BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

PHD THESIS

Brno, 2019 Ing. Radim Luža

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

INTELLIGENT MOBILE ROBOT

INTELIGENTNÍ MOBILNÍ ROBOT

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR **Ing. RADIM LUŽA**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. FRANTIŠEK V. ZBOŘIL, CSc.**
ŠKOLITEL

BRNO 2019

Abstract

The thesis concerns about solution for localization of mobile robot in large outdoor environment with sparse landmarks. The emphasis is on independence on external system. The thesis describes existing localization methods and algorithms used for localization. Then the solution for localization in large outdoor areas is designed and implemented. The implemented solution is tested by set of experiments described further in the thesis. Finally the conclusion is made - results are confronted with goals of the thesis and also ways of future development are proposed.

Abstrakt

Tato práce se zabývá řešením lokalizace mobilního robota v rozsáhlých venkovních prostředích s řídkými orientačními body. Důraz je kladen na nezávislost na externích systémech. Práce popisuje stávající metody a algoritmy používané pro lokalizaci. Dále je popsán návrh a realizace řešení pro lokalizaci v rozsáhlých venkovních prostředích. Implementované řešení bylo otestováno sadou exprimentů. Na závěr jsou výsledky experimentů konfrontovány s cíli práce a jsou navrženy možnosti dalšího vývoje.

Keywords

Robotics, outdoor localization, feature-based mapping.

Klíčová slova

Robotika, lokalizace ve venkovním prostředí, feature-based mapování.

Reference

LUŽA, Radim. *Intelligent mobile robot.* Brno, 2019. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. František V. Zbořil, CSc.

Intelligent mobile robot

Declaration

Hereby I declare that this PhD thesis was prepared as an original author's work under the supervision of doc. Ing. Františka V. Zbořila, CSc. The supplementary information was provided by Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D. and Dr Frederic Labrosse PhD. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Radim Luža
June 25, 2019

Acknowledgements

I express my thanks to my supervisor doc. Ing. František V. Zbořil, CSc. for his help and guidance of my effort. I express my thanks to Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D. and Dr Frederic Labrosse PhD. for consulting specialised scientific topics and help with obtaining experimental data. faculty

Contents

Chapter 1

Introduction

In todays world the robots are becoming more and more frequent tool for solving tasks from various areas of human effort. In some areas like manufacturing they already have a tradition while in other the usage of the robots is still in experimental phase. With growing performance and decreasing energy consumption of computers the complexity and performance of robots control systems grow rapidly. This allows robotic researches and developers to apply a more sophisticated attitudes and use more advanced algorithms for controlling the robots. Thus the robots can operate in more complicated environments and solve more complex tasks. In general robots are becoming more autonomous and more helpful.

First autonomous robots appeared in industrial plants and depots where they operated in well defined conditions and well known environment. Using robots in industry was a great success and it was a matter of time when the robots will spread into other areas. As the technology advances the robots can deal with uncertainity, changing conditions and non-deterministic environments. One of the most complicated environments for controlling the robots is outdoor environment. The environment has many variants according to particular place on the planet. Moreover it is often dynamic with both slow and fast changes happening in real time. Thus it is not easy to make generic assumptions about the outdoor environment. This fact complicates design of robots for outdoor environment in terms of both hardware and software.

For most of the tasks the robots accomplish it is essential to receive feedback for executed actions. Robot needs to know how the environment reacts on its actions and in case of mobile robots where in the environment the robot is at the particular time. The problem of estimating actual location of the robot is called localization. Despite the latest advances in robotics the localization in the outdoor environment is still a challenge. There is probably no generic solution that would fit all purposes. It is usually necessary to adapt localization to particular use case. There are several support systems for outdoor localization. Probably the most frequently used is network of satellites - so called GNSS (Global Navigation Satellite System). Thanks to signal from satellites it is possible to localize the mobile robot with very good precision. Unfortunately there are situations in which it is not possible to use these satellite networks.

Creating a generic localization solution without support of external systems is not easy. The environment usually has to be restricted to particular variant or set of variants. This thesis concerns about solution for localization in large outdoor areas like meadows, fields or airport runways. Typical aspect of these environments is that there are only few objects that can be used as landmarks. The solution described in this thesis tries to deal with this

situation using sensors installed on the robot - particularly camera, laser rangefinder and odometry.

Chapter 2

Theoretical background of robot localization

Localization is one of the essential components of autonomous robot behaviour. Most of missions of autonomous robots depend on the knowledge of robot's location relative to the mission goals and obstacles. This chapter deals with theoretical background for localization and SLAM (Synchronous Localization And Mapping) process.

2.1 Localization problem

The localization problem can be intuitively described as finding location of the robot in given environment or more precisely finding location of the robot relatively to the origin of given coordinate system. For stationary robots the localization of the endpoint can be solved by precise measurement of joint poses. Unfortunately this approach is insufficient in case of mobile robots due to slippage of undercarriage. Still the technique of measuring robot pose according to speed or change of pose of motivators is used as a one of location sources. In mobile robotics the data from the chassis are called odometry. Technique of estimating new position of the mobile robot according to known last position and relative change of position measured by sensors is called dead-reckoning.

The localizaiton problem can be cathegorized according to several points of view:

Local vs global Local localization can be intuitively described as correcting prior pose information or trajectory following. The prior pose information with limited error is essential for local localization. The robot pose uncertainity can be approximated using unimodal distribution - usually Gaussian. The global localization on the other hand can not rely on any limit of pose error in prior pose information. For many global localization algorithms the initial pose is chosen randomly. It is obvious that the global localization problem is more complicated than local localization problem. Another cathegory of localization problem is introduced in [78] - the kidnapped robot problem. The word „kidnapped“ in this context means that the robot was moved to another pose without knowing that. Sudden transportation of a robot to new location does not reflect a typical real world problems but algorithms that can deal with the kidnapped robot problem can also deal with situations when robot gets lost (for example due to insufficient amount of interesting objects in its surrounding or sensor failure).

Static environment vs dynamic environment In the static environment the only object that change its pose over time is the robot. All other objects stay at their poses and also the objects keep their features and robot observing the environment can rely on it. The static environment is typical for simulators. Most of the real world environments are dynamic. In dynamic environments the features of objects including shape (for example tree in wind), pose (for example open vs closed door) and surface color (due to changing light conditions) change with the time. Despite the real world environments are almos always dynamic the static algorithms work there too - small changes in the environment can be considered as errors and so they can be filtered out without significantly affecting performance of localization and SLAM methods.

Passive vs active In passive approach the localization algorithm works as a passive observer - it takes data from sensors and tries to estimate location of a robot that is controlled by another entity. In active apporoaches the localization algorithm also controls robot motion to adapt its trajectory according to its needs. With active localization attitude the trajectory is usually affected by sensor capabilities and structure of the environment (for example robot stays close to the walls and obstacles because of limited reach of sensors).

2.2 Indoor vs outdoor environment

Localization environmnet differ in many aspects but the most significant one is distinguishing between indoor and outdoor environment. The first one is inside human created buildings. Typical aspect of indoor environment is regular geometry of rooms and obstacles in it. In most cases we can rely on a flat floor and perpendicular walls. We can relatively easily recognize some well known elements of the environment like doors and windows. Noise in the environment is relatively small. There is no wind blowing, light conditions are usually relatively stable and shapes and patterns once detected can be re-observed with good success rate. We can choose sensors according to properties of the environment. In the indoor environment we can use ultrasonic sensors for localization as the free areas without any obstacles reflecting ultrasound are limited in size. We can use 2D LIDAR because walls are perpendicular to the ground so the Z coordinate has minimal effect on X and Y coordinates. And moreover odometry information is usually quite reliable in indoor environment as the surface is mostly homogenous and predictable.

All these facts about indoor environment allow usage of simplified localization and mapping methods. Typical abstraction for indoor environment is 2D map despite the environment is three dimensional. Another advantage is usage of less robust sensoric system. We can afford to use sensors that do not operate properly in a day light like inexpensive infrared rangefinders or sensors with limited reach. Moreover the process of obtaining sensoric measurements do not need to deal with roll and pitch motion. It is also easier to build localization based on artificial markers as there is good stability of such markers in indoor environments. Artificial markers in various forms are frequently used inside industrial buildings as described later.

The outdoor environments on the other hand are more demanding in terms of map representation, landmark observation and sensoric system robustness. Despite there are simplified cases like parks or roads that can be mapped in 2D in generic outdoor environment the 2D mapping is not sufficient. Due to structure of terrain it is necessary to keep record of landmarks in three dimensions to be able to distinguish between particular landmarks.

Some of the algorithms that work in 2D mapping very well can be extended to 3D but usually there is significant performance impact. For example occupancy grid based representation of map (described below) can be extended to 3D grid but asymptotic memory complexity as a function of map size is now cubic instead of quadratic in case of 2D variant of the algorithm. Also processing such amount of data is much more complex. Moreover maps in outdoor environment are usually larger than those covering indoor environments. Another difficulty is that odometry information in outdoor environment is usually a way less realiable than in indoor environments due to drifting of wheels. All these aspects need to be considered in approaches used for outdoor localization.

2.3 Sensors

In todays robotics there are many sensors usable for purposes of localization of the robot and for creating maps of surrounding environment. Listing all of these snesors is out of scope of this thesis. Existing sensors can be cathegorized by principle of operation. Following cathegories of sensors can be defined:

Rangefinder Rangefinder in general is a device that can measure distance to obstacle. There are several principles of ranging used in todays robotics that differ in its properties and performance. Probably the cheapes rangefinders are based on ultrasonic principles. These are the only rangefinders that do not use electromagnetic waves - they use mechanical waves instead. Ultrasonic rangefinder generates a short density change in the air usually by pulsing membrane. This change spreads through the medium - usually air or water as advancing mechanical wave. When the wave hits obstacle it is reflected and travels back to rangefinder that is already listening. The delay between sending the wave and receiving it is directly linearly related to distance the wave had to travel. The relation is defined in equation 2.1. The distance has to be divided by 2 as the wave travels from the rangefinder to the obstacle and back.

$$d = \frac{\Delta t \cdot c}{2} \tag{2.1}$$

Speed of the mechanical wave in a gas c including air varies according to density of the media. The relation is described by Newton-Laplace equation 2.2 cited from [2]. The K_s is so called elastic bulk modulus that is defined as multiplication of pressure in the media and temperature of the media. ρ is density of the media. Speed of mechanical wave increases with growing pressure and decreases with growing density of the media. One important attribute of ultrasonic rangefinder comes from this equation: Its calibration is valid only for particular temperature and pressure condition in particular media. With changing conditions the precision of the rangefinder drops.

Another aspect of ultrasonic rangefinding is a rather wide cone in which the ultrasonic rangefinder detects obstacle. The rangefinder can not find exact position of the object. It can only find object laying somewhere on the equidistant spherical surcface. Of course this applies also to any other beam but in case of ultrasonic rangefinder the cone is too wide to be neglected during finding exact position of particular object. Range of ultrasonic rangefinders varies from tens of centimeters to tens of meters. The range is limited by accoustic power of the transmitter and also by noise generated by other mechanical waves (sounds) in the media.

$$c = \sqrt{\frac{K_s}{\rho}} \tag{2.2}$$

Radar rangefinding is another principle used for measuring distances. It works in a very similar way to ultrasonic rangefinder but it differs in physical principle of the transmitted wave. Radar rangefinder use electromagnetic radio wave instead of mechanical wave. Again the wave reflects from obstacle and it is received again by the radar sensor to measure time difference. As the wave is electromagnetic it is immune to interference mechanical waves on one hand but it interferes with electromagnetic waves. As the attenuation of electromagnetic wave in the air is smaller than attenuation of mechanical wave the range of the radar sensors can be higher reaching hundreds of meters. High performance radar system can reach tens of kilometers far.

Laser rangefinders use electromagnetic wave as radar rangefinders does but it uses it in a different way. As first - it uses light instead of radio - the frequency of electromagnetic waves differ a lot. And secondly the wave is focused into narrow beam. The beam is so narrow that in most of applications it can be considered as a single point. This is one of the most significant differencies in abstraction of laser rangefinder and other rangefinders described above. Reach of laser rangefinder can vary from tens of meters up to kilometers. To measure time that light needs to reach the obstacle, reflect and arrive back a more sophisticated attitude is needed as the delay is very small and speed of light is similar to speed of electrons in conductors. Naive approach of sending pulse and waiting for arrival of the reflection is not very usable. Instead a phase shift of modulated signal is used. The carrier wave that is transmitted is modulated with a known pseudorandom code.The receiver continuously receives the wave and reads the code. The distance is measured as a phase shift of the transmitted and received code sequences. The resolution is given by wave lenght and lenght of code symbols in wave periods. This attitude brings one shortage: The range of rangefinder is limited not only by transmitting power but also by lenght of code sequence. When the code sequence starts to repeat the rangefinder „sees a ghosts“ which means that it sees distant objects much closer to the rangefinder than they really are.

Single point ranging is often not sufficient for many applications. This is why 2D and 3D laser rangefinders were invented. Principle of these devices is based on rotating a single point laser rangefinder or rotating a precise mirror in front of static single point laser rangefinder. This way it is possible to obtain from tens to millions of distance measurements in one scanning period (usually a revolute of device head). Such a dense data can be used to construct precise maps and model surrounding of the robot.

The laser transmitter has very narrow spectrum - in theory a single wavelength only. This property can be used to avoid interference of several laser rangefinders but it also allows to mix several frequencies in one scan. Each color of the spectrum has a different properties especially in terms of reflection. Particular color reflects the best from the surface of the same color. This can be used to obtain color information together with distance information but light of different frequencies differ also in penetration capabilities. Some frequencies can penetrate water or grass while other reflect from its surface. This way it is possible to obtain additional information about

terraing and surroundings in a single, alligned scan. One of sensors offering such functionality can be found in [15].

Last in this cathegory are infrared rangefinders. There are two principles of measuring distance used in existing infrared rangefinders. First of them is time of flight principle - the same as the laser rangefinders use. Infrared light is also eletromagnetic signal so processing works in a very similar way. Another principle is to project matrix of infrared points to the surrounding and detect them with infrared camera. According to deformation of the matrix the distances of particular points in the 3D space can be computed. This principle described in [47] is used by Kinect 1 sensor. Advantage of infrared rangefinders over laser rangefinders are mainly manufacturing costs. Drawback of infrared sensing is that it is significantly affected by daylight that contains significant infrared component.

Cameras Cameras are sensors that provide data with a very high information density. Unfortunately the information is not structured and finding the required information in camera image is still important research topic. Basicaly the camera is a sensor that compounds of matrix of light sensitive cells and optical aparatus. The chip with matrix of light sensitive cells converts light intensity into electric signal according to conversion technology used. The signal is evaluated by camera control electronics and converted to digital value. For common cameras the value is usually 8 bits per channel. According to detected information the cameras can be distinguished to grayscale cameras (light intensity), color RGB cameras (intensity of red, green and blue channel) and special cameras (for example thermocamera that receives infrared electromagnetic signal).

Depth cameras Cathegory of depth cameras covers cameras extended with depth information for every pixel of the image.

Stereocamera Stereocamera is a sensor composed of two RGB or grayscale cameras that work in a similar way as pair of eyes works. In the same scene observed by two cameras that are slightly shifted one from another the objects are slightly shifted too in camera images. Objects that are closer to the pair of cameras have larger shift than objects that are more distant. The shift is called disparity. In the best case we can define disparity for almost each pixel in the image. Knowing the disparity and camera parameters we can compute distance of given point from plane given by camera centers and normal parallel with camera axes.

If we recognize pair of corresponding pixels in left and right image we can compute the disparity using eqaution 2.3. In the equation the x_l and x_r are coordinates of corresponing pixels in left and right image and x_{cl} and x_{cr} are horizontal center points of images. The f is focal length of both cameras (we suppose that the cameras have the same parameters) and B is baseline - the distance between camera centers.

$$d = (x_l - x_{cl}) - (x_r - x_{cr}) \tag{2.3}$$

Knowing the disparity we can reconstruct $< x, y, z >$ coordinates of the point in observed scene using following set of equations. For the equations to be valid there are some presumptions about the cameras. First the cameras has identical parameters. Fortunately todays manufacturing methods allow us to get cameras

that are practically identical in point of view of camera model. Second assumption is that the camera images are perfectly rectified. The equations above do not count with deformation of image due to camera optics characteristics. The rectification is usually achieved with software postprocessing of camera image. Correct values for postprocessing are obtained by process of calibration of the cameras.

$$z = \frac{fB}{d} \tag{2.4}$$

$$x = \frac{(u - u_c)z}{f} \tag{2.5}$$

$$y = \frac{(v - v_c)z}{f} \tag{2.6}$$

Limitation of stereocamera performance is usually given by methods of detecting correspoding points in both images. In real situations only minority of pixels can be detected properly. Still the image can be segmented into regions with the same depth so the scene can be reconstructed with good precision. More details can be found in [20].

RGB camera with matrix depth sensor The matrix depth sensor can be based on several technologies. Probably the most frequently used ones are pattern projection and time of flight measurement. The pattern projection [47] is a method of projecting known pattern to the scene and compute 3D profile of the scene according to deformation of the pattern. The projection and observation of the pattern usually happens outside the visible spectrum to avoid noise in the RGB camera view. This method is used in first generation of Microsoft Kinect sensor. Limitation of the sensor is given by limited output power of the pattern projector that can reach up to ten meters in good conditions and also by small resitance to light noise. In case of Kinect the sunlight usually makes the patter completely unreadable and even in indoor environment the daylight or artificial light significantly affects stability of the depth data.

Another approach is measuring the time of flight of modulated light beam [58]. The modulated beam is projected to the scene and received back with receiver. According to phase shift of the projected beam in particular point the distance is computed. This approach is more robust in changing light conditions but it still has its limitations. Except for output power of the beam projector there is interesting problem when measurement exceeds the maximal distance the sensor was designed for. If the object reflecting the beam is so far that the time of flight is longer than the period of the transmitted signal then the sensor might observe „ghost“ measurements. The ghost measurement appears to be very close despite the real object is much further.

Encoders Encoders in general are sensors that provide information about position of moving components using digital code.

Optical Optical encoders [52] compound of three components: Ligh emmitor, light receiver and moving part with sequence of contrast areas. The contrasting sequence can be achieved by alternating areas with high and low light reflection or

by alternating areas with hight and low opacity. In the first case the light emmitor and reciever aim in the same direction and receiver receives reflected light and in the second case the emmitor heads against the receiver and beam of light is blocked by opaque areas. Optical encoders are convenient for measuring position of moving components because the output depends only on actual intensity of reflected or passing light beam. Light beam is resistant to electromagnetic noise and vibration so position feedback provided by optical encoder can be very stable and reliable. Beam of light can be very narrow so it is possible to combine several one-bit encoders into more complex encoders. For motor shafts the Gray code or simple AB phase encoder is typically used. Advantage of these encoders is ability to determine direction of rotation together with to revolute speed. For manipulators with rotary joints and for other rotary joints with limited range of rotation the N-bit binary encoder is often used to provide information about absolute joint position. For purposes of localization the AB phase encoders are typically used for odometry and N-bit encoders are typically used as a feedback for sensor positioning.

Hall probes Hall probes use effect of force interaction of magnetic field on charged particles moving through it called Hall effect [26]. When beam of electrons flows through semiconductor and the semiconductor gets into magnetic field the flowing electrons are affected by magnetic field. Electron trajectory is deviated by magnetic field. If the magnetic field is orthogonal to electron beam the effect is maximized. As a consequence of electron beam deviation one plane of the semiconductor is charged positively due to lack of electrons and due to it the opposite plane gets negative charge. This charge creates a barrier in the semiconductor that electrons flowing through have to overcome. The barrier appers as Hall voltage between semiconductor planes. If a moving part contains small magnets we can detect the moment when magnet moves around the hall sensor by change of the voltage. Output of hall sensors used as encoders are usually filtered by threshold circuits that form rectangular output of the encoder. Hall sensors are usually used as encoders on motor shafts on AC motors and other rotary components. Hall probes are convenient especially for motion speed regulation as they react especially on change in magnetic field and the have short reaction times.

Electromagnetic Electromagnetic encoders use effect of electromagnetic induction to detect position of moving part. There are many approaches [23] using different encoding area structure with different codings. Particular designs have its cons and pros. Basically these encoders are used mainly for measuring speed of motion and phase of revolute period but they can be also used for absolute position measurement.

IMU IMU is acronym for Inertial Measurement Unit - a composite sensor intended for measuring orientation and acceleration of the moving object in 3D space [51]. The IMU typically compounds of three accelerometers, three gyroscopes and sometimes additonally of three magnetometers. By integrating acceleration vector in time while knowing orientation of the moving object we can compute its velocity and position in time. This is used for estimating pose of the mobile robot during its operation. Process of estimating position of the robot from IMU and odometry is called dead-reckoning. The dead-reckoning is used as one of robot pose information sources for

localization and SLAM algorithms. The most important advantage of dead-reckoning is that it does not rely on changing surrounding environment. It relies on mostly on gravity and magnetic forces of the planet to estimate where did the robot get from the initial position. A serious disadvantage of dead-reckoning is error accumulation. Every measurement is affected by error. No matter how large the error is every upcoming measurement brings additional error to pose estimation. Due to this the dead-reckoning looses precision over the time. When the dead-reckoning is fused with other localization approaches using external data from the environment the growing error can be „reset“ to low value from time to time. In such mode of operation the dead-reckoning provides very useful source of localization data. Even in the cases when dead-reckoning error is not reset by any other localizaiton system it provides at least relative location of the robot for a limited distance travelled.

GNSS GNSS is acronym for global navigation satellite system. This generic acronym covers navigation systems that use network of satellites orbiting the Earth to measure exact position of rover on planet surface. In todays world there are three most important satelite navigation systems: GPS, GLONASS and GALILEO [9] . GPS is controlled by USA, GLONASS by Russian federation and GALILEO by European union.

GNSS in general are complex systems compounding of satellites on Earth's orbit, ground control and rover receivers. Describing each system in the detail is out of scope of this thesis so only GPS will be briefly described. Other GNSS use similar concepts. As described in [29] position measurement using GPS is based on measuring distances of the rover from GPS satellites. GPS satellites and also ground rover have code generator that generates pseudorandom code from cathegory of Golden codes. In GPS each satellite has its own seed for pseudorandom code generator so generated sequences are uniqe for each satellite. The rover generates the same code sequences for satellites it observes. As the rover is receiving signal from a satellite the signal is modulated by the code sequence unig for the satellite. The rover compares the sequence with the one it generated itself. As the time is synchronized between rover and the satellite the rover can compute time shift between received and generated code. The timeshift has a linear dependency with the distance between rover and the satellite. The distance is radius of sphere about the satellite on which the rover lies. As rover knows the position of each satellite it can compute its own position as intersection of spheres. To get unambigous possition the rover needs to observer three or more satellites. With two satellites the rover can at least estimate its position on intersecting curve of two spheres given the model of Earth geoid but this estimate can be ambigous. Practically more satellites are needed (at least 3 and geoid model) as it is difficult to synchonize rover's clock with satellites so the clock skew is another unknown variable as described below.

Precision of position estimation is limited by code baudrate and it is also affected by atmospheric disturbance, effect of refration of the signal when entering the more dense atmosphere and also by multipath signal spreading. Especially the atmospheric disturbances and multipath signal preading bring random noise to the position measurement. This effect can be eliminated by measuring position on a spot for a longer period of time but this is not an option for moving clients like vehicles or mobile robots. The effect of random noise can be significatly reduced for mobile clients using differential GPS (DGPS) [64]. In this approach the GPS system is extended with base

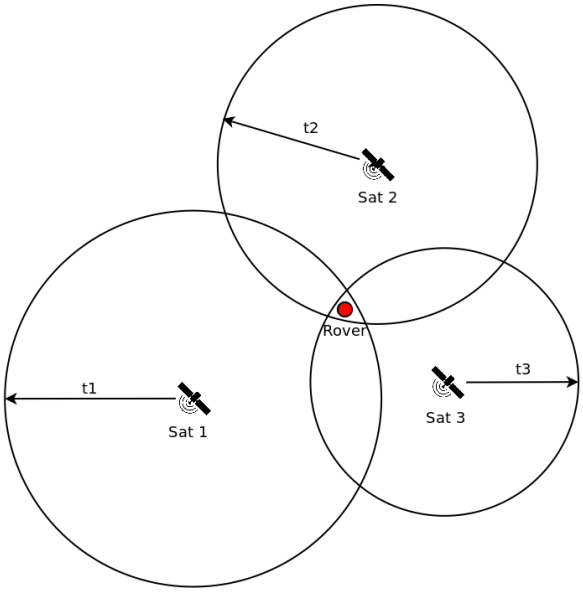

Figure 2.1: GPS principle: geometry.

station that is nearby the rover (up to couple of kilometers usually) that measures its GPS position for a long period of time and integrates noisy measurements. The integration eliminates random errors and the station gets its precise position. During mobile rover operation the base station receives its GPS position in real time and compares it with its known possition. The difference is sent to the rover that subtracts the difference from the position it measures and gets better estimation of its real position. The system works with the presumption that same effect that brings error to GPS position of the station affects also the GPS position of the rover.

Precision of GPS can be increased measuring phase shift of carier signal instead of the code period. This allows to improve precision of GPS to centimeters.

As the GPS is todays most frequent way of outdoor localization a basic mathematical background is described in following paragraphs. From mathematical point of view the GPS localization is based on N-lateration where N >= 3 as described in [79]. As mentioned before satelites travel around the planet elliptic trajectories - the orbits. Trajectories of satelites are known and so is known the exact position of satelites on the trajectories. When message from satelite is received it contains a timestamp. According to this timestamp it is possible to measure time difference between satelite and rover so it is possible for rover to compute the distance between rover and the satelite. The distance defines sphere about the satelite on which the rover's actual position lies. When distance to another satelite is measured position of the rover is restricted to curve defined by intersection of two spheres. In theory once the rover obtains third distance the exact position of the rover can be computed as depicted in figure 2.1.

The figure shows the situation with three satellites observed by rover. Time that takes the signal to arrive from each satellite is denote by t1, t2 and t3 respectively. The particular time difference measured as phase shift of pseudorandom code generated by the satellite makes a spherical surface about the satellite.

We are trying to find the particular time differences - diameters of spheres, where all three spheres intersect at rover's position. From mathematical point of view we are finding solution of set of equations 2.7. Each of the equations describe one spherical surface about one of the satellites. Identification of the satellite is given by index. All equations together define the trilateration principle: We are finding intersection point of spherical surfaces. As we can observe in the equation set 2.7 there are four equations instead of three. The reason is that we need to find four unknown variables: x_r, y_r, z_r and ϵ. The three variables define position of the rover on planet surface. The ϵ denotes a time difference of rover's clock. This another variable was added due to technical reasons. Despite the satellites have very precise time synchronization it is not easy to achieve such a precise synchronization for rover's clock - it would make GPS receiver very expensive and clock synchronization for rovers would require additional insrastructure. To avoid these technical limitations the fourth variable ϵ was added.

Function $d(t_x, \epsilon)$ is function of distance according to time difference of signal travelling from satellite to rover and difference of rover's clock. In optimal case the fourth distance is obtained by fourth satellite observation but if the satellite is not available it is possible to use three satellites only and use geometry model of Earth as the fourth distance. In this simple model the Earth is modelled as sphere. This model can be used for initial estimation without any prior knowledge. After approximate coordinates are estimated the precision can be improved using proper Earth geoid model [75]. Still this is more like a fallback solution that is less precise than observing fourth satellite.

$$\begin{aligned}
(x_r - X_1)^2 + (y_r - Y_1)^2 + (z_r - Z_1)^2 &= d(t1, \epsilon) \\
(x_r - X_2)^2 + (y_r - Y_2)^2 + (z_r - Z_2)^2 &= d(t2, \epsilon) \\
(x_r - X_3)^2 + (y_r - Y_3)^2 + (z_r - Z_3)^2 &= d(t3, \epsilon) \\
(x_r - X')^2 + (y_r - Y')^2 + (z_r - Z')^2 &= d(t', \epsilon)
\end{aligned} \tag{2.7}$$

With additional equation the required precise informations that rover needs to know to make GPS localization work are: a) Observation of three satellites, b) Exact trajectory of satellites, c) Position of the satellite on the trajectory and d) Clock with precise ticks.

Clock needs to measure time difference precisely but it does not need to be synchronized with satellites. Trajectories and position on the trajectory allows rover to compute exact position of the satellite in moment when it sent signal to the rover. This way the coordinates X_N, Y_N, Z_N are being obtained. To obtain information about GPS satellites the data encoded in the GPS signal contain almanac - the overview information about entire GPS system including trajectories (orbits) of the satellites and status of the satellites. Almanac also helps receiver to estimate which satellites could be visible according to its last known location so it speeds up search for satellites. To obtain exact position fixes for particular satellites the signal of each satellite contains ephemeris - exact spherical polar coordinates of particular satellite at given time.

We can notice that in figure 2.1 the circles denoting the spherical surfaces do not intersect perfectly. This imprecision is caused mostly by noise. There are several sources of the noise. Positions of satellites may not be perfectly precise, clock ticks

of rover may be slower or faster than satellite's one but most of the noise is caused by atmospheric disorders. The density of Earth's atmosphere slows down the signal from the satellites. Unfortunately the rover has no information about actual density changes in the atmosphere and these changes are too random to be estimed. Error caused by atmoshperic noise has normal distribution so exact position of static receiver can be obtained with good precision by averaging measurements over long period of time. This is not usable for mobile rovers of course but static receiver with known precise position can help with localization of rovers nearby. If the rover is nearby the static receiver (up to tens of kilometers) the error of the rover will be almost the same as the error of the static receiver. Static receiver can measure the error, compute correction and provide it to rover so the rover can correct its position measurements. This is the way how Differential GPS that is mentioned above works.

2.4 Localization algorithms

Localization algorithms described below are the basic algorithms that the most of existing localization algorithms are built on. They differ mainly in a way they represent map and in a way they represent uncertainity.

2.4.1 Kalman filter based localization

One of the most intuitive ways of modelling uncertainity of sensors and effectors is to use a normal probability distribution with gaussion density function given by:

$$p(\mathbf{x}) = \frac{1}{\sqrt{det\,(\Sigma)\,2\pi}} e^{-\frac{1}{2}(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)} \tag{2.8}$$

Attributes defining the shape of gaussian function are mean vector μ and covariance matrix Σ. Kalman filter is an aparatus usable for mixing two uncertain sources of the same pose information into one. For robotics localization the inputs of the Kalman filter are usually estimated poses of the robot in given map coming from several sources. Typical sources of robot localization information are dead-reckoning and a measurement-based localization. Of course Kalman filter can be used for mixing several localization sources iteratively: First two sources are mixed and in each following iteration another sources is added. Kalman filter can be easily implemented and does not have large demands on computing power. It has a serious limitation: It only works for linear motion and observations models. The linear Kalman filter is described by following set of equations:

$$\begin{aligned} \bar{\mu}_t &= A_t\mu_{t-1} + B_t\mu_t \\ \bar{\Sigma}_t &= A_t\Sigma_{t-1}A_t^T + R_t \\ \bar{K}_t &= \bar{\Sigma}_t {C_t}^T \left(C_t\bar{\Sigma}_t {C_t}^T + Q_t\right)^{-1} \\ \mu_t &= \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t &= (I - K_tC_t)\bar{\Sigma}_t \end{aligned} \tag{2.9}$$

Basically the Kalman filter in the form defined above computes the new pose of robot in two steps: In the first it computes prediction of new pose according to control action and/or feedback from dead-reckoning and in the second step this prediction is corrected with actual observations. The $\bar{K}_t$ is so called Kalman gain. The Kalman gain works as a mixing ratio between prediction and correction. Its value is given by uncertainity of sensor observation. If the observation is perfectly certain the effect of prediction is eliminated

and the final pose of the robot is given only by observation (and covariance matrix is zero matrix). If the observation is absolutely uncertain (infinite covariance matrix) then the final pose is given by prediction only and the final covariance matrix Σ_t is equal to covariance of prediction $\bar{\Sigma}_t$.

As mentioned before the linear Kalman filter is defined for linear motion and observation model. The model the Kalman filter works with is described by following pair of equations:

$$\begin{aligned} \bar{x_t} &= A_t x_{t-1} + B_t u_t + \epsilon_t \\ \bar{z_t} &= C_t x_t + \delta_t \end{aligned} \tag{2.10}$$

In equation set above the first equation describes relation between prior x_{t-1} and new x_t pose of the robot affected by control of the robot u_t, effect of the environemnt A_t and noise noise ϵ_t. The effect of control action u_t (typically defined as a vector of speed) affects the final pose of the robot indirectly via motion model B_t.

For the most frequent robot chassis with differential drive [62] controlled by speed vector u_t the motion model is not linear - it includes non-linear goniometric functions as can be seen in equation 2.11.

$$[x, y, z, \alpha, \beta, \theta] = \int [ux, uy, uz, wx, wy, wz] \cdot \begin{bmatrix} cos(\theta) & 0 & 0 & 0 & 0 & 0 \\ 0 & sin(\theta) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} dt \tag{2.11}$$

In practical scenarios a discrete version of the motion model with sampling period Δt is used (2.12). The sampling period is given by frequency of updates of the map and the robot pose estimate.

$$\begin{aligned} [x_{t+1}, y_{t+1}, z_{t+1}, \alpha_{t+1}, \beta_{t+1}, \theta_{t+1}] = [x_t, y_t, z_t, \alpha_t, \beta_t, \theta_t] + [ux_t, uy_t, uz_t, wx_t, wy_t, wz_t] \\ \cdot \begin{bmatrix} cos(\theta_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & sin(\theta_t) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \Delta t \end{aligned} \tag{2.12}$$

As we can see even a simple motion model such as differential drive model contains non-linear functions and thus Kalman filter can not be used on it. We can overcome this limitation of Kalman filter by approximating non-linear motion model with first order Taylor polynome. This practilay means that the model funcions are replaced with its first derivatives in particular point. The particular point is typically given by prior pose of the robot. In a small surrounding of given point the derivative is a sufficient approximation. This modification turns motion model of the robot - the matrix B_t in equation 2.9 into jacobian. Kalman filter with this modification is called Extended Kalman Filter.

2.4.2 Particle filter

Kalman filter allows to model only gaussian distribution. It is often sufficient but in real world distribution is only rarely a pure gaussian. Sometimes distribution of real world random variable is difficult to describe by mathematical function. It is obtaines by experiment and then it is approximated using some kind of mathematical model that models the distribution with sufficient precision for given application. One of those approaches is particle filter [45]. It allows us to model arbitrary distribution with set of particles where each particle represents one particular hypothesis of random variable. Entire set of particles models the desired distribution. The particle is defined as 2.13. The $x^{[i]}$ is the hypothesis and $w^{[i]}$ is weight of the particle. The weight says how well the hypothesis fits the distirbution we are trying to model.

$$\Psi = \left\langle x^{[i],w^{[i]}} \right\rangle \tag{2.13}$$

After updating the weights the set of particles has important property described by equation 2.14. The equation says that sum of dirac distributions of variable x with vertices at particular hypotheses weighted by corresponding weights is equal to distribution of variable x. The dirac distribution is a distribution with one infinitely high peek at given coordinate (in our case at hypothesis $x^{[i]}$) and zero value elsewhere. Integral of dirac distribution is of course equal to 1.

$$p(x) = \sum_{i=1}^{N} w^{[i]} \delta_{x^{[i]}}(x) \tag{2.14}$$

The problem of partifle filter is the method of generating samples with desired distribution. We have distribution $p(x)$ modeled by function $f(x)$. The problem is that the function $f(x)$ is usually unknown - we only have function values obtained by experiments or by expertise. We can overcome this by adapting weights of the particles generated by known distribution function $g(x)$ to model desired uknown distribution function. It is better if $g(x)$ has similar shape to $f(x)$ but it can have almost arbitrary shape with only one condition 2.15 saying that the $g(x)$ has to be non-zero in every point where $f(x)$ is non-zero. It practically means that $g(x)$ has a chance to generate sample for every possible non-zero value of $f(x)$. Typically a normal distribution is used so $g(x)$ is a gaussian.

$$\forall_x : f(x) > 0 \Rightarrow g(x) > 0 \tag{2.15}$$

When particle is generated by $g(x)$ its weight needs to be adapted to reflect how well the hypothesis represented by particle fits the desired distribution modeled by $f(x)$. The weight of newly generated particle is set by equation 2.16. The value of $f(x)$ can be estimated using simple interpolation of experimentaly obtained data.

$$w^{[k]} = \frac{f(x^{[k]})}{g(x^{[k]})} \tag{2.16}$$

Particle filter algorithm has following steps:

1. Sample particles using proposal distribution (modeled by $g(x)$).
2. Update particle weights by equation 2.16.

3. Resample according to weights. Resampling is done with probability equal to normalized particle weight.

Despite we generate particles using proposal distribution that is different from target distribution $p(x)$, the resampling step will kill particles that do not fit the target distribution and keep or multiply particles that fit the targed distribution well. After several steps of generating new particles, fitting their weights and resampling the set of particles will model the target distribution.

Advantage of particle filter is that it is non-parametric - we don't have to specify parameters of particle filter except for number of particles. Without specifying any presumptions about random variable the particle filter can deal with almost any target distribution. Limitation of particle filter is given by amount of particles. With growing amount of particles the computational complexity grows. With infinite number of particles the particle filter can model arbitrary distribution. With limited amount of particles the model of target distribution using particle fitler is imperfect. Performance problems with particle filter typically appears in high-dimensional state space where high amount of particles is needed.

2.4.3 Monte-carlo localization

Application of particle filter in localization is Monte-carlo approach [39]. In this approach each hypothesis of robot pose is modeled by particle in particle set. The Monte-carlo localization algorithm has the same steps as particle filter algorithm as Monte-carlo is application of particle filter. The particular steps are more specific. The proposal distribution $g(x)$ as mentioned in equation 2.16 is motion model of the robot. This motion model is sampled (2.17) to get new generation of particles. The motion model is typically represented by N-dimensional gaussian with number of dimensions equal to number of degrees of freedom of mobile robot chassis.

$$x_t^{[i]} \sim p(x_t|x_{t-1}, u_t) \tag{2.17}$$

How well the location proposal matches the reality is evaluated using sensoric observations. The ratio of proposal and target distribution (2.16) is expressed as probability of sensor observation given the hypothesis of robot pose generated by sampling the proposal distribution (motion model). This is expressed by equation 2.18.

$$w_t^{[i]} = p(z_t|x_t^{[i]}) \tag{2.18}$$

If the observations supports the hypothesis of robot location the particle weight is high. If the obervation z_t from robot pose $x_t^{[i]}$ is unlikely the particle weight is low. This way the particles generated by motion model will be reduced to those that are about most likely poses in the map. The particle filter naturally supports multi-modal distribution. It can maintain several hypotheses of a likely robot pose that will be reduced to only one at the moment when they can be distinguished.

The resampling step is not different from resampling step in generic particle filter. There are many resampling strategies. Comparison of different resampling strategies in genetic algorithms was described in [27]. Despite the comparison was done for genetic algorithms it is aplicable to particle colonies in particle filters as well. The most frequently used strategy for Monte-carlo localization is stochastic universal resampling ([27]). This strategy has important feature that if the weights of candidates are equal it replicates the cadidate set to newly generated set without change. This feature is useful in situation when

several equaly likely hypothesis needs to be maintained. The entire Monte-carlo algorithm is summarized in following algorithm 1.

Data: Ψ_{t-1}, u_t, z_t
Result: Ψ_t

$\bar{\Psi}_t = \Psi_t = \emptyset$
for $i = 1$ *to* N **do**
  $x_t^{[i]} \sim p(x_t|x_t^{[i]}mu_t)$
  $w_t^{[i]} = p(z_t|x_t^{[i]})$
  $\bar{\Psi} = \bar{\Psi} + < x_t^{[i]}, w_t^{[i]} >$
end
for $i = 1$ *to* N **do**
  draw j with probability $w_t^{[j]}$
  $\Psi = \Psi + < x_t^{[j]}, w_t^{[j]} >; < x_t^{[j]}, w_t^{[j]} > \in \bar{\Psi}$
end
return(Ψ)

Algorithm 1: Monte-carlo algorithm

The only moment that is not described by the algorithm 1 is initialization of the entire localization process. The initialization is described as a single step of the algorithm but for initialization we need to generate set of particles that cover all non-zero points of target distribution as can be observed in eq. 2.19. Practically it means that we need to cover with particles all places in the map where the robot can possibly appear. If there are no presumptions about robot location the uniformly distributed particles are usually used. But if we provide the algorithm a hint in form of non-uniform distribution with peek in the most likely hypothesis the algorithm will converge faster and there is also a lower probability of falling into local extreme.

$$\forall_x : target(x) > 0 \Rightarrow initial(x) > 0 \tag{2.19}$$

If there is a local extreme that temporarily gives better fit then other areas the sampling step in Monte-carlo algorithm may dispose hypotheses in other points. This will result into stucking in local extreme. Practicaly we can imagine a long corridor with two places that look very similar. Due to imprecision of map and sensor noise one place will fit better. If the robot was moving in one of those two areas and due to map imprecision this area would fit the observations better the resampling would kill out particles that represent the second area after some time. After some time the robot would meet a landmark prooving that the assumption about robot's location was incorrect. There would be no particles already that could be used to generate new hypotheses for the correct location of the robot. Simply said: Monte-carlo has no mechanism of correction if it makes a mistake in final robot location estimate.

This problem can be resolved by adding small amount of random hypotheses into each generation. This approach is used in augmented Monte-carlo algorithm [12]. The resampling step has two conditioned branches. In one of them the hypothesis is randomly chosen from $\bar{\Psi}$ with probability given by particle weight and added into final set Ψ. In the other branch the new particle is generated using some random distribution $rdist()$ and added to set Ψ instead. In the algorithm 2 there is a control mechanism for controlling ratio of particles generated by proposal distribution and particles generated randomly. If particles

in $\bar{\Psi}$ fit the target distribution well there is bigger amount of random particles in the set. If they don't fit well the ratio of random particles is lower - it can be described as „waiting for better times". This augmentation allows the Monte-carlo algorithm to overcome local extremes and always find the global extreme. The reference implementation of this approach for ROS([72]) users is AMCL ([28]).

Data: Ψ_{t-1}, u_t, z_t
Result: Ψ_t

$\bar{\Psi_t} = \Psi_t = \emptyset$
for $i = 1$ *to* N **do**
  $x_t^{[i]} \sim p(x_t|x_t^{[i]}mu_t)$
  $w_t^{[i]} = p(z_t|x_t^{[i]})$
  $\bar{\Psi} = \bar{\Psi} + < x_t^{[i]}, w_t^{[i]} >$
  $w_{avg} = w_{avg} + \frac{1}{N} w_t^{[i]}$
end
$w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$
$w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$ **for** $i = 1$ *to* N **do**
  $v \sim \mathcal{U}$
  if $v <= max(0, 1 - w_{fast}/w_{slow})$ **then**
    $x_{rnd} \sim rdist()$
    $w_{rnd} = p(z_t|x_{rnd})$
    $\Psi = \Psi + < x_{rnd}, w_{rnd} >$
  end
  else
    draw j with probability $w_t^{[j]}$
    $\Psi = \Psi + < x_t^{[j]}, w_t^{[j]} >; < x_t^{[j]}, w_t^{[j]} > \in \bar{\Psi}$
  end
end
return(Ψ)

Algorithm 2: Augmented Monte-carlo algorithm

2.5 Synchronous localization and mapping

Previous chapter concerned about localization of mobile robot in previously known map. In real scenarios it is very typical that there is no map known before the robot is launched and its up to the robot to create one. Process of creating the map and self-localizing in it in the same time is called SLAM (acronym for Synchronous Localization And Mapping). More formaly the SLAM problem can be described as finding map m and actual location of the robot together with trajectory from inital pose ($x_{1:t}$) while we know the inital pose and trajectory of control commands $u_{1:t}$ for the robot and sensoric observations since inital time to actual time $z_{1:t}$. The SLAM problem is defined in 2.20 taken from [5].

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) \tag{2.20}$$

We can distinguish between on-line SLAM where the only last pose of the robot is being estimated while the trajectory remains unchanged and full SLAM when we update also the

trajectory heading to actual pose of the robot. For most of applications finding the map and actaul location in it is sufficient as it is important where the robot is rather than how it got to this place. If we assume that the prior poses of robot are known it also simplifies the SLAM problem itself. Instead of finding trajectories we are finding actual poses only according to prior pose, control and observations. This simplification brings a significatn advantage: Map and actual pose are independent. We are still finding best map in which the robot pose estimation fits best but now we are optimizing expression 2.21.

$$p(x_t|z_{1:t}, u_{1:t}) \cdot p(m|z_{1:t}, x_{1:t}) \tag{2.21}$$

Advantage of this approach is that the complexity does not grow with lenght of trajectory traveled by robot. Of course the limitation is that if some incorrect estimation was done in the past it can not be corrected so the trajectory will be always incorrect. On the other hand in most of applications the trajectory is not needed so this disadvantage is not serious and it does not significantly affect quality of new pose estimation.

2.5.1 Kalman filter based SLAM

Kalman filter based SLAM algorithms use Kalman filter for estimation of robot pose according to motion prediction and observations in time t. For localization of mobile robot in unknown environment the extended kalman filter needs to be used as most motion and observation models are not linear. Replacing non-linear model with local jacobian allows us to use Kalman filter algorithm as defined by set of equations 2.9 anyway. The only difference is that in the prediction step the prediction of new robot pose according to control is not a simpe linear motion but it uses generic non-linear but continuous function to predict new pose of the robot $\bar{\mu_t}$. Estimation of new covariance matrix uses jacobian of motion model. A similar modification applies for observation model that is replaced by non-linear function $h(\mu(t))$ and its jacobian H_t. Entire algorithm of extended Kalman filter after modification can be observed in 2.22.

$$\begin{array}{rcl} \bar{\mu_t} & = & g(u_t, \mu_{t-1}) \\ \bar{\Sigma_t} & = & G_t \Sigma_{t-1} G_t^T + R_t \\ \\ \bar{K_t} & = & \bar{\Sigma_t} {H_t}^T \left(H_t \bar{\Sigma_t} {H_t}^T + Q_t \right)^{-1} \\ \mu_t & = & \bar{\mu_t} + K_t(z_t - h(\bar{\mu_t})) \\ \Sigma_t & = & (I - K_t H_t)\bar{\Sigma_t} \end{array} \tag{2.22}$$

In Kalman filter base SLAM algorithms the entire state is represented by one highly-dimensional guassian. The state contains pose of the robot, pose of landmarks, covariance of robot pose, covariances of landmark positions and also covariances between robot pose estimates and landmark pose estimates. The state vector is described by equation 2.23 and covariance matrix is described by 2.24.

$$x_t = [\mu_R, \mu_{l1} \cdots \mu_{lN}] \tag{2.23}$$

$$\Sigma = \begin{bmatrix} \Sigma_{R,R} & \Sigma_{R,l1} & \cdots & \Sigma_{R,lN} \\ \Sigma_{l1,R} & \Sigma_{l1,l1} & \cdots & \Sigma_{l1,lN} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{lN,R} & \Sigma_{lN,l1} & \cdots & \Sigma_{lN,lN} \end{bmatrix} \tag{2.24}$$

Initially the algorithm starts with $x_t = \mu_t$ and $\Sigma_t = \Sigma_{R,R}$. With any new observation the state vector and covariance matrix is extended with pose and covariance matrixes of the new landmark. If the landmark is re-observed its pose is updated together with update of robot's pose and of course the covariance submatrixes connected with the landmark are updated too. Kalman filter based SLAM algorithms are relatively straightforward application of Kalman filter to SLAM problem but there are several limitations that limit usage of this family of algorithms.

The most important aspect of Kalman filter based SLAM algorithms is computational complexity. Entire state of the SLAM including robot pose and map is represented by a single state vector and signle covariance matrix. As we can observe each new landmark extends the state vector by M elements and covariance matrix by M elements in each direction. For example for 3D SLAM with robot pose represented by 6D vector and landmark pose represented by 3D vector with corresponsing 6x6 and 3x3 covariance matrixes respectively the size of state vector will be $6+3N$ and dimensions of covariance matrix will be $< 6+3N, 6+3N >$. Asymptotic computational complexity of the algorithm is $\mathcal{O}(n^2)$ where n is amount of observed landmarks ($n = N$). The complexity is given by operations with covariance matrix.

Another limitation of Kalman filter based SLAM algorithms is requirement for correct landmark association. It means that once the landmark is re-observed we need to ensure that we recognize that landmark as the proper one that was already observed. If the association fails the state will be extended with a „ghost“ landmark that will increase complexity. Moreover that false detected landmark may compete with previously observed landmark for associations and thus could bring even more uncertainity instead of improving estimations. As the state of system represent only one hypothesis with uncertainity there is no recovery after this hypothesis becomes wrong.

2.5.2 Particle filter based SLAM

Particle filter is non-parametric recursive Bayes filter. Principle of particle filter was described above. In particle filter based SLAM methods the posterior is represented by set of particles so unlike the Kalman filter based algorithms the posterior distribution is not limited to gaussian. The abstract steps of the algorithm are similar to Kalman filter. First new proposal distribution is predicted according to actual pose and control. The proposal is represented by set of particles generated from actual pose of the robot x_t using motion model with uncertainity. Generating new set of particles is actually sampling of the proposal distribution as drafted in equation 2.25.

$$x_t^{[i]} \sim proposal(x_t|x_{t-1}, u_t) \tag{2.25}$$

In next step particle weights are updated. In this process the observation model is utilized to convert measurements to observations. Each particle is evaluated with update of its weight. The weight update reflects how well the particle fits into target distribution given by observations. A generic weight update equation is 2.26.

$$w_t^{[i]} = \frac{target(x_t^{[i]})}{proposal(x_t^{[i]})} \tag{2.26}$$

Finally the particles are re-sampled according to its weights to choose the best particles for next generation. This generic algorithm is the same as generic particle filter algorithm

described in chapter 2.4.2. The importance of particle filter for SLAM is semantics of particle. Each particle represents one hypothesis about pose of the robot in the environment and about model of the environment (map) - shortly the particle represents hypothesis about actual state. In case of feature-based SLAM algorithms that represent map as a set of features the state representation (particle) contain also set of these features.

Important aspect of particle filter based SLAM algorithms is that they maintain a set of various hypotheses instead of single hypothesis like Kalman filter based algorithms. Moreover new hypotheses are generated in every iteration during proposal sampling. This approach allows to correct improper hypotheses containing for example broken maps or very unlikely pose estimation. Also the landmark association process itself can be simplified with set of particles representing various association hypotheses. By the time the incorrect hypotheses will die out with high probability while the correct ones will survive.

2.5.3 FAST-SLAM

The basic FAST-SLAM algorithm is a particle filter based SLAM algorithm with feature-based map representation. Every object in the map is represented by vector of features. In FAST SLAM every particle represents actual map as a vector of features of all mapped objects and trajectory of the robot since initialization to actual time as described in 2.27.

$$i_j = \langle x_{j,1:t}, m_{j,1}, m_{j,2}...m_{j,N} \rangle \tag{2.27}$$

Figure 2.2 shows conditional dependency of landmark L_x observations on robot new pose X_{t+1} and previous trajectory of the robot $X_{t,t-1,..}$. The depencency graph corresponds to a particle representation described in 2.27. If we look at each particle as a map (set of observations) for given robot pose we can say that the particular observations are independent on each other - given the robot pose. Thanks to this presumptions we can decompose the particle representation into robot pose and set of independent N-dimensional observation coordinates. This gives us two important advantages: First - we need to sample only probability distribution of robot pose and second - we can represent each observation with independent model. This simplifies the problem a lot. The set of particles will cover only possible robot poses. Each particle will „cary“ map of the surrounding environment describing how would the environment look if the robot pose was the pose represented by the particle. And the map of the environment will be significantly simplified - we can represent each observation with a small N-dimensional extended Kalman filter where N is number of dimensions the SLAM works with (usually 2 or 3 dimensions).

The conditional probability of observation z_{t+1} is given by equation 2.28. The probability of $X_{t+1:1}$ can be expressed by equation 2.29. The equation 2.29 applies recursively to each position of the robot in the history since beginning the SLAM process.

$$P(z_{t+1}|L_2, X_{t+1:1}) = \frac{P(z_{t+1}, L_2, X_{t+1:1})}{P(L_2) \cdot P(X_{t+1:1})} \tag{2.28}$$

$$P(X_{t+1:1}) = P(X_t + 1|X_{t:1}, u(t+1)) \cdot P(u_{t+1}) \cdot PX_{t:1} \tag{2.29}$$

If we declare position in each step as known the dependency graph reduces to isolated subgraphs as can be observed in figure 2.3.

The conditional probabilty of z_{t+1} can be now expressed in much more simplified way 2.30. This makes the situation much easier as there is no dependency on history. The dependency graphs 2.2 and 2.3 apply to motion model where we predict new pose of the

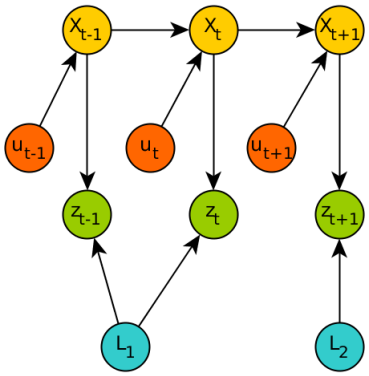

Figure 2.2: Conditional dependency of observation in SLAM process.

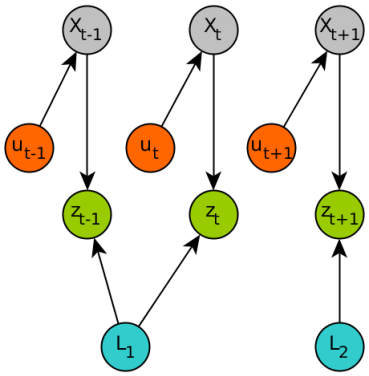

Figure 2.3: Conditional dependency of observation in SLAM process - given pose of the robot.

robot according to last pose and control action. The dependency of the observation on the pose of the robot can be expressed in oposite direction as dependency of the robot pose on actual observation using Bayes formula [46]. Saying that we know the pose of the robot in every point of its trajectory we can split complicated high-dimensional model of robot pose and all landmarks observed during robot travel into smaller independent models for each landmark.

$$P(z_{t+1}|L_2, X_{t+1}) = \frac{P(z_{t+1}, L_2, X_{t+1})}{P(L_2) \cdot P(X_{t+1})} \tag{2.30}$$

The process of reducing dimensionality without losing properties of probabilistic model is called Rao-Blackwellization. The Rao-Blackwellization is a more generic process of transforming estimator of unobservable random variable into estimator of observable random variable that satisfies the Rao-Blackwell theorem [42]. In case of FAST SLAM we model joint distribution of robot trajectories and possible maps with conditional distribution of possible maps given the robot trajectory multiplied with distribution of possible trajectories (2.31). Application of Rao-Blackwelization allows us to model the high-dimensional model compounding of robot trajectory and map by set of particles and apply particle filter on it.

$$P(x_{1:t}, m_{1:t}) = P(m_{1:t}|x_{1:t}) \cdot P(x_{1:t}) \tag{2.31}$$

The FAST SLAM algorithm compounds of four steps:

1. Prediction of new particle coordinates.
2. Update of particle weights according to observations.
3. Update EKFs for observations.
4. Resampling - generating new set of particles.

Prediction of new particle coordinates

New particle coordinates are predicted using motion model of the robot. In general new pose of particles can be expressed by eq 2.32. The new particle is obtained by sampling probabilistic model of new robot pose given prior pose of the particle x_{t-1} and control action u_t.

$$x_t^{[k]} \sim p(x_t|x_{t-1}^{[k]}, u_t) \tag{2.32}$$

Update of particle weights according to observation

New particle weight is updated using gaussian distribution with mean equal to expected observation $\hat{z}$ of each particle - eq 2.33. With growing distance between expected and measured observation the particle weight decrease. The Q is a measurement covariance matrix saying how precise the measurement is. The k is index of particle in set of particles ψ.

$$w_t^{[k]} = |2\pi Q|^{-\frac{1}{2}} exp\left(-\frac{1}{2}(z_t - \hat{z}^{[k]})^T Q^{-1}(z_t - \hat{z}^{[k]})\right) \tag{2.33}$$

The measurement covariance Q takes into account previous covariance matrix of EKF representing position of given landmark and measurement noise as described by equation 2.34.

$$Q = H\Sigma_{j,t-1}^{[k]}H^T + Q_t \tag{2.34}$$

Update EKF for observations

Updating the EKF model for observation is a bit more complicated as we need to deal with situation when the landmark was observerd for the first time. In such case we have to initialize EKF for given landmark. This step is usually tightly connected with prior step of weight update. Getting observation j in time t we get following initialization procedure for set of N particles that are indexed by k:

$$\begin{aligned} \mu_{j,t}^{[k]} &= h_{inv}(x_t^{[k]}, z_t) \\ H &= h'(\mu_{j,t}^{[k]}, x_t^{[k]}) \\ \Sigma_{j,t}^{[k]} &= H^{-1}Q_t(H^{-1})^T \end{aligned} \tag{2.35}$$

The $\mu_{j,k}^{[k]}$ and $\Sigma_{j,t}^{[k]}$ are parameters of EKF. The H is jacobian observation model h ([76]) at point $\mu_{j,t}^{[k]}$, $x_t^{[k]}$. Note that we assume that we process only one observation in one iteration of the algorithm. If we need to process more observations, we will repeat the iteration with same pose of the robot. If the landmark was observed for the first time, weight of the particle will be initialized by default value $w^{[k]} = p_0$ as we don't have mean of the EKF with which we could compare the observation.

If the landmark was already observed in the past we will update EKF according to following set of equations:

$$\begin{aligned} \hat{z}^{[k]} &= h(\mu_{j,t-1}^{[k]}, x^{[k]}) \\ H &= h'(\mu_{j,t-1}^{[k]}, x_t^{[k]}) \\ Q &= H\Sigma_{j,t-1}^{[k]}H^T + Q_t \\ K &= \Sigma_{j,t-1}^{[k]}H^TQ^{-1} \\ \mu_{j,t} &= \mu_{j,t-1} + K(z_t - \hat{z}^{[k]}) \\ \Sigma_{j,t} &= (I - KH)\Sigma_{j,t-1}^{[k]} \end{aligned} \tag{2.36}$$

For all features that were not observed simply the EKF is not updated.

Resampling - generating new set of particles

For resampling there can be used several algorithms. Probably the most straightforward is the bin algorithm where each particle has its „bin“ - an interval in range $< 0, weight_sum >$. The size of the particular bin is naturally given by weight of the particle. All the intervals are placed one next to another to form integral space without gaps. Then values from interval $< 0, weight_sum >$ are generated N-times, where N is desired number of particles in new generation. Particle is chosen into the new set if random value hits its bin. Particles with high weight will probably be selected several times while particles with low weight will be selected only one time or completely skipped.

Features of FAST SLAM algorithm

The FAST SLAM is effective in low-dimensional space that can be effectively covered by particles (typical for particle filter based algorithms). Its computational demands does not grow with growing set of mapped landmarks - it is only affected by number of particles. At this point independence of models for particular landmarks proves itself as very important feature. Probably the most important feature coming from particle based nature of the algorithm is effective dealing with data association. FAST SLAM implements the multi-modal belief by design. Each particle has its own association of observations to landmarks. If the association is incorrect the particle will sooner or later naturaly die due to low weight.

Disadvantage of FAST SLAM is that robot pose has no uncertainity. The uncertainity is modelled by set of particles but one particle with highest weight represents the robot pose and this particle represent particular coordinates with no uncertainity. This disadvantage in combination with noisy motion model leads to generating a lot of particles that are later disposed due to non-matching observations. This problem is significantly reduced in FAST SLAM 2.0 ([18]) algorithm which takes into account also observations when predicting new particle poses as described by equation 2.37.

$$x_t^{[k]} \sim p(x_t|u_t, x_{t-1}^{[k]}, z_t) \tag{2.37}$$

Chapter 3

Actual approaches to outdoor localization

Not all the outdoor localization methods mentioned above work for large outdoor areas. Large areas with small abount of usable markers are a special case for outdoor localization and SLAM. In general most of the methods count with at least one observable landmark any moment in time. This often does not apply to large sparse areas.

3.1 Cathegories of actual approaches

Actively broadcasting external localization systems: Most of actaul approaches rely on external system that provide information for localization. These external systems usually compounds of network of nodes that act as some kind of beacons. Mobile robot measures either distance (lateration) ar angle (angulation) to these nodes. After obtaining enough informations the robot solves a set of equations to find solution - its location. This cathegory covers GNSS, wireless network localization and beacon based localization systems like VOR.

Marker based localization: Another cathegory of localization approaches use observation of external markers - natural or artificial. Natural markers are markers that are recognized in the environment by the mobile robot itself. The markers are natural part of the environment. Their recognition and association is up algorithms used on mobile robot. On the other hand the artificial markers can be recognized easily - they usually come together with sensoric system and algorithm for their recognition and identification. More details about artificial marker based localization can be found in 3.6.

Localization based on external observation: Localization based on external observation uses external cameras or other sensors capable of observing the mobile robot and measure its position in defined area. More about this cathegory of systems can be found in 3.5.

Other approaches: This cathegory cover the approaches to outdoor localization that are not covered by prior cathegories. They are usually specific for particular tasks. One example belonging into this cathegory is wired navigation for autonomous lawn movers. This approach is based on sensing of electromagnetic field by coil. The coil

works as a sensor. The wire cirquit placed under ground denotes the area in which the mover can operate. The mover does not localize itself in this area but it can detect crossing the border line so it can always stay inside. Similar approach is used for navigating transport machines in industrial facilities. The wire in the ground or under ceiling serves as a guidance for mobile machines. The guiding wire may serve also as a communication channel between the mobile machine and operation center.

3.2 GNSS

First in cathegory of localization solutions that rely on external system is satellite localization. Todays there are several existing satellite networks including GPS (USA)[9], Gallileo (Europe) [9], GLONASS (Russian federation) [9] and BeiDou (China) [80]. All these system use lateration to estiamte position on the surface of the Earth. Details of GPS principle were described in chapter 2.3. Generic description of the GPS applies more or less also to other systems.

3.3 Localization using wireless networks

Using network of wireless nodes is another approach to localization ([4]). The nodes can communicate with each other using wireless technology based usually on radio signal. By analyzing properties of the signal the nodes can estimate location of rover traveling amongst them. Location using wireless networks is relative location - we can find relative location to nodes of the network. With additional information about node positions in absolute coordinate system we can compute absolute position of the rover (mobile device with receiver).

From principial point of view there are three approaches to finding relative location of the rover to wireless network nodes as described in [1]: lateration, angulation and combination of both. The lateration measures distances to nodes of wireless network. Knowing three or more distances uniqe position can be computed. Trilateration is visualized by figure 3.1. A three nodes A,B,C with known location are placed in area. Thre rover R travel through the area and measures distances from them. Getting three or more distance measurement the rover can compute its position as intersection of circles with diameters equal to measured distances. Having three measured diameters r_A, r_B, r_C and knowing positions of the nodes A,B,C the position can be computed by solving set of equations 3.1. The set of three quadratic equations with two unknown variables is redundant. Solving two equations together will give us two roots - each of them lies on intersection of circles with given diameters. This situation is visualized on figure 3.1 by possible rover positions R and R'. The third equation is used to eliminate invalid root so we get uniqe solution.

$$\begin{aligned} (R_x - A_x)^2 + (R_y - A_y)^2 &= r_A^2 \\ (R_x - B_x)^2 + (R_y - B_y)^2 &= r_B^2 \\ (R_x - C_x)^2 + (R_y - B_y)^2 &= r_C^2 \end{aligned} \tag{3.1}$$

The triangulation measures angles between wireless nodes and rover. Problem of localizing rover by measured angels is also called resection problem. Knowing three or more angles to nodes with known positions we can compute exact position of the rover. The situation is depicted in figure 3.2.

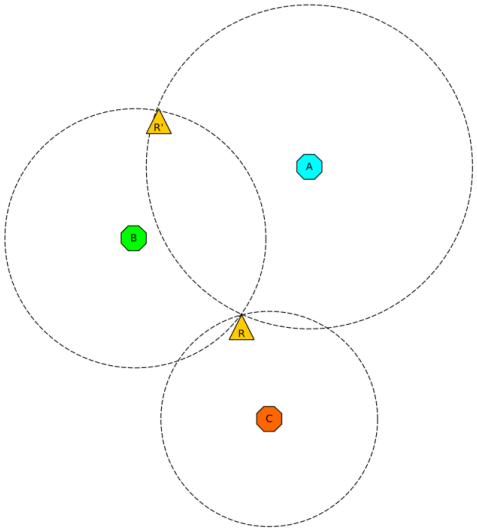

Figure 3.1: Trilateration principle.

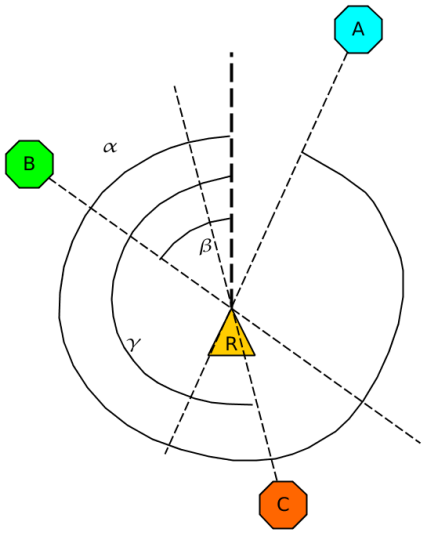

Figure 3.2: Triangulation principle.

There are several approaches of computing position of the rover knowing angles to nodes. The [69] compares 18 resection algorithms. One of effective resecion algorithms called ToTal is shown in algorithm 3. The algorithm is described in the detail in [70].

Data: $\mathbf{A}, \mathbf{B}, \mathbf{C}, \alpha, \beta, \gamma$
Result: $\mathbf{R}$

Computation of delta coordinates:
$\mathbf{A}' = \mathbf{A} - \mathbf{B}$
$\mathbf{C}' = \mathbf{C} - \mathbf{B}$

Computation of cotangens:
$cot_{AB} = cot(\beta - \alpha)$
$cot_{BC} = cot(\gamma - \beta)$
$cot_{CA} = \frac{1 - cot_{AB} cot_{BC}}{cot_{AB} + cot_{BC}}$

Computation of modified cirle centers coordinates
$x'_{AB} = A'_x + cot_{AB} A'_y; y'_{AB} = A'_y - cot_{AB} A'_x$
$x'_{BC} = C'_x + cot_{BC} C'_y; y'_{BC} = C'_y - cot_{BC} C'_x$
$x'_{CA} = (C'_x + A'_x) + cot_{CA}(C_y - A_y); y'_{CA} = (C'_y + A'_y) - cot_{CA}(C_x - A_x)$

$k = A_x C_x + A_y C_y + cot_{CA}(A_x C_y - C_x A_y)$
$D = (x'_{AB} - x'_{BC})(y'_{BC} - y'_{CA}) - (y'_{AB} - y'_{BC})(x'_{BC} - x'_{CA})$

Position of the rover
$R_x = B_x + \frac{k(y'_{AB} - y'_{BC})}{D}$
$R_y = B_y + \frac{k(x'_{BC} - x'_{AB})}{D}$

return$(\mathbf{R})$

Algorithm 3: ToTal triangulation algorithm

Most of localization systems use triangulation or trilateration technique to compute position of the rover. In some cases it is necessary to find only topology of the wireless network. Wireless signal analysis is also helpful in reconstruction of topology and locations of wireless nodes. For topology discovery algorithms based on challenge and response are used such as TopDisc [38]. To achieve a more precise localization of nodes a more sophisticated signal analysis is typically used.

Localization algorithms in wireless networks use signal strength (RSSI) [50] or time-based distance measurement using measuring of round-trip tipe, phase shift of pseudorandom code or phase shift of carrier. Localization based on RSSI use principle of trilateration to localize the rover. The RSSI ranging use relation between transmitted power of the transmitter and received signal power. As radio wave spread with constant energy and spherical surface of wavefloor grows quadraticaly the intensity of the signal in every single point of the wavefloor decrease. Relation between intensity and distance from the source of waves is given by relation 3.2. From this relation we can express the distance 3.3.

$$I = \frac{P}{4\pi r^2} \tag{3.2}$$

$$r = \sqrt{\frac{P}{4\pi I}} \tag{3.3}$$

In real world with obstacles the signal is attenuated by passing through the obstacles and spreads trhough different paths due to reflection. There is also noise from other wireless traffic that interferes with the signal. This is the reason why naive approach has usually poor results. There are several models of the environment that model realistic spreading of the signal. With environment models the precision of RSSI localization can be improved to sub-meter precision under ideal circumstances. Another problem is that the decrease of the amplitude due to distance is non-linear. In larger distances the resolution of RSSI in dB/m decrease.

Distance measurement based on time difference is in general more robust and more precise. One of the most straightforward time measuring principles is measuring the round-trip time. If we have a network of nodes with well known localtion we can measure distances to particular nodes and compute the distance using trilateration as mentioned in [16]. Localization in cellular network is convenient because there are existing networks for mobile telephony and data transfers. In todays crowded world even the round-trip time itself is sometimes insufficient. The [56] suggests augmentation to the RTT-based approach using Bayesian inference method combining SINR (Sognal to Interference plus Noise Ratio) measurements with RTT or machine learning approach when there is a map of signal strength obtained by supervised learning method.

The RTT or time delays in general can be measured using several approaches. One of them is phase shift of sent and received code sequence. The sender sends usually pseudo-random code sequence, receiver receives the sequence and sends it back immediately. The sender receives the returned code sequence and iteratively computes correlation of sent and received sequence. With each iteration receiver shifts the received sequence one bit forward and computes corellation with sent signal. When the correlation is highest the time delay is computed as number of shifts multiplied by code symbol lenght. The round-trip time is a sum of signal travel to receiver, receiver processing time and signal travel from receiver back to sender. Receiver processing time may vary. This source of imprecision can be eliminated by measuring the RTT several times and averaging it. Still the precision of code phase shift is limited by legth of code symbol. To achieve better resolution some solutions use phase shift of carrier signal. As a code symbol is representet by two or more periods of carrier signal using carrier signal brings much better resolution. On the other hand carrier signal processing brings harder demands on signal processing [14].

3.4 Localization using beacons

Principal of localization using beacons is very similar to localization using wireless networks. The difference is that the beacons are designed for purpose of localization - in contrast to GSM cellular network localization where the network is intended for a different purpose and it is just „abused“ for localization. Example of such solution is VHF omnidirectional range (VOR) [36]. VOR is a beacon based localization system that uses angulation to estimate position of mobile rover. It was designed for aerial navigation and it is still a primary purpose of VOR. To measure angle from VOR beacon to rover the beacon transmits a two signals - one is omnidirectional and the another is a directional signal. The rover detects the highest amplitude of the directional signal and in this moment it measures phase shift of the directional and omnidirectional signal. With several angular measurements the actual location of the rover can be computed.

3.5 External visual localization

Another cathegory of localization systems are systems based on observating mobile rover by cameras [11] or other optical sensors like laser rangefinders [59]. Systems based on cameras use two or more cameras for measuring position of the rover. The rover is detected in camera image and angle to the rover is computed. Knowing exact positions of the cameras and angle between camera centers and the rover the rover's pose can be computed using angulation principle. Of course the localizaiton system needs to have some prior knowledge of the rover's shape and texture to be able to distinguish it from the background. The rover is often equipped with special markers to ease the detecion. With sufficient resolution of localization system and sufficient amount of distinguishsable landmarks it is possible to track motion of particular parts of the rover. Systems like VICON [63] allow detailed motion tracking of particular parts of the rover. For precise motion tracking and localization the image from cameras is often combined with depth information provided by depth scanners or 3D LIDARs.

3.6 Marker based localization

This method of localization uses prepared markers installed in the environment. The markers are usually in the form that can be easily recognized and distinguished from each other [19]. The localization system is installed on the rover. The rover detects markers in the nevironment and recognizes particular markers. According to relative position of the rover to the marker and prior knowledge of position of the marker the postion of the rover can be computed. If the markers are supposed to be observed by camera they have predefined shape and color and they contain some kind of code that allows distinguishing particular markers. Example of such marker is QR-code or barcode.

Another method of localization based on markers is laser reflection guidance [31]. This method uses reflection of laser beam from markers adapted to particular wavelenght of the laser. This way the markers can be easily distinguished from background. This approach may use angulation and lateration principle together to improve precision and robustness of the system. Despite that this solution can be used in outdoor environments typical application of this solution is indoor industrial environment like warehouses, docks and large factories. Poses of markers are known to the rover so it can easily estimate its location when a marker is detected. Reflective markers are usually installed under the ceiling of the hall so they can be observed from most places of the area. Thanks to good observability the amount of markers can be significantly lower without affecting robustness of entire solution.

RFID [32] is a technology of wireless exchange of informations between initator of communication - interrogator in terminology of RFID - and communication slaves - the transponders. The transponders can be very small devices. There are three variants of RFID transponders: active, battery assisted and passive. The active transponders have their own power source and work as a common wireless communication devices. They listen to the wireless communication and when they are challenged they respond with requested information. All parts of the active transponder are powered from its own power source.

The battery assisted transponders have still its own source of power but this source is usually very minimalistic with limited power. The power source only powers the internal control cirquit of the transponder but not the RF part. Energy for wireless communication is taken from interrogator. When interrogator challenges a transponder it broadcasts the

request in form of electromagnetic wave. This wave is modulated and carries the content of the challenge but it also transports enegry and thanks to this it can be also considered as a minimalistic power source. Energy of the wireless signal is sufficient for the transponder to power its own transmitter and transmit the information.

The last variant of transponder is passive transponder. It has no own source of energy. When the challenge from the rover is received the power of tha signal is used to power transmitting part and also the control logic of the RFID transponder. This small amount of energy is sufficient to interpret the challenge from the interrogator, generate response and transmit the response back to the interrogator. Of course in case of battery assisted and passive RFID transponders the reach of the response is very small - tens of milimeters usually. Advantage of passive RFID transponders is their size. The antena needs to be large enought to receive enought of energy but it can be completely flat. With small internal chip the RFID transponder can be in form of a tag or label.

For purposes of localization the RFID transponders are installed at particular positions in the environment [30]. They can be observed using the RFID interrogator. Short range of RFID communication is advantage because it allows the rover to estimate its position according to observed RFID tag. With greater range the position estimation would be less precise. Disadvantage of short communication range of RFID tags is that rover discovers them only when it crosses over them. This is why are the RFID tags used for localization usually installed on the ground.

There are also less sophisticated but very precise methods of localization that can be included to marker based localization. One of them is guidance line observed by optical reflection sensor. This apporoach is used usually in indoor environments with flat floor where the line can be easily detected. Equivalent of this approach in outdoor environment is wire guidance using induction loop. The line is replaced by wire placed few centimeters under ground. The wire makes a loop that is powered by pulse power supply. Instead of using reflection sensor the wire is detected by system of coils. These approaches are described in [10].

3.7 Limitations of actual methods

It is not a surprite that each approach to the SLAM have its limitations. Particular methods based on algorithms described above fulfil particular set of requirements while they fail to fulfil others. Developing an universal SLAM method that would work in arbitrary environment is still an ultimate goal. Some of the methods work only in a flat terrain as they use 2D mapping, others based on 3D laser scanning with voxel grid map are very robust in almost any environment but very resource demanding so they can not be easily used for very large environments. Moreover 2D or 3D laser scans do not work very well in large flat areas with a very few objects with distinguishable geometry. When there is no observable object in reach of sensors the SLAM algorithm can not update posterior and has to rely on dead-reckoning only which is usually not very reliable - especially in hard terrain.

Another approach is to use localization based on network of external navigation nodes. This include global navigation satellite systems with network of satellites like GPS and network of wireless nodes that help with the navigation. This cathegory of localization methods have usually the best performance. Position error is limited by network properties so systematic error can be avoided. Also solving problem of lost robot is trivial with localization in network active nodes. Localization with wireless network could be also energy efficient if the robot only passively listens to signal broadcasted by nodes of the

network. This all applies to the situations when robot can receive signal from the network. Once the signal is cut off the robot has to rely on dead-reckoning from last known possition. Despite these limitation the localization using some kind of wireless network is the most frequently used in todays world.

There is also another problem limitation of wireless network localization. This mode of localization usually uses N-lateration to localize the mobile rover. Despite the trilateration provides very good estimation of pose in the coordinate system of the network it principially can not provide estimation of orientation. Of course if the mobile rover moves we can assume that it is moving ahead so we can estimate where is it heading but this solution can not recognize if the mobile rover is moving backward or strafe left or right. This problem can be partially mitigated with fusing wireless network or satellite based localization with inertial measurement unit and odometry as described in [67] but in extreme case when the rover would spin on a spot the error of orientation estimation would grow without limits. There are exsiting systems that use N-angulation istead of N-lateration - particularly VHF Omni Range (VOR) [36]. Despite ongoing upgrades the network of VOR beacond does not offer precision comaprabele with GPS and other global navigaiton satellite systems.

Passive artificial landmarks do not differ from localization methods that use natural landmarks but they differ in overal properties. Artifical landmarks are usually a way more reliable than natural ones - this is why they are being used. They can be reliably detected and usually also reliably associated so it helps to mitigate the re-observability and association problems. If the pose of artificial landmarks is known in advance they can also limit the maximal error of mapping - even warped map can be corrected by observing landmark with known pose. The limitation of this attitude is that network of artificial landmarks has to be created before the robot can localize. And in outdoor environments it has to be maintained. It brings additional effor if the landmarks are passive and can not report its state as satellites or wireless network nodes.

Another option is external system that observes the mobile robot and estimates its location according to the observations. The localization is not a matter of the robot but it is a matter of a network of nodes that observe particular area in which the robot operates. The robot can be equipped with special beacons that are detectable by nodes in the observation network to improve reliability. System like this can offer a very good precision of localization. With properly configured network it can reach a millimeter precision. On the other hand establishing and maintaining the network is usually reather time consuming and expensive task. Practically the observation networks are usually rather small. The network is the most significat disadvantage of such system. Example of such system is VICON [63]. Advantages are good precision, independence on robot sensoric subsystem so operation of the robot does not affect localization precision. Disadvantage is dependency on observation network and also maintenance of this network.

Chapter 4

Goals of this work

According to limitations of existing methods described above goals of this work were declared. Instead of trying to create an ultimate solution for SLAM that would work in arbitrary environment which is a goal that is really hard to achieve I decided to cover a gap in variety of existing solutions. Most of the outdoor localization solutions aim on localization in relatively dense environments with artifical structures like houses or roads. This finding helped me to define the primary goal of this work.

Develop solution of localization in large outdoor areas: The primary goal is to develop a SLAM soltion that would work in large outdoor environments with only few objects that can be considered as landmarks.

Indepence on external systems: Another very important goal is to create a self-contained SLAM solution. By self-contained I mean a solution without any external supportive systems (active or passive) - all the sensors used for SLAM are installed on the robot itself. Despite there are satellite navigation systems with good precision it is sometimes not possible to receive GNSS signal.

Improve robustness of dead-reckoning: As my solution is supposed to operate in areas with only few distinguishable objects the moments when no landmark is in sight of sensors will not be rare. Third goal is to improve dead-reckoning to make it more reliable in such situations.

Chapter 5

Novel solution - long-range localization without external support systems

According to goals declared in previous chapter a new solution for localization system was was designed. Goal of this effort is to provide solution for localization in large outdoor environments with sparse markers. As mentioned in the goals this solution needs to deal with situations when there are no observations available so it will need reliable and precise dead-reckoning. To operate without external support systems the only solutions based on SLAM techniques come to account.

5.1 Basic principles of the approach

As the SLAM solution is supposed to work in large areas it is necessary to maintain a map of the area. There are no assuptions about terrain so it needs to maintain map of the environment in 3D. Feature-based SLAM method best suits these requirements. FAST-SLAM algorithm was choosed as the core of SLAM algorithm for this solution. Map is represented by set of guassians that allow to model uncertainity. Pose hypothesis is represented by particle and uncertainity of pose is represented by set of particles.

Very important is question of markers that should be detected by this solution. The environment is an area of large size with only few objects in it. Typical aspect of he objects is that they protrude above the terrain. Finding these objects by 3D laser scan is difficult as they are too far for most of sensors with reach in tens of meters and what is more important even if the sensors could reach the objects the density of distance measurements would be probably too low to find many of objects we are interested in. Typical object that protrude above ground in large natural environment is a tree. As the branches with leaves are not very reliable marker because of reflections and parial opacity the most interesting part of the tree is the trunk. As the trunk has a cylindrical shape it can be observed from other directions without significan loss of precision. So amongst the artificial objects it would be useful and in some cases necessary to find tree trunk and measure distance to it.

This requirement limit selection of sensors used for the task. As we can barely afford to cover environment with distance measurements due to reach of sensors and density of scans the only way is to measure distances selectively. To make selective measurement we need to find the object first and then point the rangefinder at it. Finding objects in

Figure 5.1: Sensoric system installed on experimental robot RUDA.

3D environment is not easy. Scanning the environment blindly with a rangefinder would be too slow. To find the objects a visual-based method was choosed. The object can be found in camera view and then the long range single point rangefinder can be pointed to it. Only laser rangefinders offer required reach with necessary precision to measure distance to one particular distant object. To orient the rangefinder it has to be installed on a pan-tilt manipulator.

To overcome imprecision of the manipulator the camera has to be installed on it too and alligned with the rangefinder. This was the sensoric system solution used for this work: Camera with single point laser rangefinder alligned together (parallel axis) installed on top of P-T manipulator. Photo of sensoric system installed on robot RUDA that was designed and constructed at Faculty of Information Technology at Brno University of Technology can be observed in image 5.1.

One of the goals is improvement of precision and stability of dead-reckoning. Dead-reckoning is also important part of entire solution. Basic input for dead reckoning is odometry based on geometrical model of robot chassis. This can be used as initial estimate but to achieve a good precision odometry has to be fused with additional sensors. Odometry is relatively precise for linear motion but rotational precision has to be improved. To achieve this fusion with visual compass was choosed. Visual compass is not a pure dead-reckoning but it takes into account surrounding of the robot and thus it has potential of better robustness. Many visual compass solutions are based on detecting particular features between consecutive frames to estimate rotation difference. In the large relatively empty environment it could be difficult to find reasonable features. This is why a feature-less approach was choosed.

In proposed solution the odometry fused with visual compass serves as input for motion model of SLAM algorithm. Uncertainity of the fused odometry is modelled by differential drive motion model to obtain more precise proposal distribution.

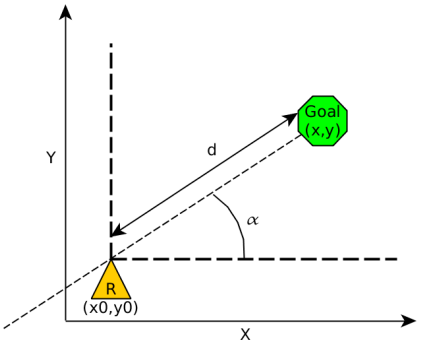

Figure 5.2: Impact of encoder error - model situation.

5.2 Orientation measurement

For proper work of SLAM approach described above it is very important to have a good information about orinetation of the robot. Moreover to deal with situations when no observations are available the proper orientation measurement is essential. One of orientation sources is odometry. Odometry quality may vary from robot to robot. With falling prices of high resolution encoders most of todays robots provide odometry feedback with high resolution and with minimal noise - especially with optical encoders in closed housing. Precise encoders provide very good feedback of linear motion. Most of todays robots use differential drive. One of aspects of differential drive is that wheels are drifting during turning of the robot. In such cases the precision of odometry drops as it is hard to estimate impacts of the drift. It is also a common approach today that encoders with direction detection are used. The direction detection helps to eliminate noise caused by spinning the weel by external forces. When robot moves through the terrain and stops at the slope the gravity pull the robot down. If the robot uses electrodynamic braking [61] that is less effective in low speeds the wheel may slowly turn and the robot will move in the direction of gravity force. The fact that the robot is moving despite the motors are not running will be detected by encoders and direction detection is essential for proper measurement of such movement.

Thanks to improvements mentioned above the encoders provide very precise feedback. This feedback from encoders usually provides a good corelation with reality in case of linear movement. In case of turning there often appears a slippage of the wheels. This situation when wheels loose contact with a terrain and slip can not be detected by encoders. Another problem is that orientation error has much more serios impact on total precision of dead-reckoning than error in linear move. In a model sitaution depicted by figure 5.2 robot turns on a spot and then travels given distance straight forward as described by equations 5.1. The effect of linear and angular error is documented by set of equations 5.2. In equation 5.1 the α is angle of rotation and d is the traveled distance.

$$\begin{aligned} x &= x_0 + sin(\alpha)d \\ y &= y_0 + cos(\alpha)d \end{aligned} \tag{5.1}$$

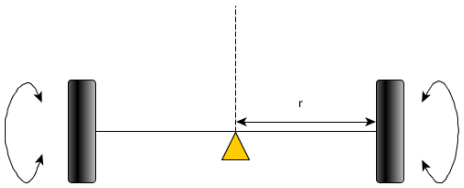

Figure 5.3: Differential drive.

The set of equations 5.2 describe the same situation as equations 5.1 but it is extended with error ϵ of encoders. As a motion model of the robot a differential drive model depicted by 5.3 was used. Parameter of differential drive is the distance between centers of wheels. When turning at the spot the center of the robot does not travel - it only rotates. For convenience of use in following equations only distance between center of the axle and wheel r was used. This distance is equal for both wheels.

$$\begin{array}{lcl} wheeltravel & = & \alpha \cdot r \cdot (1+\epsilon) \\ \alpha_\epsilon & = & \frac{wheeltravel}{2\pi r} 2\pi \\ subst.:\alpha_\epsilon & = & \frac{\alpha \cdot r \cdot (1+\epsilon)}{2\pi r} 2\pi \\ subst.:\alpha_\epsilon & = & \alpha(1+\epsilon) \\ d_\epsilon & = & d(1+\epsilon) \\ x & = & x_0 + sin(\alpha_\epsilon) d_\epsilon \\ y & = & y_0 + cos(\alpha_\epsilon) d_\epsilon \end{array} \tag{5.2}$$

The odometry error is usually not perfectly constant. The error caused by slipping can be reasonably modeled using normal distribution. In the model described by 5.3 the normal distribution models size of the wheel slippage. The ϵ is proportional error relative to the distance traveled by wheels. It can be converted to percents by multiplying by constant 100. In this model the angular and linear proportial errors have the same value despite in real situations the slipping occures more frequently during turning. Larger slip during rotation is much more significant in case of chassis with tracks.

$$\begin{array}{lcl} x & \sim & x_0 + sin(\mathcal{N}(\alpha, (\alpha\epsilon)^2))\mathcal{N}(d, (d\epsilon)^2) \\ y & \sim & x_0 + cos(\mathcal{N}(\alpha, (\alpha\epsilon)^2))\mathcal{N}(d, (d\epsilon)^2) \end{array} \tag{5.3}$$

To visualize the effect of slipping the following values were choosed. The second value of the alpha models the situation when robot executes one 360 degrees rotation in addition to the turn. Figures 5.4 and 5.5 visualize the model described above. We can see that the effect of one additional rotation to precision of dead-reckoning is significant. Of course the real parameters would vary according to particular chassis and friction coeficient between wheels or tracks and the terrain but the principle of the error will remain the same. The experiment was executed with 50 samples for each configuration.

- $d = 200m$
- $\alpha = \frac{1}{3}\pi, \frac{7}{3}\pi$
- $\epsilon = 0.005$
- $x_0 = 0, y_0 = 0$

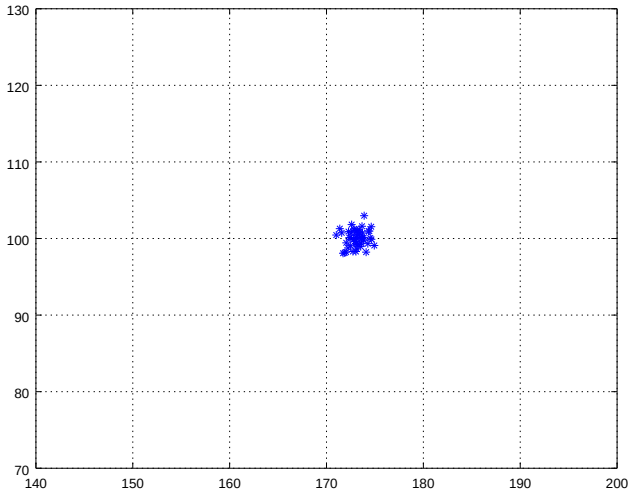

Figure 5.4: Pose variance due to slipping ($\alpha = \frac{1}{3}\pi$). Distances in [m], variance of encoders is 5mm per 1m.

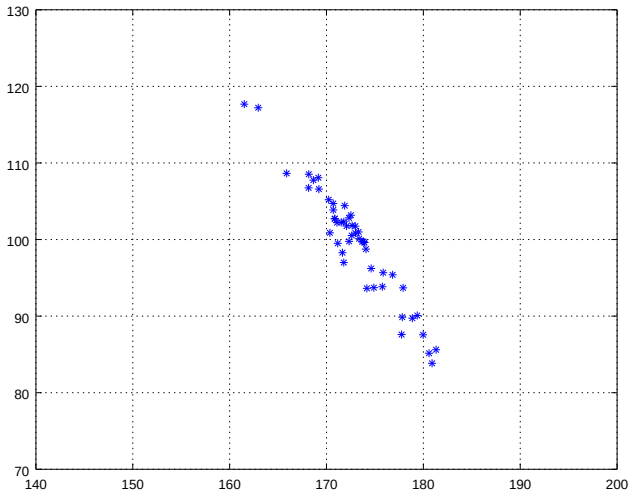

Figure 5.5: Pose variance due to slipping ($\alpha = \frac{7}{3}\pi$). Distances in [m], variance of encoders is 5mm per 1m.

Problem with wheel slippage can be solved by using fusion with additional sensors [24]. Very frequent solution is fusion with inertial measurement unit (IMU). IMU measures acceleration and orientation using gyroscopes and magnetometers. Unfortunately due to a lot of electromagnetic noise the magnetometer has usually poor performance. Accelerometers and gyroscopes provide more reliable values but they are negatively affected by vibrations when robot moves. The vibration generates a lot of noise that has to be filtered out and due to it decrease precision of dead-reckoning based on these sensors. On the other hand their output is very reliable and data from IMU are available at any moment of robot's operation so using IMU as a source of orientation data is a significant advanteage.

Typical solution for dead-reckoning based on sensor data fusion uses extended Kalman filter (EKF) to fuse the data. Odometry is usually used as prediction and sensor measurements are used during correction step [60]. The Kalman filter is effective also in cases when sensor provides data with less dimensions than the rest of the system. This is the case of optical orientation measurement. Data from orientation sensor provide rotation information only with no distance information. To make the fusion of orientation data with the rest of the system using EKF possible it is necessary to extend the output matrix to the same dimensionality. In this case the orientation sensor will be visual compass.

5.3 Visual compass

For the most generic situation with 6DOF state the data vector and covariance matrix of visual compass have following form described in 5.4 and 5.5.

$$\bar{\mu}_{visual} = [0, 0, 0, 0, 0, \theta] \tag{5.4}$$

$$\Sigma_{visual} = \begin{bmatrix} \infty & 0 & 0 & 0 & 0 & 0 \\ 0 & \infty & 0 & 0 & 0 & 0 \\ 0 & 0 & \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & \infty & 0 & 0 \\ 0 & 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\theta \end{bmatrix} \tag{5.5}$$

There are two approaches for visual compass: A feature-based approach and feature-less approach. The feature based approach monitors shift of features detected in two consecutive frames. A change of orientation is computed from shift of the same feature between two consecutive frames [74]. This approach is computationaly less demanding but there are several restrictions that limit usage for particular purposes and use cases. These restrictions may also bring significant performance improvements. One of them is visual compass using 1D SURF features for computationaly limited robot [25]. This approach is designed for flat terrain only as it detects feature in two lines in the camera image only. Advantage of this approach is significantly lower computation time that makes this solution usable in very small devices with limited memory and CPU performance.

Advantage of feature based approach (except for simplified versions) is direction robustness - it can deal with vertical shift between consecutive frames and usually also with rotation of consecutive frames. Particular type of features depend on application of the solution. Typically SIFT, SURF or ORB features are used. Obvious disadvantage is restriction to particular features. Features limit usage of compass to particular environment or set of environments.

Another approach is feature-less visual compass [53]. Feature-less approach does not detect features in the surrounding environment. Instead it compares whole image with consecutive one. The image comparison is iterational process. With every iteration the second image is shifted by 1 pixel. For every value of the offset the error is computed. The offset with the lowest error defines the rotation difference. Implementation of this solution is described in [54], [55]. Author tested the solution with Manhattan and Euclidean distance. Both distance heuristics brought equivalent performance. In my solution the Euclidean distance is used as described by equation 5.6.

$$d(\mathcal{I}_i, \mathcal{I}_j) = \sqrt{\sum_{k=1}^{hxw} \sum_{l=1}^{c} \left(\mathcal{I}_j(k,l) - \mathcal{I}_i(k,l)\right)^2} \tag{5.6}$$

The distance is defined as euclidean distance of particular color components of each pixel in compared images. The $\mathcal{I}_i$ and $\mathcal{I}_j$ denote the images with the same pixel size wxh. The images use RGB color model or another three component color model. Of course both images have to use the same color model. The distance heuristic function can be explained as quantity of different pixels multiplied by size of the difference.

The original algorithm was designed to work with 360 degrees panoramatic images. Pixel shift of the image is actually a rotation for panoramatic image. Shifting image one pixel left actually means rotation of the destination image. In our case the camera is a pinhole camera. The image can not be rotated in a loop. Due to this I had to modify the distance heuristic funtion as shows equation 5.7.

$$d(\mathcal{I}_i, \mathcal{I}_j) = \frac{\sqrt{\sum_{k=1}^{h \times w} \sum_{l=1}^{c} \left(\mathcal{I}_j(k,l) - \mathcal{I}_i(k,l)\right)^2}}{w \times h} \tag{5.7}$$

The error is normalized by count of pixels that were compared. By shifting a pinhole camera image one colum of pixels is dropped. It means that the count of pixels used during comparison decrease with each iteration. Normalization makes distance values computed during image shifting comparable. Of course this solution works only for small value of the shift. With greater value the compared areas are small and the distance function is more affected by noise due to strong effect of particular pixels to overal sum. With a reasonably small shift (up to 20 percent of image horizontal resolution) the distance heuristic works well. This number was obtained experimentaly.

To find the orientaiton difference between two images we have to find column shift α with minimal distance $d(\mathcal{I}_i, \mathcal{I}_j, \alpha)$ of images $\mathcal{I}_i$ and $\mathcal{I}_j$. The distance with colum shift is defined in following way 5.8.

$$d(\mathcal{I}_i, \mathcal{I}_j, \alpha) = \frac{\sqrt{\sum_{row=1}^{h} \sum_{col=1}^{w-\alpha} \sum_{l=1}^{c} \left(\mathcal{I}_j(row \times w + col, l) - \mathcal{I}_i(row \times w + col + \alpha, l)\right)^2}}{w \times h} \tag{5.8}$$

For given colum shift the equation 5.9 has to be satisfied.

$$\forall \alpha \in \mathbb{N} : d(\mathcal{I}_i, \mathcal{I}_j, \alpha_{min}) \leq d(\mathcal{I}_i, \mathcal{I}_j, \alpha) \tag{5.9}$$

To obtain the α_{min} we have to solve problem defined by equation 5.10.

$$\alpha_{min} = \underset{\alpha \in <-r,r>}{argmin} \left(\frac{\sqrt{\sum_{row=1}^{h} \sum_{col=1}^{w-\alpha} \sum_{l=1}^{c} \left(\mathcal{I}_j(row \times w + col, l) - \mathcal{I}_i(row \times w + col + \alpha, l)\right)^2}}{w \times h} \right) \tag{5.10}$$

The colum shift of two consecutive images needs to be converted to angular difference. In case of panoramatic loop images the solution is rather straightforward 5.11.

$$\omega = \alpha_{min} \frac{360}{w} \tag{5.11}$$

For camera with limited field of view the computation is similar but instead of 360 degrees the field of view of the camera is used as shows 5.12. The image has to be rectified to eliminate error due to image deformation.

$$\omega = \alpha_{min} \frac{FOV_w}{w} \tag{5.12}$$

The entire algorithm of visual compass adapted for pinhole camera used in this work can be observed below - alg. 4. This solution of visual compass has a significant advantage in speed. Computation of SURF or even SIFT features is very expensive. On the other hand computation of euclidean distance of pixels is rather fast especially if there is only a small range of possible column shifts.

Data: $\omega_0, \mathcal{I}_i, \mathcal{I}_j, FOV$
Result: ω

$error = \infty$
$\alpha_{min} = 0$
for $(\alpha \in <-r, r>; r \leq \frac{w}{6})$ **do**
  $\epsilon = \frac{\sqrt{\sum_{row=1}^{h} \sum_{col=1}^{w-\alpha} \sum_{l=1}^{c} (\mathcal{I}_j(row \times w + col, l) - \mathcal{I}_i(row \times w + col + \alpha, l))^2}}{w \times h}$
  if $(\epsilon < error)$ **then**
    $error = \epsilon$
    $\alpha_{min} = \alpha$
  end
end
$\omega_d = \alpha_{min} \frac{FOV_w}{w}$
$\omega = \omega_0 + \omega_d$

return(ω)

Algorithm 4: Visual compass algorithm: one iteration.

Image difference trajectory for visual compass has typical shape with one global minimum as can be observed in graph 5.6. As the robot is not moving the global minimum should reach 0. It can be observed that the minimum is almost 0. The small difference can be caused by small motion of objects in the view of camera or by changing light conditions. In the graph 5.7 is visual compass image difference trajectory during fast rotation of the robot. The error minimum is much less sharp but still evident. Another reason for less sharp minimum is relative rotation about X axis between images. The example algorithm does not compansate this rotation. Due to it if the robot is moving through the rough

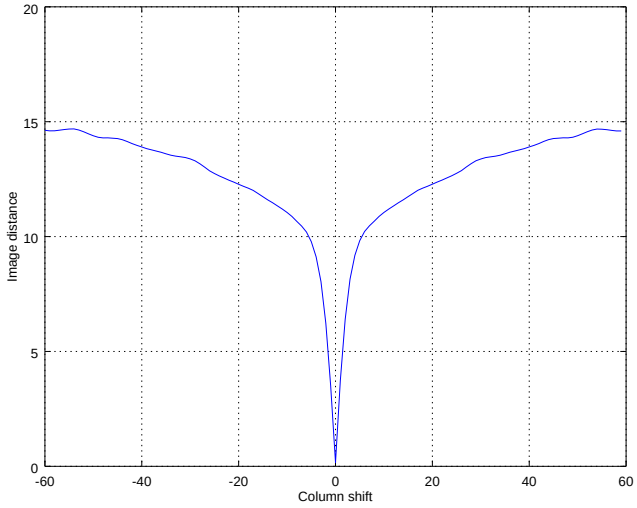

Figure 5.6: Visual compass frame difference - rover is steady.

terrain the quality of the image difference trajectory may decrease. In most cases for the robot moving on the ground the effect of rotation about X axis between two consecutive frames should be minimal.

Another aspect affecting performance of visual compass is frame skipping. According to [54] the visual compass precision is significantly affected by frame skipping. Frame skipping is a situation when instead of comparing two consecutive frames every third, fourth or maybe fifth frame is compared. With frame skip there is greater orientation error but the error is less frequent. By tuning the frame skip author of original algorithm improved precision significantly. By frameskipping we are trying to find minimum of function 5.13.

$$E_{min} = \underset{frameskip}{argmin} \left(\frac{\epsilon(\alpha \cdot frameskip)}{frameskip} \right) \tag{5.13}$$

The $\epsilon(\alpha)$ is function of rotation giving the absolute value of error for estimated rotation. When using frameskip the α would be probably different between any pair of frames. As we can not find particular α for consecutive frames we use constant estimate of α given by $\alpha = \frac{\alpha_{rot}}{frameskip}$. α_{rot} is column shift of entire rotation. Another assumption about error of rotation measurement was made: with growing colum shift α grows also the error. It is not always true as the error may slightly oscilate but in global perspective the condition 5.14 is satisfied.

$$\alpha_2 > \alpha_1 \rightarrow \epsilon(\alpha_2) >= \epsilon(\alpha_1) \tag{5.14}$$

In [54] the error of visual compass using frameskip was decreased approximately 10-times. In [54] the angular resolution of the camera was 1 deg to 1 px. In our case with camera having horizontal resolution at least 320px with FOV = 60 deg the resolution is 0,1875 deg to 1 px. Despite smaller pixel error in single frame the pixel error is accumulating with every frame. This is the reason why frame skipping is important. If we execute 20 comparisons instead of 200 in the same period of time we will get potentially 10-times smaller error.

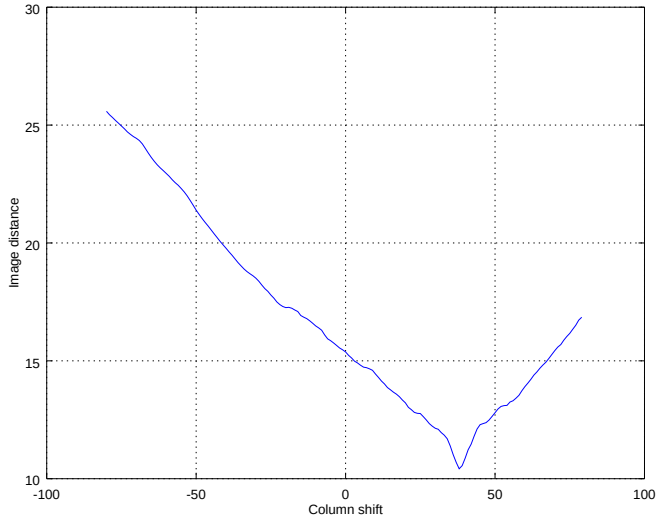

Figure 5.7: Visual compass frame difference - turning at spot.

Impact of frame skipping to total error of orientation measurement is based on the same principle as measuring with digital multimeter. The smallest resolution is given by device attributes - change by one in last digit in case of multimeter or change by 1 pixel in case of visual compass. If the measured value is displayed as 0 to 9 on last digit, the smallest change of value is 10%. If the value is displayed on last 3 digits, the smallest change of value is 0.01%. The same rules apply to pixel error during orientation measurement. If measurement is executed frequently during constant rotation with difference of 10 pixels between two consecutive frames the one pixel error will be 10% of measured value. The same situation with 10 frames skipped will result in 100 pixels difference between frames and so the error will be 1% of value. Of course in real situation the pixel error will oscilate around the real value so its impact will not be that significant but the worst case is 10-times worse than in case of frame skipping. In equations 5.15 and 5.16 the pixel error is formalized in relative and in absolute form respectively. D_{px} is a distance between two frames in pixels. FOV_h is horizontal field of view of used camera and RES_h is horizontal resolution of the camera.

$$E = \frac{1}{D_{px}} \tag{5.15}$$

$$E = \frac{1}{D_{px}} \frac{FOV_h}{RES_h} \tag{5.16}$$

Impact of various frameskip on precision of rotation measerement with constant angular speed can be observed in figure 6.6.

Desired effect of higher frame skipping is obvious but frame skipping has several limitations. First of them is given by using camera with limited field of view instead of panoramatic camera: With too many frames skipped the two images may have too small overlap or they may not overlap at all. In this case the error will grow significantly due to high matching error. This situation can be observed in figure 6.6 for frame skip 180 where negative error constantly grows as matching of consecutive frames looses resolution due to small overlap of frames. And for frame skip 200 the frames don't overlap enough to find

the real smallest error. The frame matching hits the limit of field of view and finds the smallest error in this limit but real smallest error lies beyond this limit.

Another drawback of frame skipping is smaller frequency of providing orientaiton feedback. If the camera FPS (frames per second) is equal to 30 than in case of very slow rotation the best frame skip could be about 100 which means that the orientation feedback will be provided every 3 second approximately. This could limit usage of visual compass for some applications. To overcome problem with low sampling rate new approach of conditional base image update was developed. Base image is the image that is used as a reference for comparing new images and computing orientation difference. This approach generates position feedback with every frame but it updates the base image only when angular difference in pixels is large enough. This way it reaches the precision close to optimal frame skip. Until the required difference is not reached the base image is not updated. Of course measurements right after the base image update have usually worse precision due to pixel error and small angular difference but later measurements use the same base image and so the precision of later measurements is not affected. Effect of automatic frame skipping can be observer in figures 6.8 and 6.9. Entire iterative algorithm of continuous orientation measurement is defined as algorithm 5.

Prior solution improves precision during rotary motion but real robots travel forward or backward most of the time. Precision of visual compass is very important also during linear motion. If the robot moves straight forward and orientation of robot does not change the visual compass should show the same value until robot changes its direction. This is not as trivial as it seems to be at the first glance. Of course we can just compare new image with base image and update base image from time to time but precision is usually not very good. As the robot moves forward objects in camera view become larger and further from each other due to perspective. This significantly affects precision of measurement because the images can not be matched one-to-one because of that „zoom“ effect. To decrease negative impact this problem one of the two images in comparison has to be scaled. To avoid increase of pixel error the base image is upscaled instead of downscaling the new image so the comparison runs with the same amount of pixels like without scaling.

Problem with scaling the image is that it brings another degree of freedom to pose estimation from image data. Estimating this another variables means multiplying amount of existing algorithm steps by amount of steps needed to estimate the scale. Fortunately this can be solved by hint from odometry. Despite the odometry information obtained from differential drive in outdoor environment is usually not very precise in rotation about axis z linear precision of odometry is usually relatively good. It is possible to use this hint from odometry to compute scale of base image without time consuming error minimizing matching of size in every iteration.

The principle of scaling is depicted in figure 5.8. Due to linear motion the image plane is virtually moved closer to camera. This new position of image plane is denoted as Virtual image plane in fig. 5.8. The shift of image plane caused by linear motion is L. In horizontal direction the image plane overlaps the field of view of camera by $diff$ on each side. The angle oposite the $diff$ is equal to half of field of view angle. Size of the $diff$ can be derrived from definition of tangens function as shows equation 5.17.

$$diff = L \cdot tan\left(\frac{FOV}{2}\right) \tag{5.17}$$

Scaling factor is computed according to equation 5.18. Both vertical and horizontal dimensions of the image are being scaled using the same scaling factor to avoid image

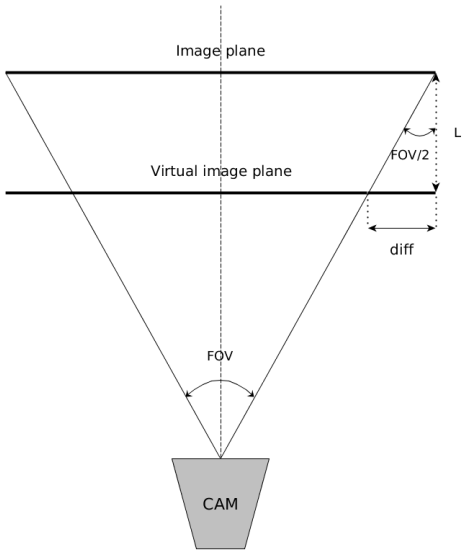

Figure 5.8: Visual compass: Principle of scaling during linear motion.

deformation. Finally the image is cropped to fit the previous dimensions in pixels. Newly created image is used as a new base image for comparison. In algorithm 5 this step is covered by *normalize* function. Effect of base image scaling according to odometry feedback is demonstrated in plot 6.10.

$$scale = \frac{RES_h + diff}{RES_h} \tag{5.18}$$

Of course the scaling is just a crude estimation how the view of the camera could look like if the robot moved forward. The main problem with scaling is that the scaling coeficient depends on distance from the camera so scaling the image will scale corectly only subset of objects equally distant from the camera. When no scaling is applied only the infinitely distant objects have the proper scale. Scaling described by equations 5.17 and 5.18 does not scale real objects but only their projection to one of set of parallel planes perpendicular to the camera axis. Its dimensions are given by camera resolution. The scaling factor trajectory is depicted in 5.9.

5.4 Solution of long-range localization

My approach is based on feature-based FastSLAM algorithm. Important part of the solution is landmark detector that is responsible for obtaining observations. In my solution the observation is obtained in two phases. In the first phase interesting objects are detected in surrounding of the robot. In the second phase the rangefinder is used for obtaining distance measurement in particular direction. Object detection and distance measurement are independent tasks that can be solved using several approaches but the data from both are merged into observation in spherical coordinates (rotation, elevation, distance). Principle of this long-range localization approach is visualised in figure 5.10.

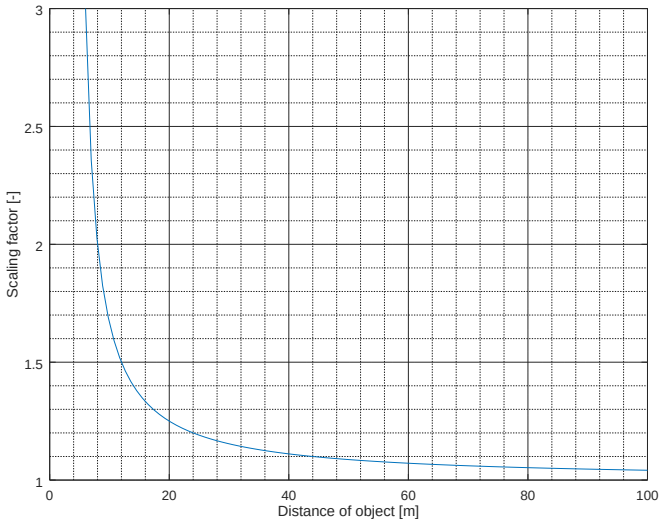

Figure 5.9: Visual compass: Trajectory of scaling coeficient for 4m traveled forward distance.

Data: $\omega_0, \mathcal{I}_{initial}, \mathcal{I}_j, FOV$
Result: ω

$\mathcal{I}_{base} = \mathcal{I}_{initial}$
$\omega_{base} = \omega_0$
while $True$ **do**
  $\mathcal{I}_{norm} = normalize(\mathcal{I}_{base})$
  $error = \infty$
  $\alpha_{min} = 0$
  for $(\alpha \in < -r, r >; r \leq \frac{w}{6})$ **do**
    $\epsilon = \frac{\sqrt{\sum_{row=1}^{h} \sum_{col=1}^{w-\alpha} \sum_{l=1}^{c} (\mathcal{I}_j(row \times w + col, l) - \mathcal{I}_{norm}(row \times w + col + \alpha, l))^2}}{w \times h}$
    if $(\epsilon < error)$ **then**
      $error = \epsilon$
      $\alpha_{min} = \alpha$
    end
  end
  $\omega_d = \alpha_{min} \frac{FOV_w}{w}$
  $\omega = \omega_{base} + \omega_d$
  $skip = \alpha_{min} - \frac{w}{/} 2$
  if $|skip| > skip_{min}$ **then**
    $\mathcal{I}_{base} = \mathcal{I}_j$
    $\omega_{base} = \omega$
  end
  $send\_update(\omega)$
end

Algorithm 5: Visual compass - complete algorithm.

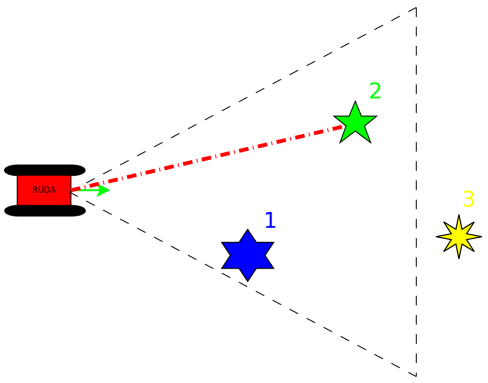

Figure 5.10: Principle of long-range localization.

Figure 5.11: Example of environment considered for long-range localization.

5.5 Considered environments for localization

As mentioned before there are some assumptions about environment where the long-range localization works. It is designed for environments with only few detectable objects. Objects have to be detectable by both camera and laser rangefinder. These requirements satisfy solid objects with significant volume and texture distinquishable from surrounding. In the environment there can be only a few objects. The localization approach could in theory also work in environments with high object density but performance would be probably worse compared to existing approaches like ICP library for 3D laser SLAM([71]) or RDBG SLAM ([41]).

5.6 Sensoric system for localization

For purpose of localization data from several sensors are fused. Each sensor input has specific attributes and particular role in entire system. The data flow is depicted in figure 5.12. This figure shows sensor inputs (green), odometry subsystem (orange), landmark detecion system (blue) and finally the SLAM algorithm that estimates robot's location (red).

Landmark detector and visual compass use both video input. Visual compass uses fixed camera installed directly on robot's hull. It serves as extension to odometry. Visual compass is fused with odometry information to decrease odometry error during rotation. Eliminating the rotation error is essential for precision of odometry as can be observerd in figures 5.4 and 5.5. The fusion is done using extended Kalmann filter. In the fusion the odometry information serves as initial estimate. The visual compass information is

used as update. The fact that visual compass does not provide information about traveled distance is modelled by very large variance in all other coordinates but rotation about Z axis as depicted in 5.20. The entire fusion algorithm is described by 5.19. The algorithm is adapted from solution proposed in [77]. The adaptation consists of different observation model and different observation covariance matrix that both correspond with visual compass capabilities.

As a prediction new predicted pose given by odometry difference is computed. New predicted covariance matrix $\bar{\Sigma}_t$ is computed from previous covariance matrix and motion noise. The F_t is a motion model jacobian. In correction step the Kalman gain K_t is computed as a weight between prediction (odometry) and correction (visual compass). Using the Kalman gain the new pose estimate and new covariance matrix are computed as a weighted fusion of odometry estimate and measured orientation.

$$\begin{array}{lrcl} \textit{Prediction step} & \bar{\mu}_t & = & f(\mu_{t-1}, u_t) \\ & \bar{\Sigma}_t & = & F_t \Sigma_{t-1} F_t^T + Q_t \end{array} \quad \begin{array}{lrcl} \textit{Correction step} & S_k & = & H_k \bar{\Sigma}_t H_k^T + R_t \\ & K_t & = & \bar{\Sigma}_t H_k^T S_k^{-1} \\ & \mu_t & = & \bar{\mu}_t + K_t(z_k - h(\bar{\mu}_t)) \\ & \Sigma_t & = & (I - K_t H_t)\bar{\Sigma}_t \end{array} \tag{5.19}$$

$$\Sigma_{visual} = \begin{bmatrix} \infty & 0 & 0 & 0 & 0 & 0 \\ 0 & \infty & 0 & 0 & 0 & 0 \\ 0 & 0 & \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & \infty & 0 & 0 \\ 0 & 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\theta \end{bmatrix} \tag{5.20}$$

Once the odometry is obtained and fused with visual compass orientation measurement the improved odometry information is obtained. This information is used as a prediction for particle pose estimates in particle filter algorithm. From image of PT camera connected with laser rangefinder the landmarks are detected. First the camera image is rectified to minimize landmark pose estimation errors due to image deformation. After landmarks are detected their poses are estimated and laser rangefinder is aimed at them to obtain more precise distance information. As the camera image does not provide information about distance of observed objects the distance of particular landmark is taken from laser measurement. The landmark pose is represented in spherical coordinates. The angular coordinates reflect position of camera manipulator. The distance coordinate is taken from laser measurements.

Landmark detection and localization

The „Landmark detection and localization“ depicted in the diagram 5.12 is visualised more in the detail by figure 5.13. The detector may consist of several parallel pipelines. Each of them for detecting landmarks of different kind. At the moment landmarks of only one type are used for localization.

We need to find landmarks that can be recognized from all directions and targeted by laser rangefinder. It basically means that we need to find objects that are rised above terrain. From camera image it is difficult to recognize such objects from a distance. We need depth information to distinguish raised objects from background or experience that will tell us that the particular part of image is interesting object. A useful depth information is difficult

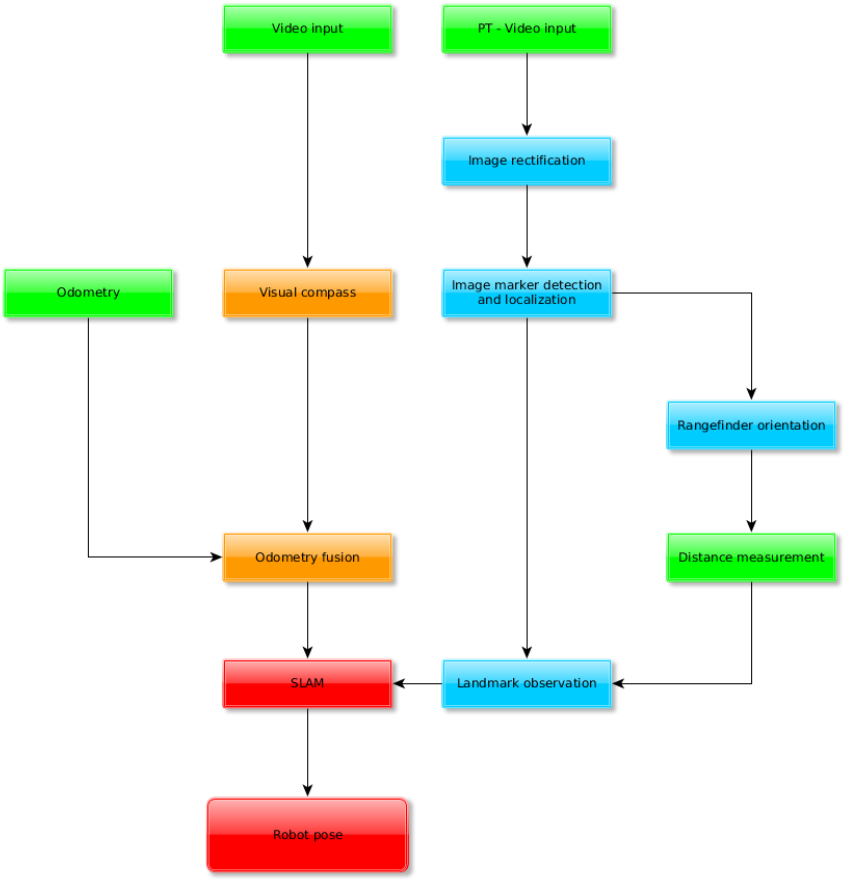

Figure 5.12: Data flow in sensoric system.

to obtain. We need to measure distance to object and to its surrounding to be able to distinguish the object from background. Approaches based on lidar or stereocamera work for rather a small distance only - usually tens of meters. For large distance a stereocamera with very high resolution has to be used. With increasing camera resolution the computation demands grow. Increasing baseline of stereocamera increase size of entire sensor system. The dependency of stereocamera performace is described in [7]. The equation 5.21 describes stereocamera resolution as function of camera parameters.

In equation 5.21 the Δz denotes depth resolution, z denotes distance of object from camera baseline, f is the focal lenght, b is the baseline and Δd denotes disparity in meters. The equation can be transformed into form 5.22 that defines dependency of disparity resolution according to object distance, depth resolution and stereocamera parameters.

$$\Delta z = \frac{z^2}{fb}\Delta d \tag{5.21}$$

As example we can describe a hypothetic stereocamera with 1m baseline and $12.5mm$ focal lenght observing object at $100m$ distance with minimal resolution $1m$. Under such circumstances the disparity resolution would be $\Delta d = 0,00000125m = 1,25\mu m$. For sensing chip format 1/3 inch the dimension of the chip is 4.8mm x 4.8mm. For disparity resolution $1,25\mu m$ the pixel resolution would be 3840 x 3840 px. For distance of $200m$ the required resolution is 15360 x 15360 px. Disparity of images with such a high resolution is difficult to process. The vertical resolution can be lower to decrease amount of data coming from the camera but required resolution of the camera grows exponencially with the distance anyway. No need to say that cameras with such a high resolution are not commonly available. These examples show that the stereocamera usage for purpose of object detection and distance measurement is limited.

$$\Delta d = \frac{fb}{z^2}\Delta z \tag{5.22}$$

For 3D lidar the limiting factor is measurement density. With small density of the measurements it will miss some distant objects. With high density of distance measurements the amount of data generated by sensor is very large. Another problem is that the sensors with high range and high measurement density are very expensive - especially if they are designed to work in outdoor environments.

Image processing pipeline

In image processing pipeline used to detect landmarks the first step is convertion to grayscale. This step has practical reasons. The grayscale image is simplified representation compared to RGB or RGBA representation. This simplifies consecutive steps and manipulation with the image in general. Also some algorithm implementations used in this work require grayscale input. Convertion to grayscale is done using luminosity method. Compared to lighness and average convertion method as described in [37] the luminosity method is closest to human perception and so it provides the most naturally looking results. Pixel of result image is given by equation 5.23.

$$P(x,y) = 0.21R(x,y) + 0.72G(x,y) + 0.07B(x,y) \tag{5.23}$$

Another step is gaussian blur. Convolution with small gaussain kernel filters out high frequency noise. The convolution has a similar effect as filtering by integration filter. Elim-

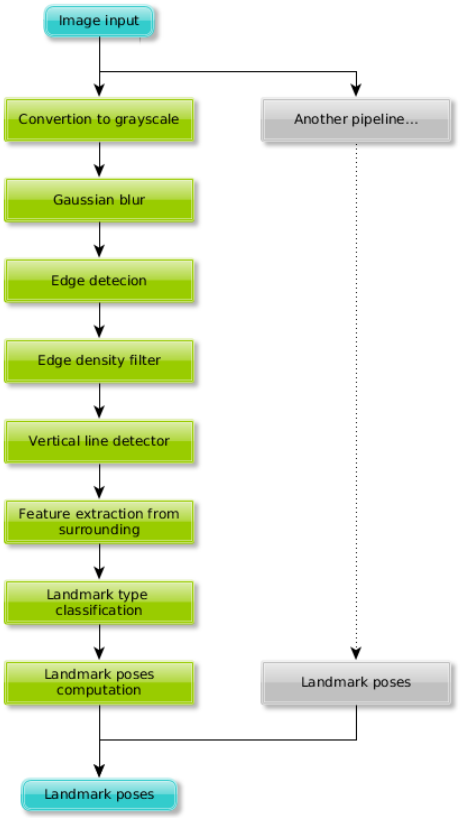

Figure 5.13: Landmark detection pipeline.

ination of high-frequency noise is essential for edge detectors based on image derivation. The noise itself is caused by random peeks in intensity. Those peeks would be highlighted by image derivation and they would appear in detected edges. Gaussian kernel in discrete variant is represented by NxN matrix. Elements of matrix are given by equation 5.24 taken from [6].

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{5.24}$$

Size of gaussian kernel and variance affects intensity of filtering. A larger kernel with greater variance σ filters the noise more effectively but may also blur edges so much that edge detector will not be able to detect them. Small kernel with low variance will not be able to filter out all the noise. Finding optimal size of kernel and optimal variance was matter of experiments. In this case the best results were achieved with kernel of size 3x3.

Edge detection

Consecutive step is edge detection. In image the edge is a place with steep change of intensity. Considering this fact the detection is based on recognition of local extremes with image derivation. The derivation is computed using approximation by convolution with NxN kernel as described in [49]. Two approaches were consiederd - Sobel and Laplacian. These two approaches differ in the way the derivation is computed. The Sobel algorithm uses two 3x3 kernels to compute first order derivatives separately - one in horizontal direction and the other in vertical. Then the results are merged. Sobel 3x3 kernels are defined in equations 5.25 and 5.26 - the first one is horizontal and the second one is vertical.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{5.25}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{5.26}$$

The Laplacian approach uses only one kernel to compute both horizontal and vertical second order derivatives in one pass. The Laplacian kernel is defined in equation 5.27. Alternatively the Laplacian kernel can be extended to compute also diagonal derivatives. This extended kernel is defined in equation 5.28.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.27}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.28}$$

Effect of convolution using Sobel horizontal and vertical kernels and basic Laplacian kernel is depicted in figure 5.14. From the figure we can observe that edges detected by laplacian have higher intensity - they can be more reliably distinguished from background. In both cases the edges are more than 1px wide. This is caused by gaussian blur that was used for filtering the image. To obtain precise location of landmarks the edges need to be

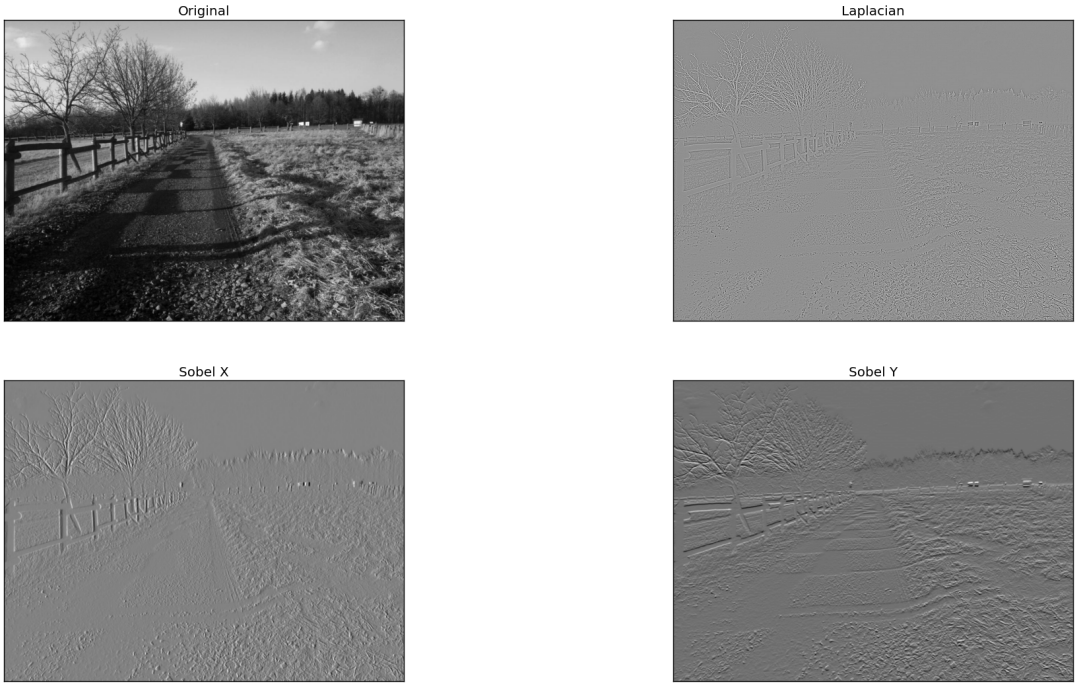

Figure 5.14: Image derivation comparison - Laplacian and Sobel.

„sharpened“ to 1px width by postprocessing. Similar results were achieved in more detailed edge detector comparison described in [13].

Common approach to sharpen the edges is application of non-maximum supression algorithm. This algorithm finds direction of edge intensity magnitude gradient and in this direction keeps only the local maximum. Other pixels are dropped (zeroed). This way every edge is shrinked to 1px width. The best estimation of real edge position is guaranteed by choosing the local maximum. The algorithm is described more in the detail in [66].

Entire process of bluring the image, detecting edges, sharpening edges and filtering edges by threshold is used in Canny edge detector algorithm. Internally the Canny edge detector uses Sobel kernels to compute image derivatives and adds preprocessign by gaussian blur and postprocessing by non-maximum supression and thresholding. The Canny edge detector uses double thresholding. There are two thresholds - high and low. If the edge magnitude in given pixel is higher than high threshold, the edge pixel is marked as strong. If the edge pixel magnitude is lower than low threshold it is dropped (zeroed). If the edge pixel magnitude lies between two thresholds it is marked as weak.

The threslholding step is followed by another step called edge tracking. This step decides which edge pixels will be kept and which will be dropped. The pixels with intesinty overcomming the high threshold that are marked as strong will be always kept. These pixels will became a seeds for hysteresis filtering. Pixels with intesity between the two thresholds will be kept only if they can be connected with strong pixels or with weak pixels already connected to cluster initiated from strong piuxels. The effect of the algorithm is visualised in figure 5.15. Pixels A are strong pixels and they will be always kept. Pixels B are weak pixels that can be connected with cluster initiated by strong pixels so they will be kept too. Pixels C are weak pixels that could not be connected to any cluster with strong pixels so

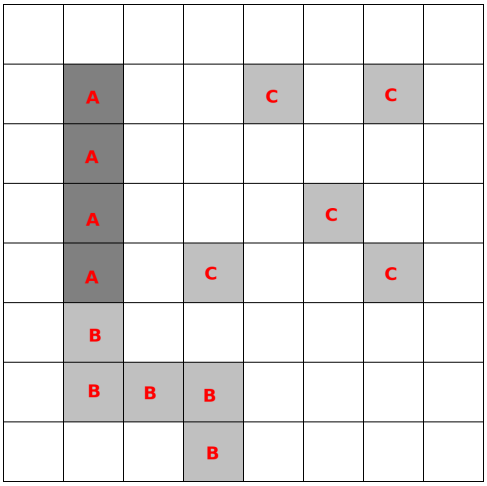

Figure 5.15: Double thresholding and hysteresis filtering.

they will be dropped. Application of Canny edge algorithm on reference image is depicted in figure 5.16. The hysteresis thresholds depend on image representation. In this case the edge image is represented by 8bits per pixel so the hysteresis has to be chosen from interval $< 0, 255 >$. Hysteresis thresholds are 100 (low) and 200 (high) for this particular image.

We can see that the performance of canny edge detector is relatively good - important edges were detected. Unfortunately in natural environment there are many edges that are useless as landmarks because they belong to too random or volatile structures like treetops or shadows on the ground. Despite we will filter vertical edges only there will be too many of those that can't be used as landmarks. Detecting landmark in treetop would result in very unstable landmark that would be probably difficult to observe from another direction. Landmarks like these would bring mostly just noise to localization process so they need to be filtered out.

Edge image postprocessing

One of approaches is based on presumption that the usefull edge (on tree trunk or on building) will be surrounded by relatively homogenous area at least from one side. It means that there will be minimum of edges in this area compared to edges that are on thin branches in tree top which are probably surrounded by random edges on leaves from both sides. Another problem is that fake vertical edge may appear as assembly of random edges in grass, leaves or another area with many edges. To filter out invalid or useless vertical lines the edge density filter was applied. Exact definition of the edge density is defined in [68]. Basically the edge density can be defined as amount of pixels belonging to edges compared to all pixels of examined area. This amount can be expressed relatively in percent to make it independent to area size. The effect of edge density filtering can be observed in figures 5.17 and 5.18.

In 5.17 there is the edge image as pure output of Canny edge detector without any posprocessing. We can see that the image contains many edges detected on tree tops and grass that belong to objects that are difficult to distinguish and re-observe. Landmarks detected on these objects would bring noise rather than useful input for localization and so they are undesired. Figure 5.18 shows the same image after edge density filtering. Despite it is not perfect we can observe that there are much less vertical edges detected on ureliable

objects. The drawback of edge density filtering is that some of the useful edges were filtered out because they were surrounded by noise and so they overcame filtering threshold.

Another problem in image processing is the vertical lines detection. The most straightforward approach is to use Hough transform to detect lines in image. Then the lines can be filtered according to its direction leaving only those that are vertical. This approach is very reliable but computation of Hough transform is rather demanding in terms of computation time. The algorithm of Hough transform is described in [44]. The core idea of Hough transform is convertion from Euclidean space to Hough space. In Hough space every single point denotes infinite line in Hesse normal form. The coordinates in Hough space correspond to direction θ and parameter r in representaiton of line defined in 5.29.

$$r = xcos(\theta) + ysin(\theta) \tag{5.29}$$

Single point in cartesian space is defined by set of lines going through this point in Hough space. In edge image the only point we consider are point belonging to edges (white in edge images). The background points are not interesting. The set of lines going through a particular point draw a curve in Hough space. For two points there are two curves that intersect. The intersection point denotes a line that goes through both of the points. For line detection we simply draw curve in Hough space for every edge point in the edge image and we accumulate number of curves in intersection point. If number of curves in intersection overcomes defined threshold the line is detected. The lines in Euclidean space appear as peeks in Hough space. The problem of Hough transform is its computational complexity. For every edge point in Euclidean space we have to draw all the possible lines going through in Hough space. For resolution of 1 deg in direction parameter *theta* we have to draw 180 lines for every point in Euclidean space. Moreover Hough transform discovers unbounded lines but we need to find particularly bounded lines. Typical approach is to project the infinite line into Euclidean space and then select only those pixels that lie on the line. This posprocessing step increase demands of entire approach.

The complexity of Hough transform is the reason for optimisation. Many optimisations of Hough transform algorithm work with random sampling of points in Euclidean space as described in [17]. By reducing amount of source points the amount of generated line representations reduce significantly. The only condition to make stochastic algorithms work is to take representative sample of source points. Good results were achieved with more than 2% of initial population of edge points as described in [17].

Another approach considered for finding vertical edges was filtering using mask. The mask is applied to image pixel by pixel. If edge shape around center of mask match the pattern they are copied to destination image. This way only the edges that fit the mask are copied to destination image. For vertical edges the mask has a column shape of dimensions 1xN. The column is vertical. This mask is put to the source image pixel by pixel. If the mask is filled by surrounding pixel from more than 80% by edge pixels the edge pixels are copied to destination image. Then N was set by experiment. Reasonable value of N is about 10. With greater size many edges that are not perfectly vertical are dropped. Edges close to vertical direction are due to discrete nature of image representation practically composed of adjoining vertical segments. With reasonable value of N the segments are accepted separately recreating the vertical line in destination image. With too small value of N the noise is copied to the destination image together with vertical edges. Disadvantages of this approach are evident: It is less flexible and less robust than Hough transform. Significant advantage of this approach is performance. It does not have to generate all

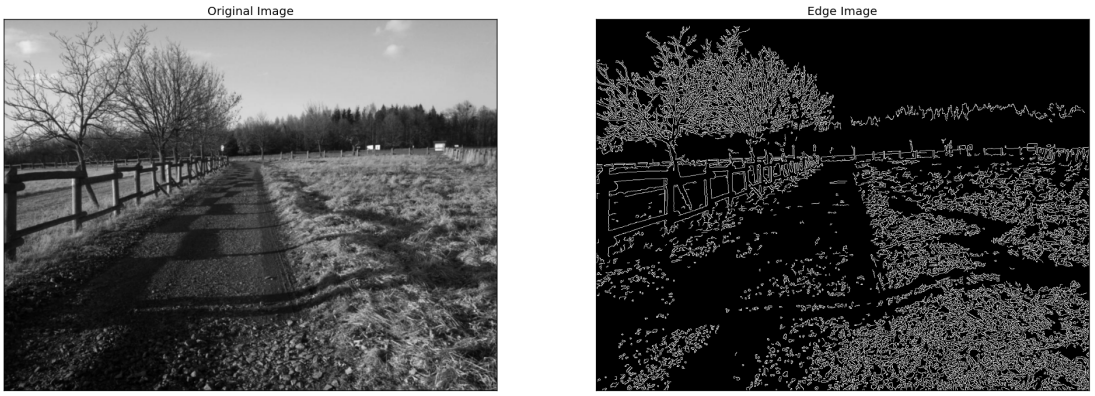

Figure 5.16: Canny edge detector algorithm applied on reference image.

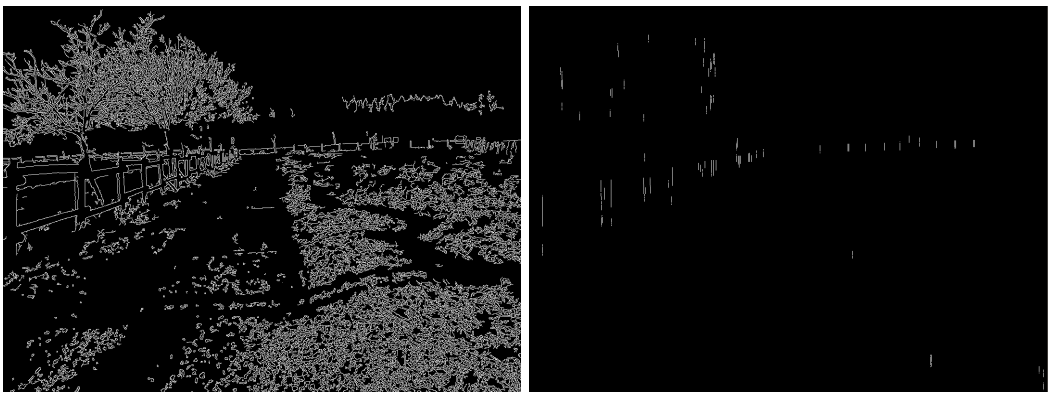

Figure 5.17: Vertical edges on unfiltered edge image.

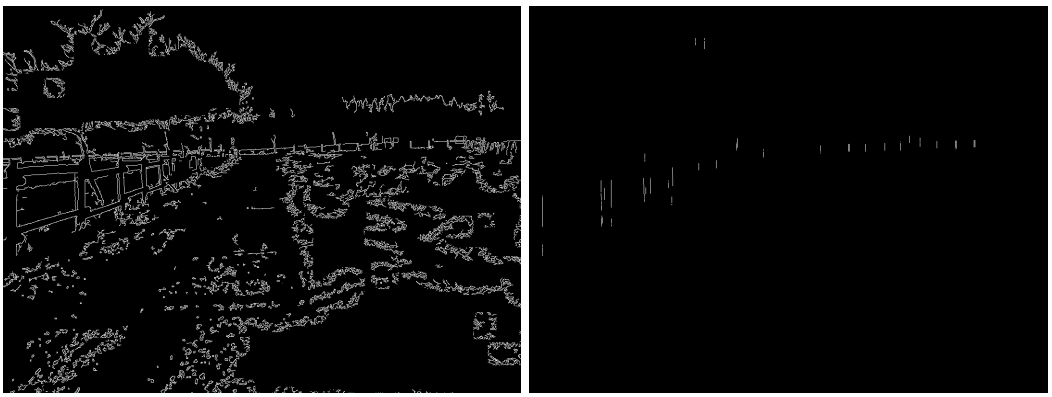

Figure 5.18: Vertical edges on edge density filtered edge image.

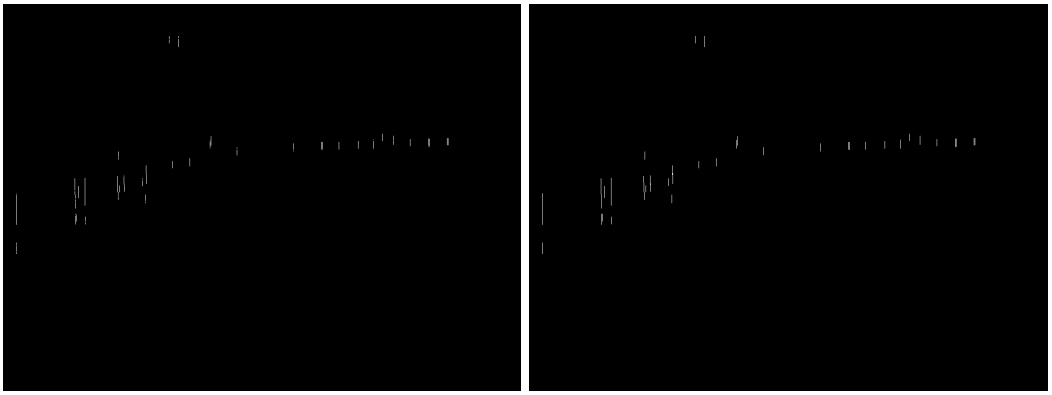

Figure 5.19: Effect of edge completition.

possible parameters of lines going through given point. Moreover result of this algorithm is a set of already limited lines that don't need to be extracted pixel by pixel.

Laser rangefinder orientation

Another non-trivial problem is orientation of manipulator with camera and laser rangefinder. Basic principle is that manipulator orients the laser rangefinder to obtain distance measurement for landmarks. The problem is obvious: laser rangefinder can take measurement at one point only and manipulator speed and acceleration is limiting number of measurements that can be taken during a period of time. Efficiency of manipulator motion can significantly increase amount of measurements taken for a fixed period of time. Still it is usually not possible to observe all detected landmarks before robot chages its position and landmarks need to be re-observed. Due to time constraints it is also important to choose landmarks that are most useful for robot localization. For best efficiency it is necessary to consider both - the efficiency of manipulator motion together with landmark quality.

The manipulator can be controlled by set of rules but the rules can not usually be applied immediately. Applying the rule takes time - usually the time between making the decision and before the manipulator reaches desired pose. To remember the decision made in the past is is necessary to keep a state of control. During decision making process in limited state space the state can be kept using final state machine (FSM) as described in [3]. The final state machine for decision making is widely used approach for this kind of tasks - it became a standard for ROS[72].

If the decision making process is represented by final state machine (FSM) each decision is represented by particular state of the FSM. The final state machine can remember finite number of prior states. In case of decision making the FSM allows to remeber sequence of prior decisions made. The decision making can be more than just reactive - new decisions may be affected by prior decisions like in case of human decision making process.

The figure 5.20 depicts the decision making FSM that is used to control motion of the manipulator. Each state depicted in the diagram represent a particular distinguishable task the robot executes. The green start state denotes entrypoint of the state machine. The „Choose target“ state represents finding of new goal for manipulator. In this state the best targed is choosed from set of observed landmarks. Once the target is choosed the state machine transits to „Travel to target“ state.

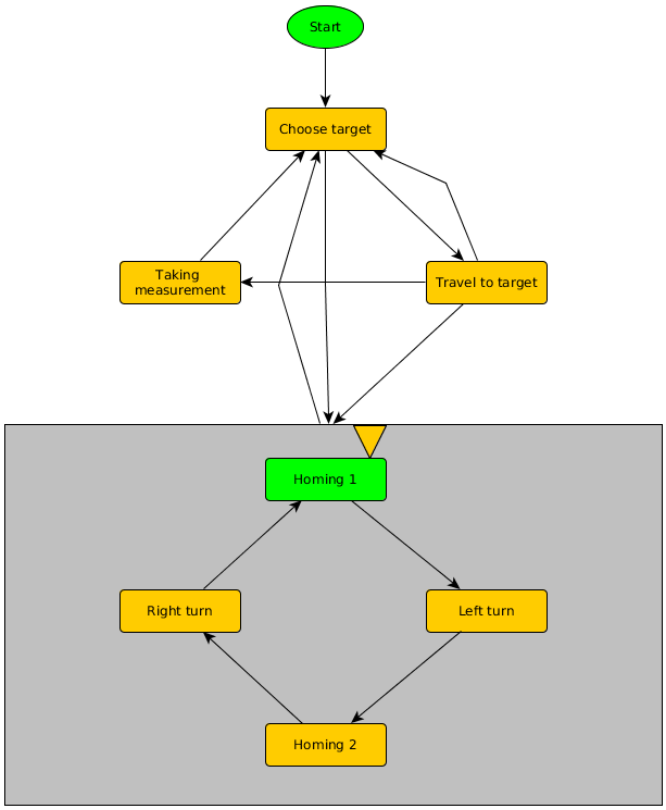

Figure 5.20: Decision making final state machine - manipulator control.

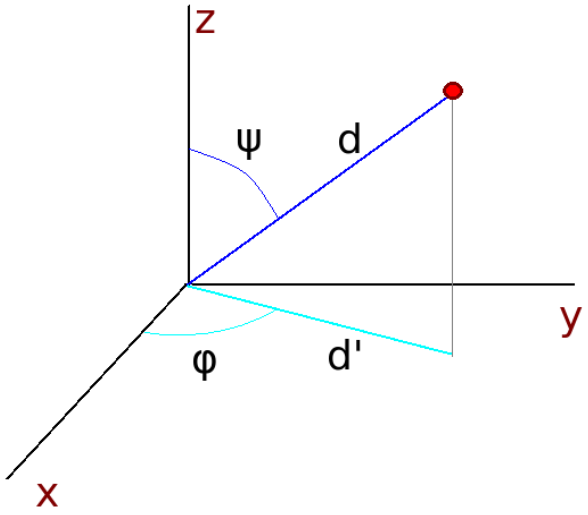

Figure 5.21: Polar coordinate system.

The „Travel to target“ state means that the manipulator travels to desired pose. Once the pose is reached within a defined tollerance the state machine transits into „Taking measurement“ state that controls process of taking distance measurement of given marker in the image. The „Travel to target“ state may also transit back into „Choose target“ if the target landmark disappears.

„Taking measurement“ is responsible for taking measurements. In this state several distance measurements are made with laser rangefinder pointing at the landmark. When the measurements are made or when counter of attempts reaches defined limit the FSM transits into „Choose target“ state and thus creates infinite loop of manipulator control.

To simplify the diagram it contains one superstate depicted as gray box with states inside. This superstate represents active searching for landmarks in situations when no landmarks can be observed by camera. This state can be accessed from „Choose target“ and „Travel to target“ states. Simply said: if there is no marker detected in the camera view in case of choosing new target for the manipulator or in case of traveling to given target the search for landmarks is started. During the search process the manipulator is first homed into initial position. Then it is turnet to the left by 180 degrees. After this turn it is homed back again. Consecutive action is to turning manipulator right by 180 degrees. This way the manipulator keeps turning left and right from its initial pose and searches for image markers that can be potential landmarks. If any marker is detected the control immediately skips into „Choose target“ state disregarding the actual state of searching process.

This global control keeps manipulator targetting the landmarks continuosly during robot's operation. As searching procedure is quite straightforward the other states will be discussed more into detail. Especially choosing best candidate for distance measurement and taking the distance measurement procedures are a bit more sophisticated.

Choosing the best candidate for distance measurement is the procedure that has the strongest impact on performance of entire solution. To make it possible to navigate the manipulator it is necessary to convert pixel coordinates in camera image to absolute spherical coordinates in which the manipulator operates. The spherical coordinate system is visualised in figure 5.21.

The convertion of position of particular point in camera image is defined by equation 5.30 below. The equation describes convertions of coordinates $< x, y >$ of point P into spherical coordinates $< \phi, \psi, d >$. In the equation 5.30 are as first computed vertical and horizontal differencies from center of camera image - $diff_V$ and $diff_H$ respective. The differencies are converted into radians using known camera field of view FOV and camera resolution. Finally the orientation of camera base is added. The final spherical coordinates are absolute - they describe a particular point in surroundings of the robot. Of course as the camera works in two dimensions only the depth information P_d is undefined.

$$\begin{aligned} diff_V &= P_y - RES_V/2 \\ diff_H &= -(P_x - RES_H/2) \\ \\ P_\phi &= Base_\phi + \frac{diff_V \cdot FOV_V}{RES_V} \\ P_\psi &= Base_\psi + \frac{diff_H \cdot FOV_H}{RES_H} \\ P_d &= undefined \end{aligned} \tag{5.30}$$

The strategy of selecting markers which distance should be measured is essential for performance of the solution as was mentioned before. Two strategies were considered for that: A reactive strategy and quality based strategy.

The reactive strategy is based on what can be observed right now. In reactive strategy the „Choose target“ procedure chooses the marker that is closest to actual manipulator pose without remembering history of landmark observations. The basic idea is to react reactively on actual observations without sophisticated processing that would require some history to build up reputation for particular markers. This way it should avoid delay caused by „Choose target“ procedure for a cost of possible low quality of landmark observations. Of course it is not possible to avoid remembering some kind of history at all even with this approach. The image marker that was visited in previous iteration is the closest one for actual iteration so without history of visited markers the one marker would be re-visited again and again. To prevent this undesired behaviour a history of N last visited markers is remembered. Entire process is visualised in figure 5.22.

The most sophisticated part of reactive marker selection strategy is recognition of already visited markers as the markers may slightly move between frames due to camera noise, changing light conditions, micro motion of objects (moving branches and leaves of trees) and also due to motion of the robot itself. To deal with this uncertainity the marker recognition has to work with tollerances. The tollerances may cause two markers that are very close to each other to be recognized as one - especially if they have similar fatures. On the other hand in this situation it is typical that these markers belong to the same object or similar object close to prior one. Still such improper recognition would negatively affect performance of localization.

To recognize the same marker the most important hint is its polar pose. If there are markers observed in two consecutive frames at the very similar pose there is a high probability that these are observations of the same marker. Another hint is a set of features describing the marker. If the marker looks similar from the point of view of features it is probably the same marker. The marker recognition uses the described principles simultaneously - if the marker pose between two frames does not overcome given tollerance and if marker features differ less than defined tollerance the marker is recognized as the same. Formal description follows in equation 5.31.

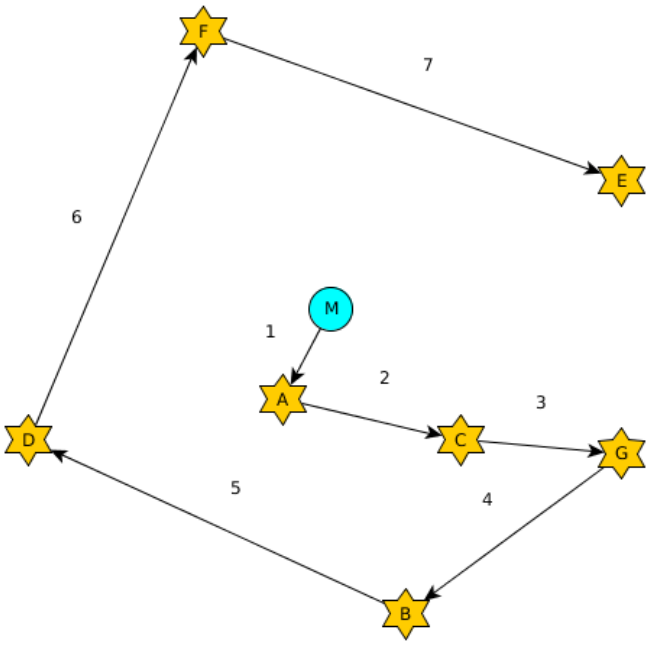

Figure 5.22: Reactive marker selection strategy.

$$abs\left(\begin{bmatrix} P_1 \\ F_1 \end{bmatrix} - \begin{bmatrix} P_2 \\ F_2 \end{bmatrix} \right) < \begin{bmatrix} E_p \\ E_F \end{bmatrix} \tag{5.31}$$

In equation 5.31 the P_1 and P_2 denote pose vectors of first and second marker in polar coordinates. The F_1 and F_2 denote feature vectors of first and second marker. The E_p denote vector of maximal errors in pose coordinates to allow the marker to be recognized as the same. The E_F is the vector of tollerances for marker features. If any value overcome the tollerances the markers are recognized as different.

Visited markers are organized in a FIFO buffer with fixed size. The size of buffer was chosen experimentaly. After several visits the markers visited at the beginning are forgotten so they can revisited. This way the reactive manipulator control algorithm can iterate between several markers in pseudorandom order. The order is affected by marker distance to other markers. Disadvantage of FIFO buffer for keeping the track of visited landmarks has impact on effectivity of „Choosing target“ phase of control. All the landmarks has to be compared with actual closest observation to decide if the actualy observed marker was already visited or not. Computation time grows linearly with FIFO buffer size.

The manipulator control strategy based on quality of marker is in principal similar to reactive one but uses a different metrics. Instead of choosing the marker according to distance to actual manipulator pose it uses a quality of landmark. The quality may have varios definitions. In this case the quality was defined as stability of the landmark in time. As can be observed in images in chapter Image processing pipeline despite various filter applied during image processing the markers can be detected on a very unreliable places like thin branches or in a grass. According to human experience we can intuitively distinguish between reliable and unreliable markers but it is not trivial to describe such experience by algorithm. As the stability in time is difficult to estime at a first glance it can be measured

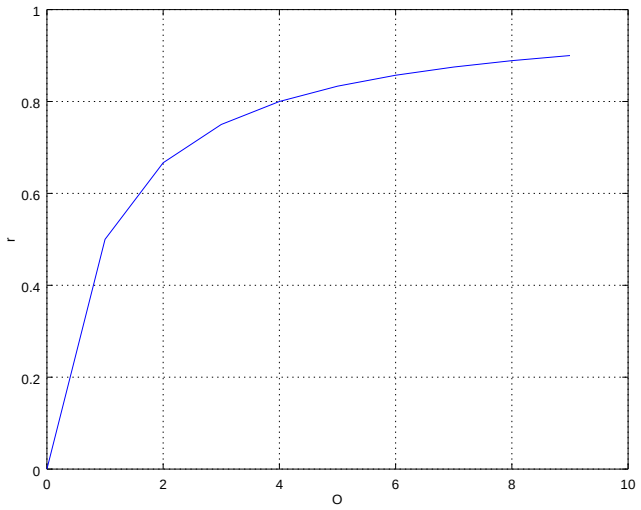

Figure 5.23: Marker reliability trajectory.

in a sequence of camera images. The particular marker can be observed in consecutive frames. If it can be observed in most of frames it can be considered as reliable. On the other hand if the marker disappear in significant part of frames in sequence - it is „blinking“ - it will be considered as unreliable.

To measure the time reliabilty of marker it is necessary to keep track of marker observations in time. Keeping historical data is generally computationally demanding. In this case the marker history was reduced to last marker pose, its features and some kind of reputation for each observed marker. The marker pose and fetures are necessary for recognition of the marker between frames. As the robot moves and the surrounding environment changes slightly the remembered marker is updated every time it is recognized by actual data. This way it is possible to track a marker that drifts away slowly or marker that changes slightly due to different observation angle.

The reputation of the marker is given as number of observations of this particular marker related to age of the marker expressed in count of frames since its first observation as described in 5.32.

$$r = \frac{O - 1}{F_{CNT}} \tag{5.32}$$

The problem with this approach is forgetting of markers that were being observed for a long period of time so their reputation r will not be significantly affected by several missing observations. To avoid catching already unavailable markers the manipulator control algorithm considers only actually observed markers as potential goals. To prevent newly observed marker from getting perfect reliability by simply $\frac{1}{1} = 1$ problem the O is set to equal 0 for first observation. The trajectory is visualised in figure 5.23.

In optimal case the function is continous and it is growing monotonously. It is defined for every valid value of O and F_{CNT}. Important feature of the function is that the newly observed markers get zero reliability but their reliability quickly grows with new observations. In case of missing observations the reliability function value does not grow. Actually it drops as can be observed in figure 5.24.

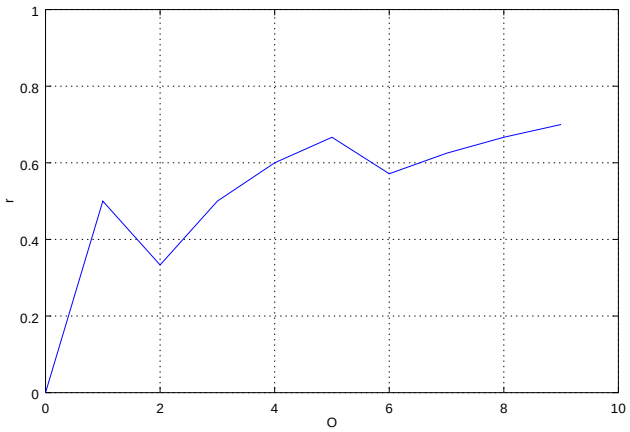

Figure 5.24: Marker reliability trajectory - missing observations.

The problem of forgetting disappeared marker is depicted in figure 5.25. We can see that the trajectory declines slowly. This efect will be more evident if the marker was observed for a long period of time (many frames). This problem is avoided by considering only those markers that are actually observed. Unobserved markers are neglected in current iteration despite their good prior reputation. Forgetting the marker is conditioned by drop of its quality below forgetting threshold. With this approach the markers that were observed for a long time will remain in the buffer for a longer time but they will not affect performance of entire solution.

The effect of quality selection instead of reactive closest marker choosing in „Choose target“ phase is visualised in figure 5.26. The trajectory of the manipulator is not affected by its actual pose. Instead it relies on value of quality function for given marker. The quality function is the reliability funcion r defined in 5.32. We can easily observe that the trajectory is a way different from the reactive approach.

The greatest advantage of the quality based marker selection is the fact that the distance to most reliable markers is measured as first which increase the probability of obtaining reliable and useful data for localization. Another advantage is that it avoids wasting time with „flickering“ markers that appear and disappear amongst iterations. In reactive approach the manipulator starts traveling to a marker that disappear in next iteration. Reactive target selection process will choose another marker as a goal for the manipulator so manipulator will change the trajectory just to change it again in consecutive iteration when the prior marker appears again. In quality based approach the less reliable markers are choosed after the more reliable ones were visited. The very unreliable markers are ignored completely.

The quality based approach brings also several drawbacks. Probably the most significant one is slower marker selection. When the marker is observed for the first time it takes several iterations for its quality rank to grow enough to be considered as manipulator goal. This problem is most significant in cases where are only a very few marker observations. In these cases the waiting for growth of marker quality rank brings only useless delay without any benefit. Another drawback of quality based strategy is higher consumption of memory and

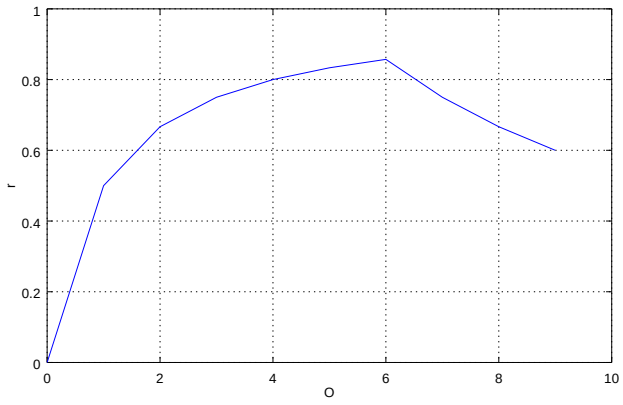

Figure 5.25: Marker reliability trajectory - marker disappeared.

CPU time. It may not be a significant drawback but in global evaluation of both algorithms it should be considered.

Important aspect of the quality based approach is the continous update of marker pose every time the remembered marker is observed. It is necessary to track the marker when robot moves. To make the marker observation independent from higher level robot control the manipulator control does not have information about motion of the robot. It uses advantage of high framerate comapred to speed of robot motion to track the markers despite the robot moves. As for reactive approach loosing association of observations of the same marker between frames is not crucial in case of the quality based approach the association is essential to build up marker quality rank. Failing to associate the marker has a way more serious impact on performance in case of quality based selection strategy. The effect of marker expected pose drift due to robot motion is depicted in figure 5.27. Continous updates of marker pose allow better association and better tracking as the manipulator goal is also continuously updated while aiming at the same marker.

The control of the manipulator motion during „Travel to target“ needs to deal with manipulator mass and inertia. Despite this is not directly a part of high level control of the manipulator the problem of dynamic system control with inertia is tightly connected to it. The problem is precise positioning of the manipulator. The controller needs to control acceleration of motivators to achieve fast but still precise motion of the manipulator. As the manipulator does not change its payload in the time (by grasping for example) we can tune up the controller precisely for given hardware configuration. Three approaches were consiedered: PID regulator, fuzzy logic regulator and neural network. To control the dynamics of the manipulator the PID regulator [22] was used.The PID controller is convenient for non-changing dynamic systems as it requires minimum prior knowledge of the regulated system comapred to fuzzy logic [21] and tuning of its parameters is a way more simple compared to for example neural network training. The basic idea of the PID regulator is described by equation 5.33.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{5.33}$$

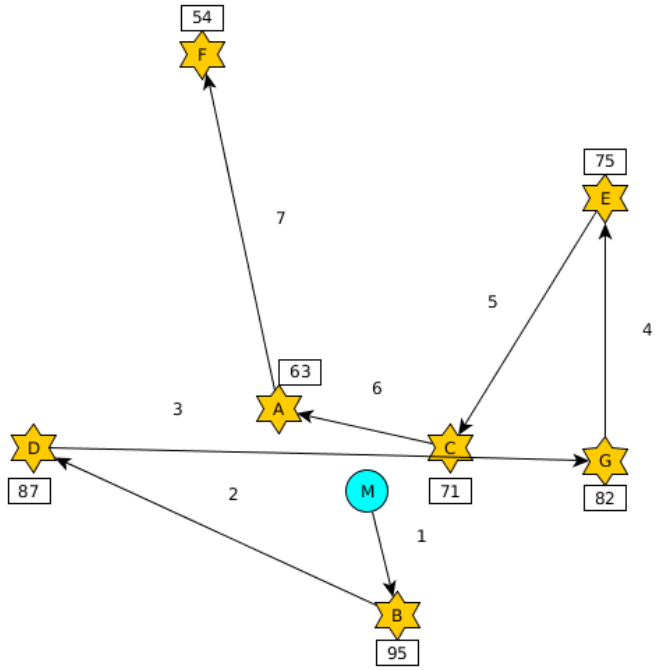

Figure 5.26: Quality based marker selection strategy.

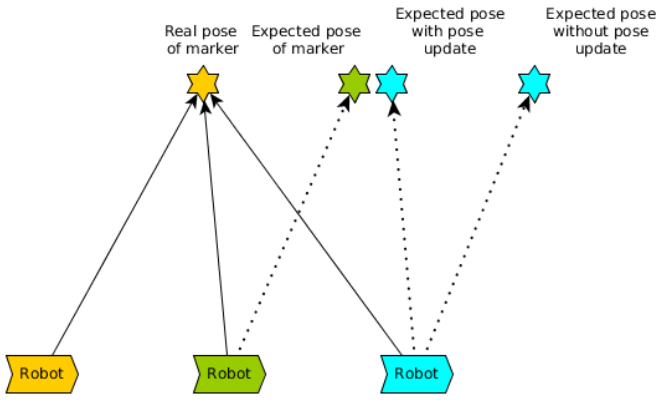

Figure 5.27: Drift of real marker pose from expected one due to robot motion.

The control signal for effector $u(t)$ is given by weighted sum of three components: The proportional, the integral and the derivative component. Each component processes the error signal of regulation feedback error measured in time $e(t)$. The proportional component simply amplifies the error signal. It takes the actual error and converts it into regulation action. The integral component works with history of regulation error. It sums the error through the time. Purpose of the integral component is to prevent oscilation of regulation loop around desired value. The problem of integral component is that the absolue value of the integral in time grows if the error does not change its sign for a long period of time. This may lead to effectively disabling the controller if too large value integrated in time. To avoid this kind of problems the integral component has a limitation of absolute value usually. The last component is the derivative component that reacts on change of regulation error. If the error grows steeply the derivative component quickly reacts on the change - much faster than the proportional component as the derivative component reacts on change.

When the manipulator arrives to the desired pose given by actual goal or within tollerance about it and marker can still be observed the „Taking measurement“ state is activated. In this state the distance information provided by laser rangefinder is used to measure distance d of the marker from the robot. The problem of distance measurement is that due to imprecision of manipulator pose and due to nature of landmarks (narrow tall object like tree trunks) the laser beam might miss the object. In case of single measurement there is a hight probability of incorrect distance measurement. Problem of the distance measurement is that we can not say if the measurement was correct because the only distance information is provided by the rangefinder.

To improve the reliability of distance measurement the advantage of high sampling rate of laser rangefinder was used. For one processed camera frame the distance is measured several times. In case of 30fps camera frame rate and 100Hz sampling rate of the rangefinder it is possible to take 3-4 samples for one processed frame. The additional distance measurements can be used to make the distance information more reliable. The landmark is by its nature an object protruding from the ground. As the object is protruding from the ground the distance to the object is smaller or equal to distance to the backgroud as depicted in figure 5.28. We can use this assumption to decide if the measurement is correct. Despite it is not completely reliable considering the smallest distance as the correct one gives proper distance measurement with the highest probability.

In the figure 5.28 the beam A hitted the object while beam B missed. The distance measurement in case B gives apparently greater distance than in case A. Of course the situation in the figure is simplified. In real world the object might be surrounded by several other objects and the beam might hit another object closer to the sensor. Situations like this are hard to detect even by human. The only counter measure is to use a robust localization algorithm that can deal with future correction of landmark pose.

Landmark covariance matrix

Last but not least important information provided by landmark detector is the covariane matrix. In case of landmark in polar coordinates particular components of pose vector do not affect each other. Thanks to zero co-variance amongst components the covariance matrix is diagonal as shown in equation 5.34. The angular variances of landmark pose related to base of the manipulator are defined by properties of the manipulator and allowed tollerances in the algorithm. This applies to angular variances σ_{phi} and σ_{psi} . The distance

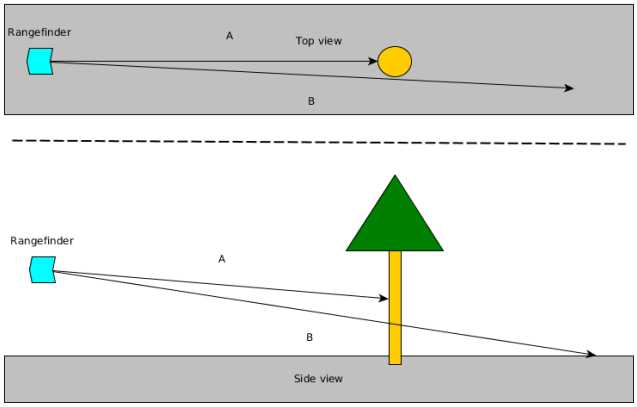

Figure 5.28: Distance measurement with laser rangefinder.

variance is given by parameters of the rangefinder. The depth variance σ_d might also be affected by procedure of taking distance measurements.

$$\Sigma_L = \begin{bmatrix} \sigma_d & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_\psi \end{bmatrix} \tag{5.34}$$

Derived from cosine sentence defined in equation 5.35 the equation for linear resolution of rotary manipulator joint according to distance to the object and resolution of the encoder is defined in equation 5.36 was defined.

$$c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot cos(\gamma) \tag{5.35}$$

$$\Delta_l = \sqrt{2d^2(1 - cos(\frac{2\pi}{ENC_{ppr}}))} \tag{5.36}$$

For demonstration purposes let's consider manipulator with rotary encoder with 4092 pulses per revolue (ppr) at its base joint for pan motion. If the desired linear resolution of pan motion is 0.30m (30cm) that should be sufficient to target trunk of a larger tree the maximal distance at which the manipilator can possible target the goal can be computed according to equation 5.37. The computed distance d_{MAX} for this example is approximately 195m which is not much. If we double the encoder resolution the maximal range at which it is possible to target the tree trunk with 30cm diameter is 391m.

$$d_{MAX} = \sqrt{\frac{\Delta_l^2}{2(1 - cos(\frac{2\pi}{ENC_{ppr}}))}} \tag{5.37}$$

The value of 390m even with extremly precise manipulator is not very good. Fortunately there is another attribute of the distance measurement system that is normally considered as drawback but in this case the drawback becomes an advantage. The attribute is beam divergence of the laser rangefinder. Every beam no matter how narrow becomes wider and wider with distance from the source. It is given by properties of photos emitted from single point source. The photons act as a spherical wave. The spherical wave spreads as a sphere with growing diameter. When the beam is emitted a small part of the spreading sphere

gets through the aperture so it looks like a narrow beam. Head of the beam is a spherical surface. With growing range the spherical surface grows in its size. Due to this effect the area covered by laser beam after hundreds of meters grow to sqaure meter and even more according to particular laser. Hitting the object with a square meter area is a way less demanding than hitting the object with a single point.

Of course the increasing diameter of the beam brings a significant disadvantage: The energy of the laser spreads accross the spherical surface so if the target is hit only by a part of the surface the reflected energy might be insufficient to take the measurement. If the energy is sufficient the rangefinder will obtain valid distance measurement. The laser beam forms a cone but without sharp edges. A frequent model for laser beam is a Gaussian beam model with gaussian power density distribution [34], [43]. The Gaussian model defines the diameter of the beam as a diameter where energy density drops to $\frac{1}{e^2}$ of the density e at the axis of the cone. So despite the beam cone has a particualr diameter at given distance the most of the energy is still concentrated at the center of the cone where axis goes through. Precise aiming with the laser still brings benefits of obtaining valid measurements for more distant objects.

The size of the cone is given by two parameters of laser emitter - the aperture diameter and the beam divergence. The relationship between aperture size, beam divergence and the range of the beam is defined in equation 5.38. The aperture diameter D_a is a constant value that affects the diameter but is usually very small. More significant is effect of beam divergence θ as it defines how much will the size of the cone grow with distance R. Beam divergence of recent good quality laser rangefinders vary from 2 to 3 mrad according to [34]. The divergence of 1 mrad corresponds to 10cm diameter growth every 100m. So for the distance of approximately 400m and the divergence 2 mrad the diameter of the beam will be approximately 0.8m. If the manipulator will be able to aim with granularity of +-30 cm and diameter of the laser beam is 40cm at the same distance it means that there are no spots that can not be covered by the rangefinder.

$$D = \sqrt{D_a^2 + R^2\theta^2} \tag{5.38}$$

The angular elements of the covariance matrix are given by precision of joints of the manipulator. There are two parameters that affect the precision: encoder precision and control tollerance. The control tollerance is a tollerance of the joint position in which the goal position is considered as reached. Its purpose is purely practical - positioning the manipulator exactly to encoder pulse unit that is the smallest distinguishable unit of rotation is usually slow as the joint has to move very slowly to avoid overshooting the goal position. Within the tollerance the „Taking measurement“ process is launched as the manipulator still moves to its goal position. In the worst case the angular pose error could reach the tollerance plus encoder error on both sides.

The error described above is the maximal error of rotationary joint. With hight probability the error will be lower. And with a very low probability caused by mechanical problems or data noise the error could be even greater. This approximately suits the Gaussian distribution of probability which is the distribution represented by covariance matrix. In this approach the maximal limit will be represented by $3 * \sigma$ distance. The angular error will fit into this interval with 99.7% probability.

Variance of the distance element is influenced by parameters of the rangefinder. The rangefinder manufacturers usually define the precision of the rangefinder as absolute tollerance of maximal error. The error of measurement is usually in tens of millimeters - for example +-20mm for RIEGL LD05e-A10[35] rangefinder. As the beam has a shape of cone

increasing its diameter with distance the possibility of taking invalid measurement grows. If the object is too small for the laser beam spot the probability of taking valid measurement decrease. If the energy reflected by the object is too small the invalid measurement may occure. If all the reflected energy is too low the measaurement will not be registered but in case the reflection form surrounding objects and background has significantly higher energy the distance of the backround or surrounding object might be readed instead of distance of object the rangefinder is pointing at. This uncertainity is difficult to model by gaussian. To model the unceratinity the model described by equation 5.39 was created.

$$\sigma_d = \frac{E_r}{3} + \alpha R \tag{5.39}$$

In the equation 5.39 the E_r represent the rangefinder error interval size. For interval $< -error_{MIN}, error_{MAX} >$ the $E_r = error_{MAX} - error_{MIN}$. The R is the measured distance and α is a weight coefficient saying how significantly the distance influence the variance. The linear growth of influence of the distance came up from the fact that the diameter of the beam cone grows linearly with the distance. The complete covariance matrix is defined by equation 5.40.

$$\Sigma_L = \begin{bmatrix} \frac{E_r}{3} + \alpha R & 0 & 0 \\ 0 & \frac{T_\phi + \frac{4\pi}{ppr_\phi}}{3} & 0 \\ 0 & 0 & \frac{T_\psi + \frac{4\pi}{ppr_\psi}}{3} \end{bmatrix} \tag{5.40}$$

5.7 SLAM algorithm

The SLAM algorithm used during localization process is based on FastSLAM algorithm described above. The algorithm was adapted to particular needs of this solution. The reason why the FastSLAM algorithm with vector map representation was chosen is given by good match of algorithm properties with requirements. One of the most significant properties is map representation. As the environment considered for the localization is mostly „empty“ in terms of the density of landmarks per square meter. Moreover the landmark is often of circular shape so representing the landmark using gaussian is convenient. Compared with occupancy grid based map representation the vecor of means can describe landmark poses exactly without growing computation time and memory demands (except for high precision value representation). This statement is valid for sparse environments.

Of course the efficiency of grid representation can be wastly improved by tree compression of the map. Still if the density of landmarks is low the octree representation will perform worse comapred to vector representation anyway. Each level of the tree of compressed occupancy grid represents particular granularity of cells. If we need to achieve better precision the size of the cell decrease. With every increase of precision at least one level of the tree has to be added. For example if one cell compounds of 4 small cells every level of the tree doubles the precision in following way: $E_{new} = \frac{E}{2}$. The figure 5.29 shows graphical representation of the tree data structure that contains description of one landmark. The figure shows representation in two dimensional space. The same landmark can be described by (μ, Σ) compounding of N-dimensional vector μ and NxN dimensional covariance matrix Σ where N is dimensionality of map representation - typically two or three.

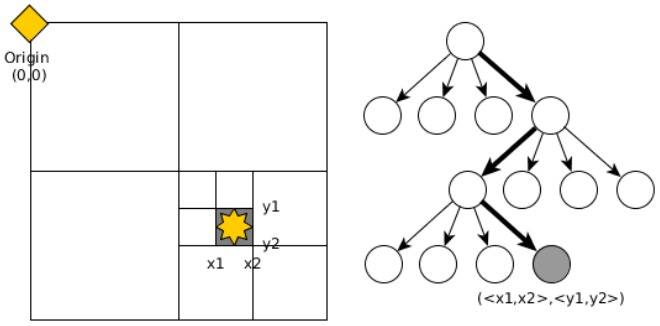

Figure 5.29: Occupancy grid representation for one landmark.

For purposes of localization the generic algorithm for Fast-SLAM described in chapter FAST-SLAM was specified by defining motion model $p(x_t|x_{t-1}^{[k]}, \mu_t)$ and observation model $h(\mu_{j,t}, x_t^{[k]})$.

Motion model

The Fast-SLAM algorithm uses probabilistic model of robots's motion. It is defined by probability distribution function. The function models estimated location of the robot in the map according to control action μ_t in time step t. The control action of robot motion can be with or without feedback. More sophisticated robotic platforms provide feedback using odometry. This allows to measure real size of the control action. With more precise information about control action the motion model can be more precise.

For ground robot with differential drive the typical motion model is gaussian distribution as described in [65] despite other models can be used. Most of the algorithms were designed for indoor environments. Typical for indoor environment is 2D flat space in which the robot is located. For exact description of robot pose in such space is 3 dimensional vector $\mu_t = (x_t, y_t, \theta_t)$ and 3x3 dimensional covariance matrix $\Sigma_t = \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_\theta \end{bmatrix}$. So the description of robot's pose is 3-dimensional. In case of outdoor localization the 2D map plane with 3D robot pose representation is usually insufficient. In outdoor terrain the ground is usually not flat and distance of landmarks from the ground may also vary. This is why the pose of the robot in case of outdoor terrain localization has to be represented by full 6 dimensional representation described by 5.41.

$$\mu_t = (x_t, y_t, z_t, \omega_t, \phi_t, \theta_t)$$

$$\begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\omega & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\theta \end{bmatrix} \tag{5.41}$$

The model counts with simplification that the components of robot pose are independend of each other. This fact is modelled by diagonal covariance matrix in form 5.42.

$$\text{diag} : \mathbb{R}^n \to \mathbb{R}^{n \times n}, \quad \text{diag}(a_1, ..., a_n) := \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{bmatrix} \tag{5.42}$$

In reality the z coordinate and rotations ω and θ of robot can not be estimated by some odometry solutions. Moreover these uncertain coordinates are affected by changes in remaining coordinates in very unspecific way. When robot moves through the terrain it will climb up the hills or go down into valleys and tilt according to terrain gradient. This is why many outoor odometry solutions use IMU augmentation of sensoric system that is capable of measuring size of tilt and vertical differences. According to particular capabilities of the robot the motion model has to be adapted. If the odometry does not provide information about some of the coordinates the typical way of representing this fact is to put very high or infinite values to covariance matrix at positions corresponding those coordinates. This solution is very effective for Kalman filter but causes problems during sampling - the samples are scattered in a very large area due to high uncertainity.

Another way is to use prior poses. This approach is based on assumption that the coordinate will not significantly change with small changes in remaining coordinates. Still the variance of unknown coordinates has to be large enough to prefer pose hypothesis update according to observation before motion model estimate. The best option is of course a situation when odometry provides information about all coordinates. Despite the uncertainity is significantly higher for some coordinates (usually z coordinate) the SLAM algorithm can deal with such uncertainity. In this case the pose and variance reported by odometry is given by fusion of sensor data. Typically the Extended Kalman filter is used for such fusion.

The coordinates that are direclty influences by control action are computed by one of following approaches according to model. In case of no odometry feedback the new pose of the robot is computed according to equation 5.43. In equation 5.43 the $\mathbf{\Delta u}$ is difference in control action and $km()$ is kinematic model of the robot chassis.

$$\mu_{\mathbf{t}} = \mu_{\mathbf{t-1}} + km(\mathbf{\Delta u}) \tag{5.43}$$

If the feedback from odometry is available the equation will change a bit. Instead of using control action $\mathbf{\Delta u}$ as a parameter to kinematic model the difference of odometry $\mathbf{\Delta o}$ will be used instead. The $conv()$ is a convertion function from odometry to control action. It is just a detail that ensures compatibility of motion model parameters.

$$\mu_{\mathbf{t}} = \mu_{\mathbf{t-1}} + km(conv(\mathbf{\Delta o})) \tag{5.44}$$

According to principles described above the motion model has to be adapted to particular mobile robot. The need for 6D robot pose came up from 3D landmark representation. For better modeling of variance of robot motion the final covariance matrix of the motion model can be generated in every step by multiplying covariance matrix of control error by jacobian of kinematic model in actual pose. If the kinematic model is linear the jacobian is invariant to parameters of kinematic model. For linear model we can avoid generating of motion model jacobian in each iteration of the SLAM algorithm. Computation of covariance matrix of motion model using kinematic model is defined in 5.45.

$$\begin{gathered} \Sigma_\mu = M(u_t)\Sigma_{u_t}M(u_t)^T \\ M = \frac{\partial km(u)}{\partial u} \end{gathered} \tag{5.45}$$

For differential drive chassis the model can be specified using equations 5.1. Considering differential drive kinematic model defined in 5.46 the model control vector $\mathbf{u}_t$ has two components - the linear speed v_d and rotational speed v_θ. This is typical control action used by high level control algorithms in ROS [72], [57]. The kinematic model defines new pose of the robot as 6D coordinate. Not all components of robot pose vector can be affected by control actions. The components that are not influenced by control action are modeled using prior value of the component and random growth $rand()$. These coordinate components can not be used for localization purposes as they provide no information about real robot pose. This fact is modeled by covariance matrix as described above.

$$mk(u_t, \mu_{t-1}) = \begin{bmatrix} x_{t-1} + \Delta_t \cdot v_d \cdot cos(\Delta_t \cdot v_\theta) \\ y_{t-1} + \Delta_t \cdot v_d \cdot sin(\Delta_t \cdot v_\theta) \\ z_{t-1} + rand() \\ \omega_{t-1} + rand() \\ \phi_{t-1} + rand() \\ \theta_{t-1} + \Delta_t \cdot v_\theta \end{bmatrix} \tag{5.46}$$

The kinematic model 5.46 jacobian M according to control vector u results in matrix described in 5.47. The components that are independent on control u vere zeroed in derivative. The zero rows cause zeroing of elements of motion model mean vector and covariance matrix. In motion model the zeroed components will be replaced by sensor parameters or their effect will be eliminated by infinite variance.

$$M(v_t, v_\theta) = \begin{bmatrix} \Delta_t \cdot cos(\Delta_t \cdot v_\theta) & -\Delta_t^2 \cdot v_d \cdot sin(\Delta_t \cdot v_\theta) \\ \Delta_t \cdot sin(\Delta_t \cdot v_\theta) & -\Delta_t^2 \cdot v_d \cdot cos(\Delta_t \cdot v_\theta) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \Delta_t \end{bmatrix} \tag{5.47}$$

By substituting the jacobian M to equations 5.44 and 5.45 the parameters μ_t and Σ_t of gaussian motion model can be computed. The estimation of new pose of the robot will be computed as a set of particles generated by this motion model.

The initial set of particles usually needs to have greater variance as the estimate of robots pose is very imprecise. If some prior estimate of robot pose exists it can be modeled by gaussain initial distribution of particles. The mean of this initial estimate models the expected initial pose of the robot and variances model uncertainity of particular coordinates. If there is no prior estimate of robot pose the initial distribution of particles can be uniform.

Another important aspect of the model that is not obvious at the first glance is the fact that the model has zero variance for standing robot. This aspect copies the reality - if the robot does not move there is no motion uncertainity. The new particles will be generated exactly at positions of existing particles.

Observation model

Observation function if transformation function from map to sensor transformation frame. Jacobian of transformation (TF) follows.

$$R_z = \begin{pmatrix} \cos(r_z) & -\sin(r_z) & 0 \\ \sin(r_z) & \cos(r_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.48}$$

$$R_y = \begin{pmatrix} \cos(r_y) & 0 & \sin(r_y) \\ 0 & 1 & 0 \\ -\sin(r_y) & 0 & \cos(r_y) \end{pmatrix} \tag{5.49}$$

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(r_x) & -\sin(r_x) \\ 0 & \sin(r_x) & \cos(r_x) \end{pmatrix} \tag{5.50}$$

$$R = R_z R_y R_x \tag{5.51}$$

$$h(\mu, x) = \begin{pmatrix} \sqrt{(\mu - \mathbf{x})^T (\mu - \mathbf{x})} \\ acos(\frac{((\mu - \mathbf{x}) \cdot R_x^T \cdot R_y^T \cdot R_z^T)[3]}{\sqrt{(\mu - \mathbf{x})^T (\mu - \mathbf{x})}}) \\ atan(\frac{((\mu - \mathbf{x}) \cdot R_x^T \cdot R_y^T \cdot R_z^T)[2]}{((\mu - \mathbf{x}) \cdot R_x^T \cdot R_y^T \cdot R_z^T)[1]}) \end{pmatrix} \tag{5.52}$$

$$H_{conv} = \begin{pmatrix} \frac{a}{\sqrt{c^2+b^2+a^2}} & \frac{b}{\sqrt{c^2+b^2+a^2}} & \frac{c}{\sqrt{c^2+b^2+a^2}} \\ \frac{a\,c}{(a^2+b^2+c^2)^{\frac{3}{2}} \sqrt{1-\frac{c^2}{c^2+b^2+a^2}}} & \frac{b\,c}{(a^2+b^2+c^2)^{\frac{3}{2}} \sqrt{1-\frac{c^2}{c^2+b^2+a^2}}} & \frac{\frac{c^2}{(a^2+b^2+c^2)^{\frac{3}{2}}} - \frac{1}{\sqrt{c^2+b^2+a^2}}}{\sqrt{1-\frac{c^2}{c^2+b^2+a^2}}} \\ -\frac{b}{a^2\left(\frac{b^2}{a^2}+1\right)} & \frac{1}{a\left(\frac{b^2}{a^2}+1\right)} & 0 \end{pmatrix} \tag{5.53}$$

Variables a, b, c are defined in equation 5.54. Robot pose $\mathbf{r}$ compounds of cartesian pose $\mathbf{r_C}$ and rotational pose $\mathbf{r_R}$ in following way: $\mathbf{r} = [\mathbf{r_C}\ \mathbf{r_R}]$.

$$\begin{aligned} \mathbf{l_{rotated}} &= (\mathbf{l} - \mathbf{r_C}) \cdot R_z(r_R)^T \cdot R_y(r_R)^T \cdot R_x(r_R)^T \\ a &= \mathbf{l_{rotated}}[1] \\ b &= \mathbf{l_{rotated}}[2] \\ c &= \mathbf{l_{rotated}}[3] \end{aligned} \tag{5.54}$$

Weight update

After every observation particle weights are updated according to how precise the observation fits into each particle's map. This step differs from the reference implementations of FAST SLAM algorithm. Typically in the FAST SLAM every sensor scan brings several observations. Weight of every particle is computed according to match of the observations. Particle weights are computed with every scan according to latest observations only. This approach works very well if there are many observations and observations come with every scan.

In our case the situation is different. The observations are rare and if the observation comes it is only a single sample. Moreover getting just some observations is not very usefull. It is necessary to get re-observation to update weight. New observation is just placed into particle's map but it does not say anything about fitting new observation into existing map. This is the reason why getting particle weights according to latest observations only is not usable. Particles have to maintain history of weight - some kind of reputation. This reputation allows to take into account last several observations instead of the only last one. This solution helps to improve the stability and also decrease variance of particles as the

particles that really represent the reality have priority during resampling all the time - not just according to „lucky hit“ of particle that does not match well otherwise.

To remember history of observations some history of weights has to be remembered by each particle. This can be achieved by keeping buffer of last N weights assigned. Another approach that works on a similar principle but it is less demanding is to keep only one historical weight value but add a new value to it proporcionally. The weight udpdate is defined in equation 5.55.

$$w_t^{[k]} = \alpha \frac{f(x^{[k]})}{g(x^{[k]})} + (1 - \alpha) w_{t-1} \tag{5.55}$$

The mixing rate α affects how much the update will be affected by new observation. reasonable results were achieved with $\alpha = 0.2$. This low number is given by re-observations erros due to noise in map. The noise in map is casued mostly by mising the object by rangefinder and measuring distance to the ground behind it. This way several false landmarks in the map can be added. Most of them will never be re-observed but sometimes these false landmars are matched. The noise was visualised in figure 6.3. The figure shows observaitons transformed into transformation frame of map. There are two clusters of observations. The more distant one is surrounded by false observations - the noise.

Resampling

Very important step of the SLAM algorithm is resampling step when new population of particles is being generated. The resampling problem compounds of two subproblems: How to resample and when to resample. The way how to resample is defined by resampling strategy. The resampling strategy defines which particles will be selected for generating new population. The resampling strategies and their properties were described in theoretical part of this work. In this case the rulete wheel strategy was chosen as it chooses particles according to their weights but does not eliminate particles with low weight completely. The survival of particles with low weight allows survival of less likely hypotheses. The less likely hypotheses may speed up localization when sudden change of the robot pose is large due to new observations.

When to resample is a bit more complex problem. According to many research conclusions there is no universal best way how to decide when the resampling should happen [73]. Too frequent resampling leads to survival of only the best fitting particles. Set of equivalent best fitting particles or set of particles very close to this situation doesn't provide additional informations about other hypotheses. The entropy of the system drops. Moreover the localization becomes less robust and less reliable. It can not effectively deal with large corrections of pose estimate as hypotheses like this are not available in actual particle set.

Without resampling on the other hand the particles spread into larger area with each iteration until they cover too large area with too low density. Under such circumstances the particle set can not model reasonable hypothesis as most of the parpticles is just too far from real pose of the robot. Efficiency of localization drops despite high amount of particles. The estimated trajectory of the robot may completely decline from the real one in the worst case.

The most straightforward approach to resampling is to resample every Nth iteration where N is set experimentaly. This approach can be used in solutions that have a constant float of observations and rather stable environment. Example of such solution is localization based on LIDAR in indoor environment. Every time the robot moves the SLAM algorithm

iteration is counted and after N iterations the resampling is executed. After resampling the counter is reset and the cycle repeats. With this approach the SLAM algorithm iteration counting for resampling is done only when the robot moves. If the robot is steady the particles do not move. Resampling under such circumstances would lead to loss of diversity of hypotheses.

Several variants of resampling were considered. Boundaries are defined by performance of SLAM with constant resampling in every iteration and by no resampling at all. The metrics used to evaluate these approaches are effective sample size and particle pose variance in each coordinate. Trajectory of effective sample size can be observed in 5.30. The effective sample size is defined in equation 5.56.

The same situation from point of view of particle pose variance can be observerved in plot 5.31. The particle pose variance proves the hypothesis described above saying that particles are getting too spread and most of them do not represent a reasonable hypothesis. In plot 5.31 we can observe that particle variance grows continously in coordinates X and Y. Z coordinate is limited by motion model that contains a minimal variance in Z axis.

$$ESS_t = \frac{M}{1+\frac{1}{M}\sum_{i=1}^{M}(M\omega_i - 1)^2} \tag{5.56}$$

Effective sample size represents how well a subset of a large set represents all samples in the set. It is widely used to create estimation of political preferences amongs people - just a few people are questioned about their preferences but member of the sample are chosen carefuly to represent entire set as well as possible. Without prior knowledge of attributes of entire set we can not compute the effective sample size precisely but it can be modelled by some kind of estimator. In case of particle filter the generation of particles can be considered as a subset of all posible hypotheses so it makes sense to compute effective sample size of generation of particles. In this case we use ditribution of particle weights as estimator. If particle weights are comparable the effective sample size is high. If weights of particle set vary a lot the effective sample size decrease. This way the effective sample size can give as a hint about performance of particle set. If some of the particles in the set have a very low weight they represent a very unlikely hypotheses. Such particles represent a „wasted hypothesis“ so effective sample size decrease. On the other hand if weights of the particles are more or less equal the all particles represent a reasonable hypotheses - there is no waste so the effective sample size is relatively high.

We can observe that the effective sample size without resampling varies a lot. In same cases it almost drops merely to 1 which means that the particle population can be effectively represented by a single particle. Under such circumstances the ability of particle SLAM algorithm to model all reasonable hypotheses is limited to capabilities of a single particle. Most of hypotheses represented by particles in the population are too far from reality.

According to data obtained by experimenting with resampling in every iteration and no resampling at all it is obvious that some more sophisticated resampling strategy is needed. The resampling can be conditioned by effective sample size. Relying on effective sample size did not bring as good results as expected due to oscilation of effective sample size parameter with new observations. To avoid too frequent resampling initiated by incorrect observation a combination of both attitudes was used. After reaching minimal number of iterations without resampling the resampling will be executed only if effective sample size is low enough. This solution provided the most stable results during practical experiments.

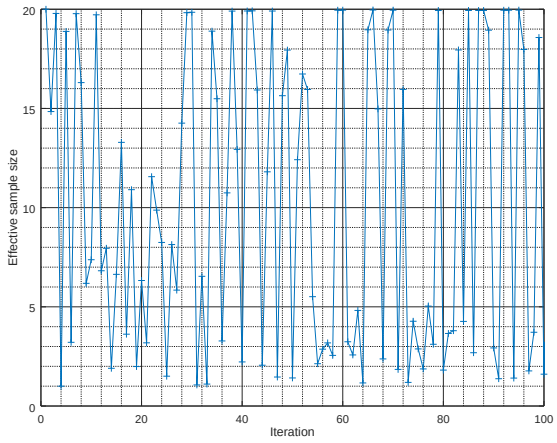

Figure 5.30: Result: Never resample - Effective sample size - Moving robot

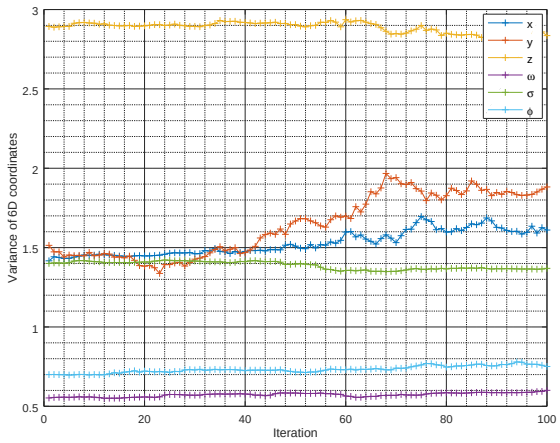

Figure 5.31: Result: Never resample - Particle pose variance - Moving robot

Chapter 6

Experiments

6.1 Simulation environment

During process of evaluating various attitudes it is not easy to repeat experiments with exactly same conditions in real environment. Moreover experimenting in real environment is time consuming due to hardware and environment maintenance. To simplify experimenting under same conditions simulation environment was prepared as a part of this work. To simulate environment for experimenting opensource simulator called MORSE[33] was chosen. This simulator is based on Blender 3D modelling environment and it is being developed by Open Initiative community. Core of the MORSE simulator computes kinematics and dynamics of rigid bodies so it can simulate interaction of the robot with environemnt. The physical model lacks advanced properties like advanced friction model based on material attributes or dilatation and contraction due to temperature changes. From point of view of sensoric subsystem MORSE contains basic set of sensors including odometry sensor, ultrasonic rangefinder, LIDAR, camera, RGBD camera, GPS or IMU amongst others. For purposes of simulation some sensor models were slightly modified or extended. The reason why MORSE was chosen is its open architecture with easy modability, easy usage and good performance. More detailed comparison of this simulator with other solutions is out of scope of this thesis.

For purpose of experimenting with SLAM in outdoor environment an environment with few trees and buildings was created. Robot can move freely in this environment in realtime and sensor observations also come in realtime. Screenshot of the environment can be observed in figure 6.1. Simulation is almost always simplification of reality and this simulation is not an exception. Essential atributes of reality were modeled to achieve realistic simulation for most of experiments. Following attributes of reality were modelled:

- Terrain profile

  Simulation contains realistic terrain profile with various gradient. The terrain gradient affects pitch and roll of the mobile robot and so it exploits all degrees of freedom of localization process in 3D environment. With completely flat terrain the situation is simplified to changing yaw angle only.

- Manipulator kinematic and dynamic model

  Manipulator is modeled more into detail including dynamic model which considers also inertia of the manipulator and acceleration of joint motivators. In real environment this phenomena significantly affects cadency and quality of observations. It is

Figure 6.1: Screenshot of simulation environment in MORSE simulator.

important primarily for landmark detecion. Dynamic model used for manipulator is defined in equation 6.1. In equation 6.1 element I denotes moment of intertia tensor of manipulator, a denotes actual acceleration, $\omega(a,t)$ denotes vector of angular speeds of all rotary joints according to its axes for all degrees of freedom. Element $p(a,t)$ denotes angular position of all axes of the manipulator.

$$\begin{aligned} \boldsymbol{\omega}\mathbf{(a,t)} &= \int I\,\mathbf{a}\,dt \\ \mathbf{p(a,t)} &= \int \boldsymbol{\omega}\mathbf{(a,\ t)}dt \end{aligned} \tag{6.1}$$

- Odometry error

  Important aspect of real environment is error of odometry. This error is actualy the reason why SLAM or other localization approaches are needed. Odometry error is a systematic error that grows in time. Odometry imprecision varies through time but most of the time the error is small and sometimes (due to drifting wheel for example) the error significantly increase for short period of time. To model this phenomena gaussian model was used. The principal of simulating odometry is the same as the principal used in motion model of SLAM algorithm. Continuous model of odometry is defined in 6.2. The model models pose and speed of mobile robot. Matrix M denotes robot drive model and $s(t)$ is change of encoder value in time.

$$\begin{aligned} \mathbf{p(t)} &= \int M\,\mathbf{s(t)}dt \\ \mathbf{v(t)} &= \frac{dp}{dt} \end{aligned} \tag{6.2}$$

  In reality odometry is usually not continuous due to digital encoders. Encoders are usually counters with discrete values and moreover they are being read over constant period of time so the value change for several impulses. The model 6.3 was used to

simulate odometry readed by encoders. Pose vector $p(t)$ and actual speed vector $v(t)$ are computed from encoder change.

$$\begin{array}{rcl} \mathbf{p(t)} & = & \sum_t M\,\Delta\mathbf{s}(1+\mathcal{N}(\mathbf{0},\,\sigma_e^2)) \\ \mathbf{v(t)} & = & \sum_t \frac{\Delta\mathbf{p}}{\Delta t} \end{array} \tag{6.3}$$

Every increase of odometry is changed by random error of size proportional to size of encoder change. Error size is given by variance σ_e^2. Error is modeled by gaussian distribution function.

- Differential drive control model

  The differential drive model - position as a function of encoder difference is described by 5.1. For simulation purposes it is necessary to model differential drive control which means rotary velocities of wheel as a function of linear and angular velocity. According to schema 5.3 the equation 6.4 was defined.

$$\begin{array}{rcl} \omega_r & = & \frac{v+\omega r}{\pi D} \\ \omega_l & = & \frac{v-\omega r}{\pi D} \end{array} \tag{6.4}$$

  In equation 6.4 the control speeds are v and ω where v is linear forward speed and ω is angular speed of rotation about Z axis. Component r is distance of wheel from center of the drive. Component D denotes diameter of wheel (we assume that both wheels have the same diameter). This model is defined for differential drive with two propelled wheels and some additional stabilization wheels without control but it also works for models with four propelled wheels on fixed axes - two on each side as can be observed in simulation screenshot 6.1.

- Laser beam model

  Laser rangefinder model was extended to better reflect the reality. In most of simulators the rangefinder takes a measurement exactly at one infinitely small point. It works for many use cases but in this particular case this simplification causes different behaviour of ranging distant objects. Real laser beam has a shape of cone with spherical surface instead of flat base. With growing distance the diameter of the cone base grows. Laser rangefinders usually considers only the nearest reflection of laser beam as a measurement. Practically it means that first object that crosses the spherical surface of cone base is taken as a measurement.

  Another aspect of laser beam is decreasing energy with diameter of cone base. If laser beam meets an object with particular surface size the reflected energy is proportional to part of spherical cone base surface that collides with the object. It means that more distant object reflects less energy that closer object of the same size. If a difference between reflected energy and noise drops below sensitivity threshold of the rangefinder the measurement is ignored as a noise. Of course noise changes through time so measurements can be sometimes accepted and sometimes ignored. For more distant object probability of correct measurement decrease. Practically larger or closer objects are better than more distant or smaller ones.

  To simulate both aspects of laser rangefinder ray casting covering volume of laser beam cone was used. All the rays initiate from the same point and they decline from the center of cone for angle growing by resolution step. This way the equidistant

measurement lays on spherical surface of cone base making a dense net of measurements. The net density decrease with growing distance. Nearest measurement is taken as rangefinder measurement. For more distant objects that are smaller than gaps in measurement net the probability of correct measurement decrease. For objects larger than gaps in net the measurement is always taken which does not reflect reality perfectly but it is close to it. Ray orientation vector pt_i is computed according to equation 6.5. The ray orientation vector is added to the orientation of center ray of beam cone to compute absolute orientation of a particular ray. Each ray orientation vector is computed as multiplication of direction vector $\mathbf{d}_j$ and angle given by index i. Direction vector $\mathbf{d}_j$ defines in which direction the angle of pose vector will grow. Indexes i and j define position on the spherical surface. Angle of maximum beam declination ϕ defines how wide or narrow the beam cone is.

$$\begin{aligned} \mathbf{pt}_i &= \tfrac{\phi \cdot i}{R}\mathbf{d}_j \\ where: \mathbf{d}_j &= (cos(\tfrac{\phi \cdot j}{R}), sin(\tfrac{\phi \cdot j}{R})) \end{aligned} \tag{6.5}$$

To obtain particular laser rangefinder measurement simply the smallest of all ray measurements is used as defined in 6.6.

$$l = \min_{i=-N \ldots N, j=-M \ldots M} l(i,j) \tag{6.6}$$

Some attributes of real environment that have significant impact on performance of localization were simplified or neglected in the simulation. The most significant of them are error of reference position measurement. For real experiments satelite navigation was used to obtain real trajectory of the mobile robot. Despite that the satelite localization has well documented error trajectory [67] it was not implemented into simulated environment. Reference pose measurement error complicates repeatability of experiments and decrease measurement precision.

Another simplified aspect of reality if non-photorealistic environment. This simplification affects performance of image processing. Environment in the simulator uses simple geometry compared to real world. On one hand edges are easier to detect, on the other ligh in the simulator is homogenous and all-directional so there are no areas of high contrast like in real environments. Simulator camera view is depicted in image 6.2. It is image from object detector with highlighted landmark detections.

6.2 Landmark detector performance

Landmark detection is a complex process with several variables including image processing and manipulator control. Aiming the laser rangefinder precisely at the distant object is not always possible. Especially with manipulators with limited precision. Manipulator with limited precision was used in simulation environment. In simulation environment the manipulator precision was decresed and range of laser rangefinder was limited to let the experiments work in smaller simulated environment due to performance reasons.

The process of obtaining observations can not be tested in simulation reliably as it relies on processing video data from camera. The visual quality in simulation is much worse than in reality. For this purpose experiments with landmark observations were executed on real hardware observing a real world. Measuring performance was difficult but visually the re-observation can be compared if the observations are put into map frame. Measuring variance

Figure 6.2: Simulation: Camera image from landmark detector with highlighted landmarks.

of observations automatically is not easy as landmarks have to be associated properly in moment of re-observation. For this purposes a visualization of map can be observed in figure 6.3. In this figure two landmarks are visible. Association of new observations to existing landmark was disabled to get overview of observation variance.

In figure 6.3 landmark observations are denoted by a red circle around it. Around the second landmark there are several incorrect observationts (circled by blue). The faulty observations were probably obtained by pointing laser rangefinder to the ground around the observed marker. These faulty observations act as a noise during SLAM process.

6.3 Visual compass precision

For orientation measurement the pinhole camera was used. With model of pinhole camera [48] we can compute theoretical best precision of orientation detection. Theoretical precision of orientation measurement based on camera is limited by two factors: Precision of feature detection and quality of the camera including image deformation and resolution. In perfect case when camera image is perfectly rectified and features are detected with one pixel precision the error is given by following equation:

$$E_R = \frac{FOV}{RES_H} \tag{6.7}$$

Where FOV is a field of view of the camera in DEG and RES_H is horizontal resolution of the camera. The equation says how large is the angle covered by one pixel of camera image. This is the best possible precision that can be achieved by approach based on color distance. To visualise precision of orientation measurement reference system was used. Comparison of orientation measurement solution with real orientation was done in simulated environment. As the visual compass is feature-less the simulated environment should be comparable to reality. At least simulated environment should not bring better results because of the visual compass compares whole images not just particular markers.

The measurement of rotation precision was done by rotating camera about its Z axis and returning it to its original position. After such experiment we can compare performance of solution used for measuring rotation with real postion of the camera.

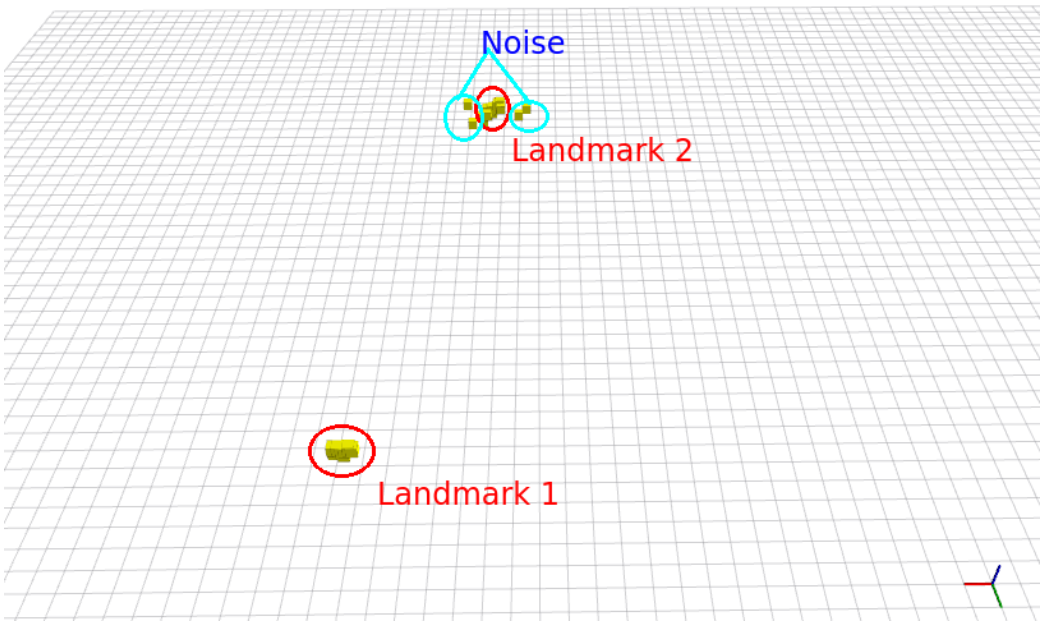

Figure 6.3: Landmark observations in map transformation frame - associtaions are disabled.

Precision of visual compass described in chapter above was measured as a comparison with reference measurement. Both reference measurement and compass measurement were put into one plot together with trajectory of orientation precision error. First sets of data were obtained from static rotation and frameskip was optimised manually. First plot 6.4 shows results for slow rotation of 0.1 radians per second, second plot 6.5 shows result for fast rotation 0.5 radians per second.

In 6.6 we can observe precision of orientation measurement with various frame skip. We can see that precision is best with frame skip about 120. Lower frame skipping has larger error because of the pixel error described above. With growing frame skipping the error decrease. After reaching the minimal error the error trajectory starts to quickly grow. It is caused by reaching frame skipping limit. The maximal skip is not a concern with 360 degrees panoramatic camera but with limited FOV camera the delay between two processed frames during rotation cause smaller overlap of frames. If the overlap is too small or none orientation difference can not be computed properly. So the error of orientation measurement grows.

Plot 6.7 shows effect of frame skipping on maximal error of rotation on a spot at speed of 0.1 radians per second. The phenomenas described above can be observed in the plot. Since frame skip equal 40 the error decrese to minimum which is about 120 and then it grows as the overcomming of maximum frame skip occures more and more frequently until it starts to happen regularly with frame skips above 200. The error trajectory does not fit the theory perfectly but it is generated by set of single runs - one for each frame skip. Maximum error is partially stochastic and it may vary with different surrounding environment, light conditions and of course with quality of the camera.

Plots 6.8 and 6.9 show the same experiments as 6.4 and 6.5 but this time with automatic frame skipping. We can observe that precision of automatic frameskipping is close to manually tuned solution. In case of smaller rotational speed the error is higher and moreover we can observe that the error grows systematically. This is probably caused by pixel error.

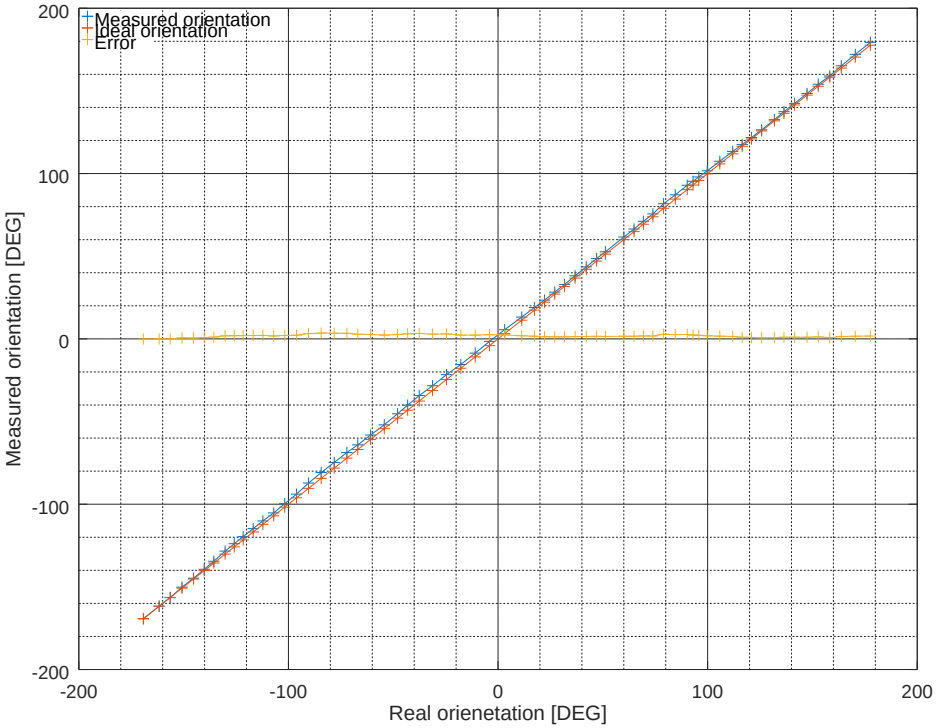

Figure 6.4: Reference: Visual compass - orientation error for interval $< -180, 180 >$ degrees, 0.1 radians per second, manually tuned frameskip.

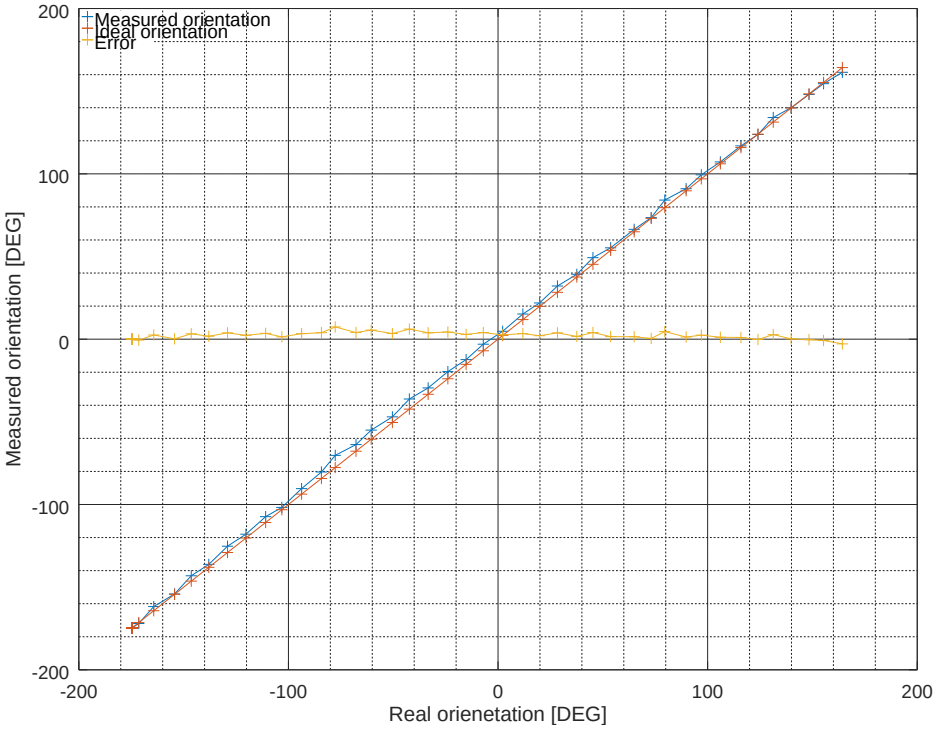

Figure 6.5: Reference: Visual compass - orientation error for interval $< -180, 180 >$ degrees, 0.5 radians per second, manually tuned frameskip.

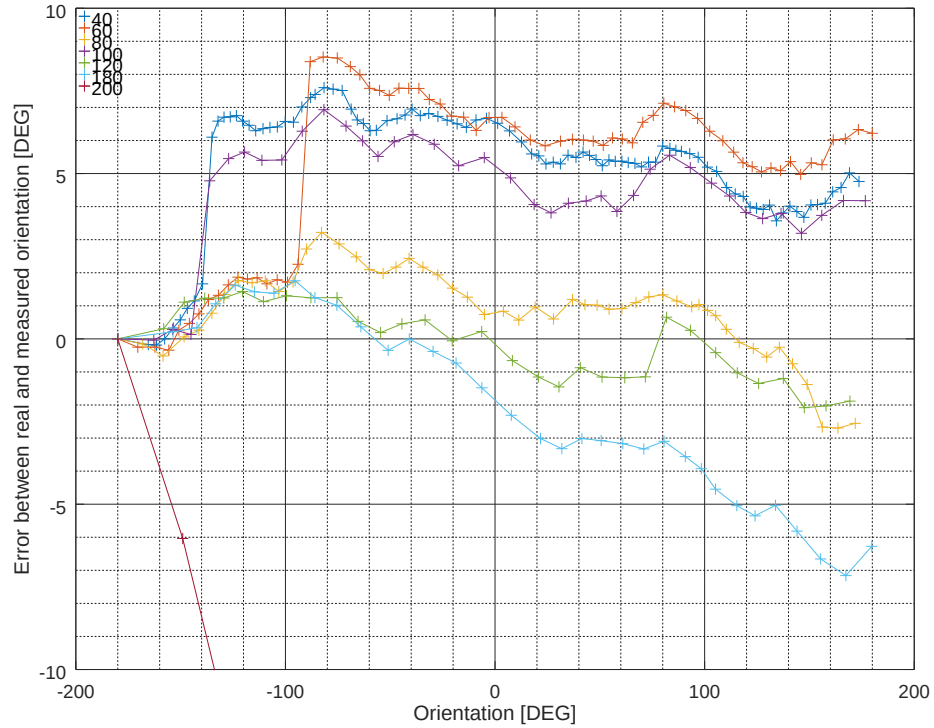

Figure 6.6: Impact of frameskip: Visual compass error for constant rotation at speed 0.1 radians per second.

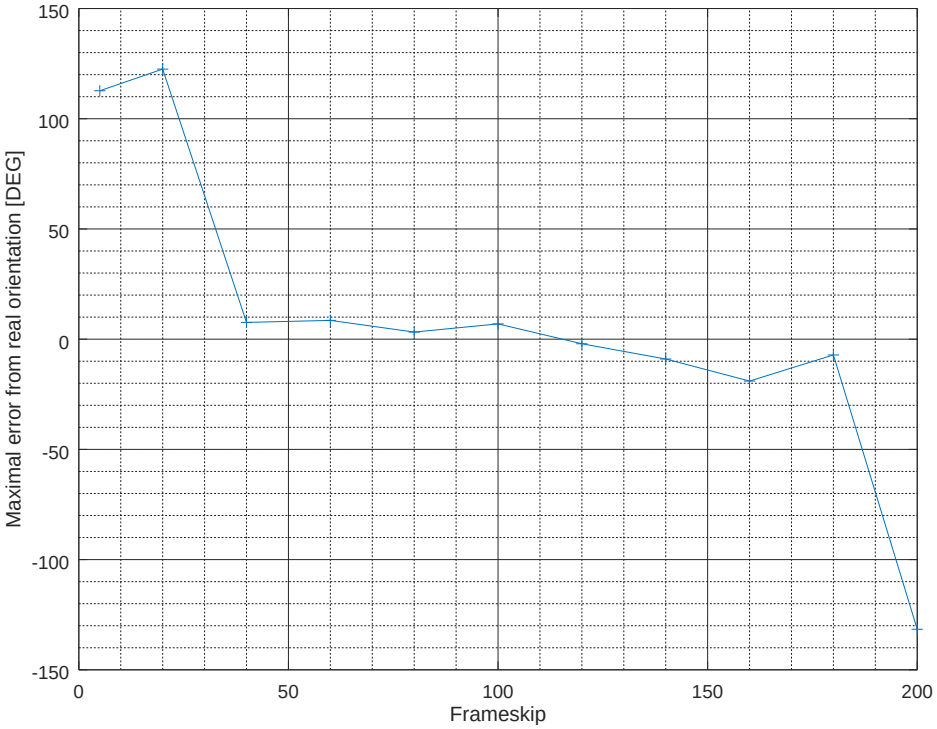

Figure 6.7: Impact of frameskip: Trajectory of maximal error between real and measured orientation for various frameskip.

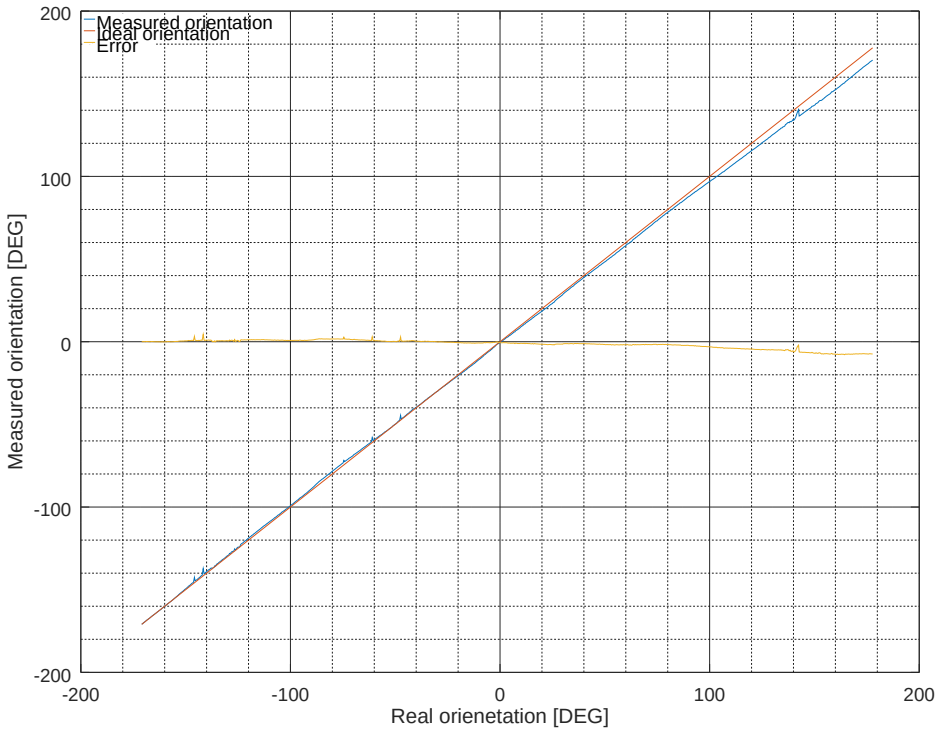

Figure 6.8: Visual compass - orientation error for interval $< -180, 180 >$ degrees, 0.1 radians per second, automatic frameskip.

Despite the automatic frame skipping base frame update is more frequent and so the average distance between two base images is smaller in pixels. The smaller distance is caused by smaller pixel distance between two consecutive frames. In case of slow rotation the two consecutive frames are shifted by few pixels. On the other hand during fast rotation the consecutive frames are shifted by five times more pixels as the speed of rotation is five times greater. This causes that the smallest value of pixel distance that overcome the pixel distance threshold for satisfying base image update condition is in average higher than in case of slow rotation.

The plot 6.10 demonstrates the impact of image scaling during linear motion. The experiment was executed in rough terrain which means that except for orientation - a yaw motion also roll and pitch motion occured during experiment. Especially falling of obstacles case a large error as it is nearly impossible to match consecutive images. From the plot we can see that without scaling the orientation error is higher most of the time. There is one big error peek about one hundreth sample which was caused by overcomming a steep terrain obstacle. Except for this moment the error keeps between +-10 degrees which is far from perfect of course but significantly better than experiment without scaling where the error started to grow from the beginning and almost 50% of time it was over 10 degrees.

In figure 6.11 we can observe dwo degrees of freedom matching error of visual compass - horizontal and vertical direction. We can see how the trajectory of horizontal match is changing with advancing vertical shift in both directions. We can observe that the shape of the horizontal error function remains unchanged but the difference between average and minimal error increase as we are approaching the proper shift. Interesting aspect is oscilation of frame difference error as we change horizontal shift. It is interesting that the

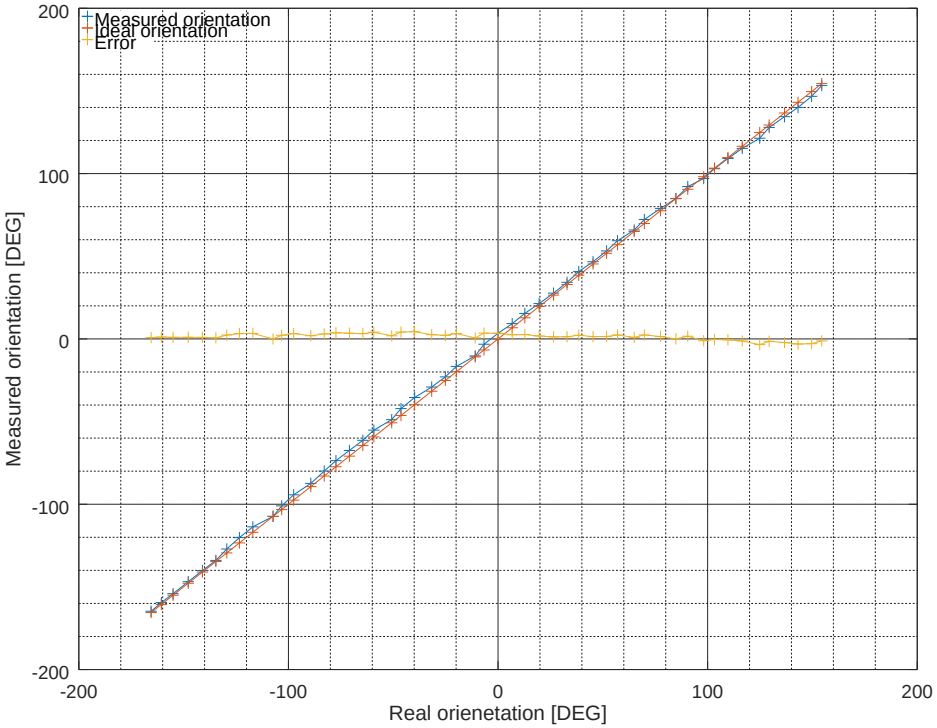

Figure 6.9: Visual compass - orientation error for interval $< -180, 180 >$ degrees, 0.5 radians per second, automatic frameskip.

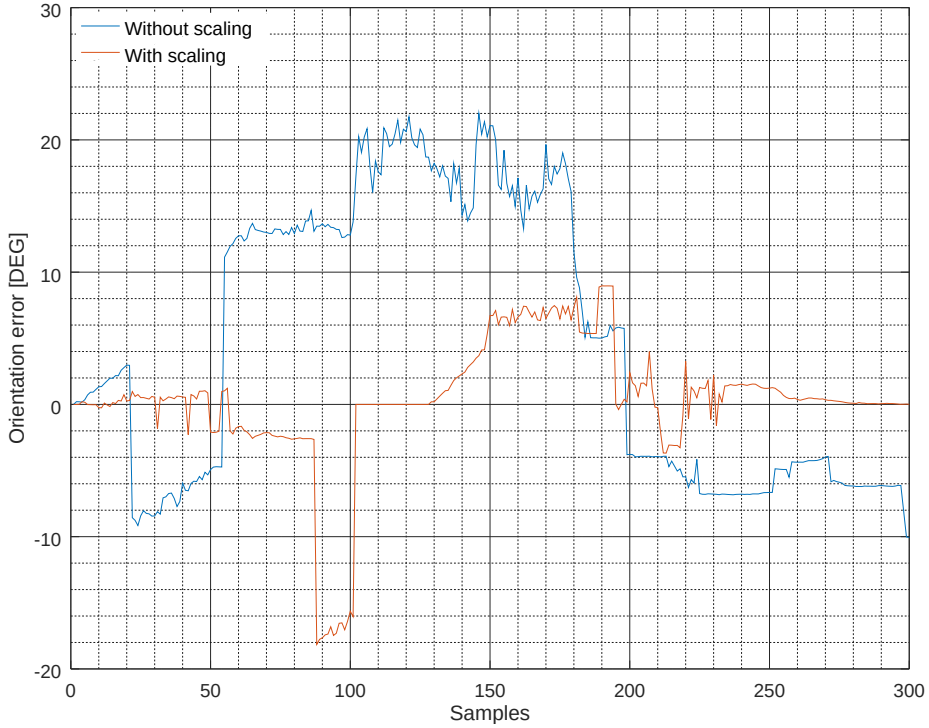

Figure 6.10: Visual compass - linear motion error - effect of image scaling - rought terrain .

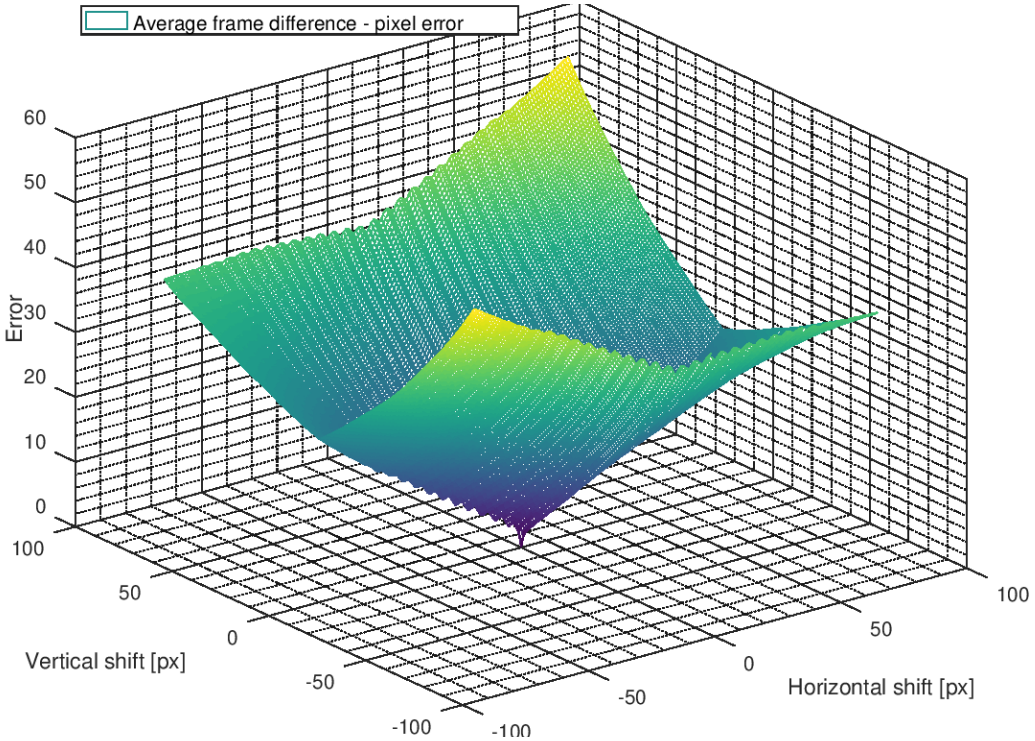

Figure 6.11: Visual compass - frame comparison error vertical and horizontal matching - static.

more we approach the optimum the more significant the amplitude of oscilation is. Reason for this oscilation is not completely clear. The plot shows the error while the camera was steady pointing at the same point all the time. This is a comparison of two practically identical images.

Plot 6.12 shows the error when the robot is moving forward in rough terrain. The plot is shown from point of view of vertical shift. We can observe interesting phenomena - the frame difference error is decreasing with extreme vertical difference. From point of view of horizontal shift (that is not visible in the plot) the plot shows that horizontal shift is incorrect for areas of extreme vertical shift. It was found by experiments that the drop of frame difference in extreme values of vertical shift comapred the rest of the trajectory grows with linear distance travelled between these two frames.

The decrease of error with greater vertical difference between two frames is probably caused by shrinking of overlapping area that moreover includes view of the sky. The sky has a very small difference as it is mostly homogenous or with only a small color variance.

6.4 Localization performance

Performance of entire localization process was measured as a comparison between real trajectory and trajectory estimated by SLAM implementation. Simulation environment was mostly used for this purpose as it is easier to realize reference pose measurement for the real trajectory. Moreover it allows to experiment with many aspects of the environment relatively easily. In limited scope it was tested that the phenomenas demonstrated in simulation experiments apply also to the reality. Grid of all the images is 1m.

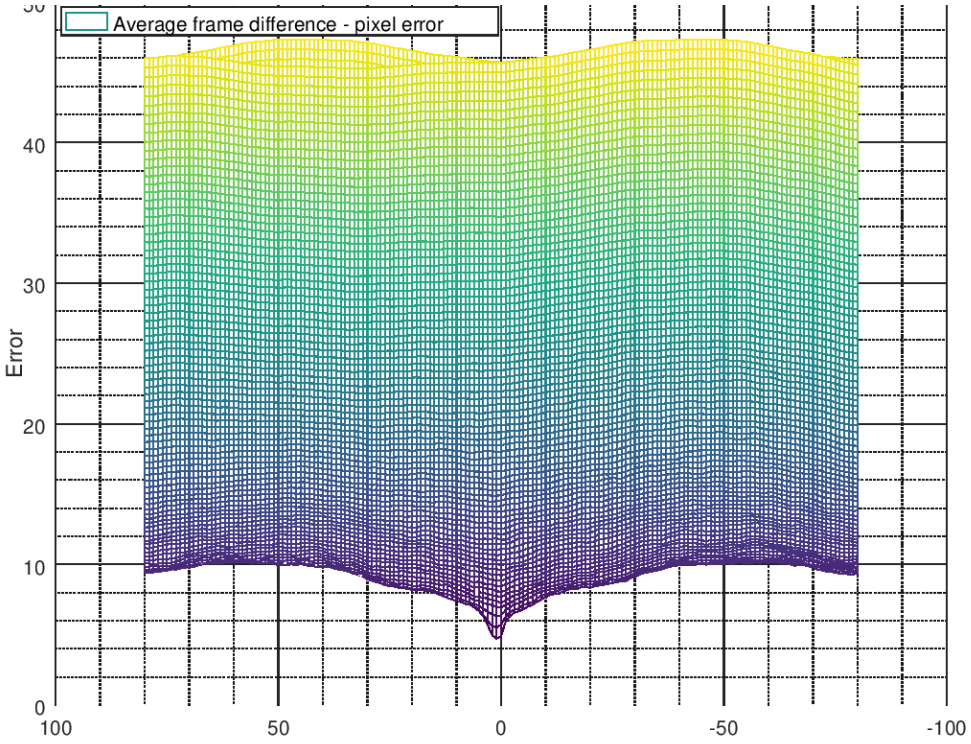

Figure 6.12: Visual compass - frame comparison error vertical and horizontal matching - moving straight through rough terrain.

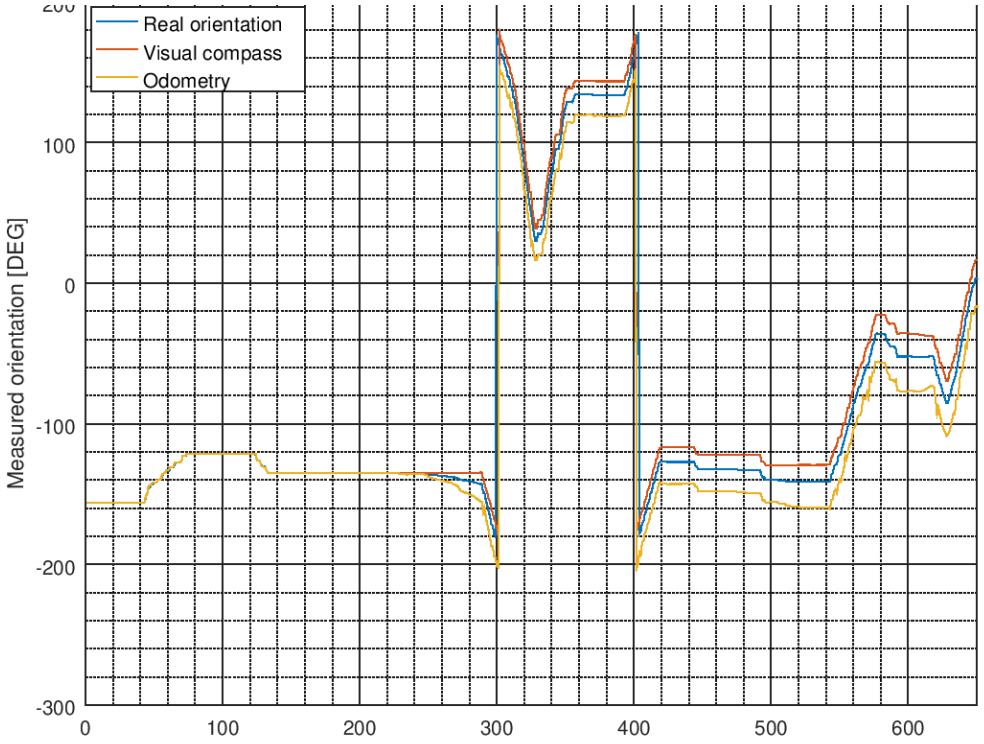

Figure 6.13: Visual compass - real orientation, orientaiton measured by visual compass and orientation measured by odometry.

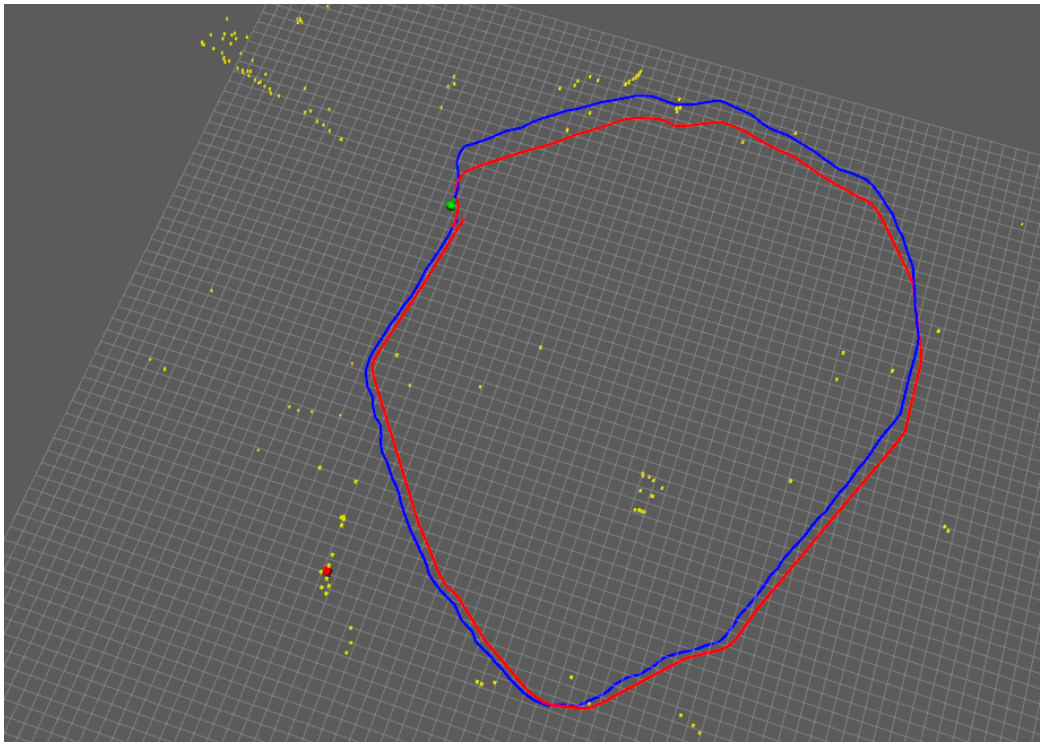

Figure 6.14: Data set 1: Localization vs real trajectory in simulation - best particle trajectory - odometry only.

As first performance of SLAM with and without visual compass was compared. The comparison was run on the same dataset. In both cases the trajectory of best particle was compared. The best particle trajectory is a historical trajectory of particle with the highest weight at the end of the experiment. This trajectory does not reflect the trajectory estimated in every moment of simulation time - the particle with highest weight vary during time as new observations come - but this experiment demonstrates the impact of different quality of odometry feedback on particle pose estimate. The experiment was run with fixed size population of 20 particles. Figures 6.14 and 6.15 show the best particle trajectory without and with visual compass respectively.

Note that the particular trajectory will change with every run even in the same data set as the posterior sampling is a stochastic process. Repeated experiments show that there was an improvement in average error but of course by nature of the algorithm there is no guarantee of error limit. With a very small probability the error may become rally large.

Another pair of trajectories on data set 2 demonstrate effect of weight computation. The data set was recorded in simulation environment - the terrain contained randomly scattered trees and few buildings that served as potential markers for SLAM. Figure 6.16 shows a comparison of trajectory recorded by SLAM and real trajectory for weight computed according to last observation. Figure 6.17 shows the comparison of the two trajectories for SLAM using integral weight with history. For the weight with history the coeficient α was set to 0.2. Such a low effect of the weight update may cause too slow update if there is an unexpected change but on the other hand the pose estimate is more stable and impact of incorrect or imperfect associations is not as significat as in case of weight update according to last observation.

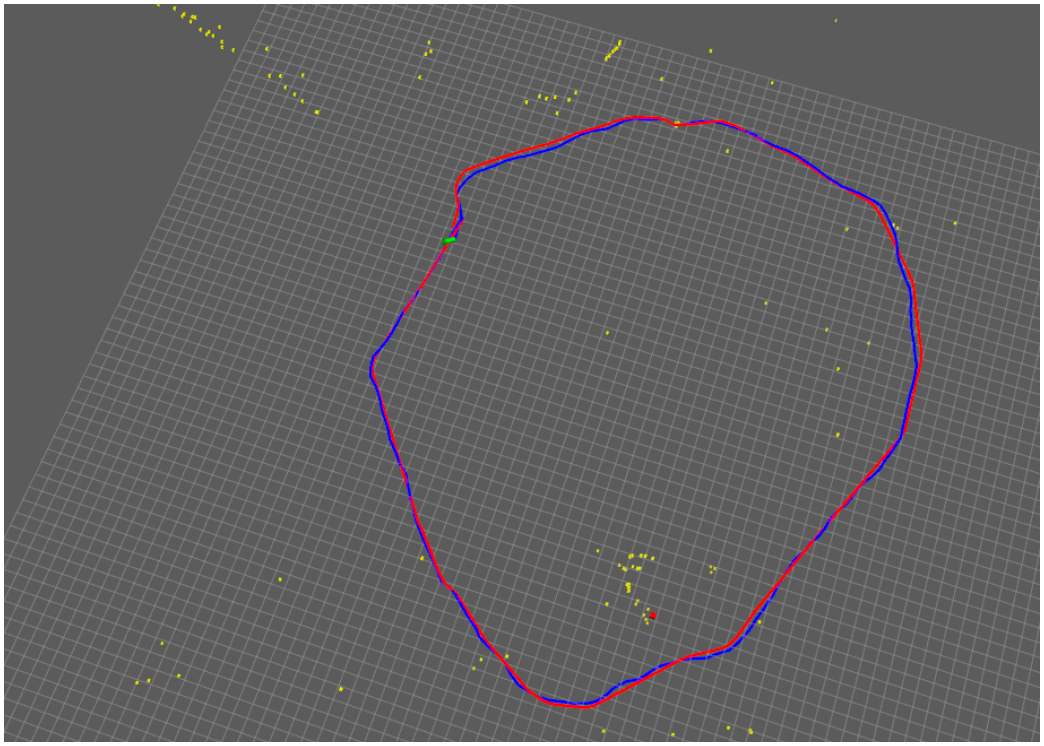

Figure 6.15: Data set 1: Localization vs real trajectory in simulation - best particle trajectory - with visual compass.

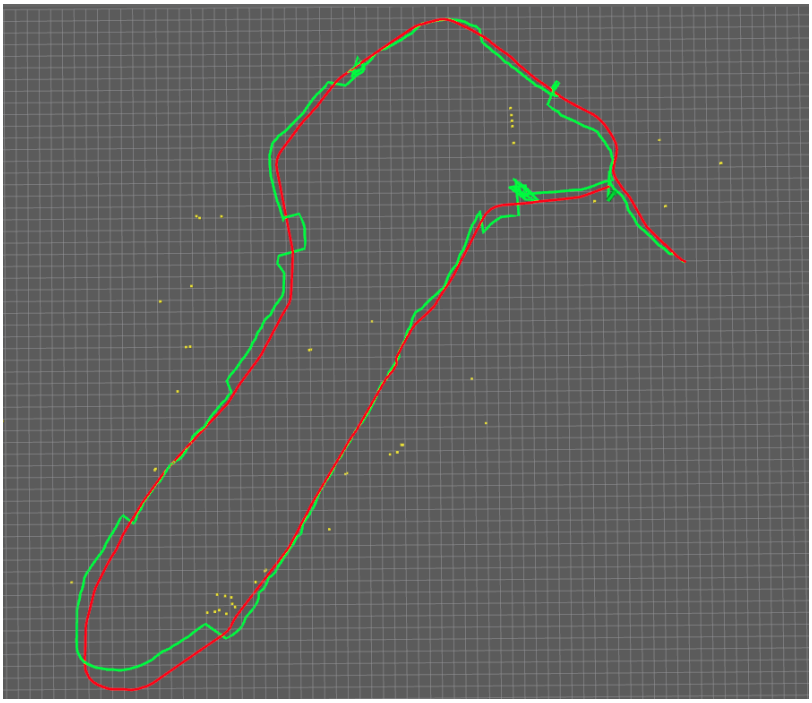

Figure 6.16: Data set 2: SLAM trajectory estimate vs real trajectory in simulation - weight by last observation.

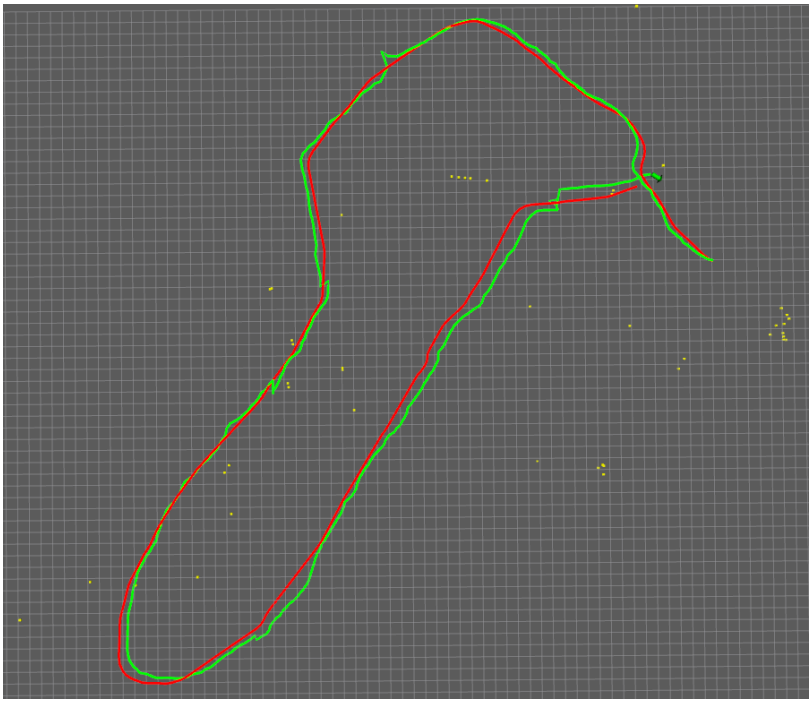

Figure 6.17: Data set 2: SLAM trajectory estimate vs real trajectory in simulation - integral weight.

In plot from datased 3 we can observe loop closing experiment with entire SLAM solution. In plot 6.18 and 6.19 is shown a trajectory of actaul best particle after experiment ended (6.18) and trajectory of robot pose estimate during the experiment (6.19). At the first glance we can observe that the trajectory estimated by SLAM during the experiment depicted on 6.19 contains glitches. These glitches are caused by two reasons. First reason why the glitches appear is that if there is no observation for some period of time the robot estimates its trajectory according to dead-reckoning. After new observation the weights of particles are corrected and particle that was the best particle after last observation suddenly becomes less important as another particle matches the reality better. The discontinuity in the trajectory reflects the priorization of another particle. We can see this happening in the left-most curve of the trajectory in the bottom left corner of the plot.

Another reason why the glitches in the trajectory appear is improper association of landmarks. We can observe this phenomena in a curve around the large red square point in the center of bottom third of the plot. In the curve we can observe a dotted line of landmarks. Most of them are actually a reflection of the same marker but it was matched incorretly as the robot was drifting from the hill so the odometry of linear motion did not reflect the reality correctly. There were many corrections of the trajectory as the landmark was sometimes associated and sometimes recognized as a new one.

Trajectory plot 6.20 shows the detail of pose correction with new observations. Yellow arrows represent particles. Each particle pose is at place where the arrow starts. The arrows point into direction where the particle is oriented.

The estimated pose changed from the top left particle into particle closer to the real pose. Despite there are closer particles that probably match better with their pose they

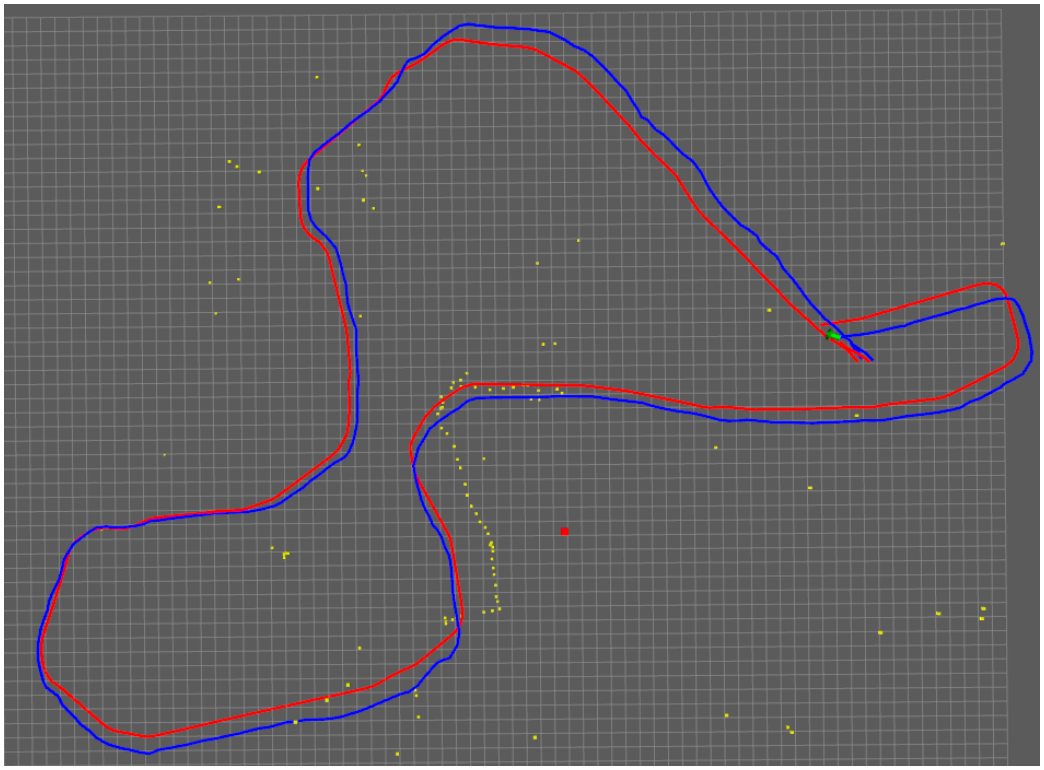

Figure 6.18: Data set 3: Actual best particle trajectory trajectory vs real trajectory in simulation - loop closing.

differ too much in orientation. This is why their weight was not greater. The experiment was run with 20 particles.

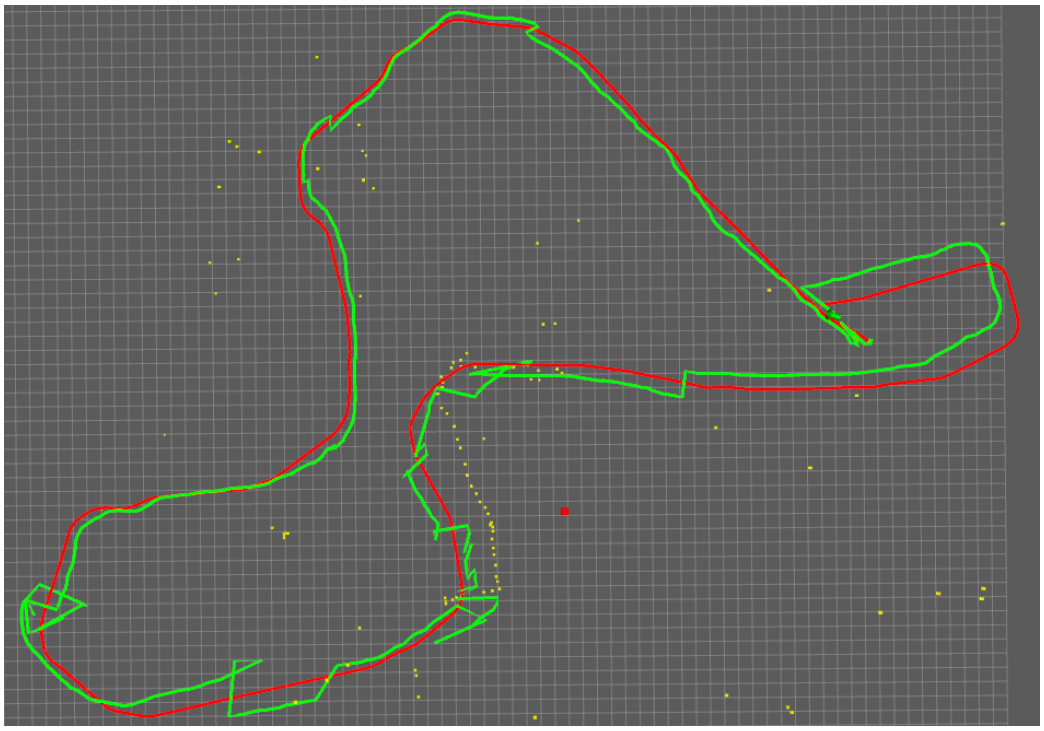

Figure 6.19: Data set 3: SLAM trajectory estimate vs real trajectory in simulation - loop closing.

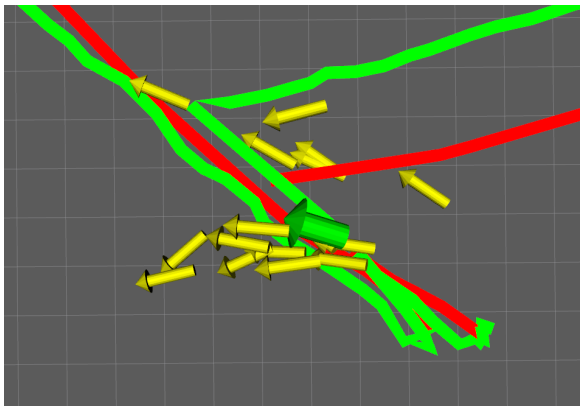

Figure 6.20: Data set 3: SLAM trajectory estimate vs real trajectory in simulation - detail of correction after re-observation of known landmarks.

Chapter 7

Conclusions

The final chapter is splitted into two subchapters. First of them contains discussion about measured results and conclusion. The second subchapter concerns about considered future development of the SLAM solution described in this work.

7.1 Conclusion of measured results

The measured results can be splitted into two parts. First of them contains results of visual compass that works as a standalone component. The second part contains evaluation of entire SLAM solution. The dependence of SLAM solution on visual compass is not a necessity but it improves the performance significantly as it can be observed in figures 6.14 and 6.15.

In this thesis a feature-less visual comapss was used. It was modified for camera with limited field of view. In real application the camera heading forward was used to minimize distorsion caused by linear motion. Visual compass precision was tested with several options including various frame skipping, linear motion compensation and also vertical shift correction. Experiments have shown that proper frame skipping is essential for visual compass precision. Further experimenting showed that reliability of visual compass can be improved by vertical shift compensation. Another improvement was achieved by using conditional update of base frame for orientation comparison. The base image update is triggered either by changing angle (if the angle between base and actual image overcomes defined threshold) or according to pixel difference of the best matched shift of the new frame.

The base algorithm for SLAM solution developed as the main goal of this work is the Fast SLAM algorithm with feature-based map. For a sparse map with only few landmarks the feature-based map is convenient. Moreover it allows to interpret landmarks as objects with additional features if needed. The SLAM method was tested on several datasets. Many data sets were obtained from simulated environment. Given by principle of the algorithm the simulation significantly affects the landmark detector but impact of simulation environment on non-visual aspects of the SLAM is minimal - this is why I could afford to use simulation for testing. Advantage of simulation is flexibility and possibility to simulate particular conditions that are much harder to arrange in real environment.

The experiments proved that the SLAM method provides position feedback with precision in units of meters (up to 3 meters according to depicted experiments). Despite the precision is far from perfect especially in comparison with differential GPS solution it provides a usefull starting point for local reactive navigation like approaching the door or

picking up an object observed by camera. In context of the environment (a large area with only few potential navigation points in it) the localization precision is reasonable.

Following list goes through the declared goals of this work. Each goal is discussed and compared with results.

Develop solution of localization in large outdoor areas. It was the main goal of this work to develop SLAM solution that can operate in large outdoor areas. SLAM solution was designed according to the requirements and implemented. Tests have shown that the solution is principially correct and that it works for testing data sets obtained from both simulated and real environment. Localization error varies according to circumstances but in areas where markers are observable the error is usually up to 3 meters. The greatest impact on error has imperfect measurement of manipulator orientation. The manipulator control approach is based on reactive navigation that is simply trying to point the laser rangefinder to the marker observed by camera. During the process of taking measurement the smallest distance value in close neighborhood of the marker is considered as the distance. The marker is supposed to protrude above ground. Another source of error is detection of the landmark. There are two sources of error. One of them is imprecision of obtaining marker pose. This imprecision is usually given by resolution of the camera - the edge separating object from background can be detected with error of 1 pixel in the best case. Another source of error is noise - the landmarks detected by the detector that do not exist or exist for a short period of time like shadows, thin branches of trees or edges in grass that can be re-observed with just a very low probability. Another problem is relatively low rate of observations per unit of time. Improving the precision and observation rate of landmark detector could bring a significant improvement of SLAM performance.

Indepence on external systems. The entire solution was designed with independence on external systems in mind. The visual compass uses odometry and camera as input for computation of rotation about Z axis. Both sensors are present on the robot and use only informations obtained directly from the robot itself about the environment. Moreover they are both passive - they do not transmit any signal that can be observed by external observer. This feature is important for some aplications including military usage.

The entire SLAM solution add another camera, encoders on manipulator and laser rangefinder. None of the sensors rely on external systems. The only potential problem is that laser rangefinder is an active sensor. Its beam can be spotted by external observer. The informations for localization are all taken from the environment by sensors of the robot. No communication with external system is needed. To conclude this goal: The solution satisfies the requirement of independence on external system.

Improve robustness of dead-reckoning. Dead-reckoning was improved by fusing odometry data with visual compass. As figures 6.14 and 6.15 show the SLAM trajectory is estimated with smaller error with dead-reckoning incorporating visual compass. Despite the visual compass precision is not perfect it is still more reliable than estimating orientation from odometry. Experiments show that error of visual compass grow with angular change and partially also with distance travelled. Visual compass has no fixed point so without data from another localization system the error of visual compass can not be reset and comapass can not be re-calibrated. Important fact is that the

error of visual compass is smaller than rotation error measured by pure odometry as shows figure 6.13.

To conclude: The goals of the work have been met. There is still a lot of space for future development of the existing solution. Possible extensions and future development is described in following subchapter.

7.2 Future development

Several aspects of the solution can be improved to get better performance and higher precision of localization. The considered improvements are discussed in this subchapter.

7.2.1 Visual compass optimization

Visual compass based on total frame error as described above can be slow as it requires a lot of processing time to compare each pixel with each another during various frame shifts - especially with vertical compensation. All the comparisons are principially similar and they run with the same input data - base frame and actual frame so the preconditions for paralellization are met. The algorithm contains two nested loops that can be paralellized as it is drafted in algorithm 6. The outer loop iterates over all posible shifts of actual image to base image. In real implementation it would be several nested loops but they were linearized into one loop to make description more clear. The inner loop iterates over all pixels in the image frame and executes error computation. The error that is summarized over all steps is later compared to smallest error and rotation is computed according to it but the comparison can be paralellized too. Comparing the partial results returned by particular computation nodes can be done with logarithmic time cost. In the finding of minimal error the length of input is equal to number of threads which is usually relatively small so the impact of finding minimal error will not be significant compared to time needed to iterate over two loops mentioned above.

Data: $\mathcal{I}_{base}, \mathcal{I}_{actual}$
Result: ω_z

for $(shift = -N \ to \ N)$ **do**
  $Err = 0$
  for $(pixel = 1 \ to \ M)$ **do**
    $Err = Err + compute\_error(\mathcal{I}_{base}, \mathcal{I}_{actual}, shift, pixel)$
  end
end

Algorithm 6: Visual compass: Abstract algorithm.

The inner loop overlaps in data regions so paralellization of inner loop depends on shared data. Requirement for shared data limits paralellization options to multi-threaded processing on single machine due to memory bandwidth. Each thread processes particular region of base image with overlapping region of actual image. In next iteration of outer loop the overlapping regions change - this is why the data has to be shared amongst all the parallel runners. Even if the node has complete images so it doesn't need to distribute much data in each iteration the aggregation of results is happening in each iteration and therefore the process efficiency is lower.

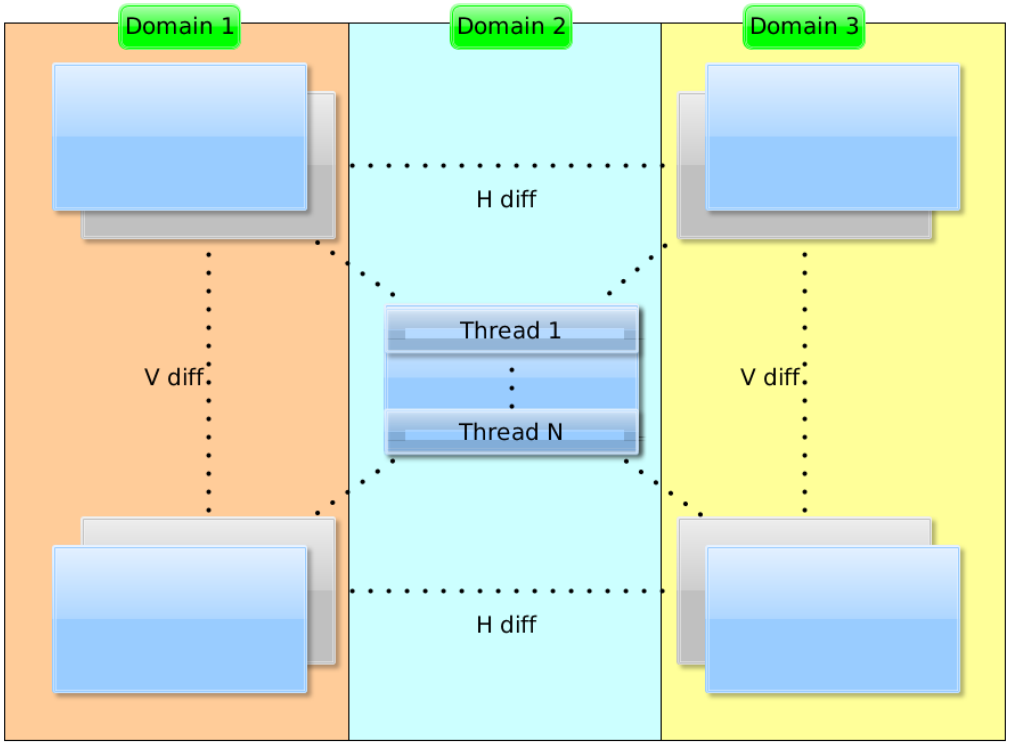

Figure 7.1: Visual compass paralellization.

On the other hand the outer loop works with entire images but it uses the same data in each iteration. Every execution is independent of the other in terms of data. Moreover a lazy aggregation is possible: Each runner finds minimum error with corresponding frame shift and once the runner processes its entire payload the data from runners can be aggregated and minimum can be easily found. These aspects of outer loop allow distributed computation of the minimum error. The outer loop can be paralellized on NUMA (Non-Unified Memory Access) architectures, grids and clusters. The payload for parallel runner is defined by range of frame shift vectors. In figure 7.1 there is depicted paralellization splitted into several domains by horizontal shift. Each runner receives base image, actual image and frame shift domain in which it execute the computation. After finishing the execution it sends the minimal error with corresponding shift to the agregator that finally finds the global minimum. The data distribution costs can be lowered by storing base image in memory of each runner and distribute only actual image with every image frame processing iteration. Processing inside the runner can be paralellized with threads if the hardware architecture supports efficient multi-threading to gain additional performance boost as is suggested in figure 7.1. Processing time is in indirect proportion to number of equal runners. Non-eqaul performance of the runners can be easily balanced by adapting size of corresponding domains.

7.2.2 Neural network for landmark recognition

Detecting the potential markers according to vertical edges is relatively simple principle but it has several drawbacks. One of them is that in noisy environments where are many trees, branches and bushes the edge detector is not very robust. Using vertical edge without

context is not perfect solution. Despite there was added a sort of context evaluation in form of computing average color and color variance around the edge there is still space for improvement.

Quality of detected markers can be improved by using context in a more sophisticated way. Instead of recognizing usefull edge according to color atributes it can be recognized according to pattern appearing in neighborhood of the edge. As there is no intuitive description of interesting object the neural network can be used for this purpose as described in [8]. Limitation of this approach is a necessity of big amount of annotated data. Annotation means a lot of images with denoted trees, building edges and possibly other interesting objects that can be used as landmarks.

Another attitude of using neural networks is based on edge importance. Neural network trained to recognize important edges can be used to detect edges of dominant objects or edges that separate foreground from background. The neural network has to be trained on large data set but recognition of important edges can be generalized. Once trained the neural network can be used to detect edges of casual objects like trees and buildings as described in [40]. This solution could be used as input for landmark detector. Detecting only important edges could be a right step towards detection of useful landmarks.

7.2.3 Feedback from SLAM to landmark detector

Re-observing the landmarks that were observed before is essential for correction of estimated pose of the robot. This makes re-observations highly desired. Actual solution of landmark detector reactively observes nearest landmarks in the view of camera. Advantage of this solution is that it makes the landmark detector independent. On the other hand the most valuable observations could be missing due to different actual orientation of the manipulator. Connecting the SLAM with landmark detector could improve frequency of re-observations and thus provide more valuable data.

There are several problems that need to be solved to make this extension work and bring added value.

Identification of landmarks to re-observe: Identification of landmarks that should be re-observed is not a trivial task. Of course the re-observation makes sense only for those landmarks that are in range of the rangefinder but there are other aspects. First of them is that the landmark may be hiding behind objects or terrain profile so it might be unobservable despite it is in range of sensors. Another problem appears in situation when there are several landmarks that can be re-observed. Choosing the best landmark is essential. In this case the closest one could be used but if the closest landmark can not be re-observed or if it was observed many times the controller should start observing other landmarks. The decision making process has to consider history and also timing of its previous actions and results of these actions.

Balance between observing new landmarks and re-observing: It is important to balance the re-observing with observing of new landmarks. Too intensive re-observing may leed to reduction of observed landmarks and loss of precision due to it. Moreover there is always a potential risk of re-observing noise that appears due to imprecise rangefinder pointing. Loosing new observations due to useless attempt to re-observe noise is the worst case.

Interconnection with manipulator control: It is not simple to provide the landmark detector a guidance to the previously observed landmarks and at the same time pro-

vide it enough autonomity to control the observation process. One of the solutions could be defining area in the orientation of the manipulator in which the marker corresponding landmark should be found. The area can be limited by uncertainity of the landmark. If the marker is found the re-observation process can be initiated. So far the idea is an abstract proposal and requires proper testing in the future.

Bibliography

[1] Alrajeh, N. A., Bashir, M. , Shams, B. : Localization Techniques in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*. 2013.

[2] Bannon, M. , Kaputa, F. : The Newton-Laplace Equation & Speed of Sound. 2014.

[3] Bohren, J. , Saito, I. I. Y.: SMACH. <https://wiki.ros.org/smach>. 2018.

[4] Bulusu, N., Heidemann, J. and Estrin, D. : GPS-less Low Cost Outdoor Localization For Very Small Devices. *IEEE Pervasive Computing Magazine*. vol. 7, no. 5. 2000: pp. 28–34.

[5] Çulha, U., Turan, B.: Particle Filter Based Fast Simultaneous Localization and Mapping. Technical report.

[6] Chung, M. K., Alexander, A. L., Lu, T.: Connection Probability in Diffusion Tensor Imaging via Anisotropic Gaussian Kernel Smoothing. In *9th Annual Meeting of the Organization for Human Brain Mapping*. 2003.

[7] Collective of authors - National Instruments: What to Expect from a Stereo Vision System. 2013. url: http://zone.ni.com/reference/en-XX/help/372916P-01/nivisionconceptsdita/guid-10d358bd-3dcd-4ccd-a73c-672e48aed39a/.

[8] Egmont-Petersen, M. , de Ridder, D. , Handels, H. : Pattern Recognition: Image processing with neural networks. vol. 35, No. 10. 2002: pp. 2279–2301.

[9] Eisfeller, B., Ameres, G. , Kropp, V. and Sanroma, D. : Performance of GPS, GLONASS and Galileo. *Photogrammetric Week '07*. 2007: pp. 185–200. doi:10.1109/MRA.2012.2205651.

[10] Feledy, C., Luttenberger, M. S.: A State of the Art Map of the AGVS Technology and aGuideline for How and Where to Use It. Technical report. 2018.

[11] Giuffrida, F. , Morasso, P. , Vercelli, G. and Zaccaria, R. : An Optical Localization System for Mobile Robots Based on Active Beacons. *IFAC Intelligent Components for Autonomous and Semi-Autonomous Vehicles, Toulouse, France*. 1995.

[12] Hiemstra, P., Nederveen, A. : Monte Carlo Localization. Technical report. 2007.

[13] Kaur, H., Kaur, L.: Performance Comparison of Different Feature Detection Methods with Gabor Filter. *International Journal of Science and Research (IJSR)*. vol. 3(5). 2014: pp. 1879–1886. iSSN (Online): 2319-7064 Impact Factor (2012): 3.358.

[14] Lee, W. Ch., Hung, F. H., Tsang, K. F. and Wu, Ch. W. and Chi, H. R. and Chui, K. T. and Lau, W. H. : High Accuracy Localization of Long Term Evolution Based on a New Multiple Carrier Noise Model. *Sensors (Basel)*. 2014 Dec. 22613-22618.

[15] Ma, L., Chen, Z. , Li, Y., Zhang, D. , Li, J., Chapman, M. A. : Multispectral airborne laser scanning point-clouds for land cover classification using convolutional neural networks. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W13. ISPRS Geospatial Week, Enschede, The Netherlands. 2019.

[16] Malik, Y., Khaliq, K. A., Abdulrazak, B. and Tariq, U.: Mobile node localization in cellular networks. *International Journal of Wireless & Mobile Networks (IJWMN)*. vol. 3, no. 6. 2011: pp. 91–100.

[17] Matas, J., Galambos, C. and Kittler, J. : Progressive Probabilistic Hough Transform. Technical report. 1998. czech Technical University, Karlovo náměstí 13, Praha, Czech Republic.

[18] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B. : FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. *Proceedings of IJCAI 2003*. 2003.

[19] Nitsche, M., Krajník, T., Čížek, P. and Mejail, M. and Duckett, T. : WhyCon: An Efficent, Marker-based Localization System. *IROSOSAR*. 2015.

[20] Rueß, D. , Luber, A. , Manthey, K. and Reulke, R. : Accuracy evaluation of stereo camera system with generic camera models. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXIX-B5, 2012 XXII ISPRS Congress, 25 Augus - 01 September 2012, Melbourne, Australia*. 2012. german Aerospace Center, Rutherfordstraße 2,12489 Berlin-Adlershof, Germany.

[21] Singhala, P. , Shah, D. N. , Patel, B. : Temperature Control using Fuzzy Logic. *International Journal of Instrumentation and Control Systems (IJICS)*. vol. 4, no. 1. January 2014: pp. 1–10.

[22] van de Logt, E.H.W. : PID Controller Calculus for HERMS home-brewing system. Technical report. PID Controller Calculus, V3.20. 2011.

[23] Zhang, Z. ,Dong, Y., Ni, F. and Jin, M. and Liu, H. : A Method for Measurement of Absolute Angular Position and Application in a Novel Electromagnetic Encoder System. Technical report. School of Software Engineering, East China Normal University and School of Mechatronics Engineering,Harbin Institute of Technology, Heilongjiang 150001, China. April 2015.

[24] Alatis, M. B.; Hancke, G. P.: Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter. *Sensors*. vol. 17. 2017. s17102164.

[25] Anderson, P.; Hengst, B.: Fast Monocular Visual Compass for a Computationally Limited Robot. Technical report. 2013.

[26] Armen, D. G. B.: Hall Effect Experiment. Technical report. Department of Physics and Astronomy, The University of Tennessee. 401 Nielsen Physics Building, Knoxville, Tennessee 37996-1200. 2007.

[27] Bala, A.: A Review of Selection strategies in Genetic Algorithm. *International Journal of Advance Research in Computer Science and Management Studies*. vol. 5. 2017. iSSN: 2321-7782.

[28] Berg, N.: AMCL ROS package. <http://wiki.ros.org/amcl>. cit. 2016-1-1.

[29] Blewitt, G.: Basics of the GPS Technique: Observation Equations§. Technical report. Department of Geomatics, University of Newcastle. Newcastle upon Tyne, NE1 7RU, United Kingdom. 1997.

[30] Bouet, M.; dos Santos, A. L.: RFID tags: Positioning principles and localization techniques. In *2008 1st IFIP Wireless Days*. Nov 2008. ISSN 2156-9711. pp. 1–5. doi:10.1109/WD.2008.4812905.

[31] Bouguerra, A. ,Andreasson, H. ,Lilienthal, A. and øAstrand, B. and Rögnvaldsson, T.: An Autonmous Robotic System for Load Transportation. *ETFA09*. 2009.

[32] Chechi, D.; Kundu, T.; Kaur, P.: The RFID technology and its applications: A review. *International Journal of Electronics, Communication & Instrumentation Engineering Research and Development (IJECIERD)*. vol. 2. 09 2012: pp. 109–120.

[33] Collective of authors - Opensource initiative: MORSE. <https://www.openrobots.org/morse/doc/stable/what_is_morse.html>. 2018.

[34] Collective of authors - Riegl: RIEGL LD05 Series - General Description and Data Interfaces. Technical report. 2016.

[35] Collective of authors- Riegl: Digital laser distance meter LD05e-A10 - Technical data. Technical report. 2016.

[36] Cook, A.: *European Air Traffic Management: Principles, Practice, and Research*. Ashgate Publishing, Ltd.. 2007. ISBN 0754672956.

[37] Cook, J. D.: Three algorithms for converting color to grayscale. 2009. url: https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/.

[38] Deb, B., Bhatnagar, S., and Nath, B.: A topology of discovery algorith for sensor networks with application to network management (short paper). Technical report.

[39] Dellaerty, F., Foxy, D., Burgardz W. and Thrun, S.: Monte Carlo Localization for Mobile Robots. *Georgia Tech Library*. 1999.

[40] El-Sayed, M. and Estaitia, Y. A. and Khafagy, M. A.: Edge Detection Using Convolutional Neural Network. *International Journal of Advanced Computer Science and Applications*. vol. 4. 11 2013: pp. 11–17. doi:10.14569/IJACSA.2013.041003.

[41] Endres, F.; Hess, J.; Engelhard, N.; et al.: An evaluation of the RGB-D SLAM system. *Robotics and Automation (ICRA), 2012 IEEE International Conference on Robotics and Automation*. 2012. isbn: 978-1-4673-1405-3.

[42] Galili, T.; Meilijson, I.: An Example of an Improvable Rao-Blackwell Improvement, Inefficient Maximum Likelihood Estimator, and Unbiased Generalized Bayes Estimator. *The American Statistician*. vol. 70. 10 2015. doi:10.1080/00031305.2015.1100683.

[43] Galvez, E. J.: Gaussian Beams. Technical report. 2009.

[44] Goldenshluger, A., Zeevi, A.: The Hough Transform Estimator. In *2000 AMS Subject Classification (Primary): 62F12; 62F35; 68T45*. October 2003.

[45] Gustafsson, F.: Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*. vol. 25, no. 7. July 2010: pp. 53–82. ISSN 0885-8985. doi:10.1109/MAES.2010.5546308.

[46] Hanlon, B., Larget, B.: Probability and Conditional Probability. Technical report. Department of Statistics, University of Wisconsin - Madison. 2011.

[47] Hoiem, D.: How Kinect works - Computational photography. Technical report. University of Illinois. December 2011.

[48] Hoiem, D.: Projective Geometry and Camera Models. Technical report. University of Illinois. January 2011.

[49] Hong, K.: Image Edge Detection : Sobel and Laplacian. 2016. url: http://www.bogotobogo.com/python/ OpenCV_Python /python_opencv3_Image_Gradient_Sobel_Laplacian_Derivatives_Edge_Detection.php.

[50] Jiuqiang Xu, F. L., Wei Liu; Zhang, Y.; Wang, C.: Distance Measurement Model Based on RSSI in WSN. *Wireless Sensor Network*. , no. 2. 2010: pp. 606–611.

[51] Kim, R. M. C. H., J.N.; Yang, Y. S.: Anatomy Calibration of Inertial Measurement Unit Using a Principle Component Analysis. *International Journal of Bio-Science and Bio-Technology*. vol. 5, no. 6. 2013: pp. 181–190. iSSN: 2233-7849 IJBSBT.

[52] Kimbrell, J.: Fundamentals of Industrial Encoder Sensing Technologies, Motion Detection Theory and Methods, and Signal Output Styles. Technical report. AutomationDirect.com, Inc.. April 2013.

[53] Labrosse, F.: Appearance-based heading estimation: The visual compass. Technical report. Department of Computer Science,University of Wales. Aberystwyth,Ceredigion SY23 3DB,U.K.. 2006.

[54] Labrosse, F.: The visual compass: performance and limitations of an appearance-based method. *Journal of Field Robotics*. vol. 23(10). 2006: pp. 913–941.

[55] Labrosse, F.: Short and long-range visual navigation using warped panoramic images. *Robotics and Autonomous Systems*. vol. 55(9). 2007: pp. 675–684.

[56] Laoudias, C.: Cellular Network Localization: Current Challenges and Future Directions. *IEEE ICC Workshop on ANLN*. May 25, 2017.

[57] Lee, J.: ROS Documentation: global_planner. <http://wiki.ros.org/global_planner>. cit. 2016-4-3., updated 2014-10-21.

[58] Li, L.: Time-of-Flight Camera - An Introduction. Technical report. Texas Instruments. April 2014. sLOA190B - January 2014 Revised May 2014.

[59] Linga, S., Roy, B., Asada, H. and Rus, D. : An Optical External Localization System and Applications to Indoor Tracking. *IROS*. 2008.

[60] Liu, J.; Deng, Z.: Information Fusion Kalman Filter for Two-Sensor System with Time-Delayed Measurements. *Procedia Engineering*. vol. 29. 2012: pp. 630 – 636. ISSN 1877-7058. doi:https://doi.org/10.1016/j.proeng.2012.01.015. 2012 International Workshop on Information and Electronics Engineering.
Retrieved from:
<http://www.sciencedirect.com/science/article/pii/S1877705812000252>

[61] Liudvinavičius, L., Lingaitis, L. P. : Electrodynamic braking in high-speed rail transport. *Transport*. vol. 3, no. 22. 2007: pp. 178–186.

[62] Malu, S. K., Majumdar, J. : Kinematics, Localization and Control of Differential Drive Mobile Robot. In *Global Journal of Researches in Engineering: Robotics & Nanotech*, vol. 14. Global Journals Inc. (USA). 2014. ISSN 0975-5861.

[63] Merriaux, P.; Dupuis, Y.; Boutteau, R.; et al.: A Study of Vicon System Positioning Performance. *Sensors*. vol. 17. 07 2017: page 1591. doi:10.3390/s17071591.

[64] Morgan-Owen, G. J.; Johnston, G. T.: Differential GPS positioning. *Electronics Communication Engineering Journal*. vol. 7, no. 1. Feb 1995: pp. 11–21. ISSN 0954-0695. doi:10.1049/ecej:19950104.

[65] Naveed, S., Ko, N. Y.: Analysis of Indoor Robot Localization Using Ultrasonic Sensors. *International Journal of Fuzzy Logic and Intelligent Systems*. vol. 14, no. 1. March 2014: pp. 41–48.

[66] Neubeck, A.; Gool, L. V.: Efficient Non-Maximum Suppression. Technical report. 2014.

[67] Nguyen H.Q.P., Kang HJ., Suh YS. and Ro YS.: INS/GPS Integration System with DCM Based Orientation Measurement. In *Huang DS., Jo KH., Lee HH., Kang HJ. and Bevilacqua V. (eds) Emerging Intelligent Computing Technology and Applications. ICIC 2009. Lecture Notes in Computer Science*, vol. 5754. 2009. springer, Berlin, Heidelberg.

[68] Phung, SL. and Bouzerdoum A.: Detecting People in Images: An Edge Density Approach. In *International Conference on Acoustics, Speech and Signal Processing, 2007 (ICASSP 2007), Honolulu, Hawaii, USA*. IEEE. 15-20 April, 2007.

[69] Pierlot, V.; Droogenbroeck, M. V.: 18 Triangulation Algorithms for 2D Positioning (also known as the Resection Problem): benchmarking, software, source code in C, and documentation. Technical report.

[70] Pierlot, V.; Droogenbroeck, M. V.: A New Three Object Triangulation Algorithm for Mobile Robot Positioning. Technical report.

[71] Pomerleau, F.; Colas, F.; Siegwart, R.; et al.: Comparing ICP variants on real-world data sets. *Autonomous Robots*. vol. 34, no. 3. April 2013: pp. 133–148.

[72] Quigley, M.; Gerkey, B.; Conley, K.; et al.: ROS: an open-source Robot Operating System. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. Kobe, Japan. May 2009.

[73] Rozman, J. , Orság, F. : Course of Robotics - lecture slides, Brno University of Technology. 2018.

[74] Scaramuzza, D.; Siegwart, R.: Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. In *IEEE TRANSACTIONS ON ROBOTICS*, vol. 24. Octoboer 2008.

[75] Soycan, M. , Soucan, A.: Surface modeling for GPS-levelling geoid determination. Technical report.

[76] Stachniss, C.: Introduction to Robot Mapping. Technical report. University of Freiburg. 2012.

[77] Surrécio, A., Nunes, U. and Araújo, R.: Fusion of Odometry with Magnetic Sensors Using Kalman Filters and Augmented System Models for Mobile Robot Navigation. In *IEEE ISIE 2005*. Institute of Systems and Robotics, University of Coimbra. June 20-23 2005. dubrovnik, Croatia.

[78] Thrun, S.; Burgard, W.; Fox, D.: *Probabilistic robotics*. MIT Press, Cambridge, Massachusetts, England. 2005. ISBN 0-262-20162-3.

[79] Tucson, R. B.: Global Positioning System: The Mathematics of GPS Receivers. Technical report.

[80] Yang, Y.; Xu, Y.; Li, J.; et al.: Progress and performance evaluation of BeiDou global navigation satellite system: Data analysis based on BDS-3 demonstration system. *Science China Earth Sciences*. vol. 61, no. 5. 2018: pp. 614–624. iSSN: 1869-1897.