CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

# Bachelor's Thesis

Robert Pěnička

## The Acquisition and Representation of Spatial Knowledge in Project Replicator

**Department of Cybernetics**

Thesis supervisor: **Ing. Tomáš Krajník, Ph.D.**

Prague, 2013

**České vysoké učení technické v Praze**
**Fakulta elektrotechnická**

**Katedra kybernetiky**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

**Student:** Robert Pěnička

**Studijní program:** Kybernetika a robotika (bakalářský)

**Obor:** Robotika

**Název tématu:** Získávání a reprezentace prostorových znalostí v projektu Replicator

**Pokyny pro vypracování:**

1. Seznamte se s metodami lokalizace a mapování používanými v mobilní robotice.
2. Seznamte se s modulárními roboty využívanými v rámci projektu Replicator.
3. Navrhněte, implementujte a experimentálně verifikujte navrhnutou metodu.

**Seznam odborné literatury:**


[1] Thrun Sebastian - Probabilistic Robotics - The MIT Press, 2005
[2] Paul Levi - Symbiotic Multi-Robot Organisms – Springer 2010


**Vedoucí bakalářské práce:** Ing. Tomáš Krajník, Ph.D.

**Platnost zadání:** do konce zimního semestru 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
**vedoucí katedry**

L.S.

prof. Ing. Pavel Ripka, CSc.
**děkan**

V Praze dne 10. 1. 2013

**Czech Technical University in Prague**
**Faculty of Electrical Engineering**

**Department of Cybernetics**

# BACHELOR PROJECT ASSIGNMENT

**Student:** Robert Pěnička

**Study programme:** Cybernetics and Robotics

**Specialisation:** Robotics

**Title of Bachelor Project:** The Acquisition and Representation of Space Knowledge in Project Replicator

**Guidelines:**

1. Get acquainted with localization and mapping methods used in mobile robotics.
2. Get acquainted with modular robots used by the project Replicator.
3. Design, implement and experimentally verify the suggested method.

**Bibliography/Sources:**

[1] Thrun Sebastian - Probabilistic Robotics - The MIT Press, 2005
[2] Paul Levi - Symbiotic Multi-Robot Organisms – Springer 2010

**Bachelor Project Supervisor:** Ing. Tomáš Krajník, Ph.D.

**Valid until:** the end of the winter semester of academic year 2013/2014

L.S.

prof. Ing. Vladimír Mařík, DrSc.
**Head of Department**

prof. Ing. Pavel Ripka, CSc.
**Dean**

Prague, January 10, 2013

## Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o održování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24.5.2013 ..... .....

## Acknowledgements

I would like to thank my supervisor, Tomáš Krajník, for his excellent leadership and the opportunity to deal with such an interesting project. I also thank my family including my girlfriend for their tolerance and support throughout writing this thesis.

## Abstract

This thesis presents simultaneous localization and mapping algorithm for a small mobile robot. The designed solution enables to determine a mobile robot's and landmark positions using self-localization and camera based landmark detection. As landmarks, roundel patterns are used that are positioned statically on obstacles or borders of the used arena. One of the key features is the suppression of odometry error in long term use of robot. As hardware platform, modular robots were used from the Replicator and Symbrion project, both funded by the European Commission. Proposed method uses the extended Kalman filter for merging data from odometry and camera landmark detection for building landmark-based map.



## Abstrakt

Tato práce prezentuje software určený k současné lokalizaci a mapování pro malého mobilního robota. Navrhované řešení je schopno určovat polohu mobilního robota a orientačních bodů fúzí odometrie a kamerové detekce orientačních bodů. Jako orientační body jsou použity kruhovité terčíky, které jsou pevně umístěny na překážkách nebo na hranicích operačního prostoru robotu. Jednou z hlavních schopností je potlačení chyby odometrie při dlouhodobém užívání robota. Jako hardwarová platforma byly použity modulární roboty z projektu Replicator a Symbrion financované Evropskou komisí. Navrhovaná metoda používá rozšířený Kalmanův filtr pro nelineární systémy. Fůzí dat z odometrie a kamerové detekce orientačních bodů je schopna vytvářet mapu orientačních bodů.

# Contents

# List of Figures

# 1 Introduction

Acquiring information on the surrounding environment is one of the crucial elements in mobile robotics. Every mobile robot must interact with its surroundings in order to reach its goals. This interaction consists of path planning for collision avoidance, recognition of goals in space or simple movement. Due to all these interactions, a robot has to collect information its surroundings. In the context of continual information acquisition on surroundings, we talk about Map Building or Mapping.
For precise map building, the exact knowledge of mobile robot position is required. If a robot has no self-localization system, then the resulting map can not grow out of the initial position of the robot. A robot in a previously unknown environment without localization infrastructure must rely solely on self-localization and must build its own map using the positions from the self-localization system. On the other hand, by localizing the robot inside the mapped area, both map and the robot position can be determined more precisely.
This thesis uses the robot hardware platform from the Symbrion and Replicator projects. Since these projects are focused on the cooperation of many robot modules independent or in connected structures, the spatial knowledge is required even more. Modular robotics that is studied in these projects uses an outstanding number of small modules equipped with docking interfaces. Every module can connect to each other and make big complex organisms. The main advantages of such connection and cooperation of modules are the ability to adapt to a wide specter of environments and also the high collective computation resources.
In the thesis we introduce the solution of simultaneous localization and mapping generally known as SLAM. The implemented algorithm that can run on all robotic modules is able to map man-made roundel landmarks as well as to localize itself according to previously mapped landmarks. The solution of SLAM is based on the Extended Kalman filter, that uses data from odometry measurements and camera landmark detection. This method is intended for cases when robots are separated, but with additional geometry transformation of camera measurements, it could be used for the localization of robots in connected organisms as well.
Both robot self localization and landmark camera measurements involve measurement uncertainties that reflect in map precision. These uncertainties in the map are reduced by repeated detections of the same landmarks from different or even the same locations and increased by unknown effect of robot movement on it's position. Dead reckoning, used for self-localization of robot modules, involves uncertainty that rises with the traveled path and is reduced by the localization of the robot in the already mapped area.
In this thesis we will firstly describe the localization and mapping methods that are used in mobile robotics. Then we will introduce the Symbrion and Replicator projects as well as modular robotics. The following two chapters analyze Dead reckoning self-localization and camera landmark detection. In the last two chapters is introduced implemented SLAM algorithm and experimentally verified its capabilities.

# 2 Mapping and localization in mobile robotics

## 2.1 Localization

Localization is the problem of determining a robot position inside an environment. It is one of the fundamental problems in mobile robotics. There is a wide range of localization methods that are used nowadays. The key factor that influences the selection of a methods is the target environment of operation. We distinguish static environments, where only the robot movement is possible, and dynamic environments [26], where also movement of other objects must be taken into account. The number of robots influences the selection of a localization method too. A multi-robot localization can be solved identically as a single-robot localization. With the detection of other robots, there is an opportunity to improve the location of many robots from the others' positions and vice versa [39].
For a better and more accurate localization, some of the following methods are commonly used, best with the combination of incremental and absolute localization, while their results are merged according to their credibility.

### 2.1.1 Incremental localization

Incremental localization uses the known initial position, or this position is set as the origin of the robot localization. Every little robot movement of the robot is evaluated and reflected in its new position. This new position is then used in the subsequent movements, so the robot position is incrementally determined with every motion.
The main problem of this method comes out of its principle. If every new position is determined from the previous ones, then any small error in the previous positions will also occur in the future positions. This phenomenon may be reduced by accurate sensors, but in a long term use, the incremental localization needs a correction of this cumulative error.
Modifications of incremental localization are mainly in the measurement of the robot movement. In this thesis, a method was used, which determines robot motion from its speeds setting on the motors and the elapsed time during the same speed. This simple way is applicable if the motor speed setting is nearly linear and stable, but without a proper feedback, the wheel speeds always differ a little from the speeds that are set on the motors. This method belongs to a class of Dead reckoning.
The most used feedback mechanism for robots with wheels or tracks are motor encoders. These sensors incrementally count wheel rotation and give us a better estimate of the traveled path than only the speed setting on the motors. This method is referred to as wheel odometry [38] and is widely used for its simplicity of implementation. Despite the fact that odometry is the name of the solution with motor encoders, we use it also for self localization in this thesis due to the future change of localization method from dead reckoning to odometry.
Other methods rely on different types of sensors that somehow the measure robot motion.

These sensors include accelerometers[23] and speed sensors[30] of many kind. By integrating the data from these sensors, we can determine position changes.

### 2.1.2 Absolute localization

Absolute localization could determine the robot position inside a covered area with its own reference frame established. This localization can also determine the initial position of the robot (that usually differs from zero). The method determines position inside a covered area usually with the same accuracy. In contrast to the incremental localization the accuracy is usually lower. On the other hand, this method does not integrate the position error, and every position measurement is independent of the previous one.
The present solutions of the absolute localization include the worldwide used GPS[6], GLONASS[5] and the future European Galileo[4] global navigation satellite systems (GNSS). All these systems use the trilateration for determining a position on the Earth. The accuracy of these systems for general purposes is from 1 to 3 meters depending on the used system.
For indoor applications, in ideal conditions, systems with statically positioned transmitters are used. Every robot is then equipped with a receiver that computes its position from the distances to transmitters by triangulation [15] or trilateration. This kind of system is also used in the Replicator and Symbrion projects with the approximate accuracy of tens centimeters. The opposite approach with transmitters on robot and receivers statically positioned around is also possible.
Another solution which can be defined as absolute is map base localization. This type uses a previously created map of space and compares it with the robot sense of surroundings. Individual map-localizations differ in the type of sensors that sense the surroundings and the type of map (2.2). With the growth of computing resources of mobile robots, the use of camera for map based localization also grows. Other methods use for example laser distance measurements or a sonar to determine obstacles around the robot, which are then compared with a global map.

## 2.2 Mapping

Mapping is a process of generating a map. Map usage depends on its contents, but generally in mobile robotics it is used for motion planning, localization purposes and sharing with other robots. Especially in robot swarms and modular robotics is spacial knowledge very useful for collision avoidance between two robots.
Maps in robotics are usually represented by sets, vectors or matrices of mapped features. In some cases, a map may also consist of all measured data from the sensors. However, this type of map is difficult for management and its usability is low. The extraction of salient sensory patterns is more common, for example landmarks inside the sensor(in our case a camera) vision.

According to map representation and stored knowledge inside the map, we can distinguish two types of maps - Feature-based maps and Location-based maps[37].

### 2.2.1 Feature-based maps

A feature based map is a vector or set $m = \{m_1 \dots m_N\}$ that consists of N mapped features. Each feature is represented by its Cartesian location or optionally other properties of the feature. This type of map contains information about the occupancy of the mapped area, but from the map we can-not distinguish whether locations not presented in the map are unmapped or unoccupied.
Feature based maps include the already mentioned Sensor maps [20] that simply contain all sensor data. Sensor data that are used in such maps are in most cases distance measurements, taken by laser sensors, that determine distances to surrounding objects in every position of the robot. These maps usually need future processing and after that they can be used in future operations in the same area.
Another type of such map representation is a Landmark map [32]. Landmark is a key pattern or an easily differentiable object in the environment. In indoor environments, landmarks may include wall corners, tables or other easily detected objects. In some cases, landmarks may be also man-made for better and faster recognition. If the man-made landmarks are active, then we call them beacon.
A geometrical map is the last widely used Feature-based map type. These maps use geometrical primitives, usually lines, to represent the surrounding area. The resulting map is then a set of lines that represent borders of obstacles. These maps are very useful for path planning, as landmark maps do not necessarily provide information about obstacles, while sensor maps are too heavy for computation [7].

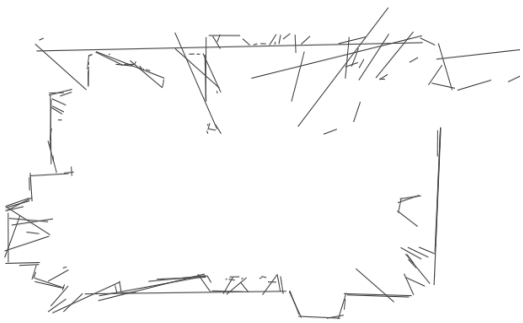

Figure 1: 2D geometrical map of room [7]

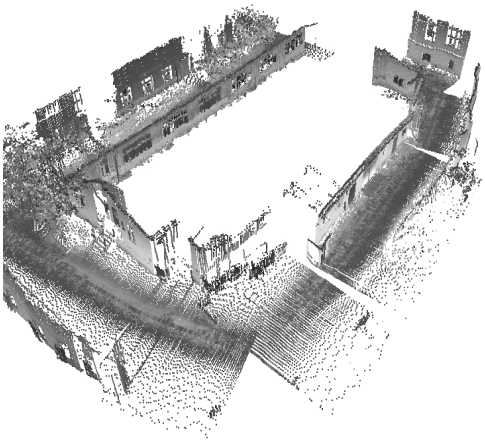

Figure 2: 3D Sensor map(Point Cloud) with courtesy of [20]

### 2.2.2 Location-based maps

In contrast to Feature-based maps, Location-based maps consist of mapped features indexed by specific location. For planar maps, one map element is denoted by $m_{x,y}$ [37]. The representation of the map is a matrix that enlarges quickly with the enlarging mapped area. These maps provides knowledge not only of the occupancy of the mapped space, but also of the free area. The problem of this representation is usually large memory usage for a relatively small area, which prove even more in 3D mapping.
The main map type among Location-based maps is a Grid map. In these maps, the environment is represented by a grid with regular cells. These cells may store the probability of occupancy [34] or can contain some physical property of this area [25]. Besides storage demands, grid maps must split the environment into small parts. With parts too big, the resulting map has a lower resolution, but the size of grid map has cubic(for 3D map) increase with finer granularity [34, 12].

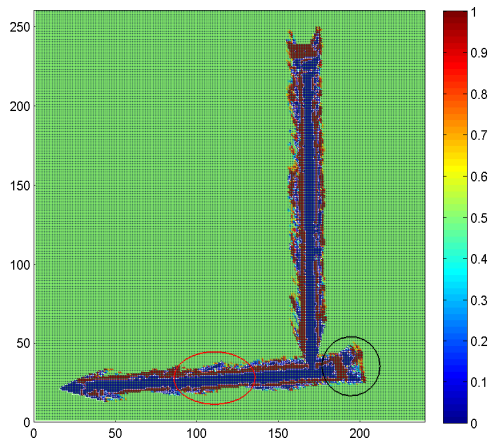


Figure 3: 2D Grid Map with probability of occupancy [34]

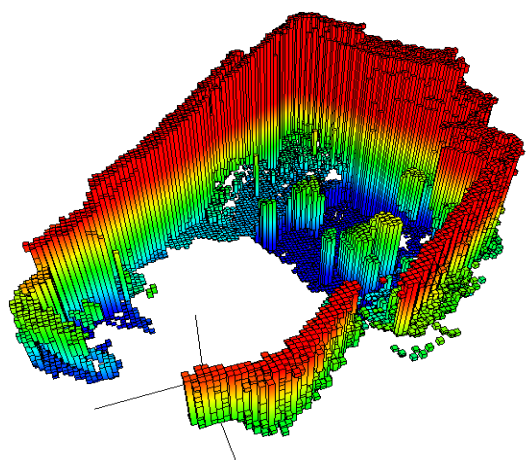

Figure 4: 3D Grid Map [12]

## 2.3 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a problem of incremental building of a map, while the robot position is simultaneously determined from both self-localization and the map.
The beginning of probabilistic SLAM dates back to the 1986 IEEE Robotics and Automation Conference held in San Francisco. An important element of the first works on probabilistic SLAM was the proof of correlation between different landmarks that grow even more with successive observations. This factor enables successful mapping during bad landmark observation or inaccurate robot localization [13, 8].
All SLAM solutions consist of self localization (see chapter 2.1), map-base localization from

a generated map and mapping (see chapter 2.2). Nowadays there are two main computational solutions of the SLAM problem [13]. The first one uses the Extended Kalman filter (EKF-SLAM)[37]. The other solution uses particle filters [3, 36].

### 2.3.1 Particle filters

The Particle filters use Monte Carlo sampling methods for state-space model estimation. The actual state is evaluated in discrete time intervals from previous state by using the Markov chains. Present the most used particle filter SLAM algorithm is a FastSLAM [28]. The FastSLAM is the reaction to time complexity of EKF-SLAM discussed in next section(2.3.2), which was the first SLAM algorithm. EKF-based SLAM solutions have a covariance matrix that has $K^2$ elements with $K$ mapped landmarks. The maintenance of such a matrix with thousands of mapped landmarks is really slow and requires $O(K^2)$ for each landmark observation.
FastSLAM decomposes the SLAM problem to path estimator, realized by particle filter, and landmark position estimators realized by Kalman filters. For each landmark, a separate Kalman filter is used. This decomposition led to a significant reduction of filtering time. On the other hand, during the observation of one landmark, the positions of others are unchanged.
The particle filter used for position estimator is similar to Monte Carlo Localization [37]. With $M$ particles that represents the most recent part of the robot path, we incrementally determine a probabilistic guess of the robot position.
The landmark location estimator uses $K$ Kalman filters for each particle. The total number of Kalman filters is thus $KM$. In each Kalman fiter, the most likely position of landmark and particle(robot path part) is generated and the results are weighted and merged to the map and the robot position.
FastSLAM requires time $O(M \cdot log(K))$ for each iteration.

### 2.3.2 EKF-SLAM

The Extended Kalman filter (EKF) is based on the Kalman filter introduced by Rudolf E. Kálmán in 1960 [1]. In contrast to the Kalman filter, the extended version solves filtering of non-linear systems. As odometry equations and landmark geometry equations are not linear, using of EKF is required. Bases of EKF-SLAM are probability of robot motion (1) and landmark observation model (2) [13].

$$P\left(x_k \mid x_{k-1}, u_k\right) \Longleftrightarrow x_k = f\left(x_{k-1}, u_k\right) + w_k \tag{1}$$

Where $x_k$ is a state vector consisting of both robot position and a map. Function $f$ describes kinematics of the robot with input $u_k$. Zero mean motion noise is $w_k$.

$$P\left(z_k \mid x_{k-1}, m\right) \Longleftrightarrow z(k) = h\left(x_{k-1}, m\right) + v_k \tag{2}$$

In the equation of observation model (2), function $h$ describes the geometry of observation according to the state vector and observation measurement $m$. Observation also suffers from measurement error $v_k$.
These two equations with conditional probabilities defines the best estimations for the robot and mapped features positions according to their error models (probabilities).
The whole algorithm is described by the following function(1) [37]. The most important variables in this function are state vector $\mu_k$ and associated covariance matrix $P_k$. There are both the robot and the mapped landmarks positions in the state vector. The robot position as well as the landmarks positions are represented by its Cartesian coordinates $(x, y)$ and also by their rotation $\phi$.

**Input**: ( $\mu_{k-1}$ , $P_{k-1}$ , $u_k$ , $dr$ ) – data from previous run of algorithm, $\mu_{k-1}$ is state vector consist of robot and landmarks positions, $P_{k-1}$ is covariance matrix of state vector, $u_k$ is camera landmark measurement and $dr$ is change of robot position based on Dead reckoning.

$\overline{\mu}_k = \mu_{k-1} + \begin{pmatrix} dr \\ \mathbf{0} \end{pmatrix}$ `// predict state vector from robot odometry change`

$\overline{P}_k = P_{k-1} + \begin{pmatrix} P_{od} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ `// add covariance matrix of robot movement` *find corresponding landmark in map to observed landmark*;

**if** *Landmark* $u_k$ *not yet observed* **then**

$\overline{\mu}_k = \begin{pmatrix} \overline{\mu}_k \\ g\left(u_k, \mu_k\right) \end{pmatrix}$ `// add new landmark to end of state vector`

$\overline{P}_{mr,new} = G_r \left(\overline{P}_{rr}\ \overline{P}_{mr}^T\right)$ `// cross-variance between robot and new landmark`

$\overline{P}_{mm,new} = G_r \overline{P}_{rr} G_r^T + G_y Q G_y^T$ `// covariance of new landmark position`

$\overline{P}_k = \begin{pmatrix} \overline{P}_k & \overline{P}_{mr,new}^T \\ \overline{P}_{mr,new} & \overline{P}_{mm,new} \end{pmatrix}$ `// enlarge covariance matrix for new landmark`

**end**

$z_k = u_k - h(\overline{\mu}_k)$ `// determine innovation` $z_k$

$K = \overline{P}_k H_k^T \left(H_k \overline{P}_k H_k^T + Q\right)$ `// determine Kalman gain`

$\mu_k = \overline{\mu}_k + K z_k$ `// calculate new state vector`

$P_k = (I - K H_k)\, \overline{P}_k$ `// calculate associated new covariance matrix`

**Algorithm 1:** EKF-SLMA

The first two equations in the algorithm are reactions to the robots movement. The new state vector is predicted with respect to the robots movement. The uncertainty of the robot position after this movement is reflected in the prediction covariance matrix. For accurate determination of the robot odometry change $dr$ and the covariance of odometry change $P_{od}$, see chapter Dead reckoning (4).
In the next part, the new landmark position is added to the state vector if the landmark had not yet been observed. The covariance matrix is also adapted to the new landmark.
The last two equations in the algorithm (1) determine the new state vector and the covariance matrix from the Kalman gain $K$ and the difference between the landmark obser-

vation and its map position $z_k$. In the calculation of the Kalman gain, matrix $Q$ is also used, which is the covariance matrix of camera measurement(experimentally determined in chapter Landmark detection 5).
As was already mentioned in section 2.3.1, the time complexity of EKF-SLAM is known as $O(K^2)$ with $K$ mapped landmarks. Detailed information and equations of the EKF-SLAM are described in the chapter about its implementation 6. In this section, all variables in the algorithm are described as well.

# 3 Modular robotics

Modular robotics is part of mobile robotics that focuses on developing small, interconnectable robotic units called "modules" [29]. Every module is equipped with docking interface in order to join other modules and create large structures together.
The advantages of modular robots are adaptability, reliability and potentially lowered costs. Adaptability relates to responses to a changing environment. For example, if a robot wants to overcome an obstacle, the whole structure can reconfigure to a form that can handle it. Robots can also create a rolling structure for faster downhill movement. Reliability refers mainly to the ability to change faulty components and stay operational in the long term. Modular robots also offer the opportunity to produce a large number of identical modules and to reduce the total cost by mass production.
There are many types of modular robots, but all of them have the common ability to somehow connect to other modules. We can distinguish two basic approaches to modular robots' geometry, modularity and docking [18].
The first approach uses lattice-type modular robots with cubical shape. Each robotic module can slide along other units and thus create and reconfigure large structures from cubes. Telecube [35] is one example of a lattice-type modular robot. Its modules consist of linear actuators on each side of the cube that allow connection and sliding over other modules. Power and data transmitting is also possible through the docking system on each side.
The other modular robot type, chain-type, uses less docking interfaces to generate structures like a snake, a legged robot or a rolling robot. These systems focus mainly on the whole body locomotion in connected state. Moreover creating of these structures is studied in order to perform faster locomotion than an individual module or to overcome the obstacles that are higher than a single robot. The examples of chain-type modular robots are PolyBot [40, 41] and SuperBot [31].

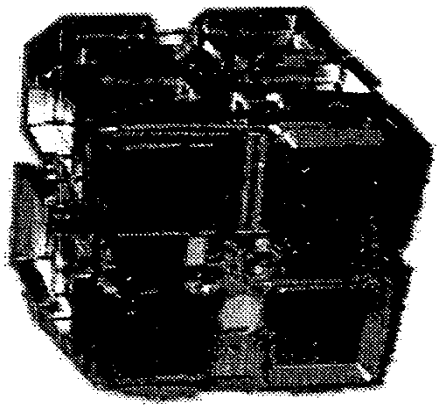

Figure 5: Telecube module [35]

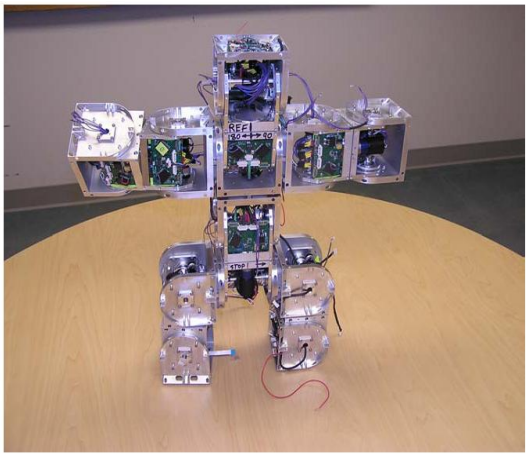

Figure 6: Six connected SuperBots [31]

PolyBot organisms consists of graph-like structures with two main module types. The first module called a segment has one degree of freedom and two docking connectors.The other, called a node, has no degree of freedom, but six possible connection interfaces.
The SuperBot project has a one type module with two end effectors and one central part. This design has three degrees of freedom that are enough for locomotion in connected state.

## 3.1 Replicator and Symbrion

The Replicator and Symbrion projects are European research projects. Their funding of around 11M € in sum makes them some of the largest grants in collaborative and evolutionary robotics [22].
The Symbrion project is aimed mainly at artificial evolution and bio-inspired behavior of robot population. This includes artificial immunology, embryology and genetic self-reprogramming [2].
The Replicator project focuses on module development with powerful on-board computing resources. Moreover, every module must be equipped with a large number of sensors(camera, laser, RFID, distance sensors) [22].
Both projects use identical hardware platforms for their research. The modules can be referred to as chain-type. All of them have the same types of docking interfaces with power and network sharing. The main processing unit of all modules is the dual core Blackfin BF561 processor. uClinux with 64MB ram runs on the processor. All sides with PCBs are equipped with sensors whose data are processed by the MSP430F2618 microcontrollers and sent to the processor over Serial Peripheral Interface (SPI). The microcontrollers serve as an interface between the linux operating system and low level sensor measurements and actuator control [19].
In the following sections, three developed modules are introduced which all use the same(previously mentioned) systems and docking interfaces in order to create connected structures.

### 3.1.1 Scout module

The Scout module is a robot actuated by two tracks that allow movement of a single unit. Also its name refers to the exploration purposes of the Scout module. The robot has a cubical shape of $10 \times 13 \times 10$cm size and the approximate weight of one kilogram including the battery. Despite its main exploration purpose, the robot is also usable in multi robot organisms. With one central rotational joint, the robot has one degree of freedom (DOF) in connected structures. There are four docking interfaces, one on every lateral side, that allow the Scout module to have three modules firmly connected and one adjustable on the central joint. The robot can move in $\pm 90°$ with this joint [17].

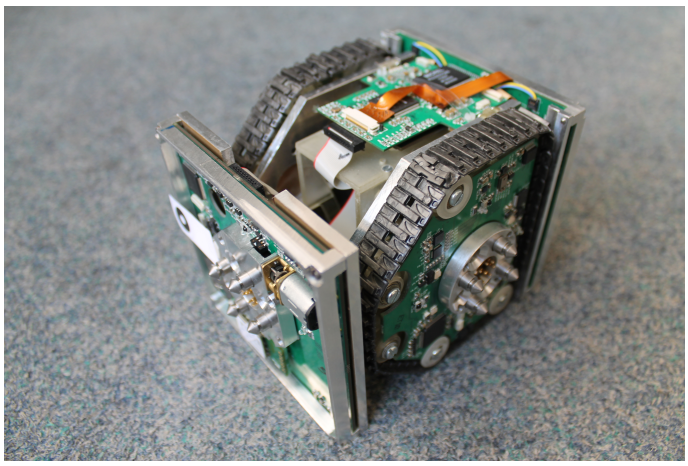

Figure 7: Scout module

### 3.1.2 Active Wheel module

The Active Wheel module is designed mainly as a module for transportation and replacement of faulty modules. With three omni wheels with 120° shift to each other, the robot can move with 3 DOF during on-floor movement. The module consists of two arms connected by a central joint (hinge) that is capable of 180° rotation. One arm is equipped with two wheels and the other with one(in previous versions also two). Two docking interfaces, able to rotate, are placed on the sides of the central joint.
This modular platform is not ideal for multi robot structures, but in special cases it may serve as a foot(a robot between other modules and floor that actuates the whole body).

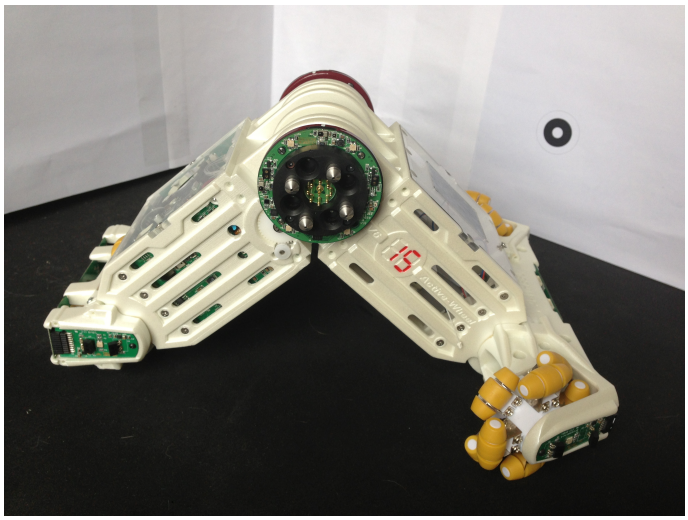

Figure 8: Active Wheel module

### 3.1.3 KaBot module

The last robotic module is KaBot(or Backbone). This platform specializes in the whole body locomotion in multi robot organisms. With a strong central joint with one degree of freedom, it is better suited in organisms than the Scout module. On the other hand, KaBot is equipped with a 2D screw drive that allows only slow movement on special surfaces. The robot has cubical shape and, like the Scout module, four docking interfaces.
At the time of writing this thesis, KaBot platform is under development and therefore all suggested algorithms have been tested only for the Active Wheel and the Scout module.

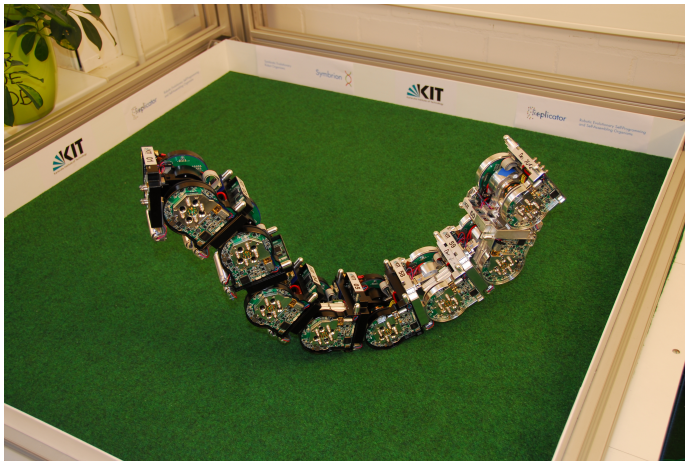


Figure 9: KaBot platforms in snake-like organism

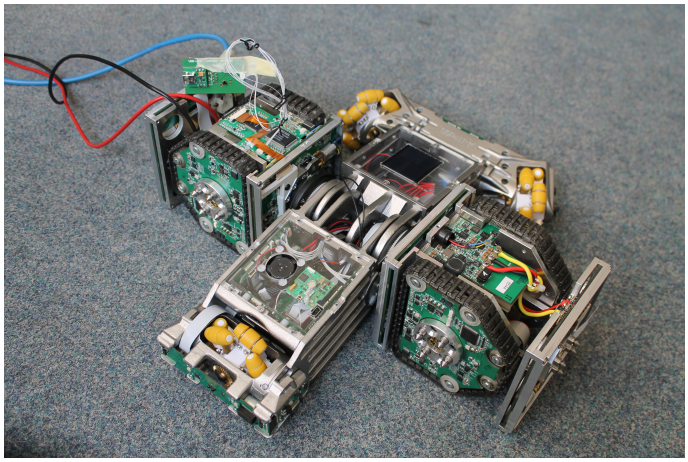

Figure 10: Organism with Active wheel and two Scout modules

# 4 Dead reckoning

For both Scout and Active Wheel robot modules, a self-localization algorithm was implemented using dead-reckoning(odometry) that was described in section 2.1. All robots in the Symbrion and Replicator projects are equipped with encoders on motors. These encoders are still being developed, therefore the determination of traveled path must be counted only from motor speed setting and elapsed times. In the future, dead-reckoning will be replaced by odometry.

## 4.1 Scout Robot

The odometry in the Scout robot has been implemented for vehicles with differential drive [24, 11]. Such vehicles use two motors with different speeds for circle paths and the same speed for straight drive. The Scout robot is equipped with tracks of nearly the same behavior as the differential drive wheels. This type of movement has only two degrees of freedom, so it is non-holonomic. Due to this fact, the that robot can-not be driven to any position straight, but must be first turned for movement to the required position.

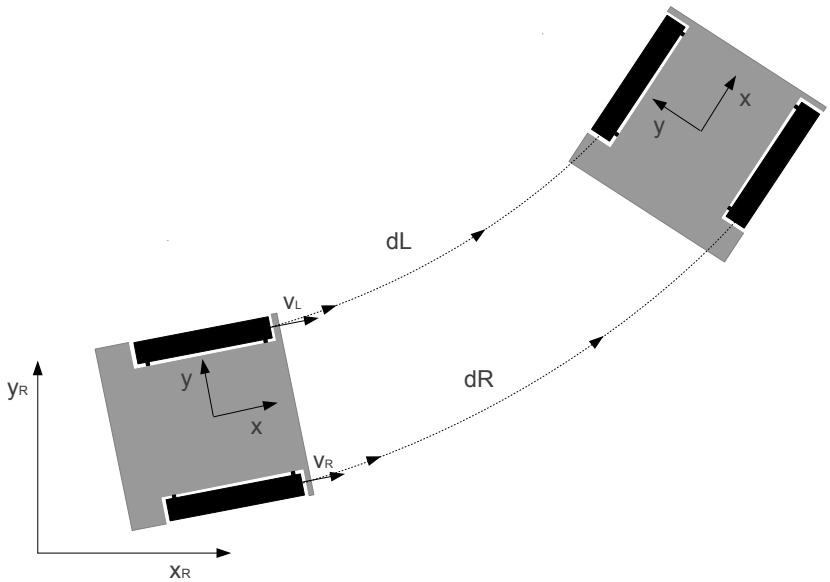


Figure 11: Model of the Scout Robot movement

As could be seen in figure 11 speeds of tracks $v_L$ and $v_R$ are tangents of the traveled path of tracks $dL$ and $dL$. Track speeds can be set in the range $[-100, 100]\%$, where 100% corresponds with more than one body length per second [19]. At the same track speeds, the

robot's main program measures time intervals $dT$ in milliseconds. The traveled distances $dL$ and $dL$ of the left and right tracks are calculated according to equations (3) and (4).

$$dL_k = l \cdot dT \cdot v_{L,k} \tag{3}$$
$$dR_k = -l \cdot dT \cdot v_{R,k} \tag{4}$$

Constant $l$ in the previous equation depends on many parameters like track length, motor gear, real motor speed, surface material, track material and so on. For that reason, $l$ was determined experimentally in section 4.1.1. In the equation for the traveled distance of the left track, constant $-l$ is used because of the reverse speed setting on the right motor. As the robot usually travels on circular paths, its position could be derived from this knowledge. The following equation for a change in the robot position $(dr_x, dr_y, dr_\phi)^T$ uses transformation from the old robot position to the new one through the center of the circular path [24].

$$\begin{pmatrix} dr_x \\ dr_y \\ dr_\phi \end{pmatrix} = \begin{pmatrix} \frac{b(dL+dR)}{2(dR-dL)} \cdot (sin(r_{\phi,k-1} + dr_\phi) - sin(r_{\phi,k-1})) \\ -\frac{b(dL+dR)}{2(dR-dL)} \cdot (cos(r_{\phi,k-1} + dr_\phi) - cos(r_{\phi,k-1})) \\ \frac{dL-dR}{b} \end{pmatrix} \tag{5}$$

The calculation of position change also uses parameter $b$ which is the track (distance) between the tracks. This parameter was determined directly by measuring this distance, but also improved during rotary movement of the Scout Robot.
If the traveled path is nearly straight, then the traveled distances of tracks are nearly the same, which led denominator $2\,(dR - dL)$ to be close null. In this case, we must use the equations for $dr_x$ and $dr_y$ different from the previous. During straight movement, the change in the robot position is calculated as follows [24].

$$\begin{pmatrix} dr_x \\ dr_y \\ dr_\phi \end{pmatrix} = \begin{pmatrix} dL \cdot cos(r_{\phi,k-1} + dr_\phi) \\ dL \cdot sin(r_{\phi,k-1} + dr_\phi) \\ \frac{dL-dR}{b} \end{pmatrix} \tag{6}$$

### 4.1.1 Determination of Dead reckoning parameters

From the measurements of the distance really traveled on straight paths, parameter $l$ was determined. This parameter converts set speeds $v_L, v_R$ and time of movement $dT$ in milliseconds to path traveled by each track $dL, dR$. Dependency was observed between these data that is shown in figure 12. With assumed functional dependency expressed by equation (3) and (4) we obtain parameter $l$.

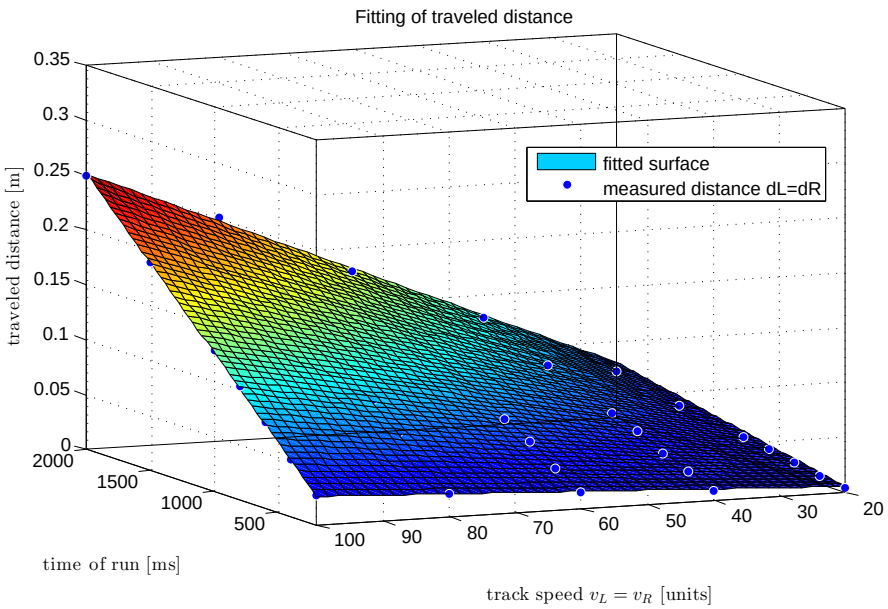


Figure 12: Dependency of traveled distance on set speed and traveled time

After fitting the measured data on equations (3) and (4), parameter $l$ was determined to be $l = 1.253e^{-06} \frac{\mathrm{m}}{\%\mathrm{ms}}$.

### 4.1.2 Scout self-localization error modeling

Our method used for self-localization belongs among incremental localization methods. This is why this method has an additive localization error. The error must be declared in our SLAM algorithm in time update 6.1 of the EKF and therefore we must determine an error model. As an indicator of the robot self localization error, covariance matrix $P_{od}$ is used.

We assume that parameters $l$ and $b$ are determined accurately, and that all generated localization errors are not due to wrong determination of these constants. Variances $\sigma_{dL}$,$\sigma_{dL}$ of traveled distances are considered to be equal. Both were determined with the same data as constant $l$ from the previous section. The outcome of equation 6 for position change was used as mean $\mu_{dL} = \mu_{dR}$ of traveled distance. The error in the determination of position after traveling straight path is displayed in figure 13.

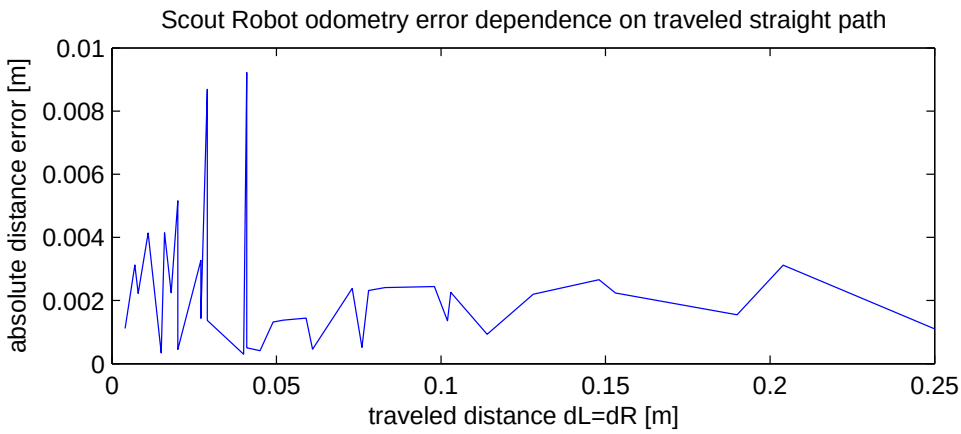


Figure 13: Dependency of odometry error on traveled distance

The figure shows that the errors are nearly constant and also that are are higher in short distances. In short distances, the error is usually bigger error due to the delay in constant speed setting. Using the standard equation for population variance (7) on measured traveled distances, we get variances $\sigma_{dL}^2 = \sigma_{dR}^2 = 9.287e^{-06}\mathrm{m}^2$ which corresponds with standard deviation $\sigma_{dL} = \sigma_{dR} = 0,00305\mathrm{m}$.

$$\sigma_{dL}^2 = \sum_{i=1}^{n} (x_i - \mu_{dR})^2 \tag{7}$$

Odometry covariance matrix $P_{od}$ that is used in the EKF-SLAM time update is of the following form:

$$P_{od} = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{pmatrix}. \tag{8}$$

Variances of position $\sigma_x^2$, $\sigma_y^2$ and $\sigma_\phi^2$ are calculated from the above measured variances $\sigma_{dL}^2$ and $\sigma_{dR}^2$ using Taylor expansion of odometry equations (5) for circular path or (6) for straight path.
For arbitrary function $z = f(x, y)$ with variables $x$ and $y$, variance $\sigma_z^2$ could be determined using the following formula (9).

$$\sigma_z^2 = \left(\frac{df}{dx}\right)^2 \sigma_x^2 + \left(\frac{df}{dy}\right)^2 \sigma_y^2 \tag{9}$$

For odometry equations we determined position variances for circular path as:

$$
\begin{aligned}
\sigma_x^2 &= \left(\frac{bdR}{(dR-dL)^2}\left(sin(r_{\phi,k}) - sin(r_{\phi,k-1})\right) + \frac{dL+dR}{2(dR-dL)}cos(r_{\phi,k})\right)^2 \sigma_{dL}^2 + \\
&+ \left(\frac{-bdL}{(dR-dL)^2}\left(sin(r_{\phi,k}) - sin(r_{\phi,k-1})\right) - \frac{dL+dR}{2(dR-dL)}cos(r_{\phi,k})\right)^2 \sigma_{dR}^2 \\
\sigma_y^2 &= \left(\frac{-bdR}{(dR-dL)^2}\left(cos(r_{\phi,k}) - cos(r_{\phi,k-1})\right) + \frac{dL+dR}{2(dR-dL)}sin(r_{\phi,k})\right)^2 \sigma_{dL}^2 + \\
&+ \left(\frac{bdL}{(dR-dL)^2}\left(cos(r_{\phi,k}) - cos(r_{\phi,k-1})\right) - \frac{dL+dR}{2(dR-dL)}sin(r_{\phi,k})\right)^2 \sigma_{dR}^2 \\
\sigma_\phi^2 &= \frac{1}{b^2}\sigma_{dL}^2 + \frac{1}{b^2}\sigma_{dR}^2
\end{aligned}
\tag{10}
$$

The resulting equations are quite long for running every loop inside the control program of the robot. This disadvantage could be eliminated by separating identical parts of the equations and re-using of sub-results.
In the case of straight paths, the position change equations are simpler, which also leads also to the simplification of the position variances equations (11).

$$
\begin{aligned}
\sigma_x^2 &= \left(\frac{1}{2}cos^2(\phi_{r,k}) + \frac{(dL+dR)^2}{2b^2}sin^2(\phi_{r,k})\right)\sigma_{dL}^2 \\
\sigma_y^2 &= \left(\frac{1}{2}sin^2(\phi_{r,k}) + \frac{(dL+dR)^2}{2b^2}cos^2(\phi_{r,k})\right)\sigma_{dL}^2 \\
\sigma_\phi^2 &= \frac{2}{b^2}\sigma_{dL}^2
\end{aligned}
\tag{11}
$$

With the above equations we could determine an additional error after every change of the Scout robot position. This uncertainty is expressed by covariance matrix $P_{od}$ and is added to the state covariance matrix in the EKF time update.

## 4.2 Active Wheel

The model of the Active Wheel module's odometry differs from the Scout module. All three wheels are omnidirectional with inclination of 120° to each other. The kinematics of this module is also dependent on the actual position of the central joint (located in $S_r$ in the diagram 14). The decrease of the central joint angle $\alpha$ implies a reduction of distance $L_3$ which is parameter in Active Wheel's odometry equations (15).

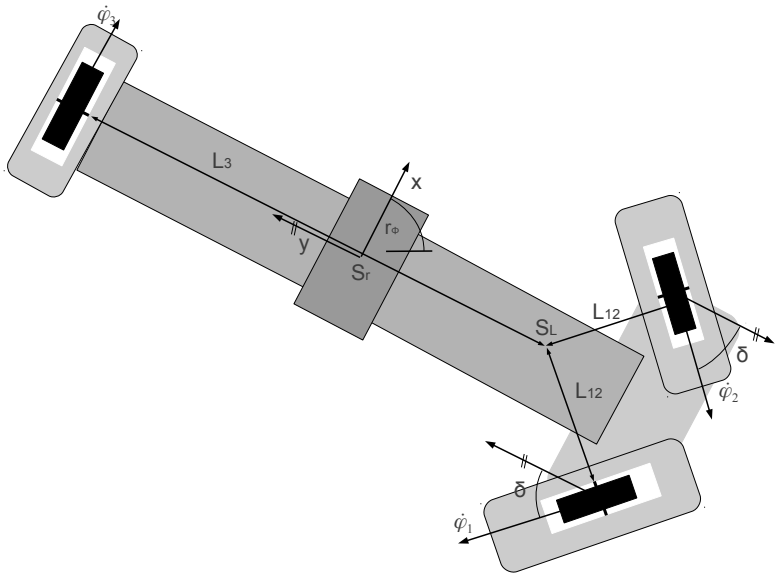


Figure 14: A diagram of Active Wheel kinematics from top view

From the robot global speed $(\dot{x}, \dot{y}, \dot{\phi})^T$, the robot rotation $r_\phi$ and the placement of wheels on robot, the actual speed of wheels $\varphi_i$ is determined as follows:

$$\begin{pmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{pmatrix} = \frac{1}{r} A \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \frac{1}{r} \begin{pmatrix} -sin(\delta+\phi) & +cos(\delta+\phi) & -L_{12} \\ -sin(\delta-\phi) & -cos(\delta-\phi) & -L_{12} \\ cos(\phi) & sin(\phi) & -L_3 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} [21]. \tag{12}$$

For self-localization purposes, an inverse expression is important which determines the robot global speed in $x, y, \phi$ from its wheel speeds. As in [27] we derived this expression using the inverse of matrix $A$.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \frac{r}{D} \begin{pmatrix} -L_{12}sin(\phi) - L_3cos(\delta-\phi) & L_{12}sin(\phi) - L_3cos(\delta+\phi) & 2L_{12}cos(\delta)cos(\phi) \\ L_{12}cos(\phi) + L_3sin(\delta-\phi) & -L_{12}cos(\phi) - L_3sin(\delta+\phi) & 2L_{12}cos(\delta)sin(\phi) \\ -cos(\delta) & -cos(\delta) & -sin(2\delta) \end{pmatrix} \begin{pmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{pmatrix} \tag{13}$$

where $D$ is the discriminant of matrix A.

$$D = 2cos(\delta)\left(L_{12} + L_3sin(\delta)\right) \tag{14}$$

After the integration of equation (13), we obtain an expression for a change in the robot position $(dx, dy, d\phi)^T$ according to the robot wheel velocities and time step $dT$. This simplification expects constant velocity during the time step and led to substitution $d\varphi_i = ndT\dot{\varphi}_i$. Variable $d\varphi_i$ is then the traveled distance of a specific wheel in meters and constant $n$ is for the conversion between set motor speeds, time $dT$ in milliseconds and one wheel traveled path in meters.

$$\begin{pmatrix} dS_{L,x} \\ dS_{L,y} \\ dS_{L,\phi} \end{pmatrix} = \frac{r}{D} \begin{pmatrix} -L_{12}sin(\phi) - L_3cos(\delta-\phi) & L_{12}sin(\phi) - L_3cos(\delta+\phi) & 2L_{12}cos(\delta)cos(\phi) \\ L_{12}cos(\phi) + L_3sin(\delta-\phi) & -L_{12}cos(\phi) - L_3sin(\delta+\phi) & 2L_{12}cos(\delta)sin(\phi) \\ cos(\delta) & cos(\delta) & sin(2\delta) \end{pmatrix} + \begin{pmatrix} d\varphi_1 \\ d\varphi_2 \\ d\varphi_3 \end{pmatrix} \tag{15}$$

The previous equation calculates the change in the position of point $S_L$ which is intersection of wheel axes. The real position of the Active Wheel module is measured to the center of the hinge joint $S_r$. In order to get the change of the real robot position $(dr_x, dr_y, dr_\phi)^T$, we need to convert the position of point $S_r$ to $S_L$ with the old angle $r_{\phi,k-1}$ and back with the new angle $r_{\phi,k}$. The final odometric change equation is of the following form.

$$\begin{pmatrix} dr_x \\ dr_y \\ dr_\phi \end{pmatrix} = \begin{pmatrix} dS_{L,x} \\ dS_{L,y} \\ dS_{L,\phi} \end{pmatrix} + \begin{pmatrix} (sin(r_{\phi,k-1}) - sin(r_{\phi,k}))\, sin(\frac{\alpha}{2}) \\ (-cos(r_{\phi,k-1}) + cos(r_{\phi,k}))\, sin(\frac{\alpha}{2}) \\ 0 \end{pmatrix} \tag{16}$$

The previous conversion between these points depends on the actual angle on the central joint $\alpha$, that also changes the size of distance $L3$ during its change.

### 4.2.1 Determination of odometry parameters

For the optimal calculation of odometry change, parameter $n$, the dependency between wheel speed during the time interval and traveled distance should be correctly determined. We used two scenarios of traveled distance measurement. The first, using all three wheels, was movement along the robot's x axis with a zero rotation of the robot. For this movement, speed 50% on first two wheels and speed -100% on third wheel was used. By adding these speeds, we get speed 200% in the x-direction. In the second method, only the two first motors were used. With opposite speeds on these wheels, movement in the robot's y-direction was achieved.
The measured traveled distances in the x and y directions are in the following figure. The data from both methods of movements were fitted to equation 13.

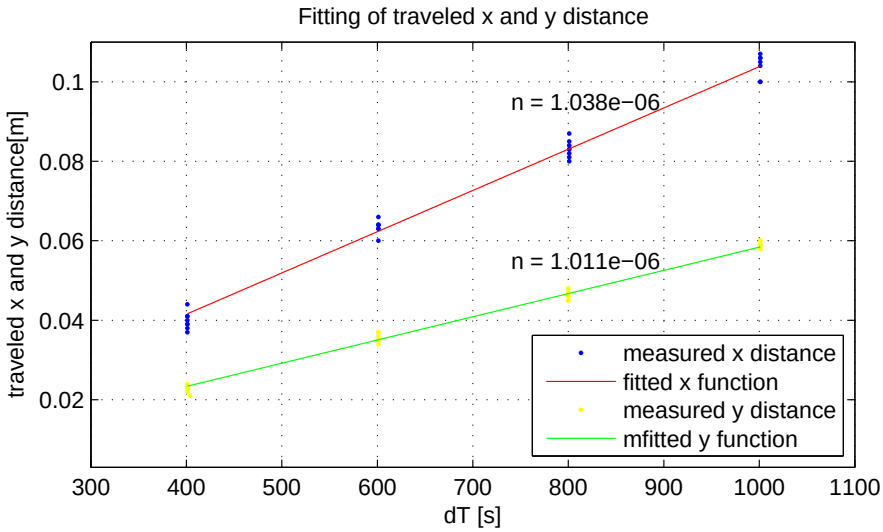


Figure 15: Fitting of measured traveled distance in the x and y directions

For movement in the y-direction, half speed was used, so the traveled distances are shorter. For motion with the Active Wheel, we only use two or three wheels, so the average constant between two wheel and three wheel constant $n = 1.025e^{-06}\frac{\mathrm{m}}{\%\mathrm{ms}}$ will be used as the resulting constant.

### 4.2.2 Modeling of the Active Wheel self-localization error

The Active Wheel odometry error model is different from the model in the Scout robot. The traveled distance of one wheel is badly measured, therefore we used the same data for odometry modeling as for parameter determination. Since we expect cumulative error of odometry, we use variance proportional to the traveled distance [10]. For motion in the x and y directions, we assume the same increase of variance $\sigma_x^2 = j_{xy}^2 \mid d_x \mid$. Constant $j_{xy}$ is determined from the observed errors. In figure 17, squared errors are displayed with dependence on traveled position both in the x and y directions.

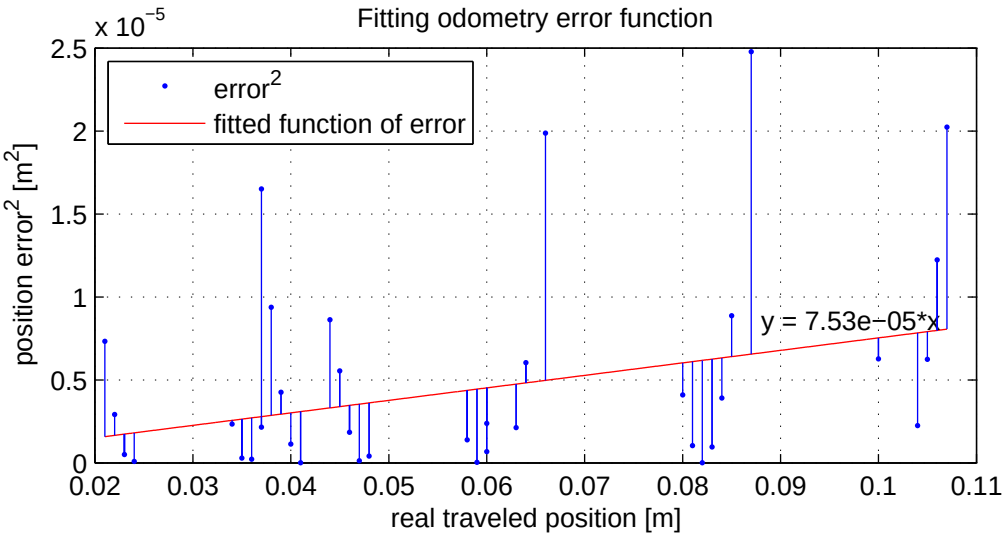


Figure 16: Fitting of error for the x and y motion

The linear equation of the resulting variance is $\sigma_x^2 = 7.53e^{-5}dx$ m$^2$ and $\sigma_y^2 = 7.53e^{-5}dy$ m$^2$. For the calculation of error in the robot rotation, additional measurement was done only with rotation movement of the robot. Constant $j_{xy}$ is not applicable to rotation angle $\phi$, because the error of this angle was proved to be larger.

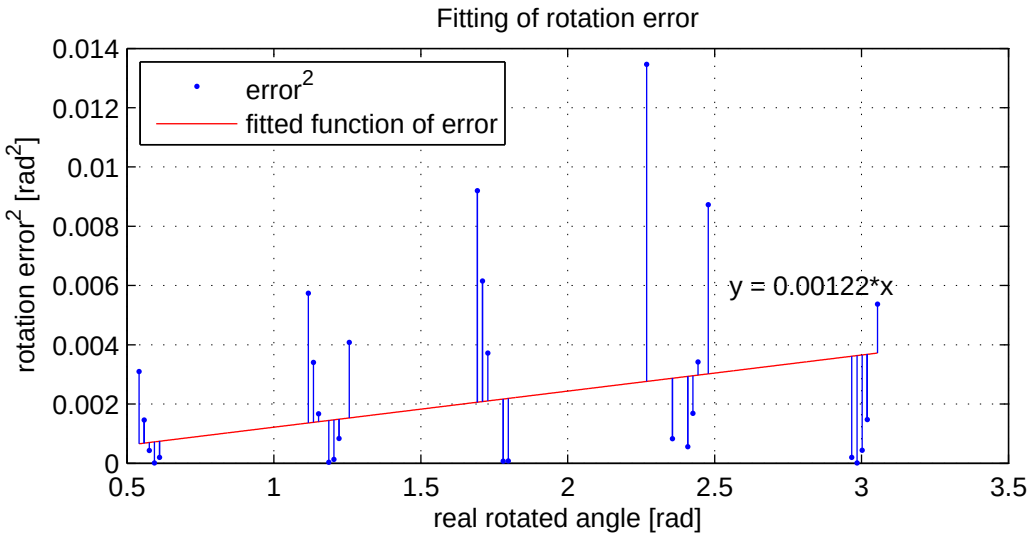


Figure 17: Fitting of robot rotation

The odometry error of rotation movement is approximately one hundred times bigger than that for movement in the x and y directions. The equation that is used for the robot rotation variance is $\sigma_\phi^2 = 1.22e^{-3} d\phi$ rad$^2$.

# 5 Landmark detection

In this thesis roundel pattern was used as a landmark. The detection of this pattern in camera vision is described in [16]. The algorithm can calculate $x$,$y$,$z$ and $\phi$ coordinates relatively to the camera position.

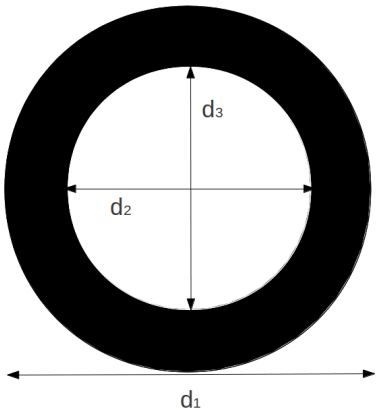


Figure 18: Landmark shape with outer diameter $d_1$ and inner diameter $d_2$

Firt, the detection algorithm finds continuous areas of dark pixels in camera image. Then the isolated regions of such areas are determined. These regions are tested for their circularity and whether they have a white region in the middle [14].
From a detected landmark inside an image and its image size, $x$,$y$,$z$,$\phi$ position parameters of the pattern relatively to the camera position can be determined. Coordinate $x$ is calculated by comparing the landmark's real size and the landmark's size inside the image. Parameters $y$ and $z$ are determined from the position of landmark in the image(x and y distance from image center). The last parameter that could be calculated is $\phi$. This angle, between the camera and the blob, is measured from the difference between horizontal $d_2$ and vertical diameter $d_3$ of the pattern. Especially the horizontal diameter is reduced if the landmark is viewed from a non-zero angle and a circle is observed as an eclipse.
So the algorithm for circle detection returns very precisely landmark angle in an absolute value. An angle sign is determined by comparing the positions of white and black circle middles. If the white circle becomes only a dot in the eclipse, then this dot will not be in the center, but will be shifted to the side closer to camera.

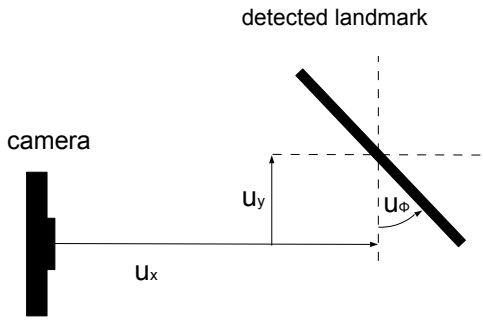


Figure 19: Camera and landmark scheme with measured parameters $x$,$y$ and $\phi$ from top view

For better a visual differentiation of the detected landmark in camera vision, the blob is colored after recognition. In the following pictures, shots of the robot's visions during landmark detection are shown.

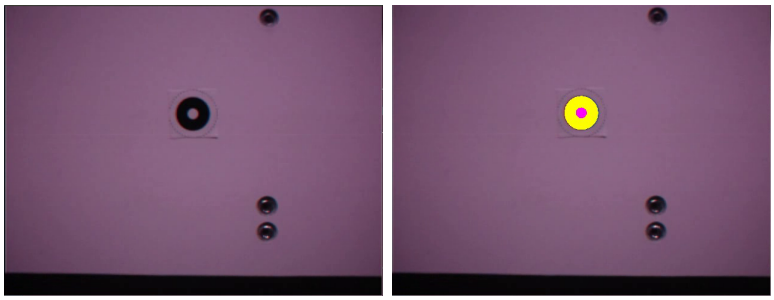

Figure 20: Landmark in the robot's vision before detection(left) and after detection(right)

In our further work, we do not use coordinate $z$ (except conversion of detection on Active Wheel 5.3.2), because we only expect movement on a flat area. Also, we used only coordinates $x$ and $y$ to distinguish between two landmarks due to the inaccuracy in $\phi$ determination.

## 5.1 Camera calibration

The camera was calibrated using the Camera calibration toolbox for matlab [9]. For optimal determination of a landmark relative position, calibration file *Calib_Results.m* was generated. This file contains data about the camera's focal length, principal point, and distortions. In the camera calibration toolbox, various various camera images of planar checkerboard were used. On every one of these images, corners were extracted as in figure 21. From this extraction and the real size of squares, data were computed for calibration file.

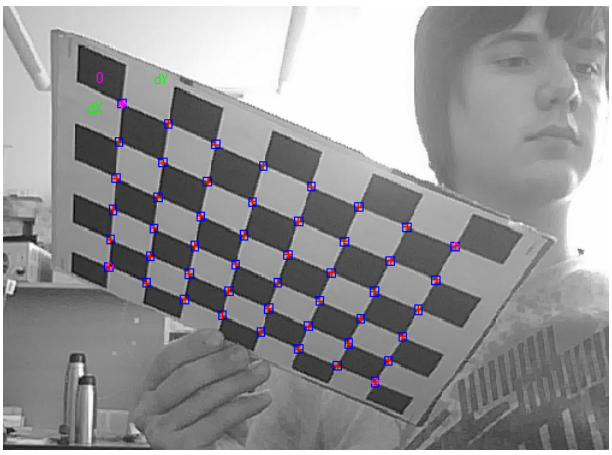

Figure 21: Extraction of square corners in Camera Calibration

## 5.2 Pattern measurement accuracy

The extended Kalman filter, used in our SLAM algorithm, calculates input data with their uncertainties. In this case, the input data are $x$ , $y$ and $\phi$ positions of landmark. Suppose that the camera measurement has normal distribution and that the three measured parameters are independent. In the Kalman filter we use covariance matrix $Q$ as the declaration of measurement uncertainty. The form of covariance matrix is:

$$Q = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{pmatrix}. \tag{17}$$

Since we assumed that the measured data are independent, the covariance matrix has a diagonal form. For the determination of $\sigma_x^2$ , $\sigma_y^2$ and $\sigma_\phi^2$ several measurements were done from different positions and with different angles. The variances of measured coordinates x and y could be considered to be dependent on the camera-landmark distance, but according to figures 22 and 23, they are independent of the distance.

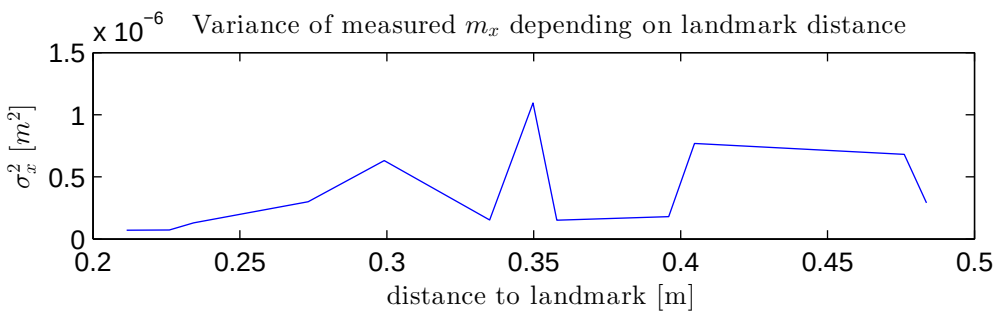


Figure 22: Variance of measured x distance

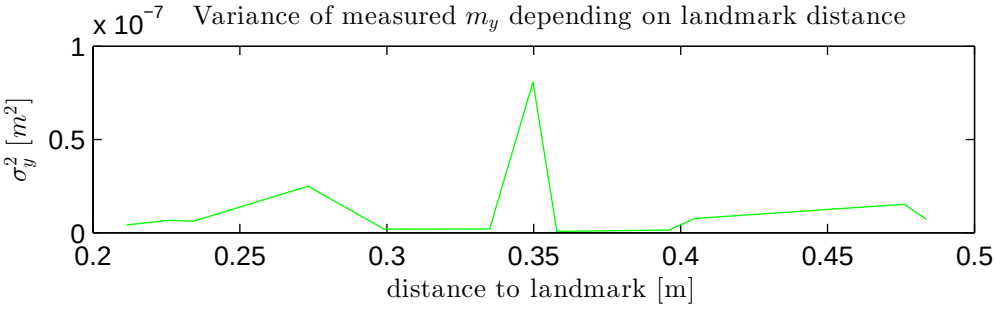


Figure 23: Variance of measured y distance

Variance $\sigma_x^2$ has a maximum value of $1.094e^{-6}\text{m}^2$ that corresponds with an average error of 1.04mm. This error is made by the computation of it's value from a small and inaccurate camera image. For further filtering inside the EKF, this precision is fairly sufficient compared to other inaccuracies. Variance $\sigma_y^2$ is ten times smaller, approximately $0.808e^{-7}\text{m}^2$. The measurement of $m_y$ is actually more precise due to determination from the position inside the image and not only the pixel size of the pattern as $m_x$.
The most inaccurate measured parameter is angle $\phi$. For angles in the interval $(-0.2, 0.2)$ rad, the measured data oscillate around zero with a minimum of $\pm 0,106$rad. This effect led to high variance in this interval.

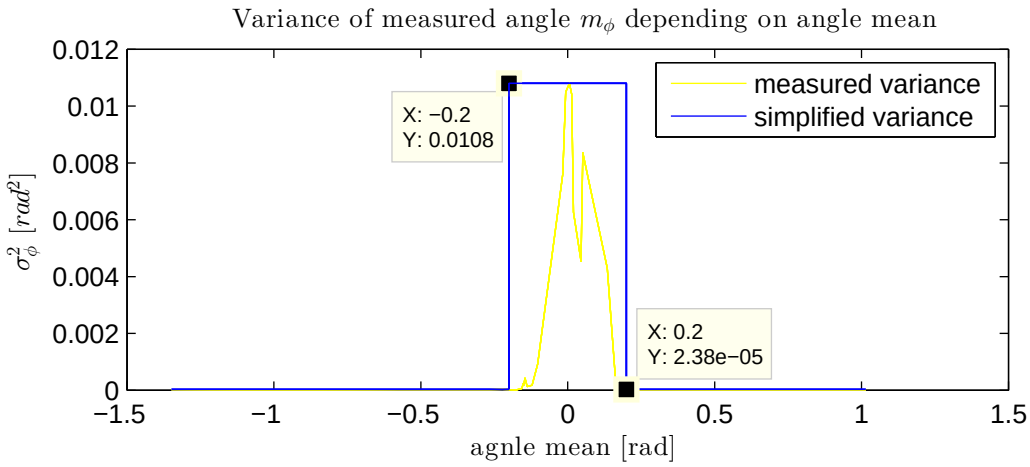


Figure 24: Variance of measured angle $\phi$

Due to the big difference in variance in interval $(-0.2, 0.2)$ rad and outside of it, variance will be simplified and considered to be constant $2.38e^{-5}$ rad$^2$ outside this interval and $1.08e^{-2}$ rad$^2$ inside.

## 5.3 Reusability of camera measurement

As there are two main modules in the Replicator and Symbrion projects (3.1), the measured position of the circle pattern has to be interpreted according to the camera placement on the robot module and the module structure. The detected circle position will always be transformed to the robot movement center.

### 5.3.1 Scout Module

The Scout robot (figure 7) has a camera placed on its front immobile board. The movement center of the scout module is in intersection of two lines. The first line goes in the middle between tracks parallel to the track movement.The other one goes through the middle of the contact area of both tracks with the ground. To this position, the landmark $u_x, u_y, u_\phi$ measurement is transformed according to equation 18.

$$\begin{pmatrix} u_{x,S} \\ u_{y,S} \\ u_{\phi,S} \end{pmatrix} = \begin{pmatrix} u_x \\ u_y \\ u_\phi \end{pmatrix} + \begin{pmatrix} 0.049 \\ -0.018 \\ 0 \end{pmatrix} \tag{18}$$

This transformation is simple translation from the position of the camera to the movement center. Angle $u_\phi$ remains unchanged.

### 5.3.2 Active Wheel Module

For the ActiveWheel module, the transformation of measurement is not only translation. As the main central joint could be arbitrarily positioned, the transformation has to count with the actual position of this joint. Camera calculation is complicated by the camera's of placement in one of the robot's arms. In case, the angle $\alpha$ of the middle joint is different from 180°, the camera image is also turned $\frac{\alpha}{2}$ degrees. The following equation defines the transformation of the camera measurement to center of movement that is placed in the central of the mentioned joint.

$$\begin{pmatrix} u_{x,AW} \\ u_{y,AW} \\ u_{\phi,AW} \end{pmatrix} = \begin{pmatrix} u_x \\ u_y \\ -u_\phi \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ cos\left(\frac{\pi}{2} - \frac{\alpha}{2}\right) & -sin\left(\frac{\pi}{2} - \frac{\alpha}{2}\right) \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_y \\ u_z \end{pmatrix} + \begin{pmatrix} 0.04 \\ sin\left(\frac{\alpha}{2}\right) * 0.06 \\ 0 \end{pmatrix} \tag{19}$$

Angle $u_\phi$ is measured correctly despite the fact that the eclipse that is seen by the camera is rotated. Only a changed sign of this angle was observed.

# 6 EKF SLAM Algorithm

Simultaneous localization and mapping (SLAM) is one of the most studied problems of mobile robotics. The SLAM algorithm is needed when a robot does not have a map of the environment where it operates. The map, in our case, consists of static landmarks. This map is built during the robot movement. The circles from section 5 are used as landmarks, placed in docking stations, or optionally on obstacles. The robot position from self-localization calculation 4 has an increasing error that is reduced during the second and following observations of a landmark.
There are four input parameters in this algorithm. The first one is the measured position of the landmark $u_t = (u_x, u_y, u_\phi)^T$ (this position is relative to the robot position). The second parameter is estimated position change $dr_t = (dr_x, dr_y, dr_\phi)^T$. The last are the state vector $\mu_{t-1}$, that holds both robot position and map, and covariance matrix $P_{t-1}$. The structure of the state vector is as in equation (20).

$$x = \begin{pmatrix} r_x \\ r_y \\ r_\phi \\ m_1 x \\ m_1 y \\ m_1 \phi \\ . \\ . \\ . \\ m_N x \\ m_N y \\ m_N \phi \end{pmatrix} \tag{20}$$

The first three rows holds the robot position and the other $3N - 3$ rows contain positions of $N$ mapped landmarks. The size of the covariance matrix is $(3N - 3) \times (3N - 3)$. It's content corresponds to the state vector.
As in the standard extended Kalman filter (EKF) algorithm described in [37] , the computation could be divided into two main parts and one optional part:

1. **Time update 6.1 ('Prediction')**
   The odometry change is projected in the state vector prediction and also the covariance matrix is increased according to the odometry error during the robot movement.

2. **Landmark initialization 6.2**
   If an observed landmark is not among the mapped landmarks, then the new landmarks's position is added to the state vector. The covariance matrix is expanded accordingly.

3. **Measurement update 6.3 ('Correction')**
   In this part, an observed position of a landmark is compared with the corresponding mapped landmark. From the difference between these two, the state vector and covariance are improved.

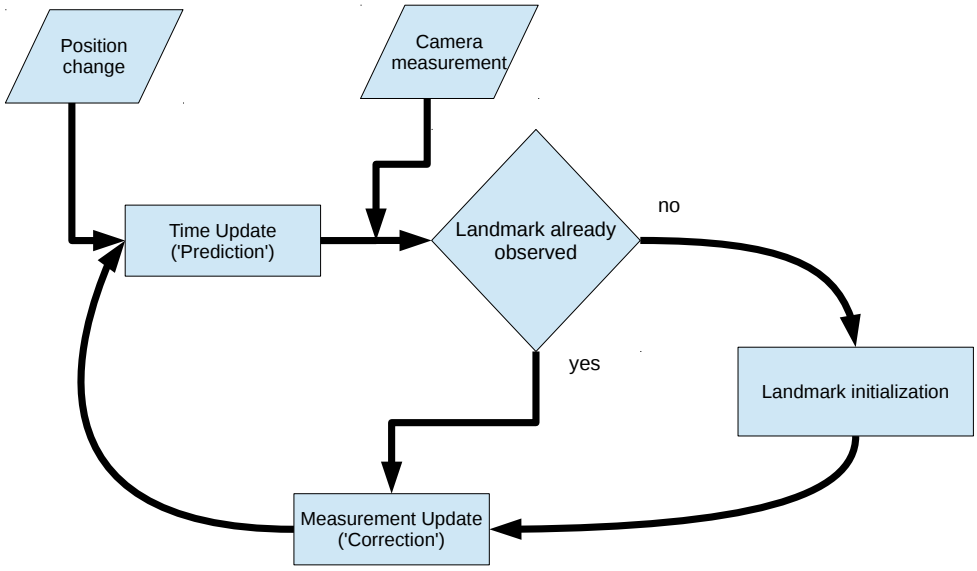


Figure 25: Diagram of SLAM algorithm

For further research of SLAM, transformations between the robot global position $r_k$, k-th landmark global position $m_k$ and landmark relative position $y_k$ (viewed position) are needed. Especially $y_k$ is important to determine the landmark's and robot's position errors during multiple observation of the same landmark.
Transformation to relative position $y_k$ (the same as camera measurement) is described by function $h\left(r_k, m_k\right)$).

$$y_k = h(r_k, m_k) = \begin{pmatrix} cos\left(-r_\phi\right) & -sin\left(-r_\phi\right) & 0 \\ sin\left(-r_\phi\right) & cos\left(-r_\phi\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_{x,i,k} - r_{x,k} \\ m_{y,i,k} - r_{y,k} \\ m_{\phi,i,k} - r_{\phi,k} \end{pmatrix} \tag{21}$$

The landmark global position is computed from the measured position of landmark $u_k$, that should corresponds with $y_k$, using function $g\left(y_k, r_k\right)$

$$m_k = g(r_k, y_k) = \begin{pmatrix} cos\left(r_\phi\right) & -sin\left(r_\phi\right) & 0 \\ sin\left(r_\phi\right) & cos\left(r_\phi\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{x,k} \\ y_{y,k} \\ y_{\phi,k} \end{pmatrix} + \begin{pmatrix} r_{x,k} \\ r_{y,k} \\ r_{\phi,k} \end{pmatrix} \tag{22}$$

The following figure shows a possible configuration of the robot and landmark in the global coordinate system $x, y$.

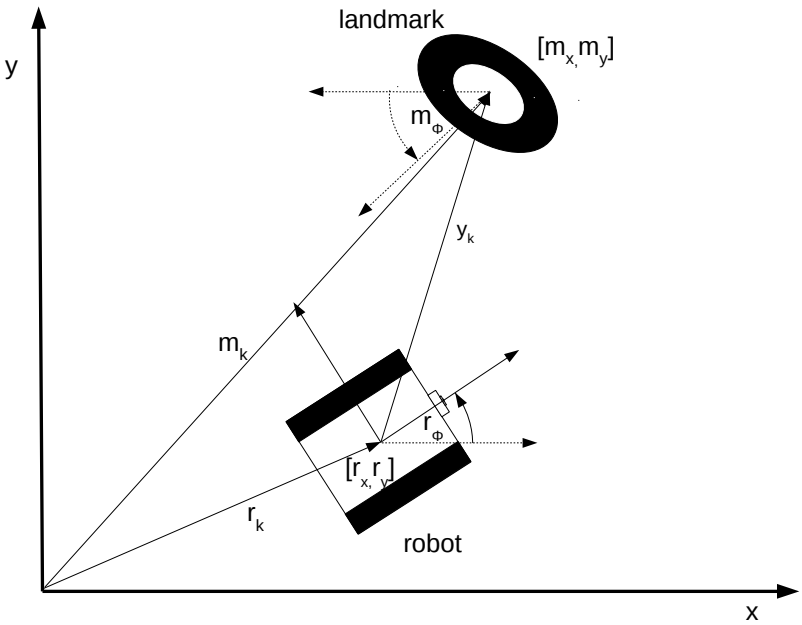


Figure 26: Transformation between landmark position $m_k$, robot position $r_k$ and viewed position of landmark $u_k \approx y_k$

For the purposes of the Kalman filter linear equations are needed rather than the equations described above. The extended Kalman filter uses linearized functions, especially their Jacobians. For both functions $g(r_k, y_k)$ and $h(r_k, m_k)$, we use a first degree Taylor expansion.
Jacobian $H_k$ is calculated in $\overline{\mu}_k$ with respect to the robot position $r_k$ and landmark global position $m_k$.

$$H_k = \begin{pmatrix} -cos(-r_\phi) & sin(-r_\phi) & sin(-r_\phi)\left(m_{x,k} - r_{x,k}\right) + cos(-r_\phi)\left(m_{y,k} - r_{y,k}\right) & cos(-r_\phi) & -sin(-r_\phi) & 0 \\ -sin(-r_\phi) & -cos(-r_\phi) & -cos(-r_\phi)\left(m_{x,k} - r_{x,k}\right) + sin(-r_\phi)\left(m_{y,k} - r_{y,k}\right) & sin(-r_\phi) & cos(-r_\phi) & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix} \quad (23)$$

$G_k$ is the Jacobian of function $g(r_k, y_k)$ with respect to the robot position and landmark measured position.

$$G_k = \begin{pmatrix} 1 & 0 & -sin(r_\phi)y_{x,k} - cos(r_\phi)y_{y,k} & cos(r_\phi) & -sin(r_\phi) & 0 \\ 0 & 1 & cos(r_\phi)y_{x,k} - sin(r_\phi)y_{y,k} & sin(r_\phi) & cos(r_\phi) & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (24)$$

In our solution of the SLAM algorithm, observed landmarks are tested for their existence

in the map using the Euclidean distance. If the distance between an observed landmark and the already mapped landmarks is longer than set distance $r$ (for most experiments was $r = 0.35$m). An other possibility is to use the Mahalanobis distance that also takes into account the actual state covariance and based on it calculates the possible area of landmark position.

## 6.1 Time Update

During time update, state vector and covariance matrix are predicted according to equations (25) and (26). The only parts that are changed during time update are those associated with robot position. Predicted state vector $\overline{\mu}_k$ is calculated form state vector in previous step $k-1$ and self-localization change $(dr_x, dr_y, dr_\phi)^T$.

$$\overline{\mu}_k = \mu_{k-1} + \begin{pmatrix} dr_x \\ dr_y \\ dr_\phi \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{25}$$

To the state covariance matrix is added covariance matrix of odometry change $P_{od}$ that was discussed in section 4.1.2 and 4.2.2. Equation (26) represents rise in uncertainty of the robot's position after movement. Parts of state covariance matrix that correspond with landmarks positions do not change, because their position is unchanged.

$$\overline{P}_k = P_{k-1} + \begin{pmatrix} P_{od} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \tag{26}$$

## 6.2 New landmark initialization

New landmark position is initialized inside map using $m_k = g(u_k, \overline{r}_k)$ from equation (22). Other option discussed in [37] is to initialize landmark positions to $(0, 0, 0)^T$. This initialization is better for situations where more observation are guaranteed after initialization. In our case, initialization using function $g$ give us the best possible estimation of landmark global position.
New landmark position $m_k = (m_{x,k}, m_{y,k}, m_{\phi,k})^T$ is added to the end of predicted state vector $\overline{\mu}_k$.

$$\overline{\mu}_k = \begin{pmatrix} \overline{\mu}_k \\ m_k \end{pmatrix} \tag{27}$$

Assuming that covariance matrix $P_k$ could be divided to

$$\overline{P}_k = \begin{pmatrix} \overline{P}_r & \overline{P}_{mr}^T \\ \overline{P}_{mr} & \overline{P}_{mm} \end{pmatrix}, \tag{28}$$

we can use subparts of this matrix during landmark initialization [33].
Part $P_r$ is covariance matrix of robot position. This part is updated during robot motion in time update of Kalman filter. Matrix $P_{mr}$ is cross-variance between robot and landmark positions. Non-zero $P_{mr}$ guarantee improvement of robot's position according to landmark's difference between landmark map position and observed position. The larges part of covariance matrix is $P_{mm}$, which is covariance between mapped landmarks. Non-zero $P_{mm}$ causes change in multiple landmark's positions if one of them is changed.
If new landmark is mapped, then parts $P_{mr,new}$ and $P_{mm,new}$ are calculated using equations (29) and (30).

$$\overline{P}_{mr,new} = G_r \left( \overline{P}_{rr} \, \overline{P}_{mr}^T \right) \tag{29}$$

$$\overline{P}_{mm,new} = G_r \overline{P}_{rr} G_r^T + G_y Q G_y^T \tag{30}$$

In previous equations were also used parts of jacobian matrix $G_k$, that can be divided into two parts.

$$G_k = \begin{pmatrix} G_r & G_y \end{pmatrix} \tag{31}$$

Part $G_r$ is derivation of function $g(r_k, y_k)$ with respect to robot position $r_k$ and $G_y$ is derivation of same function with respect to measured landmark position.
Newly created cross-variance $P_{mr,new}$ and landmark variance $P_{mm,new}$ are then connected to the end of state covariance matrix.

$$\overline{P}_k = \begin{pmatrix} \overline{P}_k & \overline{P}_{mr,new}^T \\ \overline{P}_{mr,new} & \overline{P}_{mm,new} \end{pmatrix} \tag{32}$$

## 6.3 Measurement Update

In this part, we use predicted state vector and covariance matrix from previous sections. We also assume existence of correspondence between observed landmark and mapped landmark.
From predicted robot position $(\overline{r}_x, \overline{r}_y, \overline{r}_\phi)^T$ and corresponding landmark mapped position $(\overline{m}_x, \overline{m}_y, \overline{m}_\phi)^T$ we calculate jacobian $H$ using equation (23). This matrix defines change of robot position and landmark mapped position from error of observed position.

Innovation (residual [1]) $z_k$ calculates error between observed position of landmark and mapped landmark.

$$z_k \quad = u_k - h(\overline{r}_k, \overline{m}_{i,k}) \tag{33}$$

This vector tells us how much the position of landmark inside robot vision is far from position where robot expect it.
In next step we calculate Kalman gain $K$ (34) which is $3\mathrm{N} + 3 \times 3$ matrix that define how innovation $z_k$ changes predicted state vector and predicted covariance matrix. Accurate calculation of Kalman gain is one of the crucial part of whole algorithm. Kalman gain is matrix that resolve whether improve robot position rather than landmark position. For that reason was part about landmark measurement error so important, because following equation uses landmark covariance matrix $Q$.

$$K \quad = \overline{P}_k H_k^T \left( H_k \overline{P}_k H_k^T + Q \right)^{-1} \tag{34}$$

Last part of whole algorithm is reflection of innovation $z_k$ to predicted state and predicted covariance matrix.

$$\mu_k \quad = \overline{\mu}_t + K z_k \tag{35}$$

$$P_k \quad = (I - K H_k) \overline{P}_k \tag{36}$$

New predicted state vector $\mu_k$ and covariance matrix $P_k$ are then used in future runs of algorithm. Especially covariance matrix should have smaller variances of robot position after correction using landmark measurement.

# 7 Experimental results

In this section, we will verify the abilities of the suggested SLAM algorithm. These include settling of a landmark global position, improvement of self localization, mapping capability and loop closing effect.

## 7.1 Statical observation of a landmark

In this part we prove the correct function of our SLAM algorithm on the observation of one landmark with no movement of the robot. By this, we mainly check transformation between camera measurement and the robot position. The result should be an accurate position of the landmark with lower changes(errors) than the measurement itself.

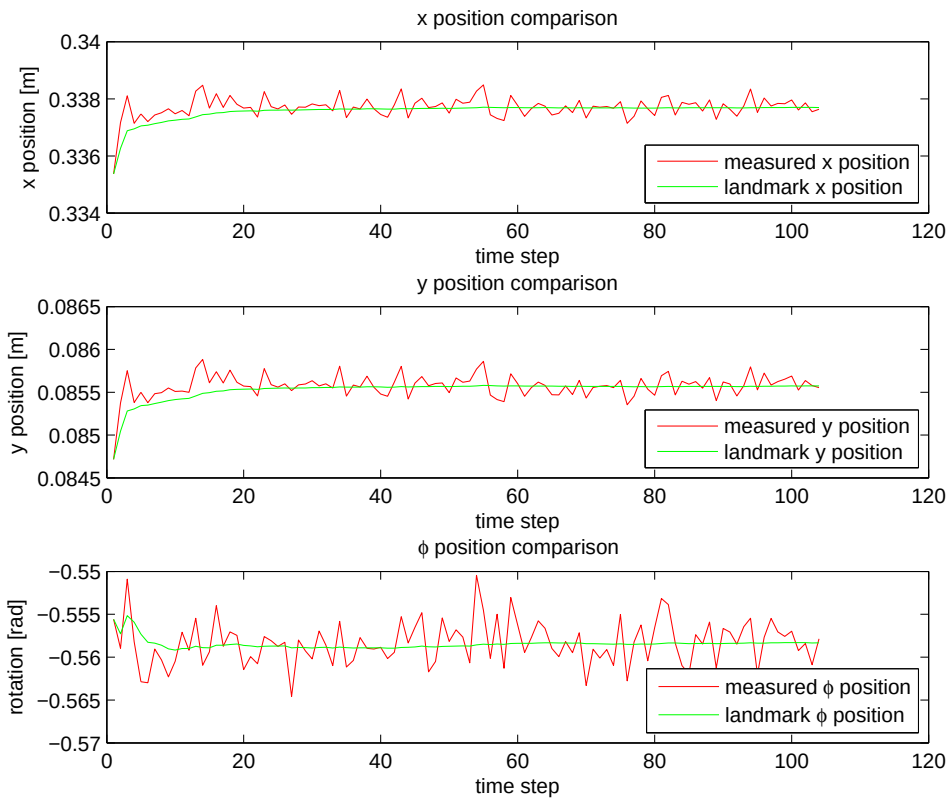


Figure 27: Comparison of measured and filtered location of a landmark

In figure 27, good filtering of the EKF-SLAM algorithm is shown. At every one time step, one camera measurement as well as filtering is done. During the first observations of a landmark, the landmark position in the map is changing quite a lot according to the measured position. After ten or less steps, the landmark position stabilizes on final the position.

## 7.2 Localization improvement using SLAM

The main advantage of the SLAM algorithm is not only building a map during the robot movement in an unknown environment, but also the improvement in self localization. As all incremental localization methods, including odometry and dead reckoning, have cumulative error, the localization must be corrected by other systems. The implemented robot self-localization relies on speed setting without feedback, therefore the error increases quite quickly.
This section demonstrates the effect of map based localization, that is a part of the SLAM algorithm on the resulting robot position.

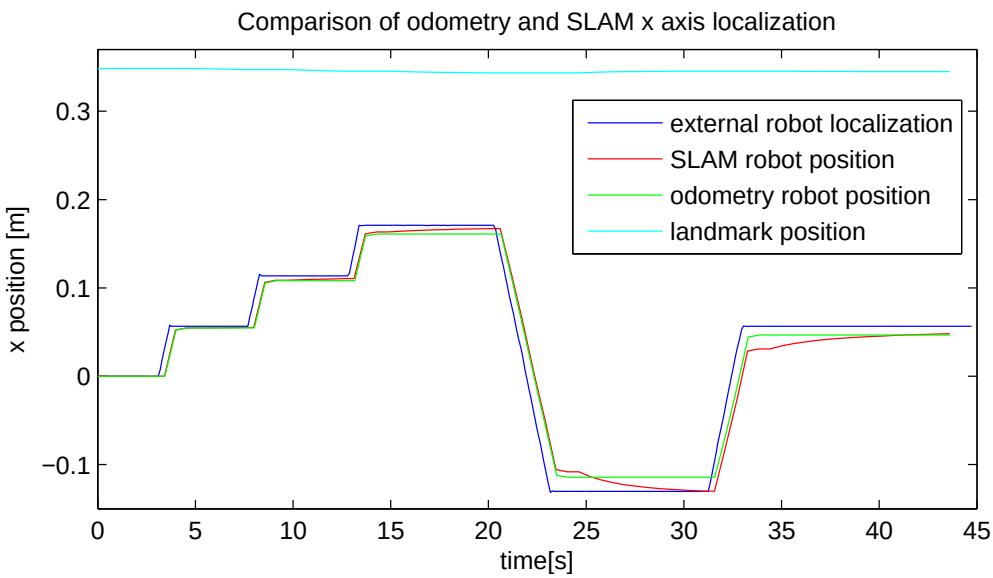


Figure 28: Comparison of odometry and SLAM x axis localization

Figure 28 shows correction of the robot position by localization both from odometry and one landmark. During this experiment, the robot moves in the x axis direction only. After driving 0.66 meters, the error was nearly a 3 centimeters difference between the robot real position, measured by external localization, and the robot odometry position .
When the robot does not move (horizontal lines in figure), the position is improved from landmark observations. The SLAM position is nearly the same as external robot localization, or visually trying to be the same. Especially in time 25-30s the red line that represents SLAM position is approaching the real position of robot, measured by external localization. During the same x-direction motion, state covariance matrix $P$ enlarges its element that represents uncertainty of the robot x position. The following figure 29 shows uncertainty progress of the robot and landmark positions.

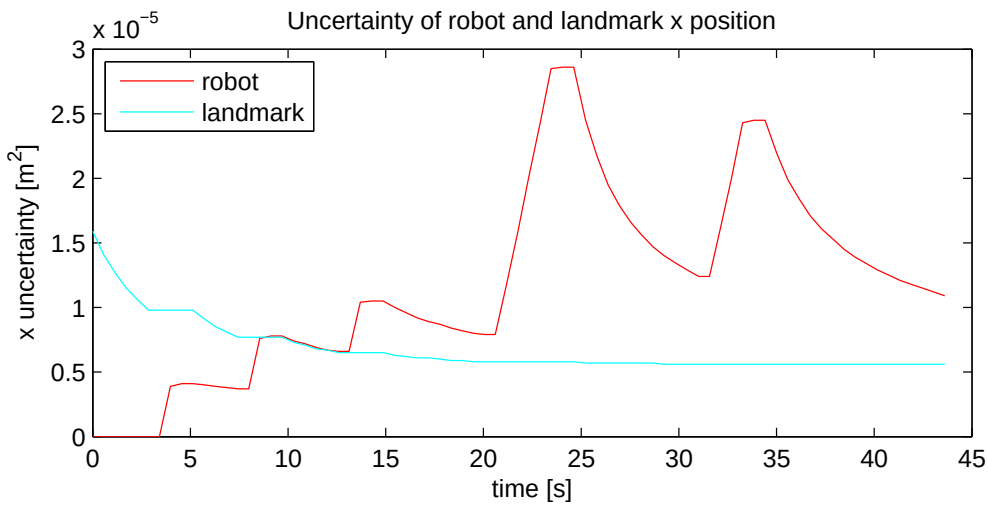


Figure 29: Comparison of odometry and SLAM x axis localization

The initial position of the robot is set at zero with zero uncertainty. After every movement, the position uncertainty enlarges. As the landmark measurement has lower variances, after every landmark observation, the robot position uncertainty is reduced. If there is enough observation, the robot's position uncertainty decreases exponentially to landmark position uncertainty.

## 7.3 Mapping with SLAM

The generated map has a vector representation with the Cartesian location of mapped landmarks. As the EKF-SLAM is a probabilistic algorithm, the covariance matrix P is also a part of the generated map, because it represents precision of position for each landmark.

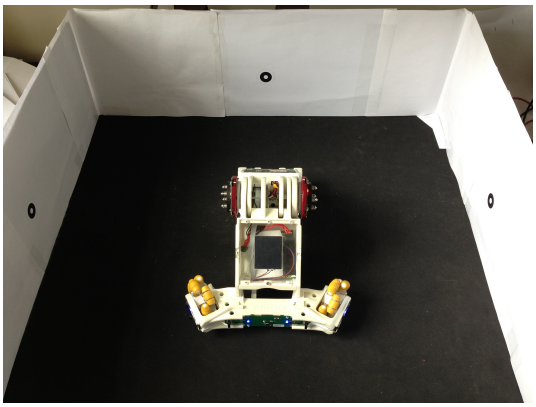

Figure 30: Experimental arena with landmarks and initial position of Active Wheel

Mapping was tested inside a square arena 30 with a landmark on each wall. Real positions of the landmarks were measured for a comparison with the generated map. The problem of measuring the real positions of landmarks is that the initial position of the robot is hard to determine, especially the initial heading of Active Wheel.
The robot performs mapping by repeatedly turning around its center of movement. When the robot again observes the first landmark (the right landmark in figures 30 and 31), we can call it "loop closing" , which is crucial, because all the previously mapped landmarks are shifted in the map according to the difference of the first.

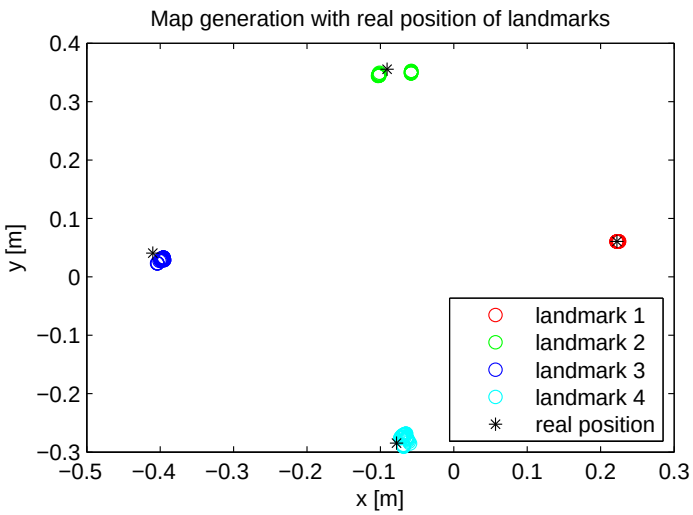


Figure 31: Resulting map

The resulting map in figure (31) shows some key factors of the algorithm. The position of landmarks stabilizes itself a little slowly, which results in a large variance of landmark positions(especially the second landmark). In spite of this fact, the resulting landmark positions have an average position error of 0.0129m. This error looks too big, but with further analysis we realized that the mapping error is caused by bad determination of initial heading (rotation of front). In figure 31 we can see that all landmark positions are rotated around the first landmark (red). The first landmark is mapped in the beginning, even before the robot moves, so the position is accurate. During the next rotation movement the robot maps other landmarks with the initial error of heading, so landmarks 2, 3 and 4 are mapped with this error.
Despite the initial error of the robot position, the resulting map is applicable in the following self-localization improvement and during possible map merging with other modules.

## 7.4 Loop closing

Loop closing denotes a situation when the robot observes the first landmark again after having added other landmarks to the map. During this, the positions of the landmarks mapped until loop closing are shifted in the map according to the difference of the first landmark. This phenomenon is caused by existing covariance between the mapped landmarks.
Loop closing is also one of the major advantages of SLAM compared to simple mapping using self-localization. Simple mapping usually only improves the position of the observed landmarks, while in loop closing the first landmark position after a long-time operation can differ so significantly that the observed landmark will be denoted as a new one. This situation can be eliminated by using SLAM where loop closing ensures the consistency of the map.
During the Loop closing some errors can occur due to the bad error models of landmark measurement or self-localization. In such cases, the mapped landmarks can change their positions inside the map so significantly that the resulting map is unusable.
In the following figure, the positions of the mapped landmarks closely before loop closing and after loop closing are compared.

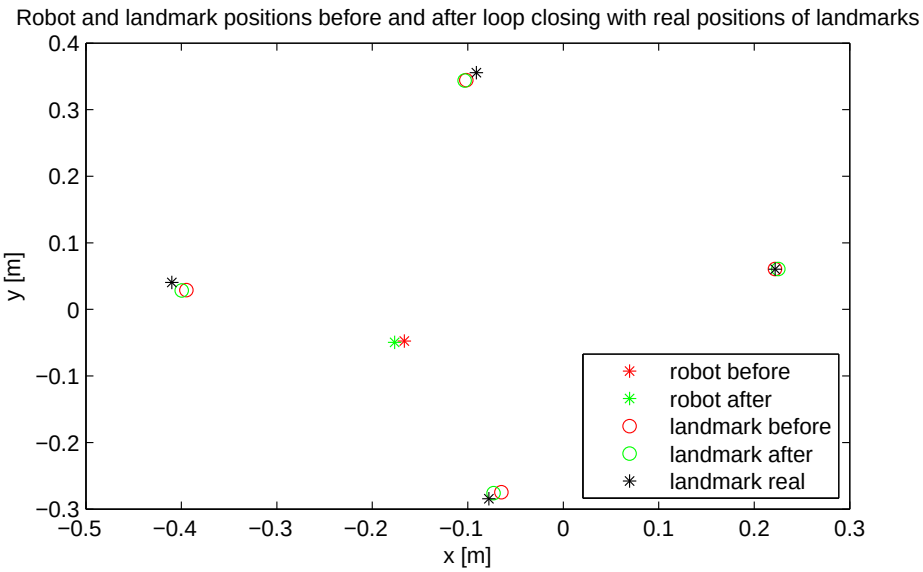


Figure 32: Loop closing effect on mapped landmarks

As seen in the previous figure, the effect of loop closing is positive. Particularly the last mapped landmark(the bottom landmark) improves its position. The left landmark (mapped as the third) changes its position as well. These two landmarks are affected by loop closing most, as they have not yet stabilized their positions and the loop closing comes immediately after their addition to the map. On the other hand, the first two landmarks

(the right and the top) stay in the same positions. The robot position after the observation of the first (the right) landmark in loop closing also changes tis position in order to balance the difference between the observation of the landmark and its map position.

# 8 Conclusion

In this thesis we developed an algorithm for Simultaneous Localization and Mapping (SLAM) based on a camera circular pattern detection. The whole algorithm was implemented on robotic platforms from the Replicator and Symbrion projects which are both aimed at the research of modular robotics focusing on bio-inspired behavior.
At the beginning, we designed self-localization of single modules using dead-reckoning. Due to faulty motor rotation encoders, the self-localization was realized with a motor speed setting and an elapsed time of movement. From the used equation for a differential drive (Scout module) and a three omni-wheel drive (Active Wheel module), we determined odometry parameters by measuring the traveled distances. Using the same data, we also determined odometry error model that is needed to establish Kalman filter Gains for SLAM.
In the section 5, we analyzed camera measurements of cycle patterns. For precise camera measurements, a calibration of the camera was necessary. An error model of camera measurement that is also needed inside SLAM was defined in the next step. Due to a different placement of the camera in robot platforms, the measurements of landmarks have to be converted to the movement center of used platforms.
In the next section we introduced extended Kalman filter SLAM algorithm. The developed solution is capable of mapping circular pattern landmarks. From the mapped landmark, robots can improve their positions during the following observation of the same landmark. As incremental self-localization using elapsed time has additive error, improvement is needed in a long-term use of the robotic module.
At the end we verified the implemented solution by several experiments. At first, we proved filtering with a statical robot on one landmark. The resulting position of landmark stabilizes though the measurements are still noisy. In section 7.2 we analyze self-localization improvement with SLAM. Using of one landmark measurement during the robot's stops between movement, the resulting robot position from the SLAM algorithm always approaches the robot real position measured by external localization. Comparing figures 28 and 29, we can see that the landmark position is also improved and the improvement is bigger until the robot position uncertainty is not higher than the landmark position uncertainty.
In chapter 7.3 is verification of the SLAM mapping capabilities. Although the generated map differs a little (1.29cm) from the real map, the resulting map is consistent and could be used for other purposes.
The last experiment analyzed Loop closing effect on the generated map. Positions of both landmark and robot are affected by loop closing. The most affected were the last two mapped landmarks and also robot position.
Future work on this SLAM solution will include application of motor encoders for better self localization of individual modules. Also, the map generated during on-floor movement could be used in multi-robot organisms. With additional modifications of camera measurement and by incorporation of the z-axis position of the robot and landmark, the algorithm could be used as an estimator of the robot position in robotic organisms. Additional improvement that allow landmark detection during robot's motion will be done.

# References


[1] Kalman filter. http://en.wikipedia.org/wiki/Kalman_filter, Nov. 2012.

[2] Kalman filter. http://www.symbrion.eu/tiki-index.php, Nov. 2012.

[3] Particle filter. http://en.wikipedia.org/wiki/Particle_filter, May 2012.

[4] Galileo. http://en.wikipedia.org/wiki/Galileo_%28satellite_navigation%29, Mar. 2013.

[5] Glonass. http://en.wikipedia.org/wiki/GLONASS, Mar. 2013.

[6] Gps. http://en.wikipedia.org/wiki/Global_Positioning_System#Basic_concept_of_GPS, Mar. 2013.

[7] D. Austin. Non-monotonic geometric mapping for mobile robots, 2000.

[8] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, Sept. 2006.

[9] J.-Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, Feb. 2013.

[10] K. S. Chong and C. L. Kleeman. Accurate odometry and error modelling for a mobile robot. In *In IEEE International Conference on Robotics & Automation*, pages 2783–2788, 1997.

[11] N. Doh, H. Choset, and W.-K. Chung. Accurate relative localization using odometry. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1606–1612 vol.2, 2003.

[12] I. Dryanovski, W. Morris, and J. Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1553–1559, 2010.

[13] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006, 2006.

[14] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart. Vision based position control for mavs using one single circular landmark. *Journal of Intelligent & Robotic Systems*, 61(1-4):495–512, 2011.

[15] J. Esteves, A. Carvalho, and C. Couto. Generalized geometric triangulation algorithm for mobile robot absolute self-localization. In *Industrial Electronics, 2003. ISIE '03. 2003 IEEE International Symposium on*, volume 1, pages 346–351 vol. 1, 2003.

[16] J. Faigl, T. Krajník, J. Chudoba, M. Saska, and L. Přeučil. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *Proceedings of 2013 IEEE International Conference on Robotic and Automation*, Karlsruhe, 2013. IEEE.

[17] K. Harada, S. Russo, T. Ranzani, A. Menciassi, and P. Dario. Design of scout robot as a robotic module for symbiotic multi-robot organisms. In *Micro-NanoMechatronics and Human Science (MHS), 2011 International Symposium on*, pages 511–513, 2011.

[18] A. Howe. Self-assembling modular robotic structures. *Robotics Automation Magazine, IEEE*, 14(4):26–33, 2007.

[19] S. Kernbach, F. Schlachter, R. Humza, J. Liedke, S. Popesku, S. Russo, T. Ranzani, L. Manfredi, C. Stefanini, R. Matthias, C. Schwarzer, B. Girault, P. Alschbach, E. Meister, and O. Scholz. Heterogeneity for increasing performance and reliability of self-reconfigurable multi-robot organisms. *CoRR*, abs/1109.2288, 2011.

[20] T. Krajník. *LARGE-SCALE MOBILE ROBOT NAVIGATION AND MAP BUILDING*. PhD thesis, Czech Technical University in Prague, 2011.

[21] L.-C. Lai, C.-N. Ko, T.-L. Lee, and C.-J. Wu. Time-optimal control of an omnidirectional mobile robot. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–6, 2006.

[22] P. Levi and S. Kernbach. Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, 2010.

[23] H. Liu and G.-H. Pang. Accelerometer for mobile robot positioning. *Industry Applications, IEEE Transactions on*, 37(3):812–819, 2001.

[24] G. Lucas. A tutorial and elementary trajectory model for the differential steering system of robot wheel actuators. `http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html`, Apr. 2013.

[25] T. Marks, A. Howard, M. Bajracharya, G. Cottrell, and L. Matthies. Gamma-slam: Using stereo vision and variance grid maps for slam in unstructured environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3717–3724, 2008.

[26] T. Matsuoka, A. Motomura, and T. Hasegawa. Real-time self-localization method in a dynamically changing environment. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1234–1239 vol.2, 2003.

[27] H. Moballegh, P. Amini, Y. Pakzad, M. Hashemi, and M. Nanniani. An improvement of self-localization for omnidirectional mobile robots using a new odometry sensor and omnidirectional vision. In *Electrical and Computer Engineering, 2004. Canadian Conference on*, volume 4, pages 2337–2340 Vol.4, 2004.

[28] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[29] P. Moubarak and P. Ben-Tzvi. Modular and reconfigurable mobile robotics. *Robotics and Autonomous Systems*, 60(12):1648 – 1663, 2012.

[30] R. Rogers. Improved heading using dual speed sensors for angular rate and odometry in land navigation. In *Position Location and Navigation Symposium, IEEE 1998*, pages 177–184, 1998.

[31] B. Salemi, M. Moll, and W.-M. Shen. Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3636–3641, 2006.

[32] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks, 2002.

[33] J. Solá. Simulataneous localization and mapping with the extended kalman filter. `http://www.joansola.eu/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf`, Feb. 2013.

[34] A. Souza and L. Goncalves. 2.5-dimensional grid mapping from stereo vision for robotic navigation. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pages 39–44, 2012.

[35] J. Suh, S. Homans, and M. Yim. Telecubes: mechanical design of a module for self-reconfigurable robotics. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 4, pages 4095–4101 vol.4, 2002.

[36] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[37] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[38] Z. Winkler. Odometrie. `http://robotika.cz/guide/odometry/en`, Jan. 2013.

[39] D. Wu and H. Su. An improved probabilistic approach for collaborative multi-robot localization. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1868–1875, 2009.

[40] M. Yim, D. Duff, and K. Roufas. Polybot: a modular reconfigurable robot. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 514–520 vol.1, 2000.

[41] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw. Connecting and disconnecting for chain self-reconfiguration with polybot. *Mechatronics, IEEE/ASME Transactions on*, 7(4):442–451, 2002.

# Appendix

## CD Content

In table 1 are listed names of all root directories on CD

| **Directory name** | **Description** |
|---|---|
| bp | bachelor thesis in pdf format. |
| sources | source codes |
| data | measured data for experiments |

Table 1: CD content