction of turning is selected in a pseudo-random manner at each newly encountered obstacle. In this way, the system is able to explore the whole working environment.

4.5. Testing environment and results

Summary

Bibliography

[1] B. Cyganek and J. P. Siebert., *An Introduction to 3D Computer Vision Techniques and Algorithms*. John Wiley & Sons, Ltd., 2009.

[2] http://www.ensenso.de/?portfolio=n10.

[3] L. Li, "Time-of-flight camera – an introduction," Tech. Rep. SLOA190B, Texas Instruments, January 2014.

[4] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time of Flight Cameras: Principles, Methods, and Applications*. SpringerBriefs in Computer Science, Springer, 2013.

[5] `http://www.extremetech.com/`.

[6] G. Sansoni, M. Trebeschi, and F. Docchio, "State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation," *Sensors*, vol. 9, no. 1, pp. 568–601, 2009.

[7] J. Salvi, J. Pagès, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, no. 4, pp. 827 – 849, 2004. Agent Based Computer Vision.

[8] G. Borenstein, *Making Things See: 3D Vision with Kinect, Processing, Arduino, and MakerBot*. Make: books, O'Reilly Media, Incorporated, 2012.

[9] http://www.asus.com/pl/Multimedia/Xtion_PRO_LIVE/specifications.

[10] R. B. Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[11] http://en.wikipedia.org/wiki/RANSAC.

[12] M. Zuliani, "RANSAC for Dummies," tech. rep., UCSB, 2012.

[13] T. K. Sunglok Choi and W. Yu, "Performance evaluation of ransac family," in *In Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[14] L. A. Alexandre, "3D descriptors for object and category recognition: a comparative evaluation," in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Vilamoura, Portugal), October 2012.

[15] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *The IEEE International Conference on Robotics and Automation (ICRA)*, (Kobe, Japan), 05/2009 2009.

[16] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation.," *IEEE Robot. Automat. Mag.*, vol. 19, no. 3, pp. 80–91, 2012.

[17] http://en.wikipedia.org/wiki/Iterative_closest_point.

[18] A. Aldoma, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, M. Vincze, and G. Bradski, "Cad-model recognition and 6 dof pose estimation," in *ICCV 2011, 3D Representation and Recognition (3dRR11)*, (Barcelona, Spain), 11/2011 2011.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.