UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Kristjan Laht

Robot Localization with Fiducial Markers

Bachelor's Thesis (9 ECTS)

Supervisor(s): Karl Kruusamäe, PhD

Tartu 2022

Robot Localization with Fiducial Markers

Abstract:
Non-industrial robotics is a relatively new field with great potential for growth, but there are still many active research problems that prevent it from becoming ubiquitous. A robot not getting lost is one of those problems. Robots tracking their position based on wheel movement are subject to drift due to uneven surfaces and wheel slippage. One way for robots to determine their position and orientation (pose) without knowledge of its? drift is to use QR code like printable tags called fiducial markers that are simple and fast to recognize from images. These tags must be installed in the environment beforehand. This work integrates global pose from fiducial markers and local pose from wheel rotations into one package to create a system that accounts for the weaknesses of both. A simple environment where the use of this package was successful and a complicated environment where it failed are demonstrated. Fiducial markers used for finding the current pose of the robot are robust solutions in applicable environments, but improvements to robustness are still needed.

Keywords:
Robotics, fiducial markers, localization

CERCS: T125 - Automation, robotics, control engineering

Roboti lokaliseerimine koordinaatmärkidega

Lühikokkuvõte:
Robotid, mida kasutatakse väljaspool tootmisjaamu, on üpris uus valdkond, millel on suur potentsiaal kasvuks, aga siiski on veel mitmeid lahtised probleeme, mis takistavad selliste robotite laialdast kasutuselevõttu. Üks neist probleemidest seisneb robotite asukoha tuvastamises. Robotitel, mis arvutavad oma positsiooni ruumis rataste pöörlemise alusel, võib ebaühtlaste pindade ületamise ja rataste libisemise tõttu esineda triivimist/erinevusi mõõdetud ja reaalse asukoha vahel. Üks viis enda positsiooni ja rotatsiooni (poosi) leidmiseks ilma triivimiseta, on kasutada QR koodi laadseid prinditavaid koordinaatmärke, mida on lihtne ja kiire piltidelt üles leida. Need märgid tuleb eelnevalt installeerida roboti keskkonda. Selle töö panus on integreerida roboti ümbruskonna absoluutse poosi ja rataste suhtelise poosi leidmine ühte paketti, et teha süsteem, mis katab mõlema meetodi vigu. Töös demonstreeritakse lihtsalt keskkonda, kus see õnnestus, ja keerulist keskkonda, kus see ebaõnnestus. Koordinaatmärkide abil poosi leidmine on robustsne lahendus, kuid töökindluse suurendamine on endiselt vajalik.

Võtmesõnad:
Robootika, koordinaatmärgid, lokalisatsioon

CERCS: T125 - Automatiseerimine, robootika, juhtimistehnika

Contents

Unsolved issues

1 Introduction

Robotics is a field with tremendous potential in the future. The most valuable robots are currently created to solve specific tasks on specific production pipelines that cost hundreds of millions of dollars to create, while general-purpose robotics is still in the future. The goal is to have robots usable by all humans to assist in menial, dangerous, or precise tasks.

The author believes there are two major unsolved problems with general-purpose robotics: a sufficiently simple user experience and a sufficiently robust abstraction. A sufficiently simple user experience is not trivially easy, but it should be quickly learnable and guide the user to a workable solution. Microsoft Excel is a great example: it is not trivially easy, but it is the tool where people from various fields and backgrounds can leverage computing to solve their specific problems. Robotics should be analogous. A sufficiently robust abstraction means that a robot must handle failure gracefully: failures must never be catastrophic and can only happen in predefined ways. Either a robot knows how to navigate and complete tasks in its current environment or is smart enough to know not to try.

Most current value-producing robots operate in a strict environment, completing specific repeatable tasks accurately. This limits the applicability of robots. A lot more value can be captured by operating in a less strict environment. The least restrictive environment is the one that already exists, the one where humans currently operate. Teaching robots to handle all of the intricacies of human environments is a very difficult open problem [GF20]. This problem can be simplified by modifying the environment a little. One way to simplify the environment is to add uniquely identifiable images to key locations. These images, and other unique landmarks, are called markers, and allow robots that recognize them to know their location.

This enables precise robotic interactions without the robot needing to fully understand everything about its environment and how it can change. Compared to other methods of finding the current position, markers have well-defined failure states and low computational cost. This thesis aims to improve the method of determining position by using markers. There will also be other methods used to help with this problem, but, where utilized, using markers will be the most robust method that's also capable of correcting global positioning error.

All development and testing were completed on the robotont platform [Rau19], an "open-source platform for robotics education and research" [].

In chapter one, marker-based localization is introduced, giving examples for possible localization techniques. Also, an overview of stochastic filters is provided. In chapter two, functional requirements for this work are described. In chapter three, a design of the system is shown, along with a description of the testing platform. In chapter four, results of the testing and calibration are presented.

Chapter five provides some discussion on things that need improvement for the specific test robot and the robotics field in general, to increase robustness in future robotic solutions and their development.

2 Background

This chapter describes localization, fiducial markers, sensor fusion, and general principles of bayesian filters.

2.1 Localization

Localization is the process by which a robot establishes its own position and orientation, relative to a frame of reference. There are many different modalities possible for localization. It is possible to use multiple ultrasonic beacons to triangulate position (which usually does not help with orientation) [JS05]. Modern robot vacuum cleaners usually use a laser distance finder [Vau+14]. Visual systems can use a camera to construct and localize to a map [FA18]. In visually obstructed environments like fog or smoke, low cost radar can be used [Lu+20].

2.2 Fiducial Markers

Many types of image landmarks can be used for localization. Landmarks can be fiducial markers on the wall, lights in the room [ZZ17], houses in the distance [CRB03]. Some landmarks can be fast to find with image processing tools, for example fiducial markers (see figure 1), a 2D grid of black and white squares, with a black border. [Gar+14]

Figure 1. Examples of ArUco markers, with sizes of 5x5, 6x6 and 8x8. [Gar+14]

Fiducial marker-based localization has many advantages: it can work even when something is partially covering the marker, it provides global knowledge of position, it can be used anywhere where markers can be placed, and it allows for precise positioning [Ols11]. There is a requirement for specific markers to be placed in pre-mapped locations, which means that any unmodifiable environment e.g. public spaces, vacuum of space, inside nuclear reactors is unsuitable for this approach. [Gar+14]

Some fiducial markers are coded with Error Correcting Codes (ECC), which allows them to convey information even when partially occluded or at highly oblique

angles. ECC helps when one square in the marker might be mistaken for another, such as in places where the marker is not much larger than a few pixels on a robot's camera. [Gar+14]

Markers are placed in specific locations in the environment, and those locations are defined in the robot's map of the environment. By seeing a marker, it is simple to calculate the robot's position. How this calculation is done is elaborated in Section 2.6 about coordinate transforms.

2.3 Estimating errors

Estimating errors correctly is central to having a well-defined system. This usually means knowing in great detail the method of operation of sensors and potential error-causing situations. Take the example of a laser range finder. A laser range finder uses a light to measure distance. It does this by sending out a pulse of light, letting it reflect off of something, and having a sensor to register the return. Distance is calculated by measuring the time light travels [JS05]: $d = \frac{\Delta t}{2} \cdot c$, where d is distance, t is the time between sending a light pulse and receiving it, c is light speed in air. This means that the fundamental accuracy is tied strictly to how accurately we can measure the time the light pulse takes. For example, with a timer accuracy of ± 1 ms, the measurement will be off by 150 000 m and with a measurement accuracy of ± 1 ns, the measurement will be off by 0.15 m. Less fundamental errors come from the environment: artificial lights and overexposure can confuse the sensor [KPK01]. Taking into account all of these characteristics and other factors, can only work in a laboratory setting.

That is why modern *solutions* rely on multiple sensors with different error characteristics and why modelling those errors is important [Sas02]. Several sources make it possible to create simplified physical models that can be analyzed and allow the system to ignore data that lies well outside of what is expected.

There are two types of errors [SD05]:

- static errors where the measured value doesn't change and any measured changes are random measurement noise. It is often possible to simplify this to a Gaussian with some mean and variance and gain huge speedups for computations.

- dynamic errors that come from changing environmental conditions. Often these errors are non-linear.

An example of static error is a camera measuring object size in pixels. Even when the distance to the object never changes, the pixel count might vary due to the discretization of a continuous variable - camera sensors have discrete pixels that discretize continuous wave-like light.

Static errors are straightforwardly estimated using the Kalman filter. If all the errors have a normal (Gaussian) distribution, the Kalman filter is the best possible linear estimator. [HRW12]

An example of a dynamic error is a simple computer vision algorithm recognizing objects when the camera has any automatic adjustments turned on. Auto stabilisation might make an algorithm looking for straight lines fail, when it skews pieces of the frame differently and straight lines turn into bendy ones. Auto focus can remove lines altogether. Auto exposure might make an algorithm looking for specific colours or contrasts fail, since the colours and lightness values will be all different after. All of these examples are highly non-linear and can not be modelled with gaussians.

Currently many fast methods suitable for real-time robot control that do not handle non-linear errors [Sas02]. There are methods to handle specific non-linearities, like an extended Kalman filter, that only needs to assume linear errors in small local time scales and allows for not some non-linearity for the overall problem [Fox+99].

2.4 Sensor fusion

Sensor fusion is a field that uses multiple data points from different sensors with drastically different error characteristics to combine the best features of each sensor to minimise error [Elm02]. A simple example can be a train that can move forwards and backwards, with two sensors, one measuring wheel movement and the other GPS position.

Measuring wheel movement is locally accurate but wheels slip and accumulate errors. Slippage can be modelled by a linearly increasing error value in the position received from the wheel sensor.

The GPS sensor is a global measurement: not subject to drift, but still subject to noise. GPS has a relatively static error value as long as a signal is available. If there is no signal, for example in the case of the train passing through a tunnel or a forested area, there is a large discontinuity in the error signal. The discontinuity makes the error in the system non-gaussian, which makes statistical modelling and analysis difficult.

These sensors exhibit a significant difference in the boundaries where they provide useful data. This leads to the solution of combining the best aspects of multiple different sensors. One of the most popular ways to do that is by using a Kalman filter, which models gaussian problems with gaussian error distributions.

2.5 Bayesian filter

Filters in signal processing context provide a way to extract a more meaningful signal from data. The simplest of such filters is a low-pass filter, which allows all signals lower than a cutoff frequency to pass - used in speakers that are designed to produce low notes, in order to remove all the high notes from the signal and avoid damaging the speakers. In a more abstract description: when there's prior information that the signal is low frequency, cutting off all the high frequencies will reduce noise.

Stochastic filters also extract signals from data, but in a way that incorporates information from the stochasticity of measurements. A common example is weighing a person on a scale. One set of measurements might read w = [87.8kg 83.3kg 81.9kg 87.9kg 83.2kg 85.3kg 87.5] and without knowing more about the scale or how measurements are done, it is best to assume an uniform distribution and therefore the estimated weight w = mean(w) = 85.3.

2.6 Coordinate transforms

Coordinate transforms are used to convert coordinates stored in one frame of reference, to coordinates stored in another. For example, a computer vision algorithm recognizing objects in a scene will output the position of the objects relative to the camera it uses. But to interact with or avoid those objects, a robot holding the camera needs to know where the objects are located relative to itself. This is done by transforming the coordinates of the position in the camera frame to the robot base frame.

A common method to calculate transforms is to convert the wanted transform to a homogeneous transformation matrix, in which case applying the transform or composing multiple transforms together is just a matrix multiplication [Har80]. For example, a simple translation in 3 dimensions can be defined as [Har80]

$$\text{Translation}(a_x, a_y, a_z) = \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & 1 & 0 & a_y \\ 0 & 0 & 1 & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

While rotation along the x axis (often referred to as roll) can be defined as [Har80]

$$\text{Rotation}_\text{x}(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Similarly for the y and z axis (commonly known as pitch and yaw) [Har80]

$$\text{Rotation}_{\text{y}}(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\text{Rotation}_{\text{z}}(\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

These can be combined together through matrix dot product as [Har80]

$$\text{T} = \text{Translation}(a_x, a_y, a_z) R_z(\theta_z)\, R_y(\theta_y)\, R_x(\theta_x) \tag{5}$$

This product can then be precomputed for each actual transform that needs to be executed, leading to fast code with only some multiplications and additions and no trigonometric functions. Furthermore, each transform itself is composable. The example of the camera on the robot above can be described as a composition of two transforms: one from the robot base to the camera and one from the camera to the object.

$${}^{\text{robot base}}T_{\text{object}} = {}^{\text{robot base}}T_{\text{camera}} \cdot {}^{\text{camera}}T_{\text{object}} \tag{6}$$

Another useful property of using homogeneous matrices as transforms is that to get the reverse transform, only calculating the inverse matrix is needed. This means that if the computer vision algorithm from above didn't give objects with regard to (wrt.) the camera position, but instead gave us the camera position wrt. to the object, the equation above would instead look fairly similar:

$${}^{\text{robot base}}T_{\text{object}} = {}^{\text{robot base}}T_{\text{camera}} \cdot \left({}^{\text{object}}T_{\text{camera}}\right)^{-1} \tag{7}$$

2.7 Robot Operating System

Robot Operating System (ROS) is an open-source robotic middleware platform [Sta18]. Fundamentally, ROS is a message passing runtime for the communication of software processes (nodes) through predefined messages. The utility in ROS comes from the collection of robotics libraries and tools that are all interchangeable. The system contains state-of-the-art algorithms that solve complex problems in robotics. ROS also includes simulators and visualisation tools to develop and debug

new robotics applications. ROS is also a packaging system and an online repository of C++, Python and Lisp packages [Sta18].

The central concept of ROS is a node. ROS nodes are executables that use the ROS runtime to communicate with other nodes. All nodes are named. ROS nodes send messages over topics. Topics are named buses that only pipe one specified message type. ROS nodes can subscribe to multiple topics, and can publish on multiple topics. There are many standard messages that nodes communicate to other nodes, but the messages can also be user defined. Examples of messages are actuator commands, state information, and sensor data. ROS messages allow many different algorithms and implementations to be interoperable and have created an ecosystem of packages. Nodes can also advertise services, which allow other nodes to request a single action, which has a defined start and end. [Sta18]

3 Requirements

3.1 Objective

With this work, robots will be able to move in more environments that so far have not been accessible to robots. A robot equipped with a predefined map of fiducial markers will have estimates for its location and the estimates' errors and manage to navigate inside that map. A ROS node is created that transforms marker locations according to the map. A working system is integrated that resolves the robot's position and heading with a low error estimate whenever a marker is in view and with a gradually growing error estimate in all other cases.

3.2 Functional requirements

For a well functioning system, the following requirements must be fulfilled:

- A ground robot is able to determine its pose in marker-rich indoor area
- Error estimation for the current position and heading is plausible at all times.
- A robot with degraded sensor performance lowers the prior trustworthiness for that sensor's measurement.

3.3 Software requirements

Ubuntu 20.04 running ROS Noetic with Python 3.8.

3.4 Hardware requirements

A ROS robot equipped with a calibrated camera and wheel odometry sensors. This thesis used the Robotont [Rau19] robot with Intel Realsense D435 camera and brushed Pololu 12V DC motors with encoders that have 1200 counts per revolution.

4 Design

In this chapter, an overview of the complete annotated ROS node network is shown. The testing environments are described.

4.1 Software overview

This section will give the overview of how robot software wires together to fulfill the objective of this thesis.

This work depends on the following ROS packages: robotont, realsense-ros, apriltag_ros, robot_localization and all their dependencies. It also depends on the following python packages: numpy, matplotlib.

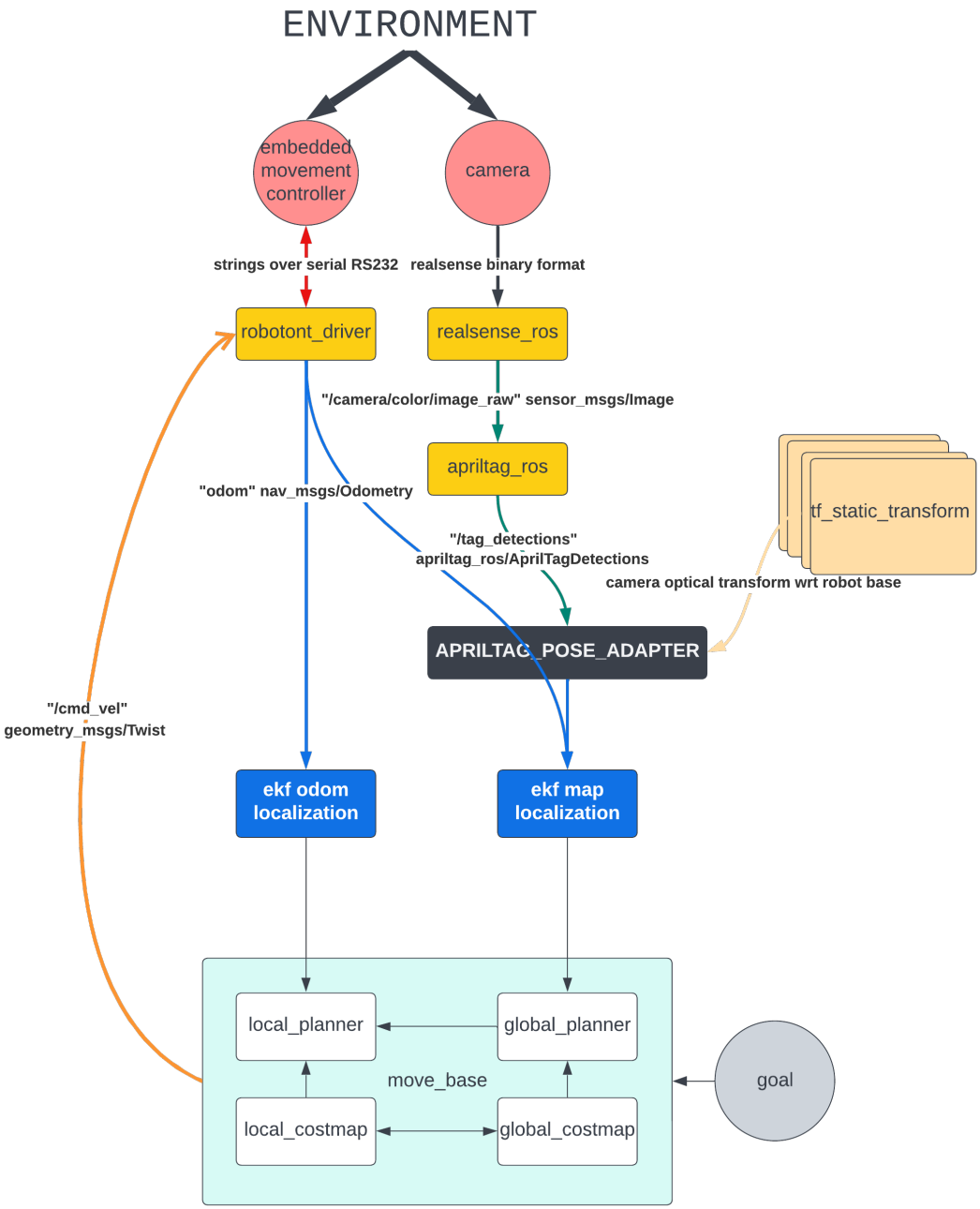


Figure 2. Colorcoded overview of ROS nodes used and how they were wired together

Firstly, data is acquired from the environment in the form of photons by the camera and in the form of static forces by the wheels.

Wheel motors have sensors called encoders attached to them that measure how much they turned. That measurement data is converted into velocities and then compared to the current set speed and based on that, the motor effort (the energy used to turn the wheel) can be increased or reduced. Movement firmware that runs the motors also transforms the encoder data into robot position and velocity and sends it back to the ROS node robotont_driver.

Camera processes incoming photons into an red green blue (RGB) image and then sends the image over USB to the ROS node realsense_ros.

Robotont_driver has two main tasks: it subscribes to a topic /cmd_vel to which any other node can send new velocity commands, after which it formats and forwards those commands to the embedded movement controller; it also sends out robot pose (position and orientation) odometry messages.

Realsense_ros publishes ros topics /camera/color/image_raw and /camera/-color/camera_info with the latest RGB images. Apriltag_ros subscribes to those images and applies a fast algorithm to detect markers (otherwise named tags) in those images. Then it publishes an array of all tag poses it found in the latest image.

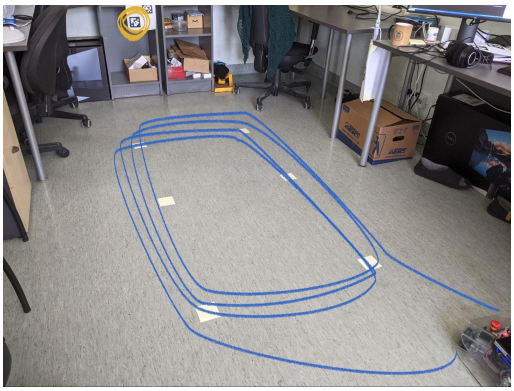
Those detected poses get sent to the apriltag_pose_adapter node and are transformed to the robot base frame of reference from the camera optical RGB sensor frame of reference. Next the tag pose wrt robot base is transformed into robot pose wrt map by using the fact that all tags poses are predefined. This is now another source of robot pose odometry messages. It also adds noise variance values to the poses contained in the odometry message.

Then two localization nodes that run the extended Kalman filter (EKF) algorithm use the odometry messages to extract the signal of how the robot really moved from noisy tag measurements and wheel positions. EKF map localization outputs a global estimate for robot pose while EKF odom localization outputs a local estimate. This is done separately because as the map localization gets more data from visible apriltags, the pose estimate may contain discontinuities. Path planners are not designed to handle such changes and that is why two path planners are needed to run simultaneously: a global one that plans the future movement in general terms, and a local one that runs in soft realtime and handles local changes quickly.

EKF odom localization messages are sent to the local planner node. EKF map localization messages are sent to the global planner node. The move_base nodes then communicate between each other and send velocity messages back to robotont_driver to move the robot as the planner suggests.

4.2 Experiment 1 - Proof of Concept

(a) Floor with markings that help keep a consistent route



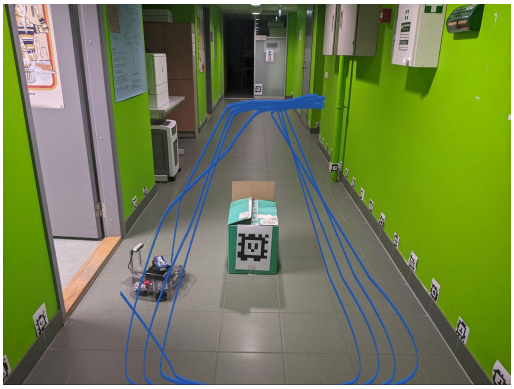
(b) Marked in blue is the Path the robot takes, marked in orange is the marker that helps correct for drift

Figure 3

The first experiment is a very simple non-simulation environment, where using an extended Kalman filter and one tag with a known location successfully corrects drift in yaw measurement for the robot. The robot was manually remote controlled to drive around in a rectangle for 5 laps. The rectangle is marked on the floor to serve as a guide to the human remote control (Figure 3a). The robot followed the path laid out in blue (Figure 3b), while using the fiducial marker circled in orange as a guide to correct for drift (the gradual increase in absolute error after integrating velocity data for position). The position and orientation of the robot was measured over time.

4.3 Experiment 2 - Quantifying results

The second experiment was conducted in a marginally more complex environment. (see Figure 4a) The robot was manually driven back and forth through a corridor, taken through the corridor turn, rotated 360 degrees to one side and even driven sideways. In contrast with the first experiment, the flooring is not smooth. (see Figure 4b) The path the robot takes is much more complicated because it consists of more complex parts than straight lines and right turns. The markers were arranged in regular intervals of 30cm to make mapping them easier. Mapping the large markers in the middle of the room was achieved by way of measuring tape.



(a) Robot in the corridor. Marked in blue is the path the robot took. There are 8cm sized fiducial markers on the walls, and 18cm sized fiducial markers at the ends of the corridor.

(b) Close up of the floor and small markers.

Figure 4

5 Results

In this chapter, the results of the two experiments for accurately estimating the robot's pose and correcting for drift are presented. In the first experiment the system successfully corrected for drift. In the second experiment the 45 markers used could not help in localizing the robot and even made the position estimation less accurate. Although the first approximately 60 attempts did not yield expected results, it does not imply that the system cannot function as intended.

5.1 Experiment 1 - Proof of Concept

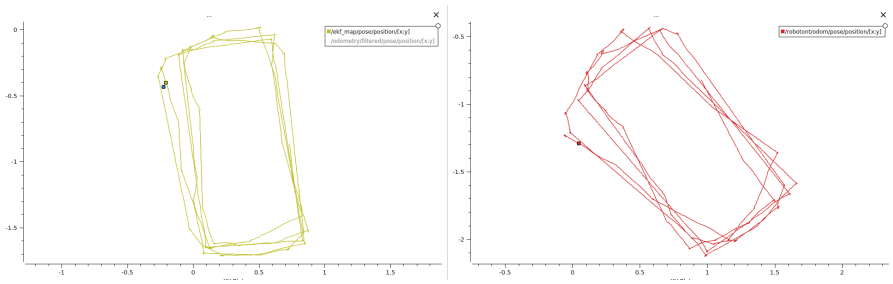


Figure 5. On the left is the robot position filtered through the extended Kalman filter. On the right is the same position without using markers for correction.

As can be observed from figure 5, without fiducial-based global correction, corners of the rectangle are significantly different at the end. Meanwhile, incorporating the data from fiducial markers into its model allows the robot to maintain a fairly accurate path for the whole.

5.2 Experiment 2 - Quantifying results

Similarly to experiment 1, the robot was manually controlled. By default, just using one marker for the whole corridor does not give good enough results - there is some improvement in orientation tracking but by only occasionally having some markers in visual range, the estimation to changes drastically and has many discontinuities (jumps). Unfortunately in this experiment, using more markers only increases the instability. These kinds of planar fiducial markers (tags) also have a fundamental ambiguity problem where small markers parallel to the image plane have two possible projections, which leads to flipping orientations [Chn+19]. (see figure 6)

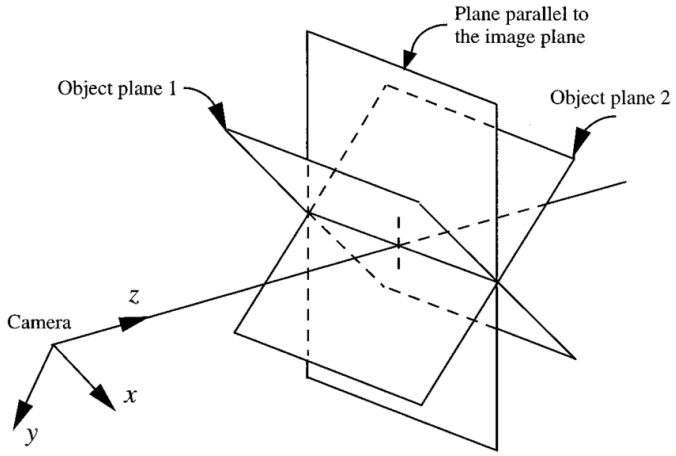
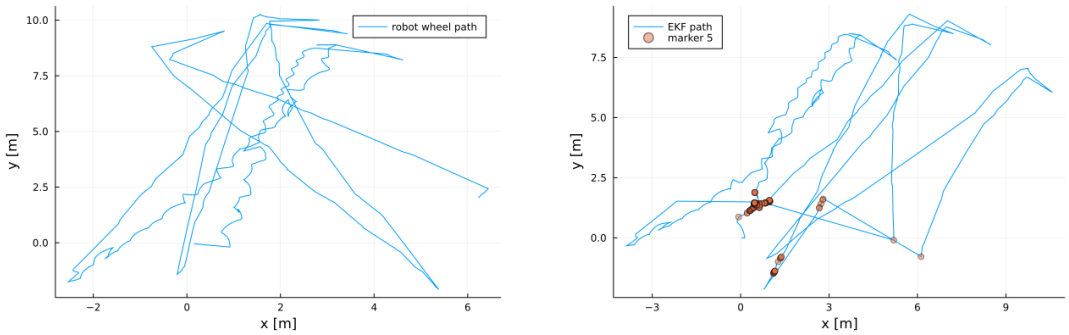


Figure 6. Tags on object plane 1 and object plane 2 are indistinguishable from each-other when the distance to camera is large or there is a lot of image noise [ODD96].

There are multiple methods to alleviate this problem: a multi-graph averaging approach [Chn+19], a method to estimate better camera positions by exploiting redundant information in the markers and carefully choosing the specific transform to do [CB14], and others [JMS17] [SP07] [ODD96]. There are performance and complexity trade-offs for each solution. This work is optimized for less complexity. Simple temporal averaging and outlier rejection is applied: any orientation received in the last 100ms that is rotated more than 1 radians compared to others, will be rejected. This solution sometimes throws away useful data and occasionally still

fails. An even less complex solution was eventually adopted by the author: printing bigger markers.

Besides marker pose ambiguity and instability, the biggest contributor to experiment 2 failing was the EKF itself. Kalman filters that handle the kind of highly sparse (nonlinear) data present in tags that only show up occasionally are very new [Piz+22].



(a) Robot position as acquired from wheel sensors.

(b) Robot position as returned by the EKF.

Figure 7. Experiment with one small marker. Note how after every time the robot sees marker 5, future movement is in a wholly different direction. This is because of large yaw uncertainty, even after seeing a marker, changes the predicted orientation of the robot, but not enough to change it to be correct.

6 Discussion

This chapter lists reasons why experiment two failed and gives an informal overview of the many ways developing a robotic solution may become encumbered by accidental complexity.

6.1 What to look out for when using mapped fiducial markers

Firstly, the poses of markers must be accurate. Even if slightly incorrect poses are not immediately recognizable, they result in a lot of hard to handle instability in calculating the pose. Two visible markers at the same time exasperate the problem.

Secondly, using small markers allowed the system to calculate the correct position, however rotation flipped between the two possible projections every 10 milliseconds. See section 5.2. While this error can be reduced with more advanced methods, it is just easier to have bigger markers.

Thirdly, using Kalman filters for sensor fusion, even those variants that can handle some non-linearity like unscented or extended Kalman filters, might not be the best option. Particle filters seem much more suited to this kind of problem. (see [Sas02] for a comparison of different methods)

6.2 Other problems encountered

Over the course of conducting these experiments, many problems arose which were not directly related to the task at hand? Some of these problems are not only relevant to robotics as a field, but they also exemplify the general problems that must be solved for a more robust abstraction. Many of these are specific to the robotics platform used for testing, but as far as the author is aware, other platforms have their own specific issues and no current platform can be claimed to work without workarounds for common issues. Robotics is especially challenging since it does not just exist in the crossroads of many disparate fields, but often requires solutions in the tail ends of the field. It does not help that non-industrial robotics is a new field with many experimental solutions which often break in novel ways. In robotics, almost all abstraction layers are leaky.

Temperature Realsense D435 camera module overheated at 40 degrees Celsius when the room temperature was 30 C. The solution was to disable infrared laser emitters and to not use depth data during the days when the weather was too hot.

Physical connections The USB-C cable between the camera and robot's primary computer ended up being non-reversible and required reseating sometimes to really reset the camera. Debugging this by using a different cable and a different USB port did not uncover the problem. Later it turned out that the other USB port was also non-functional, but because the faulty ports were not detected, considerable time was spent on searching for problems in the software stack.

Wireless connections The robot runs its own hotspot to allow use in new environments wirelessly. This is not configured to access the internet. Due to the author lacking Linux networking experience, setting up the wifi to automatically switch to an internet enabled network when in range failed and soft-bricked the robot. Fixing required the operating system to be reinstalled. Therefore most of the development of a freely moving robot happened on a physical ethernet connection, which is paradoxical but works on small scales. Another problem came up when testing the robot untethered - wheel motors create RF interference, which was the probable cause of approximately 5 second wifi latency spikes.

Underlying firmware Robot firmware had a quirk of reporting its own movement position in the wrong coordinate frame. Fixing it required piecing together an embedded dev environment from the existing codebase by travelling the tree of 'include' dependencies noting them down.

Underlying OS / environment There were also problems with automatic installation of dependencies failing, due to binary or other incompatibilities between different versions of dependencies. This required the ability to debug Linux kernel module loading; finding solutions to multiple conflicting shared libraries; breaking changes in programming languages (python 2 vs python 3). Other problems encountered included the primary package repository public key expiring, with the maintainers only noticing a few days later. In comparison, hundreds of thousands of engineers would have their work interrupted if maven or pip or npm were inaccessible for multiple days, causing damages of hundreds of millions in lost work.

Almost all packages had out-of-date or missing documentation.

Several of the above problems can be attributed to the ROS ecosystem lacking reproducible builds and installing everything in the global environment. Many features require ad-hoc fixes shared by word of mouth or workarounds found in the project's issue tracker on Github. Even in cases where Ansible was used in order to facilitate creating a well known working environment, it was still required to mix and match solutions from 3 different branches of the ansible description to get things working.

Installing everything to the global environment also breaks assumptions made by other packaging systems. Installing packages from pip that have not yet been specifically packaged for the ROS environment has a high chance of breaking working systems due to differences between version resolution algorithms and conflicting binary dependencies.

Creating and interacting with ROS packages at all requires one to learn a scarcely documented meta-meta build system called catkin that modifies and in some cases subsumes the meta build system cmake, which generates the make files that are actually used to build packages. But launching is a different story altogether as ROS programs require one to learn another totally different file format with which to describe connections and parameters between ROS programs. This tool has existed from at least 2011 and despite being widely used, also has confusing documentation on anything besides the simplest use cases.

References

[] *robotont*. URL: http://robotont.ut.ee/main?lang=en (visited on 04/19/2021).

[CB14] Toby Collins and Adrien Bartoli. "Infinitesimal Plane-Based Pose Estimation". In: *International Journal of Computer Vision* 109.3 (Sept. 1, 2014), pp. 252–286. ISSN: 1573-1405. DOI: 10.1007/s11263-014-0725-5. URL: https://doi.org/10.1007/s11263-014-0725-5 (visited on 05/06/2022).

[Chn+19] Shin-Fang Ch'ng et al. "Resolving Marker Pose Ambiguity by Robust Rotation Averaging with Clique Constraints". In: *arXiv:1909.11888 [cs]* (Sept. 26, 2019). arXiv: 1909.11888. URL: http://arxiv.org/abs/1909.11888 (visited on 05/06/2022).

[CRB03] Australian Centre, Field Robotics, and Tim Bailey. "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments". In: (Oct. 9, 2003).

[Elm02] Wilfried Elmenreich. "Sensor Fusion in Time-Triggered Systems". PhD thesis. Jan. 1, 2002.

[FA18] Maksim Filipenko and Ilya Afanasyev. "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment". In: 2018. DOI: 10.1109/IS.2018.8710464.

[Fox+99] Dieter Fox et al. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots". In: Proceedings of the National Conference on Artificial Intelligence. Jan. 1, 1999, pp. 343–349.

[Gar+14] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* 47.6 (June 1, 2014), pp. 2280–2292. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320314000235 (visited on 04/19/2021).

[GF20] Frank Guerin and Paulo Ferreira. "Robot Manipulation in Open Environments: New Perspectives". In: *IEEE Transactions on Cognitive and Developmental Systems* 12.3 (Sept. 2020), pp. 669–675. ISSN: 2379-8920, 2379-8939. DOI: 10.1109/TCDS.2019.2921098. URL: https://ieeexplore.ieee.org/document/8731700/ (visited on 08/16/2021).

[Har80] Robert M Haralick. "Using perspective transformations in scene analysis". In: *Computer Graphics and Image Processing* 13.3 (July 1, 1980), pp. 191–221. ISSN: 0146-664X. DOI: 10.1016/0146-664X(80)90046-5. URL: https://www.sciencedirect.com/science/article/pii/0146664X80900465 (visited on 05/07/2022).

[HRW12] Jeffrey Humpherys, Preston Redd, and Jeremy West. "A Fresh Look at the Kalman Filter". In: *SIAM Review* 54 (Nov. 8, 2012). DOI: 10.1137/100799666.

[JMS17] Pengju Jin, P. Matikainen, and S. Srinivasa. "Sensor fusion for fiducial tags: Highly robust pose estimation from single frame RGBD". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017). DOI: 10.1109/IROS.2017.8206468.

[JS05] Antonio Jiménez and Fernando Seco. "Ultrasonic Localization Methods for Accurate Positioning". In: (Oct. 29, 2005).

[KPK01] Ari Kilpelä, Riku Pennala, and Juha Kostamovaara. "Precise pulsed time-of-flight laser range finder for industrial distance measurements". In: *Review of Scientific Instruments* 72.4 (Apr. 2001). Publisher: American Institute of Physics, pp. 2197–2202. ISSN: 0034-6748. DOI: 10.1063/1.1355268. URL: https://aip.scitation.org/doi/abs/10.1063/1.1355268 (visited on 05/09/2022).

[Lu+20] Chris Xiaoxuan Lu et al. "See Through Smoke: Robust Indoor Mapping with Low-cost mmWave Radar". In: *arXiv:1911.00398 [eess]* (May 5, 2020). arXiv: 1911.00398. URL: http://arxiv.org/abs/1911.00398 (visited on 05/09/2022).

[ODD96] Denis Oberkampf, Daniel DeMenthon, and Larry Davis. "Iterative Pose Estimation Using Coplanar Feature Points". In: *Computer Vision and Image Understanding* 63 (May 1, 1996), pp. 495–511. DOI: 10.1006/cviu.1996.0037.

[Ols11] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[Piz+22] Ricardo Piza et al. "Nonuniform Dual-Rate Extended Kalman-Filter-Based Sensor Fusion for Path-Following Control of a Holonomic Mobile Robot with Four Mecanum Wheels". In: *Applied Sciences* 12 (Mar. 31, 2022), p. 3560. DOI: 10.3390/app12073560.

[Rau19] Renno Raudmäe. "Avatud robotplatvorm Robotont". Accepted: 2019-06-14T09:04:25Z Journal Abbreviation: Open source robotics platform Robotont. Thesis. Tartu Ülikool, 2019. URL: https://dspace.ut.ee/handle/10062/64341 (visited on 04/19/2021).

[Sas02] J. Z. Sasiadek. "Sensor fusion". In: *Annual Reviews in Control* 26.2 (Jan. 1, 2002), pp. 203–228. ISSN: 1367-5788. DOI: 10.1016/S1367-5788(02)00045-7. URL: https://www.sciencedirect.com/science/article/pii/S1367578802000457 (visited on 05/10/2022).

[SD05] Thierry Savin and Patrick S. Doyle. "Static and Dynamic Errors in Particle Tracking Microrheology". In: *Biophysical Journal* 88.1 (Jan. 1, 2005), pp. 623–638. ISSN: 0006-3495. DOI: 10.1529/biophysj.104.042457. URL: https://www.sciencedirect.com/science/article/pii/S0006349505731362 (visited on 05/10/2022).

[SP07] Gerald Schweighofer and Axel Pinz. "Robust Pose Estimation from a Planar Target". In: *IEEE transactions on pattern analysis and machine intelligence* 28 (Jan. 1, 2007), pp. 2024–30. DOI: 10.1109/TPAMI.2006.252.

[Sta18] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: https://www.ros.org.

[Vau+14] F. Vaussard et al. "Lessons learned from robotic vacuum cleaners entering the home ecosystem". In: *Robotics and Autonomous Systems*. Advances in Autonomous Robotics — Selected extended papers of the joint 2012 TAROS Conference and the FIRA RoboWorld Congress, Bristol, UK 62.3 (Mar. 1, 2014), pp. 376–391. ISSN: 0921-8890. DOI: 10.1016/j.robot.2013.09.014. URL: https://www.sciencedirect.com/science/article/pii/S0921889013001899 (visited on 05/09/2022).

[ZZ17] Chi Zhang and Xinyu Zhang. "Pulsar: Towards Ubiquitous Visible Light Localization". In: *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. MobiCom '17. New York, NY, USA: Association for Computing Machinery, Oct. 4, 2017, pp. 208–221. ISBN: 978-1-4503-4916-1. DOI: 10.1145/3117811.3117821. URL: https://doi.org/10.1145/3117811.3117821 (visited on 05/09/2022).

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kristjan Laht**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Robot Localization with Fiducial Markers ,
   (title of thesis)

   supervised by Karl Kruusamäe.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kristjan Laht
10/05/2022