TÉCNICO LISBOA

Outdoor Multi-Sensor Navigation of an Unmanned Ground Vehicle

David Miguel Elias Dias

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Pedro Manuel Urbano de Almeida Lima
Dr. Alberto Manuel Martinho Vale

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Alberto Manuel Martinho Vale
Member of the Committee: Dr. Bruno João Nogueira Guerreiro

November 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Resumo

Desde o desenvolvimento do Sistema de Posicionamento Global (GPS), o problema de localização no exterior foi significativamente atenuado devido à disponibilidade de estimativas de posição. No entanto, em certas condições, localização ainda se revela desafiante. Atualmente, em áreas como cidades, a existência de estruturas com elevada dimensão limita o número de satélites disponíveis para obter uma estimativa de posição. Nestes casos, os recetores GPS podem não ter disponíveis sinais de satélites suficientes para calcular uma estimativa. Nessas condições, é necessário confiar em métodos de dead reckoning para atualizar a estimativa de posição. Ainda assim, correções para essas estimativas precisam ser realizadas à medida que sua incerteza aumenta. Nesta tese, apresentamos métodos de localização para um veículo terrestre não tripulado. Os métodos desenvolvidos pretendem diminuir a sua dependência nas medições de GPS, confiando em diferentes configurações de odometria e duas Unidades de Medida Inerciais (IMUs) para atualizar a estimativa de posicionamento. Finalmente, realizámos um conjunto de experiências no interior e exterior para avaliar a precisão dos métodos propostos quando comparados a uma estimativa obtida usando medidas de GPS. Os resultados mostram que os métodos fornecem uma estimativa de posição e orientação fiável para curtas distâncias.

Palavras-chave: Localização, Calibração da odometria, Modelação de sensores, Fusão de dados, Veículo terrestre não tripulado

Abstract

Since the development of the Global Positioning System (GPS), the outdoor localization problem was significantly overcome due to the availability of outdoor position estimates. Yet, localization is still a difficult problem under certain constraints. Currently, in several areas such as cities, the urban canyon often limits the number of satellites in view used to obtain a position estimate. In these cases, the GPS receivers cannot obtain enough satellites signals to compute an estimate. In these conditions, it is often necessary to rely on dead reckoning methods to update the position estimate. Still, corrections to these estimates need to be performed as its uncertainty increases. In this thesis, we present localization methods for an Unmanned Ground Vehicle (UGV). The developed methods intend to decrease the resilience of GPS measurements by relying on different configurations of wheel odometry and two Inertial Measurement Units (IMUs) to update the pose estimate. A set of indoor and outdoor experiments were performed to assess the accuracy of the proposed methods when comparing these to an estimate obtained using GPS measurements. Results showed that our methods do provide a continuously reliable pose estimate over short distances.

Keywords: Localization, Odometry calibration, Sensor modeling, Sensor fusion, Outdoor unmanned ground vehicle

Contents

List of Tables

List of Figures

Acronyms

DOF Degrees Of Freedom.

EGNOS European Geostationary Navigation Overlay Service.

EKF Extendend Kalman Filter.

GPS Global Positioning System.

IMU Inertial Measurement Unit.

IP Ingress Protection.

MBZIRC Mohamed Bin Zayed International Robotics Challenge.

MEMS Microelectromechanical systems.

MOCAP Motion Capture.

MSAS Multi-functional Satellite Augmentation System.

ROS Robot Operating System.

RTK Real-Time Kinematic.

SBAS Satellite-based Augmentation System.

SI International System of Units.

UAV Unmanned Aerial Vehicle.

UGV Unmanned Ground Vehicle.

UTM Universal Transverse Mercator.

WAAS Wide Area Augmentation System.

Chapter 1

Introduction

This chapter entails a small overview of outdoor mobile robotics applications and the motivation for the work carried out during this thesis, followed by a description of the thesis' objectives. To finalize, we detail the thesis' outline, mentioning the work carried in all its chapters.

1.1 Overview

Robotics and automation, in general, have been topics with growing interest by the industry and general public. The economic benefits brought by this technology have led to numerous applications, especially in the industry [1].

In factories, warehouses and distribution centers, robotic applications have been a reality for several years, yet, activity in more complex environments is still a subject of research. While in controlled environments, such as factories, the working conditions are predictable, in outdoor environments, the unpredictability of the environment and its components represents a great challenge.

Tasks performed in dynamic environments, where a high flexibility is required, mobile robots, specifically, Unmanned Ground Vehicles (UGVs) might be the best option. UGVs have been employed in several applications. Previous research showed the employment of these robots in fields such as agriculture and, search and rescue [2, 3]. However, even in some repetitive tasks, such as in construction, we have not seen many applications of automation [4]. Due to multiple challenges, robotics applications in our day to day are still limited.

1.2 Motivation

Unstructured environments and their unpredictability pose a serious challenge for robots and humans, not only not knowing what does the environment look like, but also take into account the unpredictable behaviors that bodies might take within it. All these increase the complexity of performing outdoor tasks. For the safe and efficient deployment of UGVs, efficient methods are required to gather as much useful information from the environment as possible.

One way to address this is by increasing the interest in the subject, which generates new approaches to the problem. This can be achieved in several ways, such as, promoting discussions, funding research, creating more focused degrees programs and robotics competitions.

The latter helped popularize several robotics use cases. An example of this is iRobot's Roomba [5]. This robot's first prototype was developed in the Massachusetts Institute of Technology for participation in an event known as the AI Olympics. Later, this robot came to be a successful robotic product in the consumer market.

Robotics competitions help promote and fund, directly or indirectly, research made in educational centers and companies [6]. The competitions' challenges formulation is made in line with the competition aim and by doing so, research in specific fields is promoted, increasing the knowledge base in the specific area.

The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) is a robotics competition which started in 2017. It focuses on enabling technologies for applications in several areas, these include, disaster response, domestic tasks, transport, and construction, with the aim of promoting "robots working autonomously in dynamic, unstructured environments, while collaborating and interacting with other robots" [7].

MBZIRC 2020 will take place in United Arab Emirates' capital, Abu Dhabi, in February 2020. The challenges for this edition consist of:

- Challenge 1 - a team of Unmanned Aerial Vehicles (UAVs) has to autonomously track and interact with a set of objects (for example intruder UAVs) following 3D trajectories inside the arena.

- Challenge 2 - a team of UAVs and a UGV have to cooperate in order to autonomously locate, pick, transport and assemble different types of brick-shaped objects to build pre-defined structures, in an outdoor environment.

- Challenge 3 - a team of UAVs and a UGV must cooperate in order to autonomously extinguish a series of simulated fires in an urban high rise building firefighting scenario.

- The Grand Challenge - requires a team of robots (UAVs and UGVs) to compete in a triathlon type event that combines Challenges 1, 2 and 3.

Above the challenges inherent in performing such tasks, there are some imposed restrictions. One of these regards the use of Global Positioning System (GPS). GPS localization estimates are not available everywhere, when available, this is a widely used mechanism for lowering position error which, in a case of lack of satellites in line of sight it may fail. Although the proposed challenges will be held mostly outdoors, where GPS signal should be available, its use is discouraged.

The Institute for Systems and Robotics - Lisbon from Instituto Superior Técnico, along with the Robotics, Vision and Control Group from the University of Seville applied to jointly participate in MBZIRC 2020, and were accepted as a participating team.

1.3 Objectives

Localization still is a challenging task under some constraints [8]. The aim of this thesis is to modify an UGV for outdoor use and explore localization techniques taking into account such constraints. These include avoiding the use of GPS assisted localization methods.

Such methods must be evaluated by performing real-world experiments in an outdoor environment. The methods must be robust and deployed in a UGV capable of outdoor exploration.

The UGV should be endowed with a robotic arm which has the ability to performing tasks, such as the ones mentioned in the MBZIRC challenges' description or in any of the fields mentioned earlier.

The implemented methods do not rely on GPS measurements. However, an estimate using GPS's measurements is computed in order to obtain a pose reference to which our methods are compared to.

We have modified an existing UGV in order to be able to perform outdoor experiments. The sensor suite already implemented on the UGV was studied and upgraded. The selected sensors are investigated and its noise modeled.

Due to difficulties unrelated to this thesis work, endowing our UGV with a robotic arm was not possible. Nonetheless, we performed a technical comparison between two previously selected robotic arms. This comparison takes into account several constraints related to our problem.

The UGV is being developed for the participation in MBZIRC and for future research in the before mentioned areas.

1.4 Thesis Outline

This thesis comprises seven chapters. The present chapter introduces our work giving a brief topic overview, the motivation for the studied subject and the thesis objectives.

In Chapter 2, we go over the related work in the two topics we focused, odometry calibration and state estimation applied to localization.

Chapter 3 lays out the theory behind the methods used in our approach to localize a UGV. We describe the UGV's model and its control variables. Following, we explain the method used to calibrate the UGV's odometry and outline the noise models used to describe the sensors behavior. Finally, we detail the method used in our approach to localization.

Over Chapter 4, we describe all the methods used in our approach. First, we detail how the UGV's odometry is calibrated. This is followed by a description of the sensors present in the UGV, its characteristics and sources of measurements' noise, and the models used to model them. We describe the details of the extendend Kalman filter (EKF) implementation, specifically the filter state vector along all the parameters. Finally, we give a comprehensive description of how we compute the five localization estimates and outline the error metrics used to compare them.

In Chapter 5, we give a comprehensive description of the hardware and software used to perform our experiments. We start by presenting the UGV used to perform the experiments, its initial software suite and how it evolved into the present one. This is followed by an overview of two robotic arms which

can be use to perform tasks such as the ones from MBZIRC 2020. It also entails a description of all the software used to carry the experiments, along with its characteristics. Following this, we determine all the sensors model parameters, along with the methods used to compute them. Afterwards, sensor characteristics are discussed and to finalize, we detail the parameters used in the state estimator.

In Chapter 6, we present the results obtained and discuss the different methods. The chapter starts with a detailed overview of the experiments conducted outdoor, followed by five computed localization estimates. This ends with a discussion comparing the first four estimates to the last, which we use as a reference since it uses GPS's measurements in its computation.

The last chapter, Chapter 7, outlines the main conclusions of our work and the proposed future work.

This thesis contains two appendices. In Appendix A, we present the preliminary indoor experiments. The collected datasets are described, followed by the experiments conducted indoor.

In Appendix B, we detail the outdoor experiments' datasets.

Chapter 2

State of the Art

In this chapter, we identify the methods available in the literature which regard the two main problems we tackled. The first is calibration of the localization estimate provided by the UGV odometry. The second is state estimation applied to localization problems.

2.1 Odometry calibration

Odometry estimates are one of the simplest methods applied to localization. Odometry is the name given to a dead reckoning localization estimate obtained by integration of wheel movement measurements. The name originates from the early mechanical instruments, odometers, used to measure the distance traveled by a vehicle. With time, the methods and sensors evolved but the name stayed.

Modern odometry computation methods use an approximate model of the vehicle and information provided by sensors which measure wheel rotation, these are usually shaft encoders, sensors which measure rotation about the wheel axis. Using these and the model of the vehicle, the measurements are integrated providing a continuous position estimate. Since the model is not accurate and diverse factors affect the vehicle movement, this estimate deviates from the real position.

Two types of errors affect odometry estimation, systematic and nonsystematic errors [9]. Systematic errors are the ones which occur constantly, nonsystematic errors occur randomly. Unequal wheel diameters, misalign wheels, limited encoder resolution, and sampling rate are some examples of systematic errors in this application. Slippery floors and skidding are some examples of non-systematic errors. Due to the non-aleatory nature of systematic errors, these can be removed from our estimate by tuning the vehicle model.

Multiple methods for odometry calibration are described in the literature. One of the most widely known is UMBmark [9]. This method was proposed as a simple experiment from which the parameters of a differential drive robot could be calibrated. The method involves several steps and the execution of a predetermined path. First, it requires the UGV to execute the predetermined path, a square with four meters of side, five times, clockwise and counterclockwise. Before starting and after finishing the square, the 2D position of the robot must be measured and the center of gravity of both coordinates

computed separately for clockwise and counterclockwise experiments. Using those four values (the two centers of gravity in 2D), compute the absolute offsets of the two centers of gravity. With those compute the parameters which calibrate the model.

In [10] a different approach was taken by directly measuring the wheels velocities of a differential drive robot while its wheels were rotating without any load. Using a tachometer, an instrument used to measure rotation velocity, the real wheels velocities are measured and compared to the estimate provided by the encoders, the relation between the two is used to calibrate both wheels velocities. With measurements the robot's attitude while skid steering, the coefficient between the effective axle length is estimated.

While any of these methods has shown to provide a reliable and systematic way of estimating odometry calibration parameters, due to the existence of a Motion Capture (MOCAP) system in the laboratory, methods which take advantage of this system, without requiring a predetermined trajectory were preferred. Methods which rely on least-squares over a non-defined trajectory are one example of these.

Using methods with least-squares allows to estimate the parameters over a wide range of motions since they do not rely on a predetermined trajectory. The method presented in [11], which is the one used to calibrate the UGV's odometry, has such characteristics. Using an estimation of the path, provided by the UGV's odometry, and a ground truth, provided by the MOCAP system, it computes a calibration matrix from the estimated and ground truth 2D homogeneous coordinates difference in the robot frame. This matrix minimizes the sum of the squares of the residuals of the 2D homogeneous coordinates difference in the robot frame, approximating this way the estimated path to the ground truth.

This method was preferred due to the before mentioned characteristics and is throughly described in Section 3.2.

2.2 Localization

Localization although still a difficult problem in some situations, is a well-studied subject with multiple approaches found in the literature. There are two types of localization, relative and absolute [12]. Dead reckoning is a relative localization method, which relies on a given initial pose of the agent and on information about how it moved to update its belief. These do not update the agent pose directly but by integration of estimates of its velocities. Due to the noise present in such measurements, the estimated pose drifts from the real one. These provide a good estimate on a per step basis, but when integrating the velocities over a larger time interval and distance the estimation error grows without bound [13].

In order to correct the drift present in estimates obtained through dead reckoning techniques, measurements of the actual pose of the agent are required. These can be in relation to a map, in case of map matching, or in relation to a global frame such as the ones obtained from GPS receivers. These are then combined with the current estimate to correct the drift.

There are multiple methods to combine measurements from different sources and of different quantities. Such methods fuse multiple sensor measurements to obtain a better estimate. Example of these are particle filter, Kalman filter, Extended Kalman Filter (EKF) and unscented Kalman filter [14].

Localization can be thought as a state estimation problem where the state to be estimated contains the localization of our agent or its localization can be computed from it. EKFs have been extensively used in localization [15], they are efficient state estimators for nonlinear problems which measurements and state transition noise are modeled after white Gaussian noise.

This topic has been studied extensively, therefore recent literature has focused on studying the applications of such methods in new areas. In [15], four different estimation architectures which rely on an EKF are proposed and evaluated for six Degrees Of Freedom (DOF) pose estimation of a skid steer mobile robot used in urban search and rescue missions. The first approach, a nonlinear model which relies on an EKF to filter Inertial Measurement Unit (IMU) and odometry measurements, to estimate the robot's position, velocity, and linear accelerations. The second approach has the same structure as the previous but the attitude estimation is computed through the use of a complementary filter. The third approach, an error model [16] which estimates the bias of the IMU using the robot's position, velocity, and linear accelerations error, where the errors are obtained using an EKF. The fourth approach has the same structure as the previous but the attitude estimate is obtained using a complementary filter.

The author performs a series of experiments in different scenarios and determines that the second approach yields better results while being computationally more efficient than the error model base architectures.

Regarding the estimation architectures used in this thesis, the first architecture was used, this is simpler than the estimate which yields better results, yet it performed similarly in terms of return position and attitude error under challenging conditions.

Chapter 3

Background

In this chapter, we describe the theory in which our methods rely on and introduce the used notation. We present the unicycle model (the used UGV's model), describe the odometry calibration method and the noise models after which our sensors were modeled. This is followed by the EKF's description.

3.1 Unicycle model

Ground vehicles' appearance differs from application to application, yet vehicle models are more consistent since, in practice, simpler models are implemented more frequently. In this section, we give a detailed explanation on the model and control variables of the UGV used to perform the experiments.

The unicycle model has two control variables, v and ω, these represent the linear and angular velocities, respectively, in the fixed robot frame (subscript $_r$). Figure 3.1 illustrates the unicycle model along with its state and control variables.

The kinematics equations for the Unicycle Model are shown in (3.1). The variables x, y represent the position of the UGV in the 2D plane, while the variable α represents the orientation of our UGV regarding the positive X_w axis direction. Together, the three variables, represent the 2D pose in our world frame,

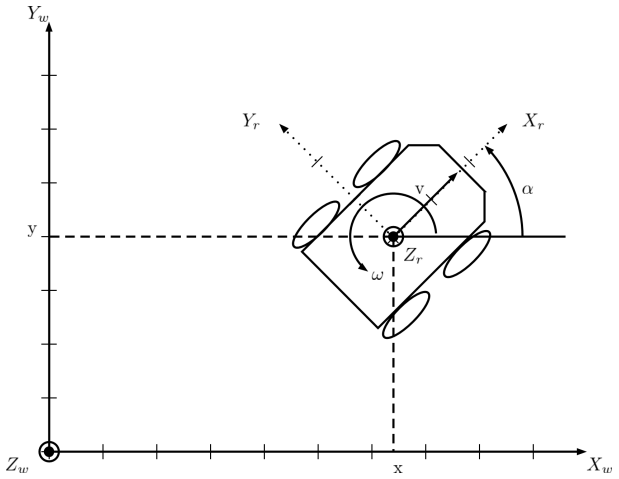


Figure 3.1: Unicycle model's state and control variables.

depicted with the subscript $_w$.

$$\begin{aligned} \dot{x} &= v \cdot cos(\alpha) \\ \dot{y} &= v \cdot sin(\alpha) \\ \dot{\alpha} &= \omega \end{aligned} \tag{3.1}$$

Where the variables $\dot{x}$ and $\dot{y}$ represent the linear velocities in the world frame along the X_w and Y_w axes respectively, and $\dot{\alpha}$ represents the angular velocity about the Z_r axis.

3.2 Odometry calibration method

Odometry 2D poses are computed using the measurements from wheel encoders and an ideal robot model. This model uses an estimate of the wheel radius and axis distance. As previously stated, the ideal robot model has persistent errors which can be removed by calibration of those two parameters.

Since a MOCAP system is available, we used a method which takes advantage of it. Considering $\mathbf{u}_t$ the ground truth pose, provided by the MOCAP system, and $\mathbf{z}_t$ the odometry estimated pose, both at time *t*, in 2D we have:

$$\begin{aligned} \mathbf{u}_t &= [x_t, y_t, \theta_t]^T \\ \mathbf{z}_t &= [\hat{x}_t, \hat{y}_t, \hat{\theta}_t]^T. \end{aligned} \tag{3.2}$$

Using these, we compute the 2D homogeneous transformation vectors between $t-1$ and t for both of our sequences of measurements, $\mathbf{u}_{0:N-1}$ and $\mathbf{z}_{0:N-1}$. This is done by computing the 2D homogeneous transformation matrix for t, A_t, then computing the homogeneous transform between $t-1$ and t, B_t, and finally obtaining the 2D homogeneous coordinates, $\mathbf{s}_t$, which correspond to the 2D homogeneous coordinates difference in the robot frame. These series of transformations are depicted for $\mathbf{u}_t$ in the sequence of operations in (3.3).

$$A_t = \begin{bmatrix} cos(\theta_t) & -sin(\theta_t) & x_t \\ sin(\theta_t) & cos(\theta_t) & y_t \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3a}$$

$$B_t = A_t A_{t-1}^{-1} \tag{3.3b}$$

$$\mathbf{s}_t = [B_{0,2_t}, B_{1,2_t}, arctan2(B_{1,0_t}, B_{0,0_t})]^T \tag{3.3c}$$

Where the function $arctan2(y, x)$ computes the angle in the Euclidean plane between the positive X_w axis and vector from the origin to the 2D point (x, y).

With these, two matrices are obtained, U and Z, which are matrices of dimension $3 \times N$, being each one the concatenation of the robot frame velocities for $\mathbf{u}_t$ and $\mathbf{z}_t$, respectively. This is shown for $\mathbf{u}$ in

(3.4).

$$U = \begin{bmatrix} s_{0_0} & \dots & s_{0_t} & \dots & s_{0_{N-1}} \\ s_{1_0} & \dots & s_{1_t} & \dots & s_{1_{N-1}} \\ s_{2_0} & \dots & s_{2_t} & \dots & s_{2_{N-1}} \end{bmatrix} \tag{3.4}$$

The problem was then formulated as a linear system of equations as in (3.5), where X is the 3×3 calibration matrix.

$$U = XZ \tag{3.5}$$

The calibration matrix X was then computed by applying least-squares minimization as in (3.6), where N is the number of 2D poses used in the minimization problem.

$$\min_X \sum_{t=0}^{N-1} (XZ_{:,t} - U_{:,t}) \tag{3.6}$$

With the calibration matrix, we calibrate the 2D homogeneous coordinates difference in the robot frame, as in (3.7).

$$\tilde{Z} = XZ \tag{3.7}$$

Taking $\tilde{Z}$ we compute the calibrated robot frame velocities, $\tilde{v}$ and $\tilde{\omega}$, and the calibrated trajectory. The first is computed by differentiation, as in (3.8).

$$\tilde{L} = \frac{\tilde{Z}}{\delta} \tag{3.8}$$

Where δ represents the measurement's acquisition period and the first row of $\tilde{L}$ is the calibrated linear velocities, $\tilde{v}$, and the third row contains the calibrated angular velocities, $\tilde{\omega}$. Both velocities are in the robot's frame.

The calibrated trajectory is computed recursively from $\tilde{Z}$ as in (3.9).

$$C_t = \begin{bmatrix} cos(\tilde{Z}_{1,t}) & -sin(\tilde{Z}_{1,t}) & \tilde{Z}_{0,t} \\ sin(\tilde{Z}_{1,t}) & cos(\tilde{Z}_{1,t}) & \tilde{Z}_{1,t} \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9a}$$

$$P_t = P_{t-1}C_t \tag{3.9b}$$

$$\tilde{\mathbf{z}}_t = [P_{0,2_t}, P_{1,2_t}, arctan2(P_{1,0_t}, P_{0,0_t})]^T \tag{3.9c}$$

Where the vector $\tilde{\mathbf{z}}_t$ is the calibrated 2D pose at the time instant t.

3.3 Noise models

EKFs rely on measurements variance to estimate the next state, for this purpose noise models are used to model the noise present in the sensor's measurements.

3.3.1 Additive white Gaussian noise model

One of the most commonly used models is the additive white Gaussian noise model, which models the noise as an additive element which is represented as Gaussian white noise, as shown in (3.10).

$$\begin{aligned} r(t) &= s(t) + w(t) \\ w(t) &\sim N(0, \sigma^2) \end{aligned} \tag{3.10}$$

Where $r(t)$ represents our measurement, $s(t)$ is the true value and $w(t)$ the white Gaussian noise with 0 mean and variance σ^2.

3.3.2 Additive white Gaussian noise model with proportional variance

In some cases, the white Gaussian noise variance tends to increase with the absolute value of the measurement's true value. For these cases, a model with varying variance was used to take advantage of this information. In this case, the variance is proportional to the estimated measurement's absolute true value. The model is depicted in (3.11).

$$\begin{aligned} r(t) &= s(t) + w(t) \\ w(t) &\sim N(0, \sigma^2(t)) \\ \sigma^2(t) &= m \cdot |\hat{r}(t)| + b \end{aligned} \tag{3.11}$$

Where $\hat{r(t)}$ represents the true value's estimate, m is the slope of the linear regression and b is the variance when $|\hat{r}(t)|$ is 0. The units of m are the same as the quantity we are modeling, the units of b are the same units but squared.

3.4 Extended Kalman Filter

EKFs are based in Kalman Filters which in time are part of a family of recursive state estimators, the Gaussian Filters family [17]. Gaussian Filters are tractable implementations of the Bayes Filter for continuous time.

Bayes Filters have been extensively applied to localization and tracking. This is due to Bayes Filter's being computationally efficient methods to state estimation, where uncertainty is described by multivariate Gaussian distributions, which is usually the case of localization and tracking problems.

Multivariate Gaussian distributions are defined as in (3.12), where the state is represented by $\mathbf{x}$, a column vector, with dimensions $n \times 1$.

$$p(\mathbf{x}) = |2\pi\Sigma|e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)} \tag{3.12}$$

Where n is the number of states being estimated, μ is the state mean vector and Σ is the state covariance matrix.

The derivation of the Bayes Filter assumes that the world is Markov, meaning that past and future data are independent if one knows the current state [17].

Algorithm 1 shows a generic representation of the Bayes filter. The vector $\mathbf{u}_t$ represents the control variables and $\mathbf{z}_t$ is the measurement vector, with dimension $k \times 1$. The last contains the state variables' measurements, both at time t.

Bayes Filters are comprised of two separate steps. In the first step, line 3, the belief over $\mathbf{x}_t$, $\overline{bel}(\mathbf{x}_t)$, is computed by incorporating the control vector. In the second step, line 4, the belief over $\mathbf{x}_t$, $bel(\mathbf{x}_t)$, is computed by incorporating the measurements vector.

Algorithm 1 Bayes Filter belief prediction and update.

1: Algorithm Bayes_Filter($bel(x_{t-1})$, u_t, z_t)
2: **for** all x_t **do**
3: $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})dx$
4: $\quad bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$
5: **end for**
6: return $bel(x_t)$

Bayes Filters per se are distribution agnostic. The Gaussian Filters family, to which Kalman Filters belong to, contain algorithms which are based on the Bayes Filter but using Gaussian distributions. The Kalman Filter's algorithm is shown in Algorithm 2.

As in Bayes Filters, Kalman Filters can be divided into two steps, the first step is the prediction step, the second step is the update step.

Algorithm 2 Kalman Filter belief prediction and update.

1: Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t)
2: $\overline{\mu}_t = A_t\mu_{t-1} + B_tu_t$
3: $\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
4: $K_t = \overline{\Sigma}_tC_t^T(C_t\overline{\Sigma}_tC_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$
6: $\Sigma_t = (I - K_tC_t)\overline{\Sigma}_t$
7: return μ_t, Σ_t

The prediction step $\overline{bel}(x_t)$, here represented by $\overline{\mu}$ and $\overline{\Sigma}$, is computed using a linear process model (3.13), given the control variables vector and the previous state, lines 2 and 3. The uncertainty of the model and control variables are encompassed in ϵ_t.

$$\mathbf{x}_t = A_t\mathbf{x}_{t-1} + B_t\mathbf{u}_t + \epsilon_t \tag{3.13}$$

The update step $bel(\mathbf{x}_t)$, here represented by μ and Σ, is also computed using a linear measurement

model (3.14), given the measurements vector and $\overline{bel}(\mathbf{x}_t)$, lines 5 and 6. The uncertainty of the model and measurements is encompassed in δ_t.

$$\mathbf{z}_t = C_t\mathbf{x}_t + \delta_t \tag{3.14}$$

The matrix R_t, with dimension $n \times n$, denotes the state transition noise and it reflects how well our state is modeled. Matrix Q_t, with dimension $k \times k$, reflects the noise present in our measurements. Finally, matrix K_t, with dimension $n \times k$, denominated by Kalman gain, specifies to which degree the measurements are incorporated into $bel(\mathbf{x}_t)$.

The differentiating feature between Kalman Filters and EKFs is the ability to use non-linear functions to model our system. For EKFs the process model is defined as in (3.15) and the measurement model as in (3.16).

$$\mathbf{x}_t = g(\mathbf{u}_t, \mathbf{x}_{t-1}) + \epsilon_t \tag{3.15}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \delta_t \tag{3.16}$$

Where $g(\cdot)$ and $h(\cdot)$ are nonlinear functions. Function $g(\cdot)$ is the process model which is used to predict the next state. While function $h(\cdot)$ is the observation model which is used to update the next state.

When estimating $\overline{\mu}_t$ the non-linear function is used, yet when estimating $\overline{\Sigma}_t$ a linear approximation is required. To overcome this problem the Jacobian of $g(\mathbf{u}_t, \mathbf{x}_{t-1})$, a matrix with dimension $n \times n$, is computed as in (3.17). For $h(\mathbf{x}_t)$, the same applies, the Jacobian, a matrix with dimension $k \times n$, is computed as in (3.18).

$$G_{ij_t} = \frac{\partial g_i(\mathbf{u}_t, \mathbf{x}_{t-1})}{\partial \mathbf{x}_{j_{t-1}}} \tag{3.17}$$

$$H_{ij_t} = \frac{\partial h_i(\mathbf{x}_t)}{\partial \mathbf{x}_{j_t}} \tag{3.18}$$

Algorithm 3 depicts the EKF's algorithm.

Algorithm 3 Extended Kalman Filter belief's prediction and update.

1: Algorithm Extended_Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t)
2: $\overline{\mu}_t = g(u_t, \mu_{t-1})$
3: $\overline{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + R_t$
4: $K_t = \overline{\Sigma}_t H_t^T (H_t\overline{\Sigma}_t H_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
6: $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7: return μ_t, Σ_t

Chapter 4

Methodology

In this chapter, we describe the odometry velocities calibration method and determine the noise model applied to every sensor. These are followed by the methods used to estimate the different measurements' variance. Finally, we detail the employed methods to compute the localization estimates and its corresponding error metrics.

4.1 Odometry velocities calibration

In order to calibrate the UGV's odometry, we used the method described in Section 3.2. We performed indoor experiments in which the UGV was driven randomly at varying speeds in order to collect data encompassing the full range of possible movements. Both the UGV's odometry and the MOCAP system's pose measurements were recorded. These experiments are described in Appendix A, data from both the first and second experiments was used in the calibration.

An OptiTrack MOCAP system with 12 cameras was used. The MOCAP system's pose measurements are obtained by the triangulation of multiple infrared reflective markers. Knowing the position and configuration of the markers, the pose of the array is estimated. The marker array was attached to the UGV by aligning it with the UGV's frame, with the marker array frame being placed directly above it. This implies that the transformation between the two is a simple translation in the Z axis. The marker array used is depicted in Figure 4.1.

The MOCAP system's pose estimates were considered as ground truth. While these estimates still contain some noise, the markers mean position errors are under $0.1\,[mm]$.

4.2 Sensor models

For each sensor, the noise was identified and the model which fit the noise best was applied and their parameters estimated. Quantities for which the ground truth was available the parameters were estimated. Quantities for which the ground truth was not available, the model parameters were estimated either by using the product datasheet or by other methods which are described in each sensor section.

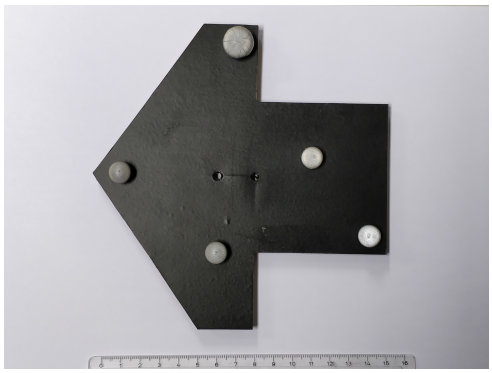


Figure 4.1: Array of infrared reflective markers used in indoor experiments.

4.2.1 Calibrated odometry velocities noise model

Odometry's velocities for a mobile robot with differential steering have two measurements, linear velocity in the X_R axis direction, denoted by v and angular velocity about the Z_R axis, denoted by ω.

Odometry measurements' noise comes from several sources. The encoder has limited resolution, which introduces a quantization error. If this was the only source of noise for our measurements we could consider the variance constant, and thus use the additive white Gaussian noise model. That does not apply since real mobile robots have different wheel radius, move on terrain with irregularities (especially outdoors) and the contact with the ground is not perfect, wheel slippage also contributes as a source of error.

Some noise sources have a non-constant effect in the measurements' noise. Sources such as slippage add to the noise an element which depends on the velocities' absolute true value, this relation can be modeled linearly in the velocities' absolute true value. We observed that these elements are large enough to dominate the overall error behavior. Considering this, we model both of our calibrated odometry's velocities, v, and ω, with the additive white Gaussian noise model with proportional variance defined in 3.11.

To show the increase in the measurements' variance with the velocities' absolute value, we performed indoor experiments which results are presented in Chapter 5.

4.2.2 Crossbow IMU noise model

The Crossbow DMU-6X-003 IMU measures angular velocity and linear acceleration. Both quantities were modeled using the additive white Gaussian noise model, defined in (3.10). This is not common practice, usually, gyroscopes and accelerometers noise models contain, beyond the elements present in (3.10), an additive one which is modeled as a random walk process [18]. To streamline the measurements' variance estimation, we compromise on this front.

Due to the lack of a datasheet for this specific model, the variances were computed assuming that they are the same at rest and while moving. Crossbow IMU's data was recorded for two minutes while at rest and the measurements' variance estimated.

During this experiment, we verified that the measurements have an approximately constant bias. In

order to better model the real value, the mean of the white Gaussian noise was not considered 0 as defined in (3.10), but equal to the estimated bias. The bias was then subtracted from our measurements in the IMU's driver.

4.2.3 MPU IMU noise model

The MPU-6050 IMU measures angular velocity and linear acceleration. With these measurements, it estimates, on-chip, the IMU's attitude relative to an inertial frame. The attitude estimate is obtained using a Digital Motion Processor developed by the Microelectromechanical System (MEMS) manufacturer, InvenSense. The measured or estimated quantities were modeled after the additive white Gaussian noise model, defined in (3.10).

The different sensor measurements variances are estimated using two different methods. The orientation's variance about the IMU's Z axis was estimated using data collected during indoor experiments where the MOCAP system was available. Considering the MOCAP system measurements as ground truth, the variance of the IMU's attitude estimation is computed from the error between the two for the first and second indoor experiments.

For the angular velocity and linear acceleration, we compute the variance using two parameters provided in the IMU's datasheet, the gyroscope noise performance and the accelerometer power spectral density.

4.2.4 NovAtel GPS receiver noise model

The Novatel OEM4-G2 GPS receiver measures, latitude, longitude and altitude. These three measurements' noise was also modeled using the additive white Gaussian noise model but with a varying variance.

In the message sent by the GPS receiver, along with the measurements, an estimate of the standard deviation for the latitude, longitude, and height is provided. The variance for the three different quantities was computed using this value.

GPS's measurements, in general, contain several sources of noise. These include ephemeris errors, atmospheric effects, such as tropospheric and ionospheric delays, clock drift, multipath signals, etc.

This specific receiver uses NovAtel's patented Narrow Correlator to estimate the measurement and its standard deviation [19].

This receiver has Real-Time Kinematic (RTK) capabilities, this technique is used to increase GPS measurements performance, yielding $1\,[cm]$ accuracy [20]. It requires the use of a dedicated base station (GPS receiver and antenna unit) which then relays corrections to the first receiver, usually over radio. Optionally, these corrections can be relayed over the Internet. A Portuguese entity (Direção-Geral do Território) maintains a network of base stations (Rede Nacional de Estações Permanentes) which corrections are available online. RTK was not used since it requires additional hardware or an Internet connection, both not available at the time we conducted the experiments.

This GPS receiver also has Satellite-based Augmentation System (SBAS) capabilities. Since this does not require any additional hardware, SBAS was used. The receiver uses corrections from Wide Area Augmentation System (WAAS), European Geostationary Navigation Overlay Service (EGNOS) and Multi-functional Satellite Augmentation System (MSAS).

4.3 Extended Kalman Filter

We used an already developed implementation of an EKF which we outline in this section. This filter's implementation is described in [21].

The estimated state $\mathbf{x}_t$, generally defined as a vector of dimension $n \times 1$ (as explained in Section 3.4), is now accurately defined with dimension 15×1. The state and its variables' physical meaning is given in (4.1). This encompasses the following fields, 3D pose (position and orientation), 3D velocities (linear and angular) and linear accelerations. The 3D pose is defined in the world frame while the velocities and accelerations are defined in the robot frame.

$$\mathbf{x}_t = [\underbrace{x_w, y_w, z_w, \gamma_w, \beta_w, \alpha_w}_{\text{World frame}}, \underbrace{v_{x_r}, v_{y_r}, v_{z_r}, \omega_{\gamma_r}, \omega_{\beta_r}, \omega_{\alpha_r}, \dot{v}_{x_r}, \dot{v}_{y_r}, \dot{v}_{z_r}}_{\text{Robot frame}}]^T \tag{4.1}$$

There are three main differences when comparing the implemented EKF's algorithm and the one described in Section 3.4, specifically in Algorithm 3.

The next state prediction $(\overline{\mu}_t, \overline{\Sigma}_t)$ is not a function of the input u_t and the previous state, $(\mu_{t-1}, \Sigma_{t-1})$, but only of the previous state. Also, the state estimate covariance matrix Σ_t is computed using the Joseph covariance equation to promote filter stability [22]. Finally, since the time between each filter cycle is not constant in the real world, the matrix R_t is scaled by this factor, here represented by δ_t. This results in higher prediction uncertainty for cycles in which the previous happened longer ago.

In Algorithm 4, we show the considered EKF implementation for this problem.

Algorithm 4 Extended Kalman Filter as implemented in robot_localization package.

1: Algorithm Extended Kalman Filter(μ_{t-1}, Σ_{t-1}, z_t)
2: $\overline{\mu}_t = g(\mu_{t-1})$
3: $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R\delta_t$
4: $K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu}_t + K_t(z_t - H_t \overline{\mu}_t)$
6: $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t (I - K_t H_t)^T + K_t Q_t K_t^T$
7: return μ_t, Σ_t

The function $g(.)$ used in the prediction step is a "standard 3D kinematic model derived from Newtonian mechanics" [21], the model is described in (4.2).

$$\begin{bmatrix} x_{w_t} \\ y_{w_t} \\ z_{w_t} \end{bmatrix} = \begin{bmatrix} x_{w_{t-1}} \\ y_{w_{t-1}} \\ z_{w_{t-1}} \end{bmatrix} + R_{ZYX_{t-1}} \begin{bmatrix} v_{x_{r_{t-1}}} \\ v_{y_{r_{t-1}}} \\ v_{z_{r_{t-1}}} \end{bmatrix} \delta + \frac{1}{2} R_{ZYX_{t-1}} \begin{bmatrix} \dot{v}_{x_{r_{t-1}}} \\ \dot{v}_{y_{r_{t-1}}} \\ \dot{v}_{z_{r_{t-1}}} \end{bmatrix} \delta^2 \tag{4.2a}$$

$$\begin{bmatrix} \gamma_{w_t} \\ \beta_{w_t} \\ \alpha_{w_t} \end{bmatrix} = \begin{bmatrix} \gamma_{w_{t-1}} \\ \beta_{w_{t-1}} \\ \alpha_{w_{t-1}} \end{bmatrix} + R_{ZYX_{t-1}} \begin{bmatrix} \omega_{\gamma_{r_{t-1}}} \\ \omega_{\beta_{r_{t-1}}} \\ \omega_{\alpha_{r_{t-1}}} \end{bmatrix} \delta \tag{4.2b}$$

$$\begin{bmatrix} v_{x_{r_t}} \\ v_{y_{r_t}} \\ v_{z_{r_t}} \end{bmatrix} = \begin{bmatrix} v_{x_{r_{t-1}}} \\ v_{y_{r_{t-1}}} \\ v_{z_{r_{t-1}}} \end{bmatrix} + \begin{bmatrix} \dot{v}_{x_{r_{t-1}}} \\ \dot{v}_{y_{r_{t-1}}} \\ \dot{v}_{z_{r_{t-1}}} \end{bmatrix} \delta \tag{4.2c}$$

$$\begin{bmatrix} \omega_{\gamma_{r_t}} \\ \omega_{\beta_{r_t}} \\ \omega_{\alpha_{r_t}} \end{bmatrix} = \begin{bmatrix} \omega_{\gamma_{r_{t-1}}} \\ \omega_{\beta_{r_{t-1}}} \\ \omega_{\alpha_{r_{t-1}}} \end{bmatrix} \tag{4.2d}$$

$$\begin{bmatrix} \dot{v}_{x_{r_t}} \\ \dot{v}_{y_{r_t}} \\ \dot{v}_{z_{r_t}} \end{bmatrix} = \begin{bmatrix} \dot{v}_{x_{r_{t-1}}} \\ \dot{v}_{y_{r_{t-1}}} \\ \dot{v}_{z_{r_{t-1}}} \end{bmatrix} \tag{4.2e}$$

Where $R_{ZYX_{t-1}}$ is a matrix of dimension 3×3 defined as in (4.3), and R_Z, R_Y, and R_X are the rotation matrices of dimension 3×3 about Z, Y and X axis, respectively.

$$R_{ZYX_{t-1}} = R_Z(\alpha_{w_{t-1}}) R_Y(\beta_{w_{t-1}}) R_X(\gamma_{w_{t-1}}) \tag{4.3}$$

This model represents a body which can move in 3D without any constraints. This is not ideal for our application since it does not reflect the unicycle model's constraints. We addressed this problem by feeding the filter an estimate for $v_{y_{r_t}}$ of 0 with a low variance. While not ideal, this has shown to be a valid option.

The measurements' vector size was defined in Section 3.4 as k. This value may differ from update to update cycle since sensors have different acquisition frequencies, which results in a varying number of measurements available at each iteration. As a result, all the matrices defined with dimensions of size k also differ.

The function $h(.)$, contrasting with function $g(.)$, is not used in the algorithm since the measurements arrive already transformed, yet the matrix H_t still needs to be defined. As previously stated, the matrix H_t has dimension $k \times 15$, since we already defined n to be equal to 15. Given this, all the matrix entries are zeros with the exception of one column for each measurement, which corresponds to the state variable the measurement will update, meaning that if the measurement i, updates the state variable j, $H_{i,j} = 1$.

4.4 Localization methods

The localization estimates are computed offline using data gathered in real-world outdoor experiments.

For the estimates obtained using the EKF, the estimation was performed for 2D measurements only, x_w, y_w, α_w, v_{x_r}, v_{y_r}, ω_{α_r}, $\dot{v}_{x_r}$ and $\dot{v}_{y_r}$. The rest of the state variables (z_w, γ_w, β_w, v_{z_r}, ω_{γ_r}, ω_{β_r} and $\dot{v}_{z_r}$) were kept at 0 by forcing this value and a low variance (1×10^{-6}) at every cycle iteration. For these

estimates, the EKF computes an estimate at a frequency of $30Hz$. Note that the odometry and the calibrated odometry are also only 2D estimates.

Method A is the UGV's odometry. The estimate is computed by integrating the wheel encoders values using an approximate model of the UGV. This model uses a wheel diameter and axes track approximation.

Method B computed by calibrating the UGV's odometry. The method by which this estimate is obtained was already presented in Section 4.1.

Method C is the first in which the EKF was used. We configured the EKF to use:

- the calibrated odometry velocities in X_r, Y_r and about Z_r;
- and the MPU IMU attitude about Z_r, angular velocity about Z_r and linear accelerations in X_r and Y_r.

Method D is built on top of the configuration from the previous one (method C). In this estimate we added:

- the Crossbow IMU measurements of angular velocity about Z_r and linear accelerations in X_r and Y_r.

Method E is also built on top of the configuration from the previous one (method D). In this estimate we added:

- the NovAtel GPS receiver's measurements along X_w and Y_w.

These measurements are converted into metric units by converting the latitude, longitude measurements to values in the respective Universal Transverse Mercator (UTM) coordinate system region.

The sensors' measurements which are not mentioned here, such as the NovAtel GPS receiver's altitude measurement, were not used. This is due to the estimation being performed only in 2D.

4.5 Error metrics

Two error metrics are used to compare the different pose estimates. Follows assuming a path formed with N samples.

For the position, we considered the mean absolute error of the Euclidean distance between the two poses, given by (4.4).

$$e_{position} = \frac{\sum_{i=0}^{N-1} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 + (\hat{z}_i - z_i)^2}}{N} \tag{4.4}$$

Since the computed estimates are in 2D, the last member $(\hat{z}_i - z_i)^2$ is always 0 since both $\hat{z}_i$ and z_i are equal to 0.

For the orientation, we considered the average of the absolute angle between the orientation estimate and the ground truth's orientation.

The error quaternion between the ground truth and the estimated quaternions is computed as in (4.5).

$$\delta = \hat{q} \otimes q^{*} \tag{4.5}$$

Then, the angle of the rotation defined by the error quaternion is computed as in (4.6).

$$\theta = 2 \cdot arccos(\delta_w) \tag{4.6}$$

Where δ_w is the quaternion scaler part.

Finally, the average of the absolute angle between the orientation estimate and the ground truth's orientation is given by (4.7).

$$e_{orientation} = \frac{\sum_{i=0}^{N-1} |\theta_i|}{N} \tag{4.7}$$

Chapter 5

Implementation

In this chapter, we describe the UGV used to perform the experiments and the sensors with which the data present in the datasets is acquired. A comparison between two proposed robotic arms is made, followed by the system's software description and the sensor model's parameters and characteristics. Lastly, the EKF's initial state and process noise covariance are determined.

5.1 Hardware

To conduct outdoor experiments, a system which can cope with various types of terrain and slopes is required. Outdoor environments introduce different challenges. An example of these are ground irregularities, these may induce vibrations which propagate through the hardware and cause unexpected behaviors.

5.1.1 Unmanned Ground Vehicle

The UGV used in this project is an ATRV-Jr mobile robot developed by iRobot. The UGV was previously used in other projects, such as RESCUE - Cooperative Navigation for Rescue Robots, to conduct research in navigation and, search and rescue[1], having suffered some minor hardware and software changes. For the purpose of this project, all the non-essential hardware and deprecated software were removed and an overall hardware revision was made.

The UGV has four wheels dependently motorized, meaning that the two left wheels are driven by one motor, motor 0, while the two right wheels are driven by another, motor 1. As a result, the UGV is steered using differential drive, meaning that is modeled after the unicycle model, introduced in Section 3.1.

The UGV's technical specifications are present in Table 5.1.

The communications with the UGV are done through a serial RS-232 interface with a board, rFLEX. This board interfaces with every sensor and actuator subsystem, it also contains a small display and a

[1] `http://rescue.isr.ist.utl.pt/publications.php`

Table 5.1: iRobot's ATRV-Jr's technical specifications.

Length	$77.5\,cm$
Width	$64\,cm$
Height	$55\,cm$
Ground clearance	$7.5\,cm$
Weight	$55\,kg$
Payload	$25\,kg$
Maximum linear velocity	$1.5\,m/s$
Maximum Angular velocity	$0.8\,rad/s$

button from which the basic functionalities of the UGV can be controlled. Such functionalities include, turning the UGV on and off, assessing the battery voltage and engaging the UGV's brakes.

Originally, the UGV was equipped with:

- an encoder in each motor;
- sonars, specifically, five forward facing, ten side facing and two rear facing sonars;
- a Crossbow DMU-6X-003 IMU;
- a compass and inclinometer unit;
- and a Garmin 35LP GPS receiver.

The Crossbow IMU, the compass and inclinometer unit, and the Garmin GPS receiver were connected to the computer through a RocketPort board which handled the serial communications. Figure 5.1 illustrates the original UGV's data flow diagram.

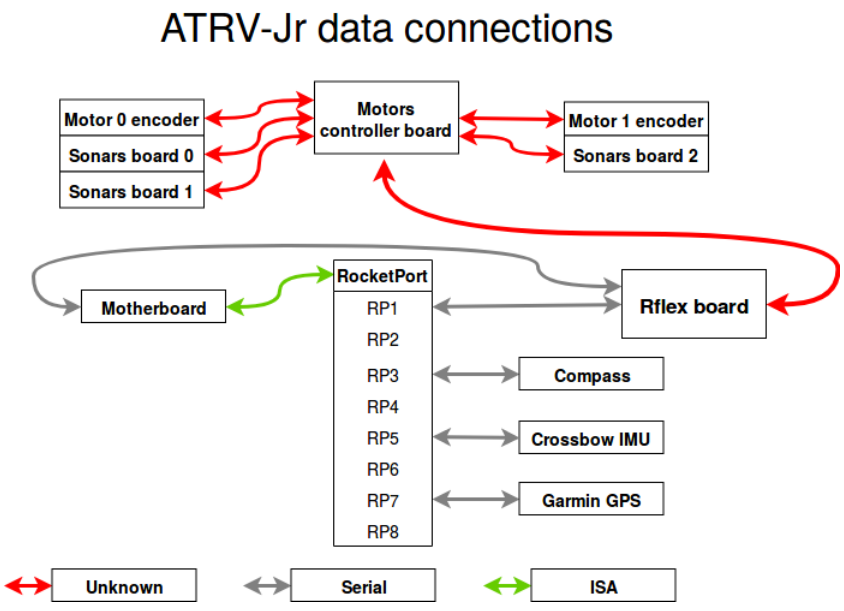


Figure 5.1: Original ATRV-Jr's data connections' diagram.

5.1.2 Sensors

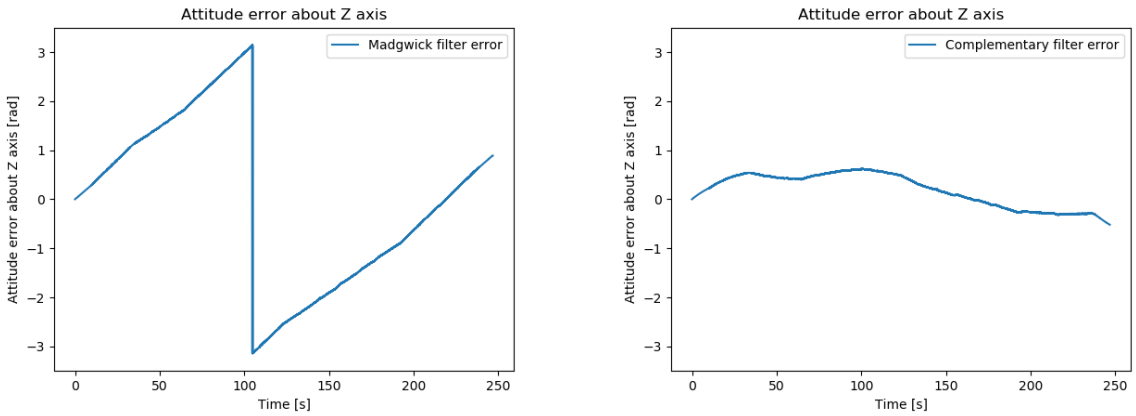
To simplify our implementation and update the UGV sensor suite several sensors were removed. These sensors were deprecated and consuming UGV's resources, such as space and power. The UGV has limited resources which will be required in the future for other systems, such as a robotic arm.

The compass and inclinometer unit, the sonars and the Garmin GPS receiver were removed. While an MPU-6050 IMU, a NovAtel OEM2-G2L GPS receiver and GPS antenna were added. The new sensors cover all the features lost when removing the old sensors, with exception to the magnetic field measurements, but they provide measurements with improved accuracy and additional characteristics.

When localizing a mobile robot with wheel odometry, one of the biggest difficulties is to keep an accurate heading. Odometry's heading, for example, drifts very quickly. In order to improve the localization estimation, a reliable source of heading is required.

IMUs are devices which measure a variety of quantities in the three frame axes, usually linear acceleration, angular velocity and, less frequently, magnetic field. Usually, all this information is fused to obtain a heading estimate.

The Crossbow IMU present in the UGV measures linear acceleration and angular velocity. Multiple filters take advantage of this information and estimate the IMU's attitude in relation to an inertial reference frame [23]. A ROS's driver for the Crossbow IMU was implemented and the sensor modeled as described in 4.2.2. In order to verify if we could rely on this sensor to obtain an attitude estimate, we estimate the UGV's attitude using two different filters, the Madgwick filter [24] and a Complementary filter [25]. Using the dataset from the third indoor experiment, described in Appendix A, we compute the estimate using both filters. These filters were chosen due to the availability of a Robot Operating System (ROS) compatible implementation. Figure 5.2 shows the estimates' attitude error relative to the ground truth provided by the MOCAP system.



(a) Madgwick filter's error.

(b) Complementary filter's error.

Figure 5.2: Attitude estimation's error using Madgwick filter and Complementary filter for the third indoor experiment.

Both filters equations are quite complex, but they are fully described in the respective reference. Note that both have different methods for fusing angular velocities and linear accelerations, or angular

velocities, linear accelerations and magnetic field, but since the Crossbow IMU does not have a magnetometer, the methods which use angular velocities and linear accelerations were used.

The Madgwick filter's attitude estimation error grows without bound at an approximately constant rate. The Complementary filter's attitude estimation error varies between approximately -0.5 and 0.61 radians. For our case, both filters' estimate error is too large. While the estimate from the Complementary filter is much better than the one from the Madgwick filter, since it does not grow without bound, it still does not provide a reliable attitude estimate. So, neither of these estimates was used in our method.

This sensor was not removed from the UGV since it did not interfere with any other hardware and since a driver had already been developed, its measurements were used in the localization estimation.

In order to obtain a reliable attitude estimate, an MPU-6050 IMU was added and additionally an Arduino, which allows to receive the IMU's measurements in the computer via USB. The estimate provided by the MPU IMU has shown to be more reliable than the ones provided by the Crossbow IMU when using any of the previously mentioned filters.

Figure 5.3 illustrates the MPU IMU estimate's attitude error for the fourth indoor experiment, relative to the ground truth provided by the MOCAP system.

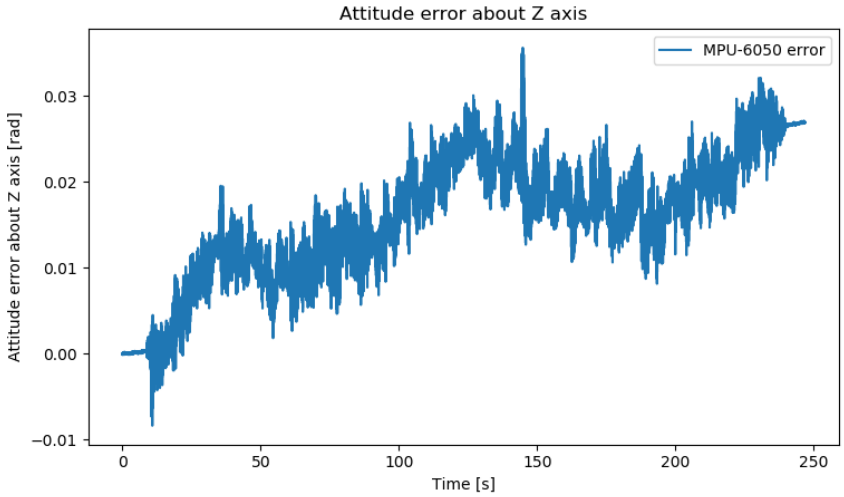


Figure 5.3: Attitude estimation's error using MPU IMU for the third indoor experiment.

While not ideal, the MPU IMU attitude estimation's error is lower than any of the previous estimates' error by more than one order of magnitude. In this experiment, the MPU IMU estimate deviates from the real heading at a rate of approximately $1.22 \times 10^{-4}[rad/s]$.

Ideally, an IMU with a magnetometer should be used. Only with a magnetic north reference, along the gravity vector is possible to uniquely define the inertial frame's three axes and thus eliminate the existent heading rate error.

At this point, the only state variables, of the vector defined in (4.1), for which we have no measurements are x_w, y_w, and z_w, the UGV's position in the world frame. To address this problem, we added a NovAtel OEM4-G2 GPS receiver, which interfaces with the computer via serial. The NovAtel GPS receiver measures latitude, longitude and altitude. From these, we compute Cartesian coordinates using

the UTM coordinate system.

The sensor suite now comprises the following sensors:

- an encoder in each motor;
- a Crossbow DMU-6X-003 IMU;
- an MPU-6050 IMU;
- and NovAtel GPS receiver.

Figure 5.4 illustrates the present data flow diagram of the UGV.

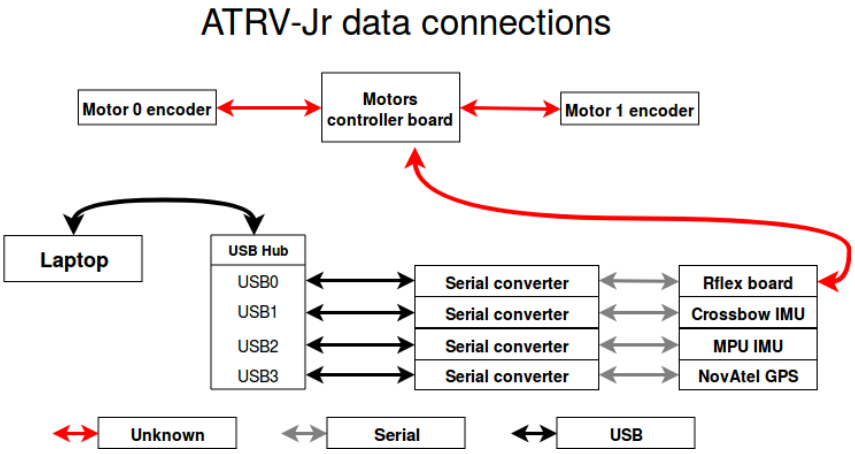


Figure 5.4: Present ATRV-Jr's data connections' diagram.

Figure 5.5 illustrates the current condition of the UGV. The disk-shaped white object with the blue stripe is the GPS antenna and, at its left, the black box which contains the Arduino and MPU IMU. The Crossbow IMU is inside the UGV roughly at the geometric center of the 4 wheels. The rest of the components such as the NovAtel GPS receiver are inside the UGV in a metal box which corner is visible below iRobot's logo.

Figure 5.5: ATRV-Jr's current configuration.

5.1.3 Robotic arm

A robotic arm is required for the challenges from MBZIRC 2020 and in all of the applications discussed. These, along with the UGV's characteristics, impose some restrictions.

Regarding the applications, there are two main constraints, the reach, and payload of the robotic arm. The robotic arm must be able to reach the floor when equipped with an end effector and to be able to lift at least $3\,kg$. We are working under the assumption that the robotic arm is placed over the UGV's frame origin.

Regarding the UGV, the main constraint is to not interfere with the UGV's normal functioning, meaning that it must be lighter than $25\,kg$, the UGV's payload.

There are other constraints which even if not prohibitive of the arm's installation should also be taken into account. The robotic arm is meant to be attached to a UGV which might move on uneven ground. This implies that the robotic arm is to be powered by batteries, so a lighter, sturdier and lower power consumption arm is preferred.

A ROS interface with the robotic arm must be available. Note that, developing such an interface is a difficult and time-consuming task, which if not executed well might lead to unexpected behaviors.

We studied two robotic arms:

- Panda, a seven DOF robotic arm manufactured by Franka Emika;
- and UR5, a six DOF robotic arm manufactured by Universal Robots.

Both are considered collaborative robotic arms by their manufactures, meaning that they are able to work alongside humans, without the need of a safety cage. These robotic arms have been previously selected, not being in the scope of this thesis to study the robotic arms in depth but to provide a technical overview of the two.

The technical specifications of both robotic arms are found in Table 5.2.

Franka Emika's Panda

Franka Emika was founded in 2016. In 2017 it launched its single product to date, a robotic arm named Panda. Figure 5.6 depicts the Panda robotic arm.

The Panda has seven rotating joints, making it a seven DOF robotic arm. It weighs about $18\,kg$ and has a payload of $3\,kg$, having a reach of $855\,mm$, when equipped with the included parallel gripper. Its control box weighs about $7\,kg$.

Panda's manufacturer open sourced a C++ library, libfranka[2], with which it is possible to interface with the robotic arm. Alongside the C++ library, it released an open source ROS package collection, franka_ros[3], which contains all the tools required to work with the Panda in ROS.

[2] https://github.com/frankaemika/libfranka
[3] https://github.com/frankaemika/franka_ros

Table 5.2: Robotic arms' specifications comparison.

		Panda	UR5
Robotic Arm	Repeatability	$\pm 0.1\,mm$	$\pm 0.1\,mm$
	Temperature range	Typical $15 - 25° C$	$0 - 50° C$
	IP classification	IP30	IP54
	Power consumption	Typical $300W$, Max $600W$	Typical $150W$, Max $325W$
	Payload	$3\,kg$	$5\,kg$
	Reach	$855\,mm$	$850\,mm$
	DOF	7 rotating joints	6 rotating joints
	Joint limits	Variable	$\pm 360°$
	Footprint	$226 \times 190\,mm$	Diameter $149\,mm$
	Weight	$\sim 18\,kg$	$18.4\,kg$
	Sensitivity	Torque sensors	Current sensors
	Mounting orientation	Upright	Any
	Cartesian velocity limit	$2\,m/s$	$1\,m/s$
Control box	Communication	Ethernet (TCP/IP)	Ethernet (TCP/IP), etc
	Power source	$100 - 240VAC, 47 - 63Hz$	$100 - 240VAC, 50 - 60Hz$
	Temperature range	Typical $15 - 25° C$	$0 - 50° C$
	Size (WxHxD)	$483 \times 89 \times 355\,mm$	$475 \times 423 \times 268\,mm$
	Weight	$\sim 7\,kg$	$15\,kg$
Gripper	Type	Parallel	(Not included)
	Weight	$\sim 0.7kg$	-

Universal Robots' UR5

Universal Robots was established in 2005 and it focuses on collaborative robotic arms. In 2008 it launched its first robotic arm, the UR5, depicted in Figure 5.7.

The UR5 weighs $18.4\,kg$ and it has a payload of $5\,kg$. The control box weighs about $15\,kg$. The teach pendant is a tablet-like device which interfaces with the UR5, allowing the realization of multiple tasks.

This arm has been the focus of various research [28, 29], having a large community. At least two ROS' drivers have been implemented for the UR5. One by the ROS Industrial community, ur_driver[4] and another, ur_modern_driver[5], developed by a student which iterated over the first and solved many issues [30].

Panda and UR5 comparison

The Panda has torque sensors in all joints while the UR5 only has current sensors. Torque sensors are viewed as a better solution, in the case of control algorithms' implementation, having torque sensors removes the overhead of estimating the torque in each joint from the current being drawn by the joint motor.

The Panda has ROS tools developed by the manufacturer, while the UR5 has ROS tools developed by the community. We can look at this point from two distinct perspectives. A tool developed by the manufacturer should be better integrated than one developed by the community since the manufacturer has full knowledge about the developed system. While community-developed tools might evolve faster.

[4] https://github.com/ros-industrial/universal_robot
[5] https://github.com/ThomasTimm/ur_modern_driver

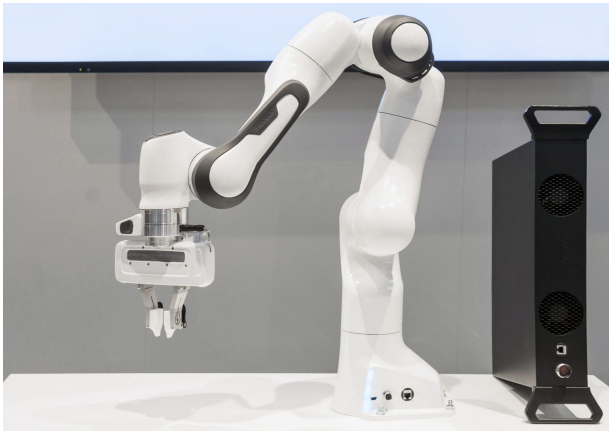


Figure 5.6: Panda robotic arm and respective control box (extracted from [26]).

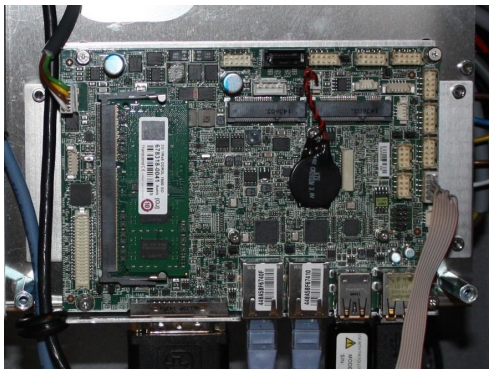
Figure 5.7: UR5 robotic arm, along its control box and teach pendant (extracted from [27]).

The Panda has an extra DOF when compared to the UR5, yet its joint limits are much narrower. Four of the Panda's joints have a limit of $\pm 166°$, one has a limit of $\pm 101°$, other of $-1°$ to $215°$ and another of $-176°$ to $-4°$. As for the UR5, all joints have a limit of $\pm 360°$.

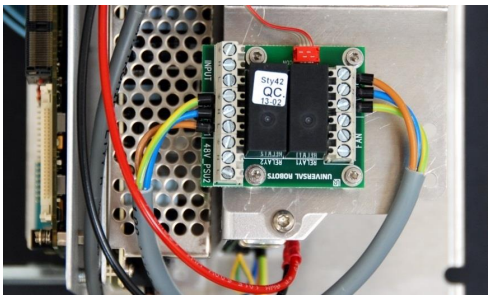
Regarding the robotic arm weight, the Panda is lighter, weighing $0.4\,kg$ less. The control box weight also has to be taken into account, since it needs to be placed in the UGV. Due to the large size and being power by alternating current, when attaching a robotic manipulator to an UGV, control boxes are usually taken apart and converted to direct current. In this process, the enclosure is replaced by a smaller one, since some components, such the AC/DC converter, are removed leaving empty space inside the control box.

In the case of the UR5, this process was studied since documentation about disassembling the control box was found in the arm's Service Manual, for the Panda this documentation was not found. The components present in the UR5's control box which are required to be kept are the motherboard, the current distributor, the safety board and the energy eater/fan unit. Figure 5.8 illustrates these components.

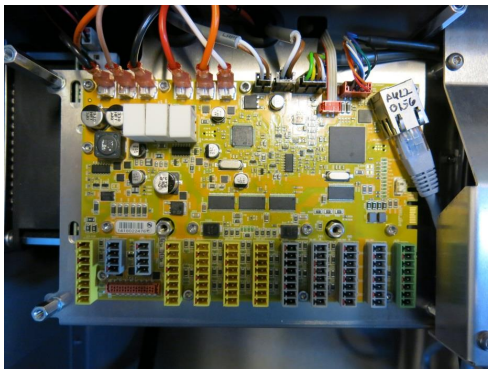
The motherboard is about $174 \times 125 \times 30\,[mm]$. The current distributor is about $50 \times 40 \times 20\,[mm]$, the safety board is about $190 \times 125 \times 40\,[mm]$ and the energy eater/fan unit is about $100 \times 100 \times 65\,[mm]$.



(a) Motherboard.



(b) Current distributor.



(c) Safety board.

(d) Energy eater/fan unit.

Figure 5.8: UR5's control box's components (extracted from [31]).

This amounts to a volume of approximately $2.3\,[dm^3]$, which is roughly 23 times less than the control box's volume. We are aware that additional hardware is required to protect, attach and connect these components. Yet, the total volume should still be much lower than the control box's.

While the weight of these components is difficult to estimate, much of the weight of the UR5's control box is due to its steel enclosure and AC/DC converters. Replacing the control box enclosure and removing the AD/DC converters will reduce the weight required to be put in the UGV.

Regarding the power consumption, since it is not clear if the typical is found by the same methods, this is not a good indicator to compare the two. Yet, the maximum power consumption can be compared, since it is an absolute value.

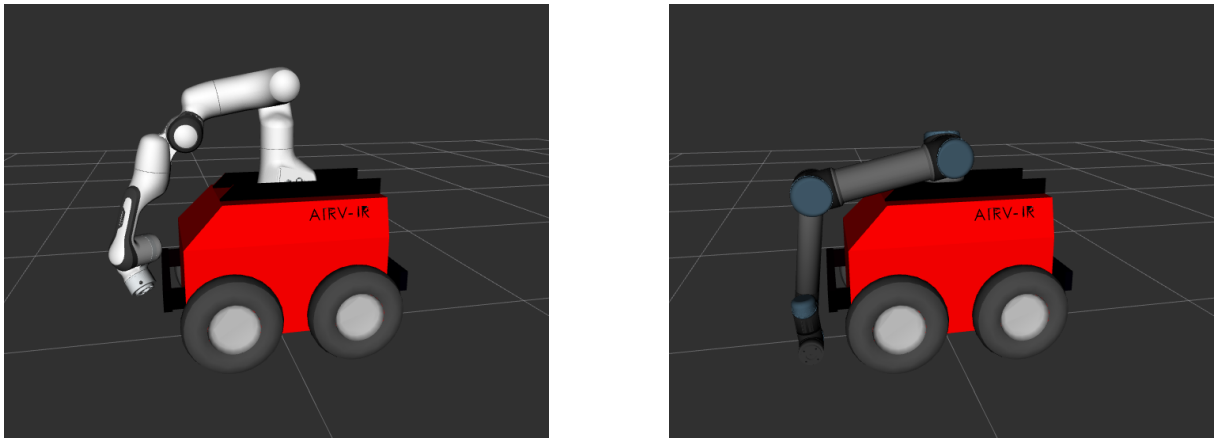
The Panda's power consumption is almost the double of the UR5's, with a maximum power consumption of $600[W]$, while the UR5's is $325[W]$. A higher power consumption for the Panda is to be expected since its Cartesian velocity limits are the double of the UR5's. To be precise, the Panda's Cartesian velocity limit is $2\,[m/s]$, while the UR5's is $1\,[m/s]$. The Panda's higher Cartesian velocity limit and the UR5's lower power consumption, can both be considered upsides. The preference for either of them depends on the specific application.

The UR5 has been adopted in mobile robotics for different purposes, having been attached to multiple UGV's. From Clear Path's Husky [32], a UGV developed for outdoor environment, and to a Mobile Industrial Robots's robot [33], used mostly in industrial environments. For the Panda, an implementation on a UGV has not been found.

Depending on the task at hand, the robustness of the robotic arm may be an important characteristic. The Ingress Protection (IP) rating of both robotic arms reflects this, while the Panda is IP30 certified, the UR5 is IP54. For context, the first digit of the IP rating denotes the level of protection against solid particles, while the second one denotes the level of protection against liquids [34]. Higher is better for either of the digits.

For solid particles, the Panda is protected down to $2.5\,[mm]$, while the UR5 is down to dust (it might enter the equipment but not interfere with its operation). For liquids, the Panda is not protected, while the UR5 is protected from splashing water from any direction. The Panda's lack of protection against water is problematic in outdoor environments since there is the possibility of precipitation. In environments witch lack structure, as in outdoor ones, the more robust build of the UR5 is also an advantage.

As illustrated in Figure 5.9, the Panda has a high placement of the second rotating DOF than the UR5. The Panda's first DOF is a rotation about Z_r, considering that the arm's axis is aligned and directly above the UGV's Z axis. This makes the last joint of the Panda stand higher when the arm is fully extended towards the ground. Note that in the figure, the second joint is in a joint limit, this is the joint with a limit of $\pm 101°$.



(a) ATRV-Jr and Panda.

(b) ATRV-Jr and UR5.

Figure 5.9: Panda and UR5 fixed on ATRV-Jr's 3D model.

Regarding the communications, both robotic arms have control units with Ethernet interfaces. So we propose to use them in either of the cases in order to communicate with the UGV's computer. The proposed data connections diagram is depicted in Figure 5.10.

Conclusions

For the challenges from MBZIRC 2020 and the applications discussed, it was found that the UR5 is a better solution. The joint configuration allows it to reach closer to the ground. If not enough, the UR5 can also be mounted in any desired position, meaning that a better placement can be studied, contrasting with the Panda which can only be mounted upright. It has also a more robust construction, at least exterior wise, with a higher protection against the elements. The fact that it has already been attached to multiple UGV's is also an important factor.

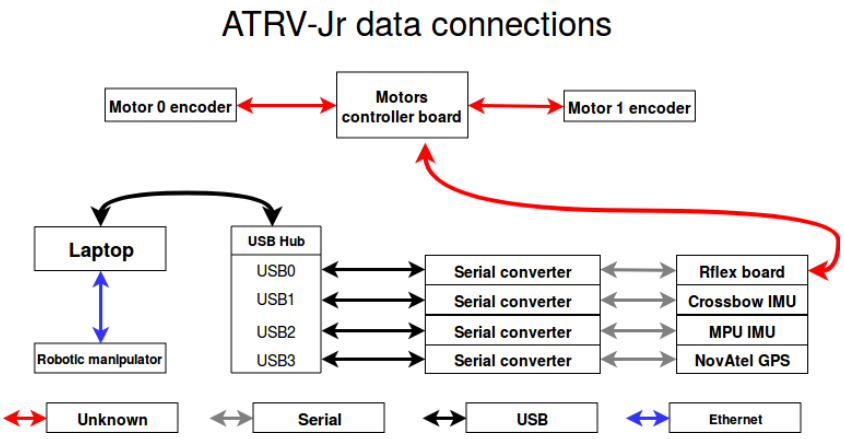


Figure 5.10: Proposed ATRV-Jr and robotic arm's data connections diagram.

5.2 Software

In this section, we describe the software employed in our experiments and its characteristics.

5.2.1 Robot Operation System

Our hardware platform is comprised of several sensors and systems, in order to interact with all of them in a standard way we used ROS [35]. ROS is a flexible framework for writing robotics applications software, but also a collection of tools, libraries and conventions.

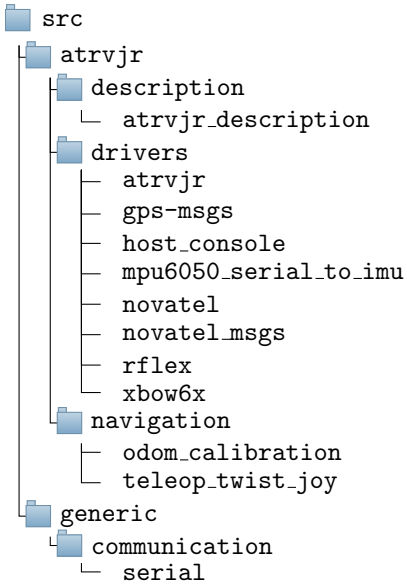
ROS software is organized in ROS packages. In order to perform our experiments in an expedite way, several ROS packages were developed, adapted or used as they were.

Figure 5.11 depicts the software's organizational structure and the ROS packages present in each subfolder.

The following ROS packages were developed solely for this application:

- *atrvjr_description*[6] - contains the description of all the frames present in the UGV in relation to the UGV's frame defined, as usual for robots modeled after the unicycle model, in the geometric center between the four wheels. The frames of both IMUs and the NovAtel GPS receiver are defined in this package.
- *atrvjr* - contains configuration and *roslaunch* files, a method used to start our system with ease.
- *host_console* - provides an interface to basic UGV functionalities, such as turn off the computer and display text in the rFLEX screen, useful when performing experiments.
- *odom_calibration* - contains the implementation of the method described in Section 4.1 and to apply the calibration to the UGV's odometry.
- *xbow6x*[7] - has the tools required to connect to the Crossbow IMU via serial and parse the incoming measurements.

[6] https://github.com/diasdm/atrvjr_description
[7] https://github.com/diasdm/xbow6x



(5.1)

Figure 5.11: ROS packages.

The ROS packages *novatel*, *novatel_msgs*[8] and *rflex* were adapted to fulfill our needs.

The package *novatel*[9] contains the driver for the NovAtel GPS receiver. We adapted it so that we could use SBAS and, for debugging purposes, to save the full NovAtel message.

The package *rflex*[10]. contains the driver of the rFLEX board which allows us to interact with most of the UGV's components. Since it had been developed to work with a previous ROS version, it was updated and several minor changes were made.

The ROS packages *mpu6050_serial_to_imu*, *teleop_twist_joy* and *serial* were used without performing any change. The package *mpu6050_serial_to_imu* interfaces with the Arduino, to obtain the measurements from the MPU IMU. The package *teleop_twist_joy* interfaces with a common joystick with which we control the robot by sending velocities commands. The package *serial* enables serial communication for the packages which require it.

5.2.2 Estimates computation

The estimates from methods B, C, D and E, were computed offline after the outdoor experiment had been carried and the datasets collected, these are described in Appendix B. However, our estimates can be computed online since the methods can be run in real-time.

Figure 5.12 gives an overview of the system used to compute the estimates.

In this figure we observe the different nodes running in our system and the connections between them. To the left, we have the system input, the commands received from the joystick used, which are them parsed and sent to the UGV driver, */rlfex/atrvjr_node*. In the following column of nodes we have

[8] https://github.com/diasdm/novatel_msgs
[9] https://github.com/diasdm/novatel
[10] https://github.com/diasdm/rflex

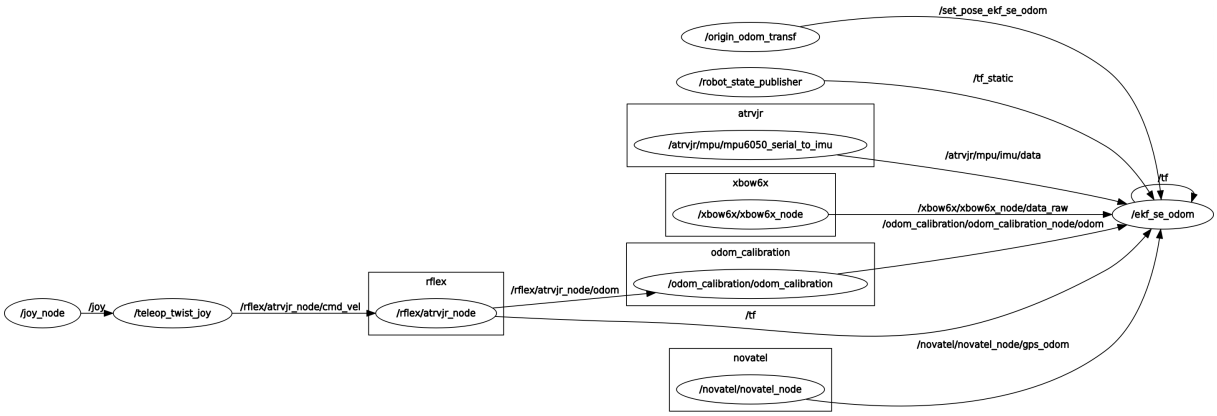


Figure 5.12: ATRV-Jr and MOCAP system infrared cameras during indoor experiment.

all the sensor drivers and the odometry calibration node, these all feed the node in which the EKF is running, */ekf_se_odom*.

5.3 Odometry calibration matrix

Using the data from the UGV's odometry and the MOCAP system poses, the calibration matrix was computed as described in Section 4.1, using the datasets from the first and second indoor experiment. Figure 5.13 depicts the UGV during the indoor experiments.

This matrix is then used by the *odom_calibration* package to compute the calibrated odometry. The estimated calibration matrix is shown in (5.2).

$$X = \begin{bmatrix} 9.470 \times 10^{-1} & -8.084 \times 10^{-3} & 1.839 \times 10^{-4} \\ -2.347 & 1.492 \times 10^{-1} & -4.779 \times 10^{-1} \\ 1.264 \times 10^{-1} & 2.348 \times 10^{-2} & 9.717 \times 10^{-1} \end{bmatrix} \tag{5.2}$$

5.4 Sensor models parameters

In this section, we define the sensor models parameters for all the modeled sensors, the parameters are computed using the methods described in 4. In Appendix A, we give a description of the preliminary indoor experiments. The data collected in the first and second experiments is used to estimate several model parameters, mostly variances.

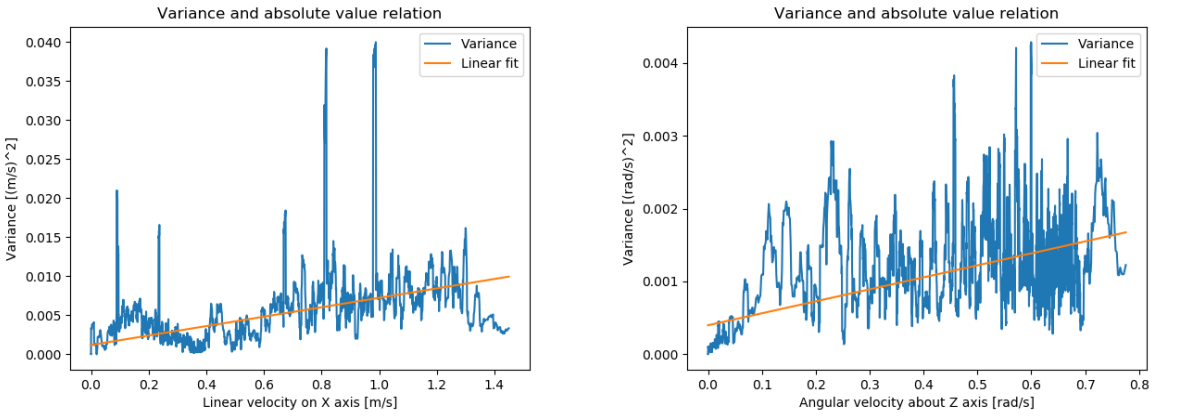
5.4.1 Calibrated odometry velocities model parameters

The noise present in the calibrated odometry velocities is modeled after the additive white Gaussian noise model with proportional variance. Using the collected data, the ground truth linear and angular velocities are computed.

Figure 5.13: ATRV-Jr and MOCAP system infrared cameras during indoor experiment.

After calibrating odometry velocities, its variance is determined by sorting them after taking their absolute value and computing the variance regarding the ground truth velocities in bins of 15 samples.

In Figure 5.14, we observe the relation between linear and angular calibrated odometry velocities absolute value and the measurements' variance. A linear regression was estimated between the two since we assume a linear relation for the increase of these measurements variance with its amplitude.



(a) Calibrated linear velocity absolute value's variance.

(b) Calibrated angular velocity absolute value's variance.

Figure 5.14: Calibrated odometry velocities' linear fit.

The noise model parameters are defined in (5.3) for the linear velocity and in (5.4) for angular velocity.

$$\begin{aligned} m_v &= 4.92 \times 10^{-3}\ [m/s] \\ b_v &= 1.80 \times 10^{-3}\ [(m/s)^2] \end{aligned} \tag{5.3}$$

$$
\begin{aligned}
m_\omega &= 1.64 \times 10^{-3}\ [rad/s] \\
b_\omega &= 3.99 \times 10^{-4}\ [(rad/s)^2]
\end{aligned}
\tag{5.4}
$$

The regressions' determination coefficient, also known as R^2, for the linear velocity variance is 0.23, while for the angular velocity is 0.31. As evidenced in Figure 5.14, and confirmed by the R^2 values of our regressions, the fitness of the model is very poor. This is a consequence of the simple methods we used to estimate the sensor noise and compute its measurements' variance.

5.4.2 Crossbow IMU model parameters

As mentioned in Section 4.2.2, the Crossbow IMU's variance is computed using values from a static experiment.

The bias and variance values for the three axes are shown in Tables 5.3 and 5.4, respectively.

Table 5.3: Crossbow IMU axes' bias.

	X	Y	Z	
Linear acceleration	-1.96×10^{-1}	-5.74×10^{-1}	5.63×10^{-2}	$[m/s^2]$
Angular velocity	-3.47×10^{-2}	8.64×10^{-2}	6.19×10^{-2}	$[rad/s]$

Table 5.4: Crossbow IMU axes' variance.

	X	Y	Z	
Linear acceleration	7.71×10^{-3}	8.94×10^{-3}	6.53×10^{-3}	$[(m/s^2)^2]$
Angular velocity	5.83×10^{-7}	1.22×10^{-6}	1.15×10^{-6}	$[(rad/s)^2]$

The bias values are subtracted to the measurements in the Crossbow IMU's driver, making the mean value of the measurements closer to the true value mean.

5.4.3 MPU IMU model parameters

The orientation estimate's variance is computed from the error present between the ground truth and the IMU estimate, during the first and second indoor experiments. Since, in the indoor experiments, we are moving in 2D, only the variance of α is estimated, and assumed the same for γ and β. The estimated variance corresponds to the value of $5.81 \times 10^{-2}[rad^2]$.

The variances for the angular velocities and linear accelerations are computed from the MPU-6050 datasheet's values, gyroscope noise performance, and accelerometer power spectral density, respectively. The gyroscope noise performance is $0.05\ [^\circ/s]$, the angular velocity variance is denoted by $\sigma^2_{ang\ vel}$ and it is defined in (5.5).

$$\begin{aligned}\sigma_{ang\ vel} &= 0.05\ [^{\circ}/s] \\ &= 8.73 \times 10^{-4}\ [rad/s] \\ \sigma^2_{ang\ vel} &= 7.62 \times 10^{-7}\ [(rad/s)^2]\end{aligned} \tag{5.5}$$

The accelerometer power spectral density is $400\,[\mu\,g\ /\sqrt{Hz}]$, where g denotes the gravitational acceleration. The accelerometer uses a first-order RC low-pass filter, for which the bandwidth is computed by $B = 1.57 \cdot f_{-3dB}[Hz]$ [36]. Knowing the filter's cut off frequency, $42\,Hz$ and assuming a flat power spectral density, we compute the root mean squared noise (RMS) as in (5.6).

$$\begin{aligned}RMS &= \sqrt{42 \cdot 1.57} \cdot 400\,[\mu\,g\ /\sqrt{Hz}] \\ &= 3.19 \times 10^{-2}\ [m/s^2]\end{aligned} \tag{5.6}$$

Assuming the noise is approximately Gaussian, the RMS is approximately the standard deviation. The linear acceleration variance is denoted by $\sigma^2_{lin\ acc}$ and it is defined in (5.7).

$$\begin{aligned}\sigma^2_{lin\ acc} &\approx RMS^2 \\ &\approx 1.02 \times 10^{-3}\ [(m/s^2)^2]\end{aligned} \tag{5.7}$$

Since these values are not specified for the three IMU axes, but on a sensor basis, the variances are the same for the three IMU axes.

5.4.4 NovAtel GPS receiver model parameters

As mentioned in Section 4.2.4 the NovAtel GPS receiver estimates the measurements' standard deviations internally. The square of these values is taken in order to obtain the measurements' variance.

5.5 Sensors characteristics

Some sensor characteristics are relevant to the problem we are tackling. One of these is the sensor sampling frequency, since, this determines the frequency at which the state variables are updated. The sampling frequencies for all the sensors are shown in Table 5.5.

The difference in sampling frequencies results in different update rates for the state variables. During some EKF cycles, there are state variables which are predicted but not updated.

Sensors measurements, such as the ones from GPS receivers, are affected differently during experiments. GPS receivers rely on multiple satellites, at a minimum four, in order to compute a localization estimate. During the experiments, time passes and the UGV's location changes, resulting in different

Table 5.5: Sensors sampling frequency.

Sensor	Frequency$[Hz]$
Calibrated odometry velocities	10
Crossbow IMU	180
MPU IMU	100
NovAtel GPS receiver	21

number of satellites in view during the experiment. With it, the variance or even availability of the estimate also varies.

When using data from such source, we are required to access its validity. The message sent by the NovAtel GPS receiver contains a flag, shown in Table B.2 as the *solution_status* field, which determines if the provided estimate is valid or not. These measurements are ignored at runtime by a similar flag in the ROS sensor_msgs/NavSatFix message, provided by the *novatel* ROS package along with the full message.

5.6 Extended Kalman Filter

In Section 4.3, we outlined the characteristics of the algorithm, but its parameters were not defined. The initial state mean, its covariance matrix and process noise covariance matrix need to be determined.

The initial state, μ_0 and Σ_0, is assumed to be known. The initial x_w and y_w are obtained from the first GPS measurement. The initial α is obtained from the first MPU IMU measurement. The rest of the state variables are 0. Since the state is known its covariance matrix is defined as in (5.8), where I is the identity matrix with dimension 15×15. This implies that in the experiment's beginning, the UGV is at rest.

$$\Sigma_0 = I \cdot 1 \times 10^{-9} \tag{5.8}$$

As previously stated in Section 3.4, the process noise covariance matrix R reflects how much we trust our model to predict the next state. It is required to estimate this matrix, but in general, there is no systematic way to calculate the process noise covariance matrix [37]. The matrix diagonal estimation was done using the knowledge we have about the process, followed by the tuning of each value, assuming that the measurements' covariance was well estimated. While not ideal, this allowed us to obtain better results.

The matrix is the same for every experiment. Meaning that it was not tuned to obtain a better result in a specific one, but to best represent the process noise covariance. The matrix R is determined as in (5.9).

$$\begin{gathered} \mathbf{r} = [0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.5, 0.5, 0.5, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3]^T \\ R_{i,i} = \mathbf{r}_i, \forall\, i \in [0, 14] \end{gathered} \tag{5.9}$$

The units of two matrices' values are not consistent among them. The diagonal values' units are the square of the state vector's International System (SI) units.

This algorithm is available as a ROS package[11].

[11] https://github.com/cra-ros-pkg/robot_localization

Chapter 6

Results and discussion

In this chapter, we present the results obtained in four different outdoor experiments. We start with a description of the conducted experiments, afterward, we detail the obtained results regarding the computed localization estimates. This chapter ends with a comparison of the experiments' estimates taking into account several evaluation metrics.

6.1 Outdoor experiments

Following the same structure as the preliminary experiments, presented in Appendix A, we conducted several outdoor experiments. These were conducted inside the IST Alameda Campus and try to cover diverse situations. The datasets collected during these experiments are described in Appendix B.

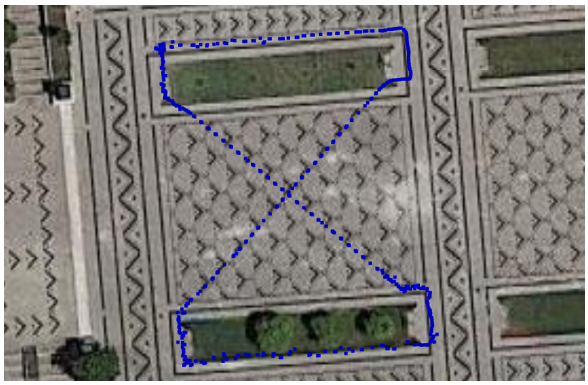
In Figure 6.1, the experiments' approximate path is represented by GPS measurements. These measurements are undersampled and represented by blue dots. They show discontinuities, usual in urban environments, where tall structures often obstruct the sky view. These obstructions, vary the number of satellites in view and introduce multipath which degrades the GPS's measurements.

The first and second experiments were conducted far from any buildings or large trees. For that reason, the measurements are relatively close to each other and do not show any large discontinuities.

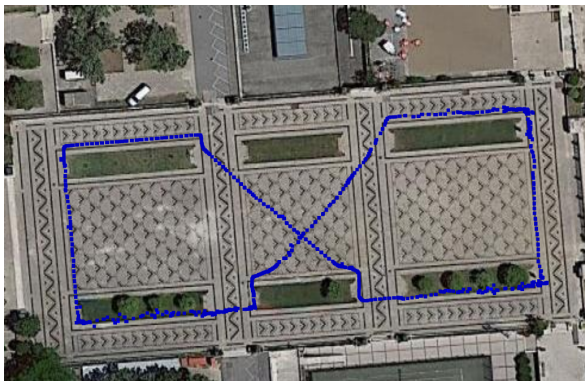
In the third and fourth experiments, the UGV's paths come very close to buildings, and in the case of the third, under large trees' canopy.

In our experiments, these structures introduced two types of errors. The trees' canopy create a dispersion in the measurements' mean, as seen in the third experiment. The buildings cause a mean offset, as seen in the fourth experiment, where the GPS measurements' mean is over the sidewalk, even though we did not drive over it. In that experiment, the UGV was driven on the asphalt along the sidewalk.

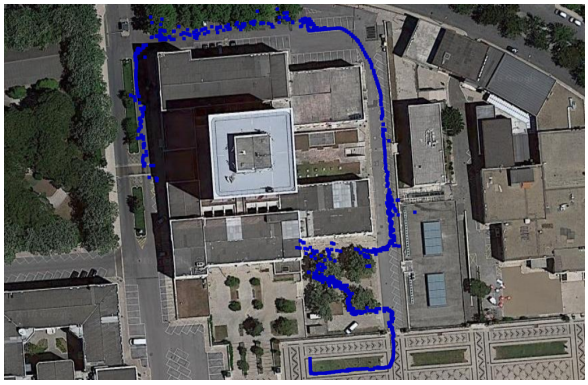
The first and second experiments were performed approximately under the same conditions. The terrain, Portuguese pavement, is not smooth and contains loose stones which inevitably affect the measurements by introducing higher amplitude vibrations when the UGV is driven over them. This terrain also contains an approximately constant slope.



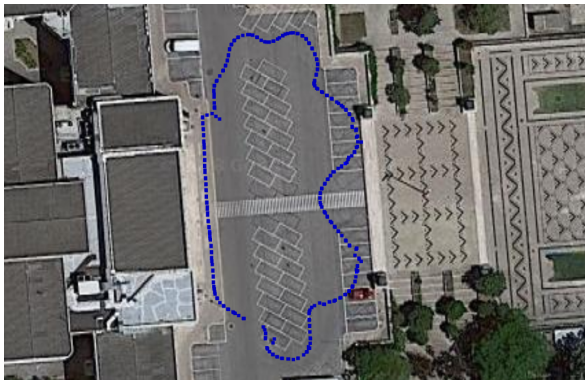
(a) First experiment.



(b) Second experiment.



(c) Third experiment.



(d) Fourth experiment.

Figure 6.1: Outdoor experiments' approximated path. Undersampled NovAtel GPS receiver samples in blue.

Table 6.1: Experiments' traveled distance.

Experiment	Traveled distance $[m]$
First	141.16
Second	275.04
Third	309.32
Fourth	137.15

In the third experiment, the terrain type, and slope changes. This experiment begins in Portuguese pavement and ends in asphalt, it also encompasses diverse slopes along the way.

During the fourth experiment, we drove the UGV on asphalt for the full duration of the test. The slope is approximately constant and leveled in the area in which the UGV was driven. This terrain is smoother than the Portuguese pavement. As demonstrated further on, this results in better overall localization estimation for this experiment.

During the outdoor experiments, we do not have access to a reliable pose ground truth, as opposed to the indoor experiments. The collected GPS measurements act a reference but note that the confidence of this position estimate is low. Also, the GPS measurements do not provide a reliable attitude estimation.

Table 6.1 contains the traveled distance during each experiment. This value is relevant because in experiments with a higher traveled distance, we expect higher pose errors for the estimates A to D, since these only predict the position but do not update it.

The experiment in which the travel distance is larger corresponds to the third one, followed by the second, first and fourth. The third experiment's average position errors are the highest among all experiments, since it is the one with the largest traveled distance, and its path encompasses the largest differences in terrain type and slope. These lead to increased measurements' noise.

We produced videos in which is possible to visualize how the experiments evolve with time. Three videos were produced:

- a video of the third experiment estimates[1],
- a video of the fourth experiment estimates[2],
- and a video of an experiment not featured in the results but in which we observe the UGV from two different view points and the estimates evolution[3].

6.2 Localization methods

Multiple localization estimates are obtained in order to evaluate the effect of the different sensors in the localization estimate. In Section 4.4, we give a detailed explanation of the methods used to obtain the following estimates. These estimates are first compared using the average position error along the experiment, taking the GPS's measurement as a reference. These errors are computed as described in 4.5.

Note that in the figures of estimates C and D the final position's 2D standard deviation ellipse is given. This ellipse represents the 2D position uncertainty with 99% confidence.

6.2.1 Localization method A

As usual with UGV's, our localization estimation starts with the study of the vehicle's odometry (localization method A).

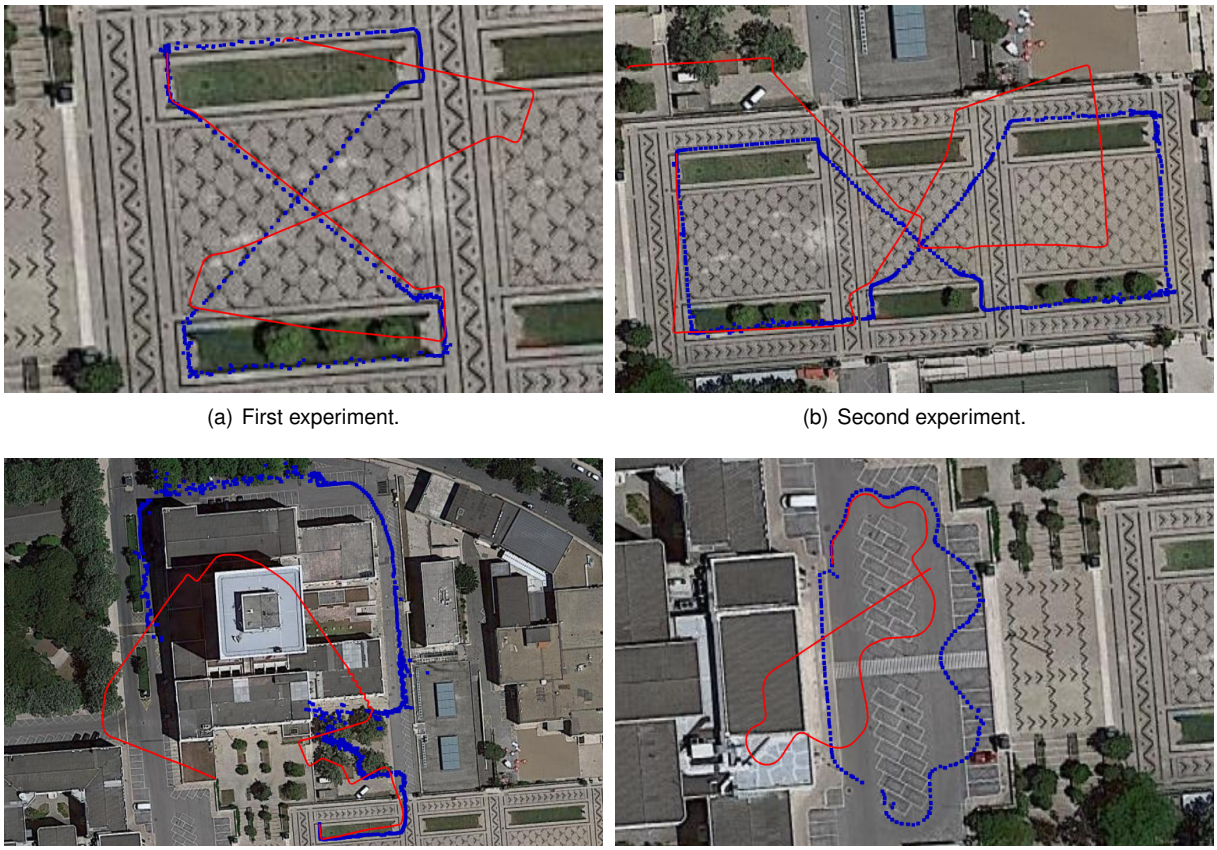
Figure 6.2 depicts the same GPS samples, along with the UGV's odometry, here shown in red. As stated before and shown here, the odometry drifts very quickly, being the drift in the estimated heading the main source of error.

The average position error considering the GPS's measurements is depicted in Table 6.2.

Table 6.2: Estimate A's average position error.

Experiment	Average position error $[m]$
First	5.98
Second	10.74
Third	25.75
Fourth	12.44

[1] https://youtu.be/CLq6-c_llyU
[2] https://youtu.be/MFZQf5r11gw
[3] https://youtu.be/UER24UXgHsc



(a) First experiment. (b) Second experiment.

(c) Third experiment. (d) Fourth experiment.

Figure 6.2: Estimate A, odometry. Undersampled GPS' samples in blue and estimate A in red.

As expected the error is higher for the experiments with higher traveled distance, although not true for every experiment. The first experiment has a lower error but a marginally higher traveled distance than the fourth.

6.2.2 Localization method B

The estimate B is obtained by calibration of the UGV's odometry.

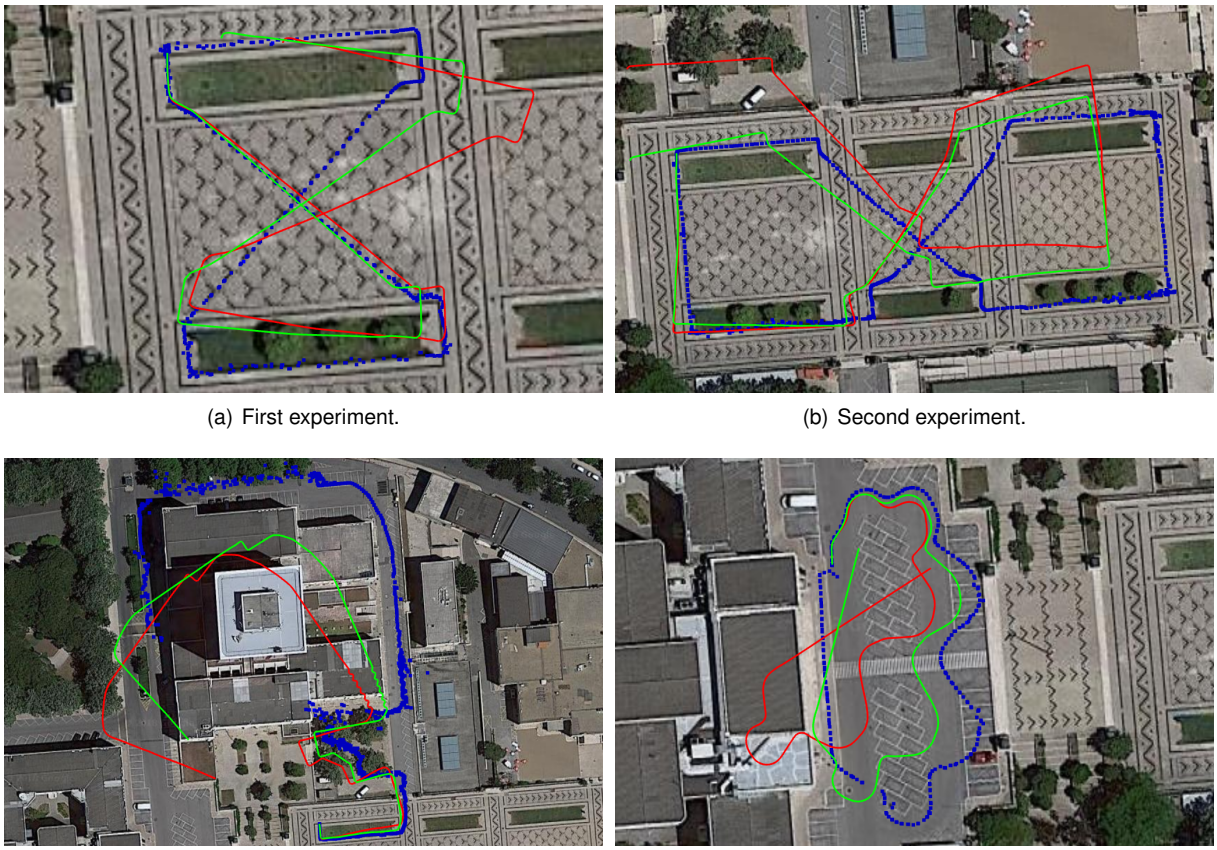
Figure 6.3 depicts the comparison between the UGV's odometry and the calibrated one. The calibrated odometry is shown in green.

The first, second and fourth experiments finish with almost the same pose as they started. For those, we observe that the calibrated odometry's last estimate is much closer to the starting position than the non-calibrated odometry.

The third experiment does not finish with the same pose as it started. Nonetheless, we observe that the calibrated odometry is closer to the approximated path given by the GPS samples.

Analyzing the path shape for the experiments, still with heavy distortion, we see that the overall calibrated path shape is closer to the GPS samples than the odometry's path (estimate A). The average position error taking the GPS's measurement is depicted in Table 6.3.

The error follows the same behavior as in the estimate A, regarding the traveled distance. The error lowered across all experiments, but for the fourth experiment the decrease is especially evident



(a) First experiment. (b) Second experiment.

(c) Third experiment. (d) Fourth experiment.

Figure 6.3: Estimate B, calibrated odometry. Undersampled GPS' samples in blue, estimate A in red and estimate B in green.

Table 6.3: Estimate B's average position error.

Experiment	Average position error $[m]$
First	3.42
Second	8.22
Third	16.37
Fourth	4.96

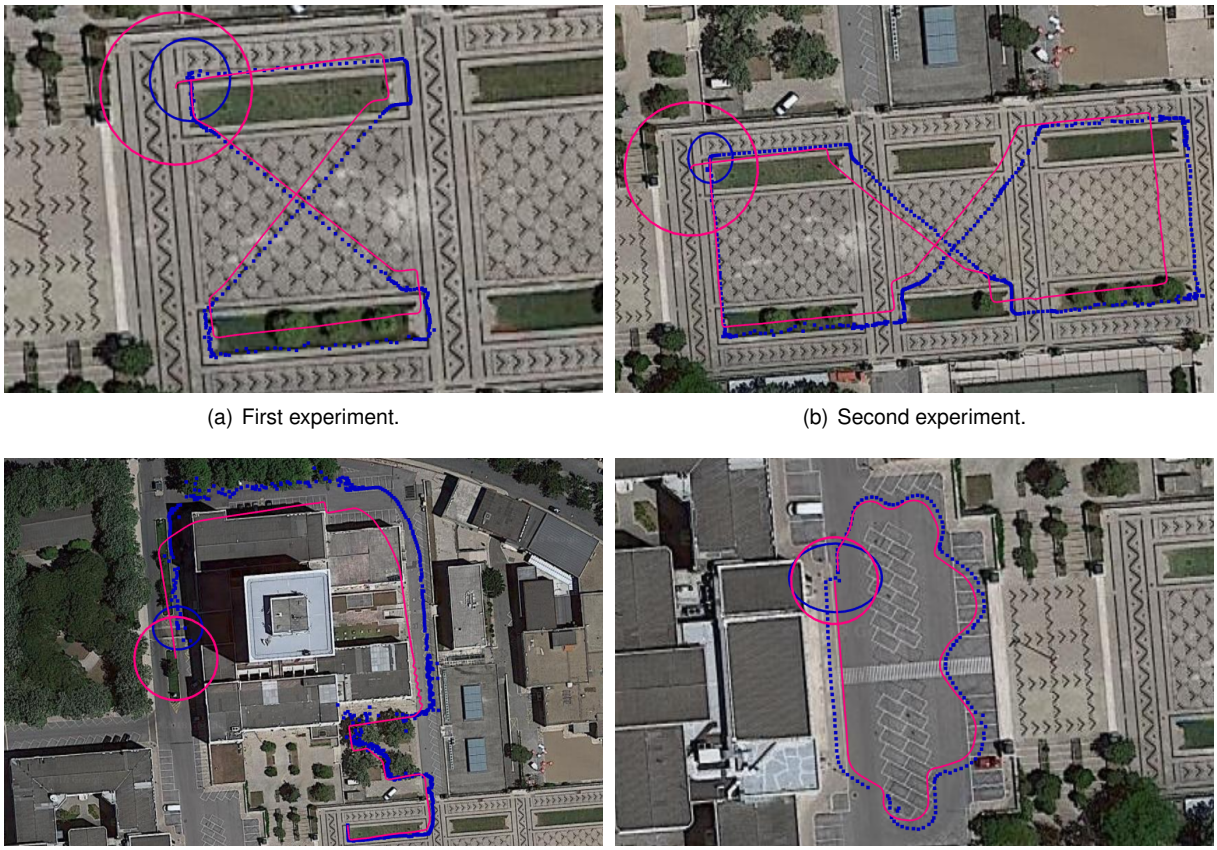
being of approximately 250% when compared to the one from the previous method. For the rest of the experiments is of approximately 150%.

6.2.3 Localization method C

Estimate C is obtained by filtering the calibrated odometry velocities and the MPU IMU measurements. This is achieved using the EKF described in Section 4.3. In Figure 6.4, the filter's estimate is illustrated, along with the final 2D position's uncertainty. The estimate is shown in magenta.

The average position error considering the GPS's measurements is depicted in Table 6.4.

The use of a reliable heading estimate, along with the angular velocities and linear accelerations, results in a decrease in the average position error of more than 350% in the case of the fourth experiment. For the remaining experiments, the decrease is above 170%.



(a) First experiment. (b) Second experiment.

(c) Third experiment. (d) Fourth experiment.

Figure 6.4: Estimate C, EKF filtering calibrated odometry and MPU IMU's measurements. Undersampled GPS' samples in blue and estimate C in magenta, along final's 2D position's uncertainty.

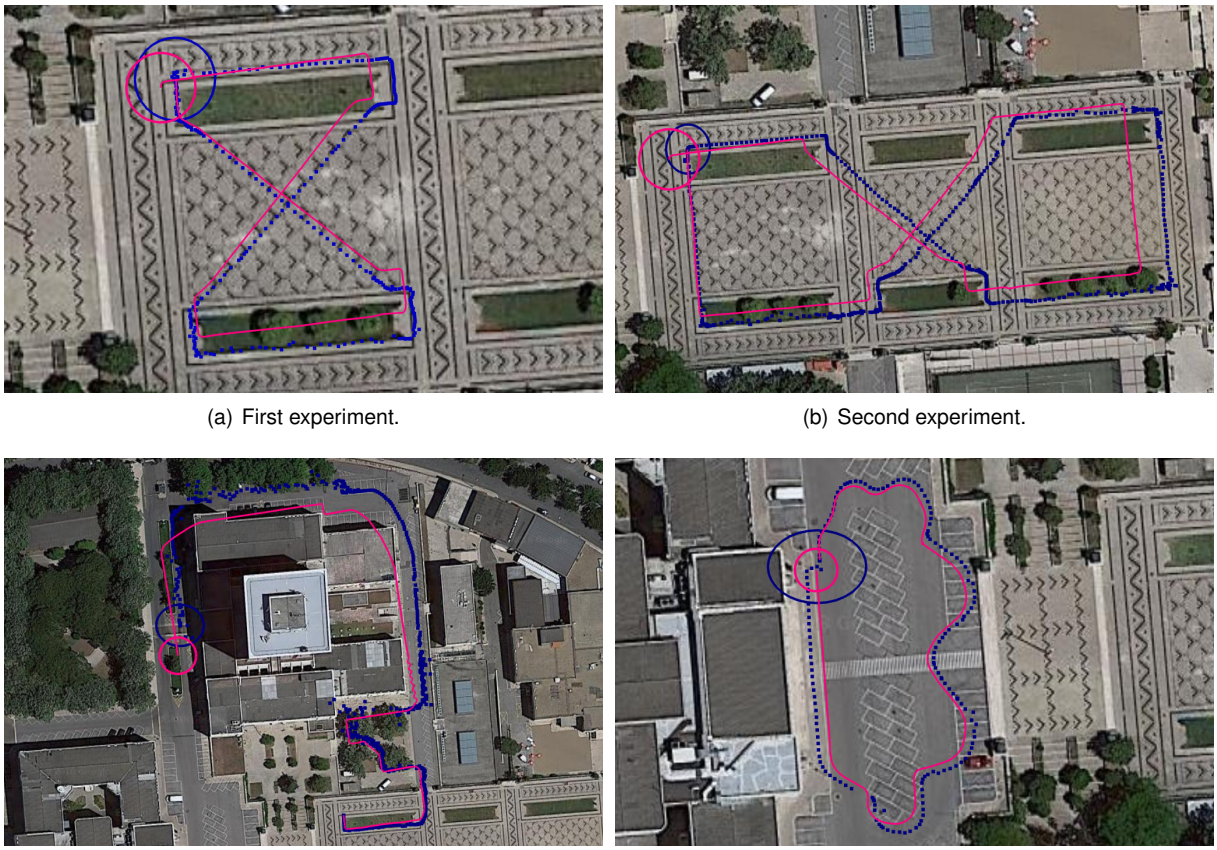
Table 6.4: Estimate C's average position error.

Experiment	Average position error $[m]$
First	2.00
Second	4.05
Third	5.67
Fourth	1.40

The error continues to have the behavior predicted in the beginning. It is to note the difference between the first and fourth experiments' error. As mentioned earlier, the first experiment's traveled distance is only marginally higher than the fourth one, yet the error is 140% higher than the fourth experiment's error. This is due to the exceptionally favorable conditions under which the fourth experiment was conducted. The first experiment was performed in an irregular terrain which contains a slope, while the fourth experiment was performed in a relatively smooth and leveled terrain.

6.2.4 Localization method D

The estimate D is obtained by filtering the calibrated odometry velocities, the MPU IMU and the Crossbow IMU measurements, using the previously mentioned EKF. Its results, along with the final position's uncertainty, are shown in Figure 6.5.



(a) First experiment.

(b) Second experiment.

(c) Third experiment.

(d) Fourth experiment.

Figure 6.5: Estimate D, EKF filtering calibrated odometry, MPU and Crossbow IMU's measurements. Undersampled GPS' samples in blue and estimate D in magenta, along final's 2D position's uncertainty.

The overall path is similar to the previous estimate. This is to be expected since the addition of the Crossbow IMU brings measurements to state variables which were already being updated by the MPU IMU measurements.

The average position error considering the GPS's measurements is depicted in Table 6.5.

Table 6.5: Estimate D's average position error.

Experiment	Average position error $[m]$
First	2.01
Second	4.03
Third	5.45
Fourth	1.42

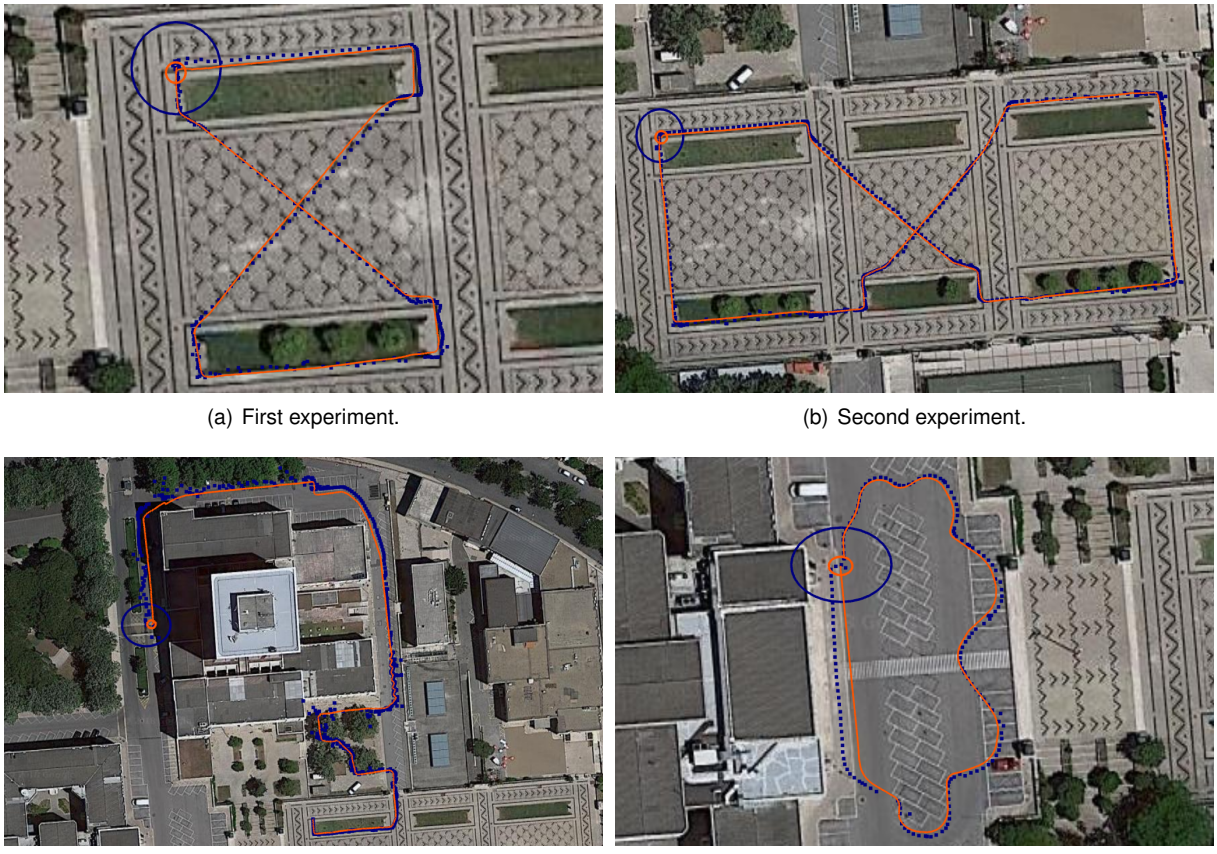
As already mentioned, this estimate's overall path is similar to estimate C's, this is also apparent in the experiments' average position error. These are similar to the previous estimate's experiments average position errors for all the experiments, having marginally decreased for the second and third experiments, while marginally increasing for the first and fourth experiments.

Table 6.6: Estimate E's average position error.

Experiment	Average position error $[m]$
First	0.52
Second	0.44
Third	1.34
Fourth	0.97

6.2.5 Localization method E

The estimate E is obtained by filtering the calibrated odometry velocities, the MPU IMU, the Crossbow IMU and the NovAtel GPS receiver's measurements. Its results, along with the final position's uncertainty, are shown in Figure 6.6.



(a) First experiment. (b) Second experiment.

(c) Third experiment. (d) Fourth experiment.

Figure 6.6: Estimate E, EKF filtering calibrated odometry, MPU IMU, Crossbow IMU and NovAtel GPS receiver's measurements. Undersampled GPS' samples in blue and estimate E in orange, along final's 2D position's uncertainty.

The average position error considering the GPS's measurement is depicted in Table 6.6.

The addition of global position's measurements results in lower errors across all experiments since we are now updating the state variables used in the errors computation. The effects of the GPS's measurements use are evident in the estimated path.

We should note the high error for the third experiment. This experiment has the highest final position's error in every estimate, this is due to being the one with the highest traveled path and only predicting

the position variables in the previous estimates. In this estimate, we are also updating the position state variables, which should result in an approximately uniform error along the four experiments. This does not occur due to the GPS's measurements higher mean dispersion in the location where the experiment finished.

Estimate E's final 2D position's uncertainty is also much lower than in any of the previous estimates.

Considering all the estimates, this is the one which we expect to have the lowest pose error. While still going over grass, when curving, in the first and second experiment, the estimate does not go over any buildings. This happened in every third experiment's localization estimate except this one.

Robustness to GPS measurements outages

GPS measurements outages can occur for multiple reasons. In order to evaluate the robustness of method E to outages in GPS measurements, effectively becoming method D during the outage, we performed an experiment in which this measurements were partially suppressed. These were suppressed in the middle of the experiment so we can observe the behavior when the measurements are lost and when they become available.

Figure 6.7 illustrates the effect of a GPS outage using data from the third experiment and the estimate obtained using method E. Due to the modularity of the system, it was not required to make any change to the EKF or to its parameters in order to compute this results.

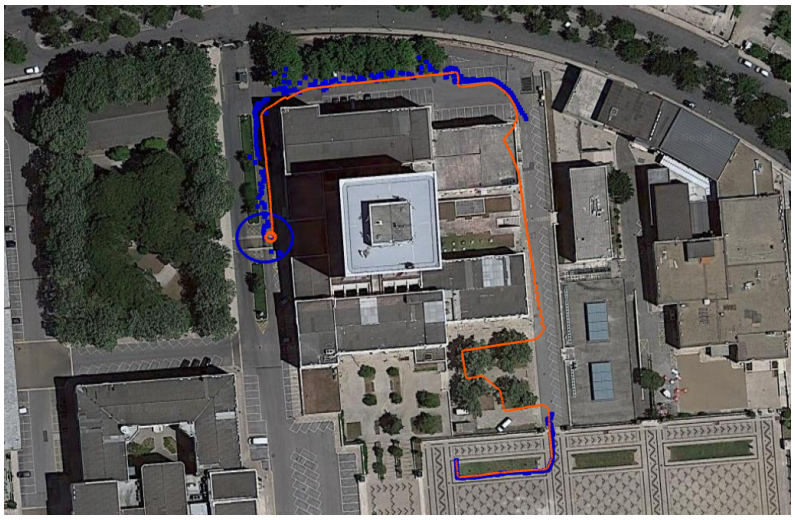


Figure 6.7: GPS outage experiment while using estimate E. Undersampled GPS' samples in blue and estimate E in orange, along final's 2D position's uncertainty.

When loosing the GPS measurements the estimate does not exhibit any unexpected behavior, only predicting the position but not updating it. Towards the position where the GPS measurements become once again available, we observe a deviation in the position estimate, characteristic of dead-reckoning techniques. Once the measurements are available, the estimate converges to the GPS measurements in a few meters.

When compared to the estimate provided by method E without the GPS measurements outages, this estimate has an average position error of $1.48\,[m]$.

6.3 Discussion

The estimates average position error is now illustrated for all the experiments in Figure 6.8.

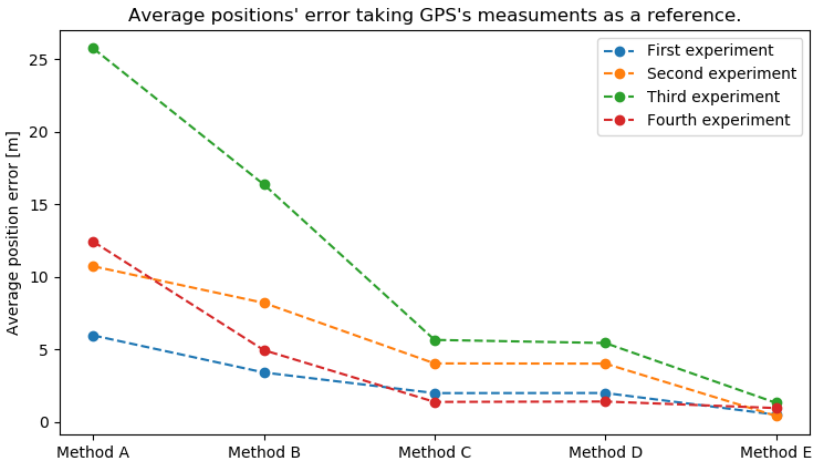


Figure 6.8: Average position's error taking GPS's measurements as a reference.

As we determined, estimate E's position estimate is the best when compared against the GPS's measurements. Yet, this estimate uses GPS and thus not making it a viable solution for our problem. Using estimate E we can reliably compare and measure the error for estimate A, B, C, and D.

The metrics described in 4.5 are now used to compare all the viable estimates to our reference, estimate E.

Estimate A to D's pose errors are computed accordingly. These, for position and attitude, are illustrated in Figures 6.9 and 6.10, respectively.

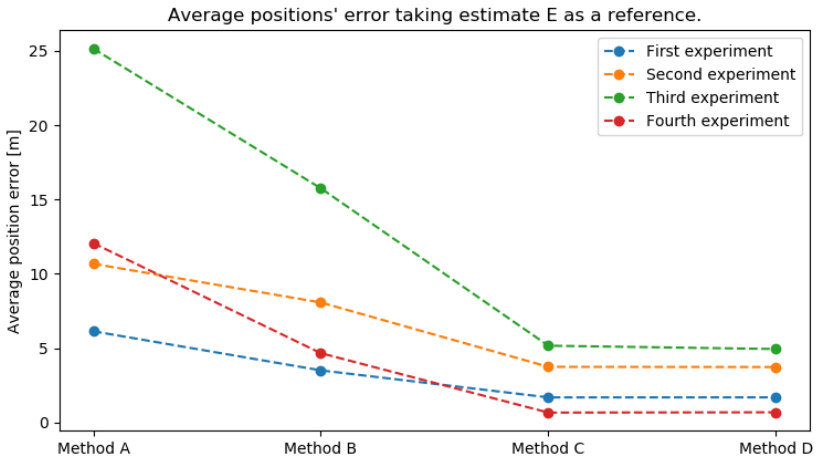


Figure 6.9: Average position error taking estimate E as a reference.

Regarding the position, we find that the experiments' average error does not converge to the same value. While still tending to decrease, estimate D's error is still very distinct between different experiments. This indicates that the error is still dependent on the experiment's characteristics, such as terrain

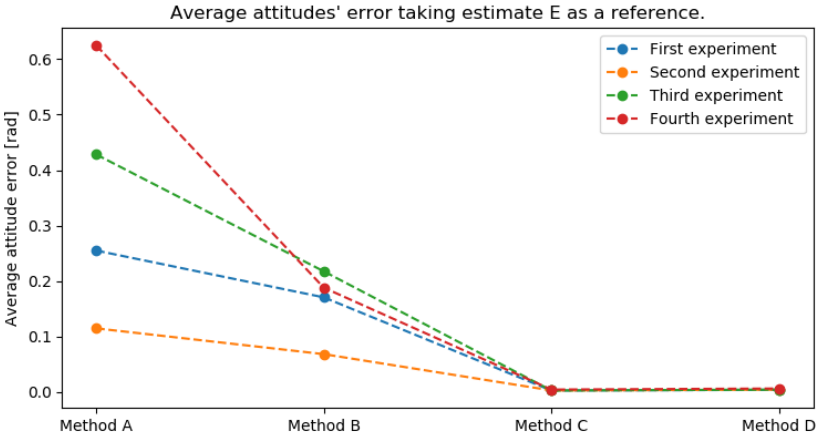


Figure 6.10: Average attitude error taking estimate E as a reference.

and traveled distance. Regarding method D, the average position error is $1.70[m]$ for the first experiment, $3.74[m]$ for the second, $4.95[m]$ for the third and $0.7[m]$ for the fourth.

Regarding the attitude, the estimates' average error behave differently than in the position's error.

In estimate A, the UGV's odometry, the experiment with the highest error is the fourth experiment. This contrasts with the previous results, where the third experiment is, by a large margin, the one with the highest errors.

In estimate B, calibrated odometry, all the errors decrease, with special significance for the fourth experiment. While these experience's error decreased significantly, this experiment still has the third highest error. The experiment with the highest is the third experiment.

In estimate C, the MPU IMU attitude estimate is introduced, lowering the error significantly in both metrics. This is expected since we identified that heading error significantly affected the localization estimation.

In the fourth estimate, the Crossbow IMU is introduced. As observed in the experiments' figures, this did not change the path shape significantly. This is reflected in the error, remaining approximately constant for estimate C and D.

The relevance of this last estimate is in the method robustness. If for some reason the MPU IMU stops providing measurements, the Crossbow IMU still continues to provide the same measurements as the MPU IMU with exception of the attitude. Also, the position covariance obtained using this method is inferior to the one estimated using method C, providing this way a more meaningful estimate.

It is to note the difference between the position and attitude's average error. In the position's error estimation, the localization estimates are compared against one which uses a global localization system. This system's measurements were not used in the previous estimates, thus estimate A to D's position state variables were not updated but only predicted.

In the attitude error estimation, the localization estimates are compared against estimate E which updates its state using the same references as D, the MPU and, Crossbow IMUs, while method C only uses the MPU IMU. This results in the different behavior observed between the position and attitude

error since for estimate D, the attitude and respective angular velocities are predicted and updated in the same way and the Crossbow IMU effect is low when used along the MPU IMU measurements.

The fact that the position error highly differs between experiments, indicates that our approach is only robust enough for outdoor environments in which we are not deprived of global position measurements.

Chapter 7

Conclusions

In this chapter, we summarize the thesis main achievements and propose future work.

7.1 Achievements

During the course of this thesis, we modified a robust system used to conduct several experiments and studied localization methods which use the available sensor suite.

We updated the UGV and studied the capabilities of its previous sensor suite, the sensors which were deprecated were removed. Working on the existent sensors, we identified the missing features by evaluating the performance of each one. Using this information, updated sensors were added and its capabilities studied.

After studying the noise sources which affect the UGV and all its sensors, we modeled the multiple sensors comprised in the attained sensor suite. Using data collected in an indoor environment, we estimated all the parameters which define the UGV's behavior, along with all the parameters required to determine the sensors models.

We conducted real-world experiments in outdoor environments in which all the data from the sensors is recorded. With it and using different methods, we obtained several estimates which performed differently. We assessed their performance using an estimate which was obtained but comprises GPS's measurements, making it an invalid solution for our problem but a reliable pose estimate. With this, we validate the previously estimated models and its parameters.

The simplicity of our system allowed for the timely development of this solution. We stress that the UGV was not operational for several years prior and a lot of effort was made to bring it to the point where is capable of reliably performing outdoor experiments.

For accomplishing such solution, we developed several software packages, this allowed us to assess and test multiple sensors which data was not available before. The ROS packages *atrvjr_description*, *atrvjr*, *host_console*, *odom_calibration*, and *xbow6x* were developed during the course of this thesis and several other studied and modified.

Given the simplicity and flexibility of this approach, the developed methods are a strong foundation

to incorporate measurements computed by other algorithms. Yet, due to its low accuracy, the obtained results only serve a limited range of applications.

7.2 Future Work

For attaining a UGV capable of performing tasks such as the ones outlined in the MBZIRC's challenges description, several sensors and systems still need to be added to the UGV's configuration. Besides the ones used, there are other sensors which are commonly used in localization such, scanning laser rangefinders, depth and, RGB cameras. While it was planned, due to time constraints, it was not possible to assess the performance of a scanning laser rangefinder, methods such as scan matching can be used to provide a second source of velocities in 2D. Depth cameras usually use infrared laser technology these are not reliable outdoor due to the presence of infrared radiation emitted by the sun, yet depth information can be obtained from other sensors such as an array of RGB cameras. Regarding the robotic arm, we studied two which were previously proposed, we are confident that this analysis will be relevant at the time of acquiring and adding such hardware to the UGV.

We prepared our method to estimate 3D pose, but due to the noisy measurements along the Z_w axis, specifically from the IMUs, the estimation diverged along this axis. With the addition of other sensors, such as visual odometry, this problem can be attenuated and a 3D pose estimate obtained.

One of the pose error sources is bad attitude estimation, this has been addressed with the addition of the MPU-6050 IMU, but as previously mentioned, an IMU with a magnetometer must be used to reliably estimate the attitude over longer periods of time. Using a magnetometer in the attitude estimation opens doors to improve the pose estimation, yet this sensors are very sensitive to interferences from different sources. Depending on the operational conditions, this might or might not provide a reliable attitude estimate.

The ROS package which has the implementation of the used EKF also contains another filter which uses most of the same configuration, but it is based on Unscented Kalman Filtering [38]. Due to time constraints, it was not possible to obtain an estimation using such filter, but improvements in the state estimation have been reported by others when using such filter [39]. Nonetheless, the improvement in state estimation is gain at the cost of loss of computational efficiency.

Bibliography

[1] M. Chui, J. Manyika, and M. Miremadi. Four fundamentals of workplace automation. *McKinsey Quarterly*, 29(3):1–9, 2015.

[2] H. S. Ahn, I. Sa, and F. Dayoub. Introduction to the special issue on precision agricultural robotics and autonomous farming technologies. *IEEE Robotics and Automation Letters*, 3(4):4435–4438, Oct 2018. ISSN 2377-3766.

[3] A. Davids. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, 17(2):81–83, 2002.

[4] T. Bock. The future of construction automation: Technological disruption and the upcoming ubiquity of robotics. *Automation in Construction*, 59:113 – 121, 2015. ISSN 0926-5805.

[5] J. Jones. Robots at the tipping point: the road to irobot roomba. 13:76 – 78, 04 2006.

[6] M. T. Chew, S. Demidenko, C. Messom, and G. S. Gupta. Robotics competitions in engineering eduction. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 624–627, Feb 2000.

[7] MBZIRC organization. MBZIRC 2020 challenge description. Technical Report v2, MBZIRC, May 2018.

[8] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios. Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys Tutorials*, 19(2):1327–1346, Secondquarter 2017. ISSN 1553-877X.

[9] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6):869–880, Dec 1996.

[10] P. Goel, S. I. Roumeliotis, and G. S. Sukhatme. Robust localization using relative and absolute position estimates. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 2, pages 1134–1140 vol.2, Oct 1999.

[11] K. A. M. B. W. B. Giorgio Grisetti, Cyrill Stachniss. Odometry calibration by least squares, 2010. URL `http://ais.informatik.uni-freiburg.de/teaching/ws10/robotics2/pdfs/rob2-05-odometry-calib-practical.pdf`.

[12] L. Jetto, S. Longhi, and G. Venturini. Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 15(2):219–229, 1999.

[13] I. J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, April 1991. ISSN 1042-296X.

[14] S. D. Gupta, J. Y. Yu, M. Mallick, M. Coates, and M. Morelande. Comparison of angle-only filtering algorithms in 3d using ekf, ukf, pf, pff, and ensemble kf. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 1649–1656, July 2015.

[15] J. Simanek, M. Reinstein, and V. Kubelka. Evaluation of the ekf-based estimation architectures for data fusion in mobile robots. *IEEE/ASME Transactions on Mechatronics*, 20(2):985–990, April 2015. ISSN 1083-4435.

[16] E.-H. Shin. Estimation techniques for low-cost inertial navigation. *UCGE report*, 20219, 2005.

[17] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[18] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, May 2016. doi: 10.1109/ICRA.2016.7487628.

[19] A. Van Dierendonck, P. FENTON, and T. Ford. Theory and performance of narrow correlator spacing in a gps receiver. 39, 09 1992.

[20] Implement lateral position accuracy under rtk-gps tractor guidance. *Computers and Electronics in Agriculture*, 59(1):31 – 38, 2007.

[21] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[22] G. J. Bierman and C. L. Thornton. Numerical comparison of kalman filter algorithms: Orbit determination case study. *Automatica*, 13(1):23 – 35, 1977. ISSN 0005-1098.

[23] A. Cavallo, A. Cirillo, P. Cirillo, G. D. Maria, P. Falco, C. Natale, and S. Pirozzi. Experimental comparison of sensor fusion algorithms for attitude estimation. *IFAC Proceedings Volumes*, 47(3): 7585 – 7591, 2014. ISSN 1474-6670. 19th IFAC World Congress.

[24] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, June 2011. doi: 10.1109/ICORR.2011.5975346.

[25] R. G. Valenti, I. Dryanovski, and J. Xiao. Keeping a good attitude: A quaternion-based orientation filter for imus and margs. *Sensors*, 15(8):19302–19330, 2015. ISSN 1424-8220. doi: 10.3390/s150819302.

[26] D. industry. Panda, 2018. URL `http://www.directindustry.com/prod/franka-emika/product-187686-1906234.html`.

[27] Z. L. Skovsgaard. Cobots - collaborative robots - industrial robots - universal-robots - ur5 - ur10 - specifications. URL `http://www.zacobria.com/universal-robots-zacobria-industrial-robot-ur5-ur10-specifications.html`.

[28] J. Chen, S. Shen, and H. Y. K. Lau. Hitting flying objects with learning from demonstration. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 55–60, July 2017.

[29] Z. Deng, J. Mi, D. Han, R. Huang, X. Xiong, and J. Zhang. Hierarchical robot learning for physical collaboration between humans and robots. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 750–755, Dec 2017.

[30] T. Andersen. *Optimizing the Universal Robots ROS driver.* Technical University of Denmark, Department of Electrical Engineering, 2015.

[31] *Service Manual*. Universal Robots, Energivej 25 DK-5260 Odense S, ur5_en_3.1.3 edition, 2016.

[32] H. Engemann, P. Wiesen, S. Kallweit, H. Deshpande, and J. Schleupen. Autonomous mobile manipulation using ros, 06 2018.

[33] R. E. Andersen, E. B. Hansen, D. Cerny, S. Madsen, B. Pulendralingam, S. Bøgh, and D. Chrysostomou. Integration of a skill-based collaborative mobile robot in a smart cyber-physical environment. *Procedia Manufacturing*, 11:114 – 123, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[34] I. E. Commission. Degrees of protection provided by enclosures (IP Code). International standard, International Electrotechnical Commission, August 2013.

[35] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[36] S. Miller. Noise measurement, 2018. URL `https://www.sensorsmag.com/embedded/noise-measurement`.

[37] H. . Nguyen and F. Guillemin. On process noise covariance estimation. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, pages 1345–1348, July 2017. doi: 10.1109/MED.2017.7984305.

[38] E. A. Wan and R. V. D. Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, Oct 2000.

[39] M. St-Pierre and D. Gingras. Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system. In *IEEE Intelligent Vehicles Symposium*, pages 831–835. Citeseer, 2004.

[40] O. S. R. Foundation. common_msgs, 2014. URL `https://wiki.ros.org/common_msgs`.

Appendix A

Preliminary indoor experiments

To assess the different systems and to have data from which several required parameters for the sensor models could be estimated from, we conducted preliminary indoor experiments. Data from a MOCAP system and several sensors measurements was recorded.

A.1 Dataset

During the performed experiments all the data was saved in $.bag$ format, recorded using $rosbag$ tool. This format stores serialized message data which can be "played" in order to obtain offline estimates as if they were computed during the experiments. The relevant saved ROS topics and their message type are shown in Table A.1. All the topic types are standard ROS messages and their description can be found in ROS documentation [40].

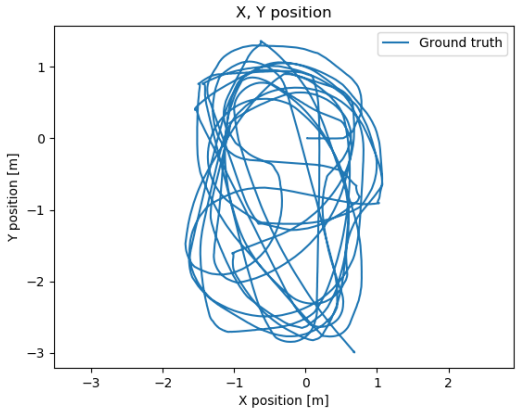
Table A.1: Relevant ROS topics in the indoor dataset and respective message type.

ROS Topic	Message Type
/atrvjr/mpu/imu/data	sensor_msgs/Imu
/rflex/atrvjr_node/odom	nav_msgs/Odometry
/vrpn_client_node/RigidBody1/pose	geometry_msgs/PoseStamped
/xbow6x/xbow6x_node/data_raw	sensor_msgs/Imu

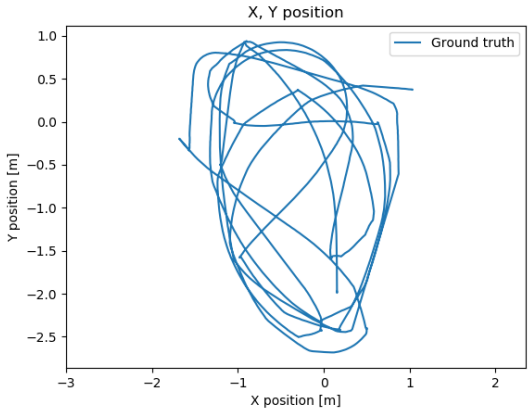
The ROS topic */atrvjr/mpu/imu/data* contains all the measurements data from the MPU-6050 IMU, */rflex/atrvjr_node/odom* contains the UGV's odometry, */vrpn_client_node/RigidBody1/pose* contains the ground truth poses provided by the MOCAP system and */xbow6x/xbow6x_node/data_raw* contains the data from the Crossbow IMU.

A.2 Experiments

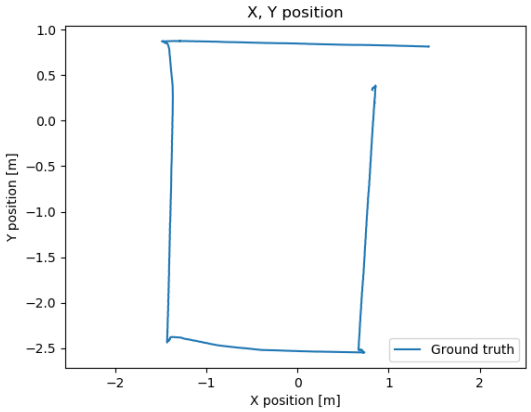
Three experiments were conducted in a room covered by a MOCAP system which is used as ground truth. The esperiments' 2D paths are depicted in Figure A.1.



(a) First experiment.



(b) Second experiment.



(c) Third experiment.

Figure A.1: Indoor experiments' path.

The first experiment consists in driving randomly at normal velocity and the second experiment consists in driving randomly at a higher velocity. The third experiment is approximately a rectangle.

These experiments' datasets were used to estimate multiple parameters for the sensor models. The travelled distance for each experiment is given in Table A.2.

Table A.2: Experiments' traveled distance.

Experiment	Traveled distance $[m]$
First	126.12
Second	57.15
Third	11.38

Appendix B

Outdoor experiments' dataset

Building on the description made from the indoor datasets present in Appendix A, we detail the additional information which is collected when conducting outdoor experiments.

B.1 Dataset

We save multiple ROS topics when conducting outdoor experiments, Table B.1 depicts the relevant ROS topics and the respective type.

Table B.1: Relevant ROS's topics in the outdoor dataset and respective message type.

ROS Topic	Message Type
/atrvjr/mpu/imu/data	sensor_msgs/Imu
/novatel/novatel_node/full_psr	novatel_msgs/BESTPOS
/novatel/novatel_node/gps_fix_psr	sensor_msgs/NavSatFix
/rflex/atrvjr_node/cmd_vel	geometry_msgs/Twist
/rflex/atrvjr_node/odom	nav_msgs/Odometry
/xbow6x/xbow6x_node/data_raw	sensor_msgs/Imu

This dataset contains two additional ROS topics when compared with the indoor dataset, */novatel/novatel_node/full_psr* and */novatel/novatel_node/gps_fix_psr*. The first contains the complete GPS message sent from the GPS receiver, this is important for debugging purposes since it contains important information such as the number of satellites in view. The second contains a standard ROS message which contains information such as latitude, longitude, altitude and respective covariance matrix.

Unlike in the indoor datasets, in outdoor experiments we use non standard ROS messages, namely to save the full message send by the NovAtel GPS receiver.

The full description of this message is found in Table B.2.

Table B.2: Description of novatel_msgs/BESTPOS message.

Field	Description
novatel_msgs/CommonHeader header	NovAtel standard header
uint32 solution_status	Solution status
uint32 position_type	Position type
float64 lat	Latitude
float64 lon	Longitude
float64 hgt	Height above mean sea level
float32 undulation	Relationship between the geoid and the WGS84 ellipsoid
uint32 datum_id	Datum ID number
float32 lat_std	Latitude standard deviation
float32 lon_std	Longitude standard deviation
float32 hgt_std	Height standard deviation
char[4] stn_id	Base station ID
float32 diff_age	Differential age in seconds
float32 sol_age	Solution age in seconds
uint8 obs	Number of observations tracked
uint8 gpsl1	Number of GPS L1 ranges used in computation