Adaptive kinematic particle filter classifier for autonomous robots

MSc Research Project
Data Analytics

Ayoola Idris-Animashaun
Student ID: x20103689

School of Computing
National College of Ireland

Supervisor: William Clifford

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ayoola Idris-Animashaun
Student ID:	x20103689
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	William Clifford
Submission Due Date:	31/1/2022
Project Title:	Adaptive kinematic particle filter classifier for autonomous robots
Word Count:	6500
Page Count:	24

Signature:	
Date:	31st January 2022

Adaptive kinematic particle filter classifier for autonomous robots

Ayoola Idris-Animashaun
x20103689

Abstract

Autonomous robots are finding more industrial and home usage. One of the challenges these robots face is how the robot can perceive its environment and move within the environment. This becomes more necessary when the environment is new or it changes. The robot uses unique landmarks within an environment to find itself. Particles, which are belief states of where the robot could be in an environment are generated, evaluated, resampled and when they converge, used to estimate the robot's position. Sometimes, there is no landmark and this causes the particles to diverge from the robot's position. Machine learning classifiers are used to detect the type of wheel on a robot, and depending on the wheel type, some constraint can be applied to the particle inputs to enable the particles continue on a set trajectory when the robot does not have any landmark to guide it within the environment.

Keywords - Particle filter, Robots, Bayes Filter

1 Introduction

1.1 Motivation

Since the earliest robots were created in the early 1950s by George C. Devol, Roberts (1999), robots continue to play major roles in helping humans accomplish tasks across various aspects of our existence and continued survival. Robots find use in space, warehousing, security, agriculture, and educationMuir & Neuman (1987)

For a robot to navigate the real world, it is required that it has a perception of its environment. Typically, a robot's understanding of the real world is represented by a map Roland & Illah (2004). With a map, the robot must localize where it currently exists on that map, and plan its navigation within the environment.

1.2 Research Objective

To begin, the author uses the term 'virtual robot' as any autonomous entity designed to function as a robot or vehicle possessing the ability to move, sense and estimate its location within a given environment. Landmarks are obstacles in that environment that the virtual robot uses to map out the environment after sensing these landmarks. It could be a door, or a wall or a pillar in the environment. Particle filters are known state representations of the virtual robot generated to estimate the probability of a robot to be in a particular location.

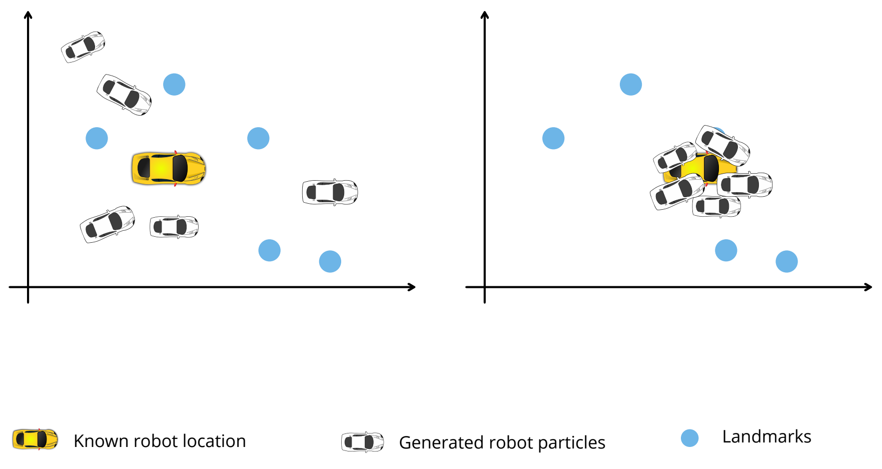

Figure 1: Project environment

Figure 1 shows an overview of the research. Since the environment is simulated, the virtual robot's position (yellow) is known. This is the ground truth. This information is not shared with the particles however, the particles receive motion input as instruction to move. The objective is to have the particle and the robot exist in the same location. This means the distance will be need to be close to zero as possible during evaluation.

In this project, the use of kinematics is proposed in developing a control mechanism for robots to use in navigating an unknown environment. Some other researchers have done something similar, Sabzevari & Scaramuzza (2016). Kinematics is the study of geometry about motion. The approach is to use the knowledge of motion and motion constraints to develop a predictive motion control software that limits itself based on the kinematic constraints related to the robot (or autonomous vehicle).

To achieve this, the author leverages a single hypothesis position representation to help the robot map its position within a region or use the multi-hypothesis representation. Although it has its drawbacks, Roland & Illah (2004), multi-hypothesis when used with probabilistic techniques have higher chances of helping the robot find its position from various possible robot positions. By selecting the position with the highest positional probability, the robot can maintain a set of beliefs that guides it. This is achieved by sending a control input and sensor measurements (contains noise) and use of particle filters to estimate the the robot location. More details on this in section 3. Some common issues with robot localization include errors in odometry measurements as well as the issue of dead reckoning. Dead reckoning is the issue where the particles diverge away from the robot when there is no landmark in sight. The further the robot moves without sensing a landmark, the more the particles disperse and the robots correct estimate of its location reduces. This research proposes a method to fix this issue by constraining the motion of the particles when there is no landmark by applying motion constraints that will force the particles to move in a specified way. By this, the particle diversion is limited and the robot's estimate does not suffer very much.

The objectives of the research are as follows;

1. How well can we predict the wheels on a robot based on the measured motion model outputs?
2. How can our prediction help the issue of dead reckoning?

In this paper, the author proposes an adaptive estimator that predicts the wheel motion on a robot based on the motion readings it gets over time. The objective is to come up with a technique to enable a robot predict the kind of wheels it has and as a result know the range of motions through which it can move thereby adjusting its localization efforts towards the possible locations when using a particle filter. Section 2 shows a summary of related works, section 3 gives the methodology we will employ while section 4 and section 5 give breakdown of the ethics and project plan for this work.

2 Related Works

2.1 Literature review

Robots are fast becoming a choice of research in the scientific community. The possibility of replacing humans for tedious and dangerous tasks make way for their acceptance in industries like emergency rescue, exploration, guides, transportation and even construction. An autonomous robot is one that is able to choose from a set of actions to carry out a specified task, Rubio et al. (2019). It uses a control system to guide, make decisions and guide its execution. We will discuss locomotion and perception in this research.

Locomotion covers how robots move. A robot can find itself in a controlled environment like a museum, retail store etc or an uncontrolled environment like a new planet. Different types of motion are available for the robot to move and factors like the surface the robot is moving on (land) or in (water), whether the robot is a wheeled or legged robot, the speed the robot can move, the weight of the robot and others contribute to considerations taken by the robot when it is moving. We focus on land based robots in this research. The wheeled land based mobile robot is popular and has the benefits of having a simple mechanism, balanced, can reach high speeds and efficiently move on man-made environments while the legged robots have a benefit of movement on terrains that are not flat. In fact, some researchers have proposed hybrid robots which have legs and wheels to address any terrain the robot finds itself. Adachi et al. (1999). A look at Figure 2 shows ANYmal, an autonomous robots that integrates four legs with four wheels for a smoother and efficient motion.

We focus on wheeled mobile robots in this research and study the kinematics involved for stable motion and control of the robot. Kinematics refers to how material bodies move without consideration of their mass or inertia and any force that produces the motion.

There are commonly six wheel types used in roboticsRoland & Illah (2004). Figure 3 shows various wheel types. We concentrate on training a model to tell the range of motions possible between a fixed wheel that can only move forward/backwards and an omni-wheel that can move in any direction.

A wheeled motor robot usually has to find its position in the world. Various techniques have been used to guide robots. One of such method is using odometry to calculate the velocity and hence find the displacement of the robot. The individual velocity of the wheels attached to the robot is calculated taking the constraint on each of the wheels

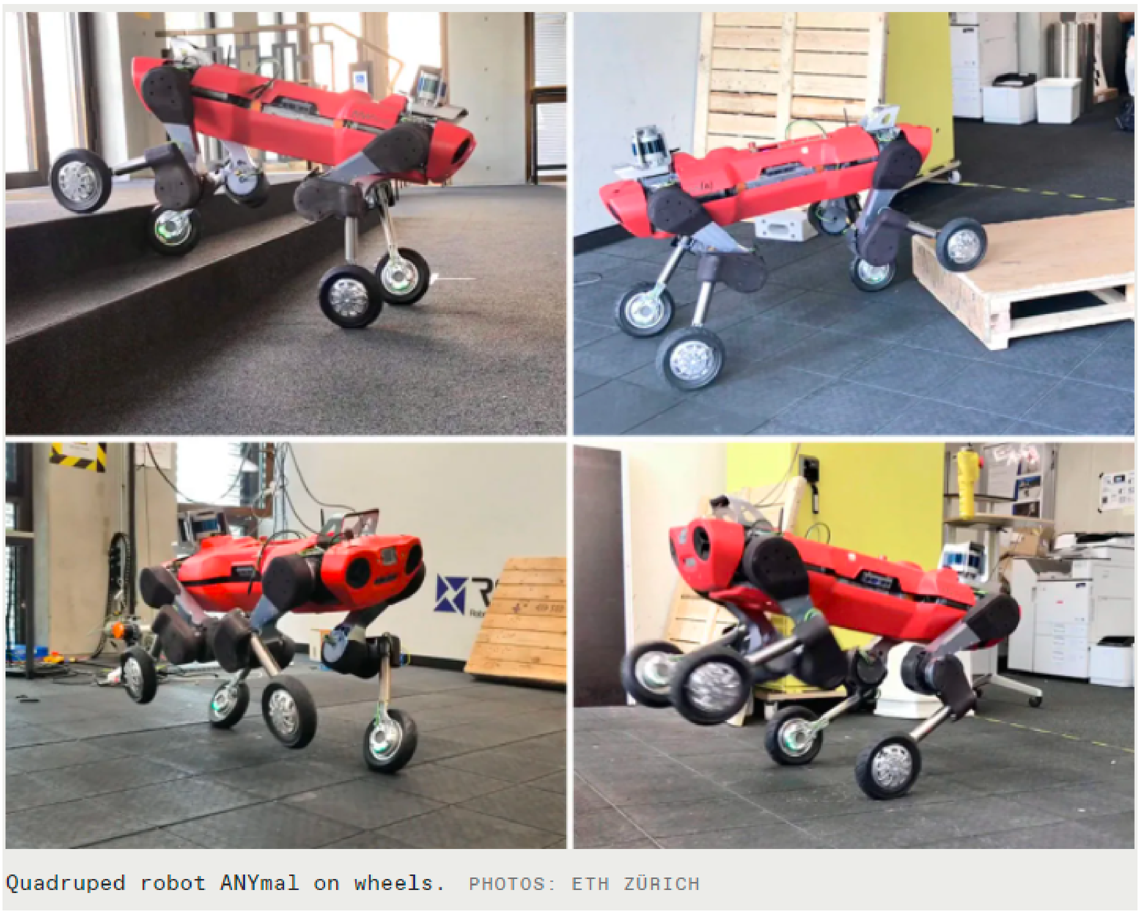

Figure 2: Anymal- Hybrid robot with legs and wheels

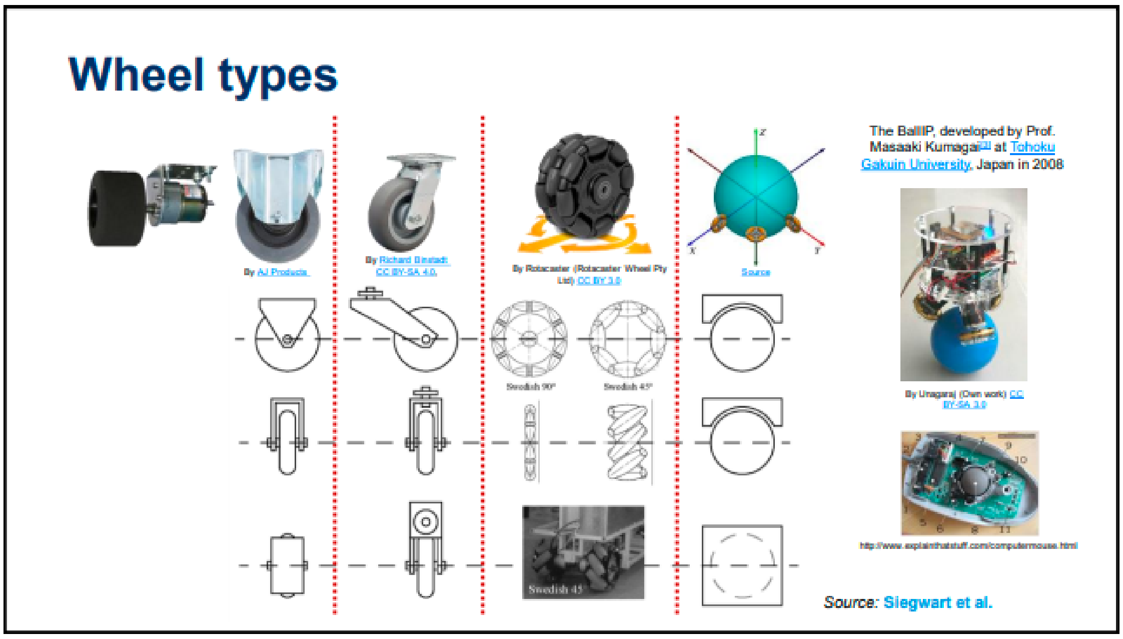
Wheel types
By AJ Products
By Richard Binstadt CC BY-SA 4.0
By Rotacaster (Rotacaster Wheel Pty Ltd) CC BY 3.0
Source
The BallIP, developed by Prof. Masaaki Kumagai at Tohoku Gakuin University, Japan in 2008
Swedish 90°
Swedish 45°
By Uragaraj (Own work) CC BY-SA 3.0
Swedish 45
http://www.explainthatstuff.com/computermouse.html
Source: Siegwart et al.

Figure 3: Robot Wheel types

into account. A global reference ξ_G usually has the position of the robot relative to a 2D plane x and y and is usually transformed into a robot reference ξ_R, see Figure 4. The formulae below captures this transformation, R(θ) is the orthogonal rotation matrix that describes the orientation of the robot on a flat 2D plane.

$$\xi_R = R(\theta)\xi_G$$

$$\begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

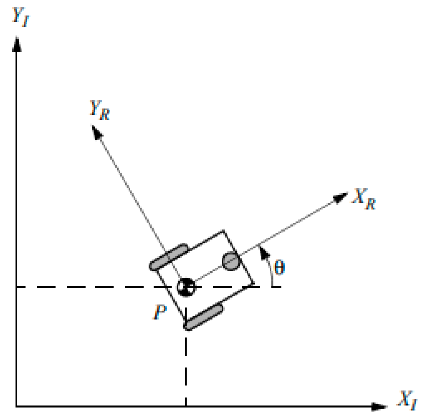

Figure 4: Robot represented on 2D plane

Perception covers measurement taken by the robot's sensors to ascertain state variables like position within the environment, proximity to obstacles within the environment, tracking its goals and mapping out its environment. Klančar et al. (2017). Sensor types like sonar, laser and vision are embedded together within robots to capture the environment and analyse it with techniques to uncover and map the environment.Types of sensors measurement relates to how the sensing mechanism works. One simple sensor that could be found on a robot is a wheel odometry. It works by counting the number of revolutions of a wheel with the ground.Aqel et al. (2016) This is translated into the linear displacement to estimate position. A wheel odometer with an Encoder can measure the wheel rotation. Another robot sensor is the global positioning system(GPS). It is a satellite-based system for navigation that enables any user determine its location anywhere on the surface of a planet body. As a result, GPS could be ineffective in indoor situations. Sonar sensors use sound or acoustic energy to find obstacles and calculate distance to those obstacles. A laser sensor uses a remote sensing technology to measure distance. It does this by analyzing the reflected light from the laser.

Odometry deals with an estimation of a robots location based on some speed measurements. It can be likened to riding a bicycle blind-folded, you can estimate how far you've ridden but you would be far off. This concept is sometimes referred to as 'dead reckoning' and robots which use odometry are susceptible to this issue. The wheel odometer can be used to estimate the position of a robot. It suffers from position drift and inaccuracy due to wheel slippages. It however shows a good short term accuracy Adachi et al. (1999)

This project will use particle filters to estimate the likelihood of a set of positions where the robot thinks it might be. The robot can draw conclusions based on how far or near a particle is from a given landmark and use this information to calculate a likelihood of its own position, Figure 5. This works well when there is a landmark, but what happens when there is no landmark in sight. The particles can be anywhere and there is less certainty about the robots location. In this research, we constrain the particle movements according to the wheels on the robot such that the particles motion will be limited to a smaller set of directions and orientations, based on the kinematic constraints. This will reduce the uncertainty in the robots estimate of its position until it finds another landmark.

Figure 5 shows the predict/update cycle for robot localization. In the first frame, we see the robot takes a step and makes a request for sensor measurement. Its sensors tell it that it is close to a column, but there are three columns in the environment and the robot can not tell which column it is near. The robot assigns equal prediction probability to the positions near the column as seen by the 'mole-hill' before each column. It takes another step, and uses the sensors measurement again to predict where it is. This time, the sensors can tell that the robot is between two pillars. The robot thus assigns more probability to the two columns. It multiples its former belief state with the current belief state to increase the likely belief state of where it is. The farthest column from the robot does not get enough possibility and will die out subsequently. By multiplying the belief updates, there is a stronger likelihood as seen in the giant 'mole-hill' in the last frame. This is how the robot is able to estimate its location within an environment.

It is possible to represent these beliefs as a set of particles where the likelihood of the particle belonging to the set of Xt is proportional to the bel(xt). Each particle represents a hypothesis of that state and is denoted as (Xt(i), Wt(i)), where Xt is the sample hypothesis and Wt is the sample weight. By this, regions with higher probability like mentioned in the previous paragraph will contain more particles. Recall, that regions with low probability will eventually die out. As a result, there is need to generate more particles so we have M set of particles at all times. Particle filtering uses importance sampling, a situation where we can only take samples from a proposal distribution to regenerate the particles. The Monte Carol Localisation algorithm with resampling can be adapted with a sample motion model and a measurement model.

2.2 Research on probability robot navigation with machine learning

In carrying out this research, the author takes a look at research using kinematics, particle filters, and classification predictor models to achieve robot localization.

Something to note about the use of odometry is that the error in each sequential measurement starts to compound. As a result, the robot experiences motion drift, where it eventually loses track of where it is. Its predicted location gradually becomes farther apart from its actual position. There are different factors that can lead to this like uneven ground, obstacles, wheel slippage. In Botta et al. (2021), the authors come up with an estimator for the kinematic behavior of a wheel mobile robot that is subject to a large lateral slip. Their research mentions that wheel slippage is quite hard to be measured directly and has to be estimated in most cases. Due to the shortcomings in odometry, the authors in Jung & Chung (2011) propose an approximation error of the conventional calibration equations by using the robots final orientation error of a test track. In this research however, the author employs the simple use of odometry inputs to propagate the robot and constrain the particle filters used in determining the robots position. By constraining, the probable space that the particles can move into is limited based on the

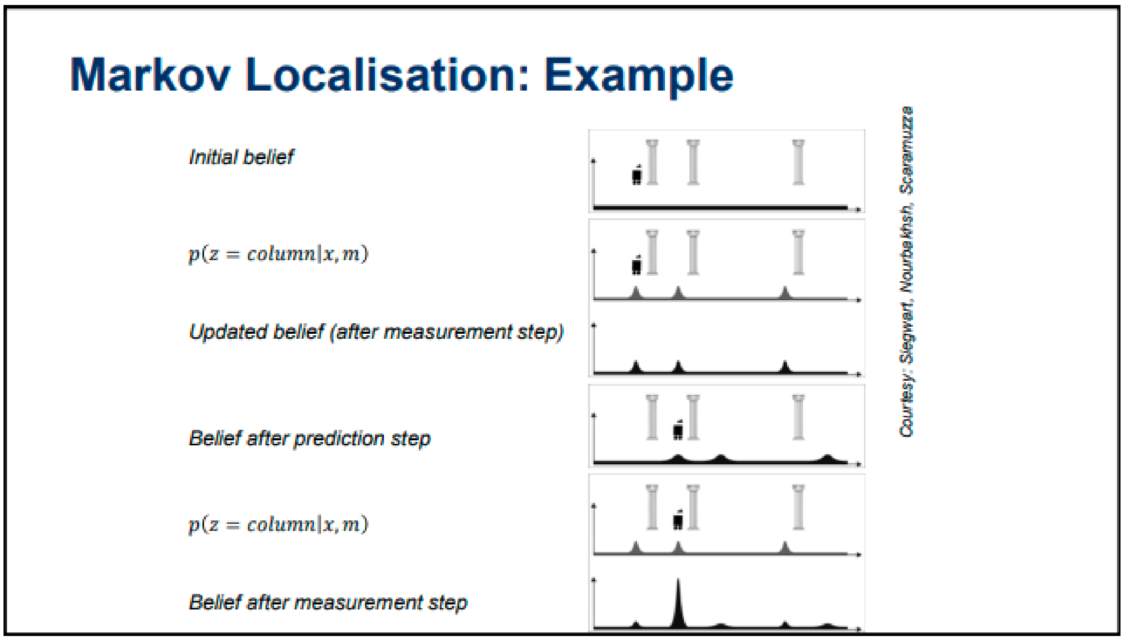
Markov Localisation: Example
Initial belief
$p(z = column|x, m)$
Updated belief (after measurement step)
Belief after prediction step
$p(z = column|x, m)$
Belief after measurement step
Courtesy: Siegwart, Nourbakhsh, Scaramuzza

Figure 5: Process of robot updating its belief of its position

kinematic constraints.

In Wang et al. (2018), the authors strategise with a model predictive control (MPC) algorithm, to build a trajectory tracker for an omni-directional wheeled motor robot(WMR) with three mecanum wheels. To develop the MPC, they establish the kinematic models of the wheels and use a target-tracking motion controller with input state and output state constraints. In comparison, we will employ a particle filter with low variance resampling to estimate the likelihood of the motion of the robot to predict where the robot might be and how it can move within the space.

In Hidalgo Carrió & Cordes (2012), the authors show how the kinematic model of a robot can be calculated to accommodate forward movement, position tracking and wheel slippage. A combination of both wheel and active systems gives the hybrid wheeled-leg solution. In Muir & Neuman (1987), a study of kinematic equations for different wheeled motor robot, we get an understanding into how to formulate motion models. The study formulates kinematic modelling for 6 wheel types as well as makes comparison between manipulator robots and wheeled robots.

In Fox (2001), the authors present a statistic approach to adapt the samples created for particle filtering, as near real time as possible. By adapting the size of the sample size, this research work is able to outperform particle filters with fixed size of sample sizes. In Burgard et al. (1998), when estimating the position of a robot, two methods are conceptually in use. One method deals with keeping track of the robot position itself while the other method covers global position estimation of the robot. Burgard et al. (1998) combines the advantage of both methods to create the Dynamic Markov Localization method. In contrast, our approach will use already trained model weights for selected number of steps. The appropriate weight will be used by the robots when it reaches the required number of steps to make informed prediction of its wheel motion.

Thrun et al. (2001) extends the traditional Monte Carlo Localization (MCL) by use of a learned KD-tree that allows for fast sampling. There is a globalization problem called the 'kidnapped robot problem' Muir & Neuman (1987), which is when a robot that has properly localized itself is taken to a a new location. The MCL extension is called a Mixture MCL, a particle filter adaptation that combines a sampler with its dual. In contrast, Zhang et al. (2020) uses a differential resampling process called Gradient Propagation Particle Filter Network. The process is used to train the motion and measurement model and the likelihood degree of the output is analysed. In this research, the low variance sampling method used, combined with a wheel predictor will help a robot recover from the 'kidnapped robot problem'.

In Hao et al. (2015), an adaptive particle filter method that uses two different resampling operations to enable fast and reliable computation is proposed. Particle filters have the drawback of high computational costs. By using two resampling techniques, the method is able to duplicate large weighted particles while reserving the remaining particles and decrease the number of particles through a particle merge. da Silva et al. (2018) explore the use of SVM, MLP and Bayes Model classifiers in robot navigation and localization. The approach uses a reject option for attributes generated from a structural occurence matrix via omni-directional images. For this research, the use of low variance sampling allows for lower computational costs. Random Forest, Decision Tree and Logistics Regression are used in this research.

Neural networks are used in Eder et al. (2021) to classify whether a robot is well localized or delocalized. Delocalization happens when a robot no longer knows its location within the environment. The research classifies patterns and features from a particle set using recurrent and non-recurrent neural networks. A convolutional neural network(CNN) model is integrated with a topological map in Foroughi et al. (2021) to enable a robot localize and navigate autonomously. The CNN is used to detect possible obstacles through vision-based localization which uses images captured from a camera available to the robot.

PF-net is a recurrent neural network proposed by Karkus et al. (2018). The research takes advantage of neural networks being able to represent complex models in large spaces to create a system model that can be optimized for an algorithm, the particle filter algorithm in this case.

Youssef et al. (2021) propose a low cost alternative for encoders attached to wheels in order to acquire the on-track speed of the wheels. Accelerometer-based Wheel Odometer for Kinematics determination(AWOK), the system developed in the research uses a single axis accelerometer mounted at the center of the wheel, in a radial manner to provide distance as well as velocity.

In looking at evaluation metrics of state of the art simultaneous localization and mapping (SLAM) methods, a review of Servières et al. (2021) is done where several SLAM methods are compared and tested on dataset benchmarks to evaluate their performance. The authors review existing SLAM methods and state that comparing them is not an easy task due to the each method targeting different objectives. SLAM methods can be seen as basic SLAM and visual SLAM (vSLAM). This research classifies vSLAM algorithms based on differences like being a filter or optimization based method, or what kind of features the model has. Evaluation metrics used for the SLAM methods includes Relative Positioning Error (RPE) and the Absolute Positioning Error (APE). The APE is said to be the calculated euclidean distance between the estimated position of the robot and the true position, while RPE is the calculated euclidean distance between consecutive position estimates.

2.3 Conclusion

The review of the mentioned works tells us that the use of kinematic modelling can achieve high success during robot position prediction. Knowing how motion drift can occur through odometry limitations, we can choose to utilize other methods to reduce the estimate space of a robots position.

3 Methodology

3.1 Research Architecture

This research is carried out in two phases as captured in Figure 6. Phase 1 begins with initializing a robot that has a sensor and N number of particle filters. Some odometry is added to the robot to move it in a forward, backward and rotational motion. It can be observed how the particles converge on the robots position when the robot senses a landmark and how the particles disperse when there is no sensed landmark. Figure 7 is a top down rendering of a virtual robot moving on a flat surface. Image *(a)* in Figure 7 shows the robot environment or virtual world, the robot (red circle and light blue lines) perceives its position using particles (purple circle and blue lines) and the landmarks (white circles) in the environment. The virtual robot has wheels, a sensor, and some control system to move it, stop it, turn it and analyse sensor readings. The control system manages the particles used in estimating its position as well. The sensor measurements predict the pose (or position) of the robot relative to landmarks. The environment is hosted on a Windows 10 machine and makes use of the following tools, Python 3, Numpy, Pandas, Matplotlib, OpenCV and CSV. Image *(b)* in Figure 7 shows the particle moving towards the robot, and in image c, convergence of the particles on the robot can be seen. Note a landmark *(with a red circle within)* has been sensed by the robot and the likelihood of the particle states has been updated. In Image *(d)* of Figure 7, it can be seen that the particles diverge away from the robot, note that there is no sensed landmark. The virtual robot is made to take 100 steps and its position recorded as well as calculating the average of the difference between the robot and each particle. The number of steps and average difference is plotted vs the number of visible landmarks. The output shows how the particles converge around the robot space. This step is carried out to compare the average distance between the robot and its particles before wheel constraints are applied on the particles. The average distance is expected to reduce when a wheeled constraint is applied on the particles

Phase 2 begins with synthesizing robot poses for three wheel scenarios. A single fixed wheel, a single omni-wheel and a two-fixed wheel. The synthesized robot poses for the wheel types are combined in a dataframe and three machine learning classifiers are applied, (logistics regression, decision tree and random forest) to determine a winning model. The winning model is applied to the robot, used to predict the wheel on it and the robot applies the necessary motion constraint for the wheel type to the particles.

A collection of a few successive poses *(or steps)* predicts the types of wheels on the robot. The prediction is used to constrain the particles such that the robot has an estimate of its position as it moves past landmarks in the environment. Another use of the prediction in managing the particles is to eliminate the particles in positions that do not fit the possible range based on the wheel type. A robot that has a fixed wheel can only move forwards or backwards. Therefore particle orientations are eliminated by not

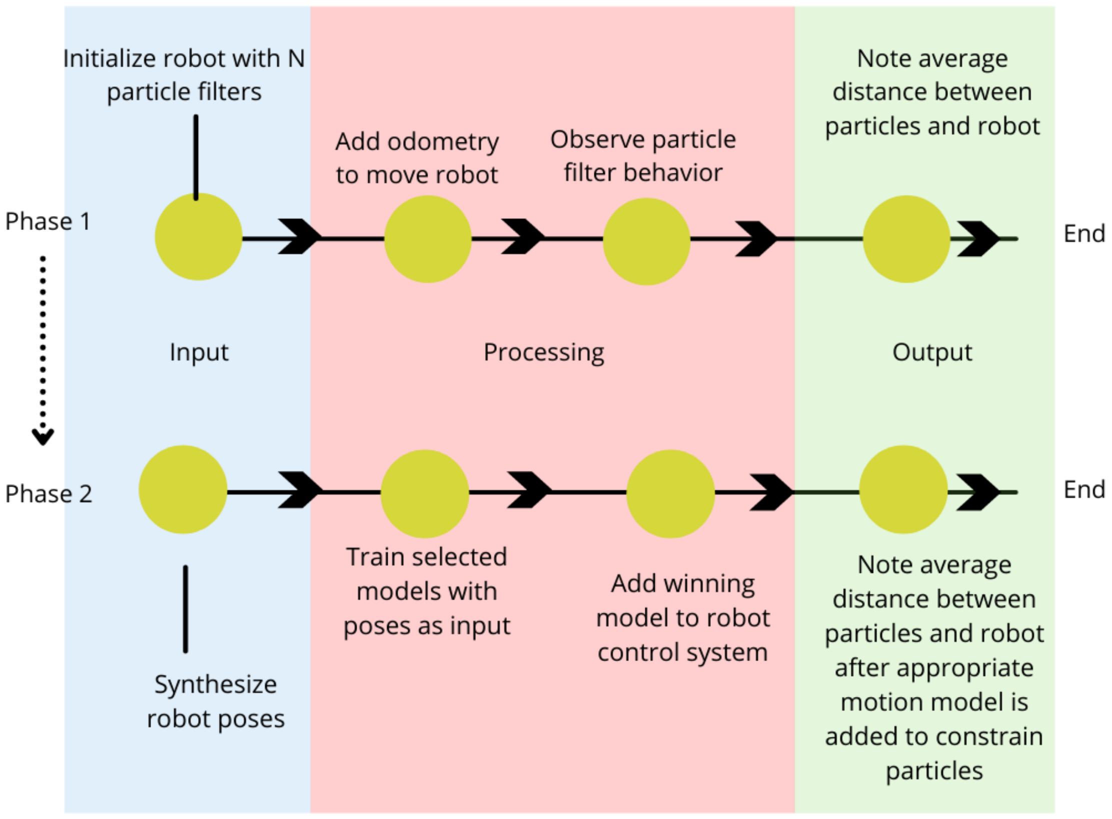
Initialize robot with N particle filters
Add odometry to move robot
Observe particle filter behavior
Note average distance between particles and robot
Phase 1
End
Input
Processing
Output
Phase 2
End
Train selected models with poses as input
Add winning model to robot control system
Note average distance between particles and robot after appropriate motion model is added to constrain particles
Synthesize robot poses

Figure 6: Research overview

considering them in the first place. Initially, particle updates were given by selecting from a normally distributed set of motion updates in all directions. These motion updates are limited such that the forward and backwards directions have higher weighting than the left-to-right directions while considering the single fixed wheel configuration. A similar approach is used for the two fixed wheels. However omni-directional wheel can move in any direction, so this keeps using the uniform distribution over all motion updates. For this research, focus is on using the prediction to constrain the particles. Further works could include particle resampling based on possible positions from the wheel type.

3.2 Data Mining Methodology

The knowledge discovery in databases (KDD) methodology is used in this research work. The methodology guides extraction of useful and non-trivial information from large databases. The methodology has eight stages. *Problem specification, Data selection, Data cleansing, Data pre-processing, Data mining, Result evaluation, Interpretation of results, Exploitation of results.* A slight modification to the generic KDD methodology is done by introducing data generation stage to replace the data selection stage. There is opportunity to expand the work with real world sensors but for now the research is interested in discussing the probability problem rather than setting it up in the real world.

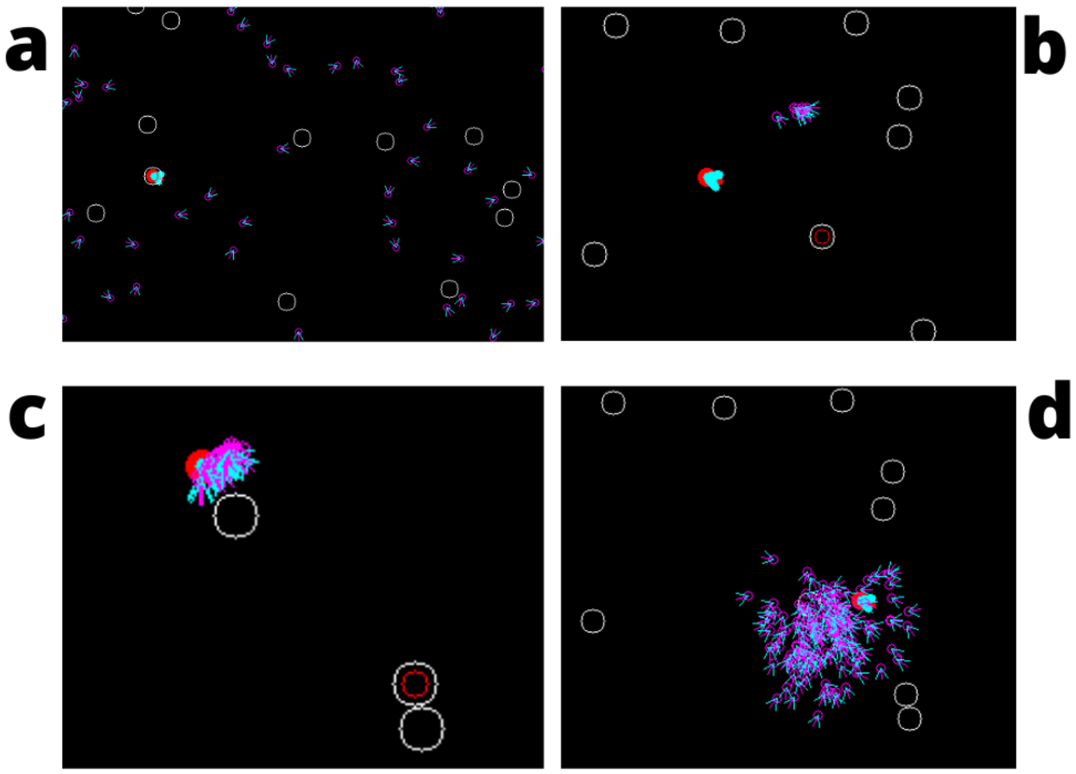

Figure 7: Robot world

3.3 Data Selection/Generation

In this research, the motion measurements that is used as input data is generated. The generated measurements include velocity, angles, wheel radius and x,y co-ordinates. From this, pre-processing and correlation confirmation between distance and velocity as well as feature extraction is carried out.

3.4 Data Preparation and Pre-processing

3.4.1 Data visualization

The author visualizes the class balance in Figure 9 which shows the same amount of samples for each class. Since there are equal number of samples in each class, the F1-score evaluation will not be as important during evaluation of models. Log loss and accuracy are selected as the evaluation metrics. The labels in the figure stands as follows, *fw - fixed wheel, ow - omni wheel, tf - two fixed wheels.*

The environment is hosted on a Windows 10 machine and makes use of the following tools, Python 3, Numpy, Pandas, Matplotlib

3.5 Implementation

In a real world scenario, the robot will receive control inputs that tell it what distance to move, and in what direction to move. This data is synthesized with some noise to mirror the real world scenario. The noise is a normally distributed set of values that are added

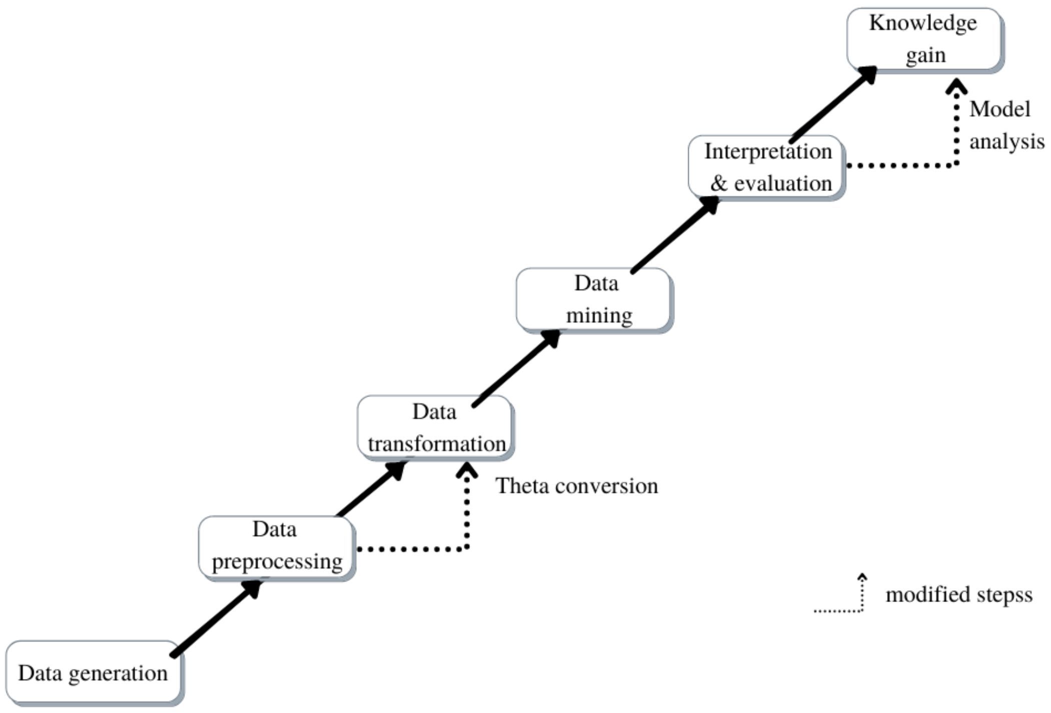

Figure 8: Modified KDD methodology

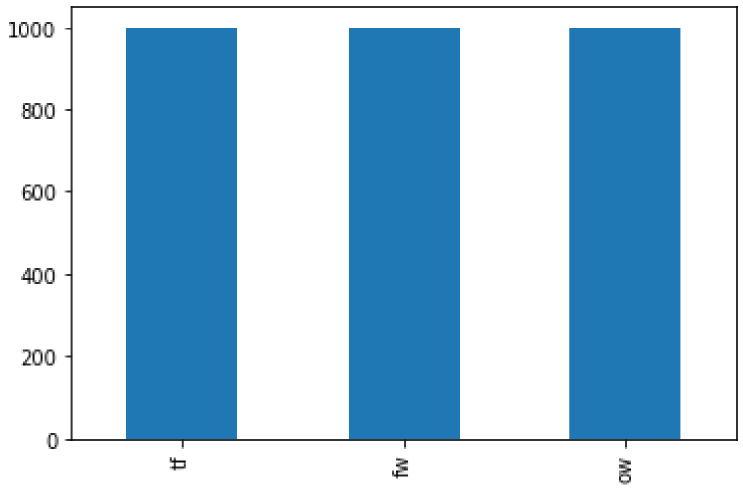

Figure 9: Wheel type class balance

to the motion update such that correct motion update becomes the mean position and the variance for the normal distribution is of some specified set size. The data is then used to generate a continuous representation that is used to gather samples from the environment. This data is collected and fed to the predictor model at a chosen interval or number of steps for which we want to derive trained weights for. In this research, the robot can request predictions after taking *5, 10, 20, 50, 100* steps. A new set of data is generated with a different random seed after the hyper-parameters are tuned to ensure the model has not been over-fitted.

The input data to the chosen model is a list containing the x-coordinate, y-coordinate and theta value of a robot in an environment represented on a 2D map as shown in Figure 11. Depending on the wheel type, the robot can either move forward/backward in a straight line, or forward/backward/sideways in a straight line. The research generates *(x,y,θ)* values for steps up to *5, 10, 20, 50, 100*. The generated data is stored in comma-separated-value files with the number of steps and type of wheel as filenames. The following wheel types are synthesized.

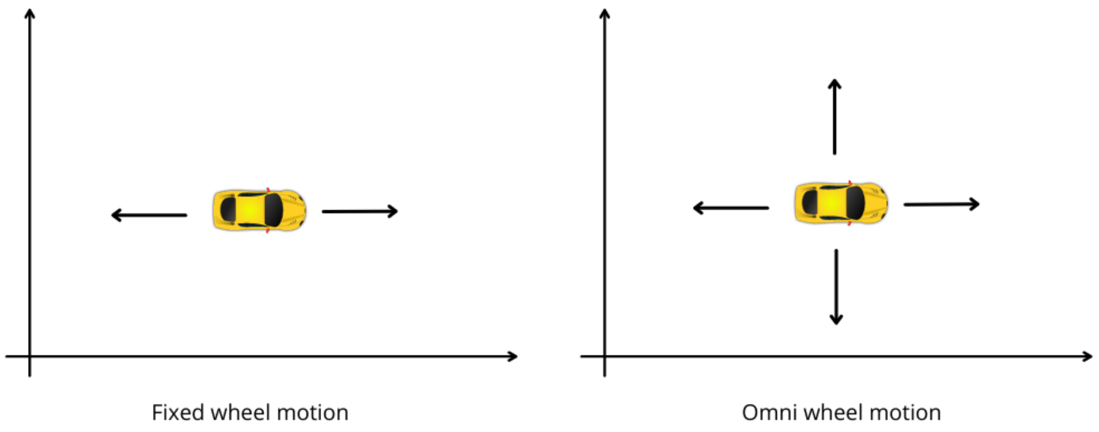

Figure 10: Wheel motion directions

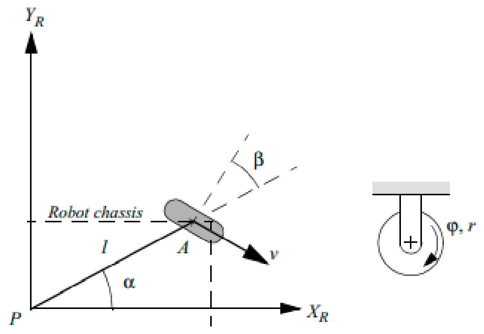

Figure 11: Fixed wheel showing motion constraint

Fixed wheel: Since this wheel can only move forward or backward, as seen on the left in Figure 10, it resembles the slope of a straight line which can be represented with $y = mx + x$. By initialising a random value to x within a sizable world-space (750 x 750 pixels), an x-coordinate can be recorded. The same is done for the m, c and theta variables. From this, the y-coordinate can be gotten by simply substituting into the $y = mx + c$ and get the value of θ through the *arctan* of the slope. The result is a point that can only move forward or backward along its x-axis. A perfect representation of a fixed wheel.

Omni wheel: In this instance, x, y, and θ get values at random within the range of numbers that support x,y ¡ worldsize. Since they can move in any direction, as seen in the right diagram in Figure 10, it is easier to generate and synthesize data to represent particles within the space.

Two fixed wheel: Captured on the left in Figure 10, here, the radius of the fixed wheels, the length between them and angular velocity of the wheels is taken into account using the formulae to get a robot position in the robot reference. We can convert this to the global reference using the transformation matrix mentioned in section 2.

3.6 Data Mining

In Figure 11, there are three wheel types in the generated data. There is no class imbalance among the wheel types. A classification task will be used to determine the wheel type when the robot is in motion. The prediction results need to be gotten as quickly as possible since the model is moving the environment and needs to correctly identify its location within the space. Another factor to consider for the model to be used is complexity. A simple model and easy to interpret model can perform the classification objective. Using the criteria of simple(non-complex) and timely result, with a good chance of high accuracy and least log loss, the following machine learning models are considered for use in this research.

To carry out data mining, four classifier algorithms are tested and three are selected for hyper-tuning based on having the lowest log loss. Table 1 shows this classifier analysis done for 100 robot steps. KNN has a high log loss compared to the others so it is excluded from hyper-parameter tuning.

Classifier	**Accuracy(%)**	**Log Loss**	**Kappa Score(%)**	**F1-Score(%)**
LogisticRegression	43.0	1.08	14.74	42.87
KNeighborsClassifier	60.56	5.3	41.26	62.06
DecisionTreeClassifier	97.78	0.77	96.66	97.77
RandomForestClassifier	99.56	0.03	99.33	99.56

Table 1: Accuracy/Log Loss Table of classifiers

4 Results

In the previous section, a description of how input data is synthesized for the prediction model to mirror the motion constraints for the wheels chosen in this research is discussed. Some classification algorithms are evaluated to see their performance on the data and KNeighborsClassifier is eliminated due to its high log loss. After tuning the hyper-parameter, a new set of data is generated and used to test the predictor model once again. The results can be found in the Figure 18, 19 and 20 in appendix.

4.1 Hyperparameter Optimization

A grid-search is used to hyper-tune the logistic regression and randomized search for Random Forest and Decision Tree. The same 100 robot steps dataset used in the classifier analysis is used for the optimization. Randomized search instead of grid-search is used for Random Forest and Decision Tree due to how long it will take to test all the parameters in a grid-search. Randomized search selects parameters at random. finds the best parameter values and delivers results quicker.

4.1.1 Logistic Regression

On initial run of logistic regression for 100 steps, an accuracy of 43% with a log loss of 1.08 is observed. On further hyper-parameter tuning, an accuracy of 47.89% with improved log loss of 0.52 is observed, when the following parameters from the best accuracy is used. *c = 0.01, penalty = L2, solver = newton-cg.* It can be seen from the plots in Figure 12 how the accuracy increases, and how the log loss reduces, as the number of steps increases. To ensure model over-fitting does not happen, new datasets are generated and tested on the predictor model. The predictor has an accuracy of 51.03% and log loss of 0.49. The model has not been over-fitted.

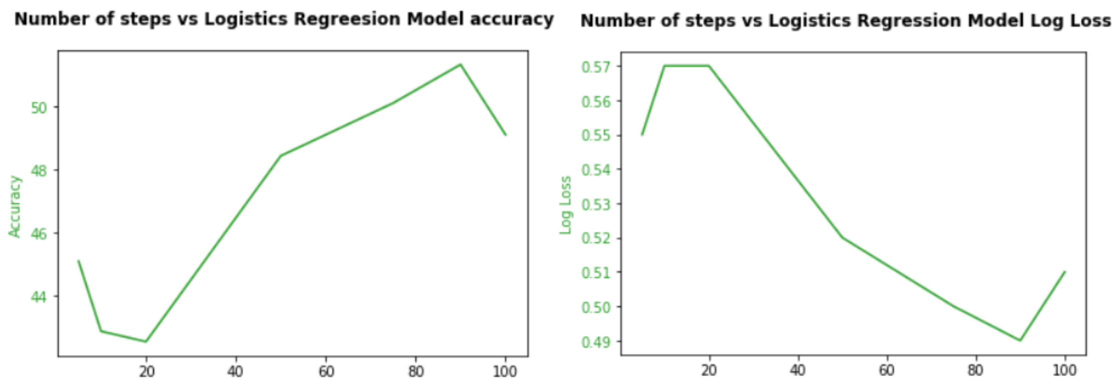

Figure 12: Number of steps vs hyper tuned Logistic Regression accuracy and log loss

4.1.2 Decision Tree

When decision tree is applied to the dataset, an initial value of 98.22% for accuracy and 0.60 for the log loss is observed. After tuning the hyper-parameters using randomized search, 96.67% for accuracy and 0.03 for log loss with tuned parameter values of *min_samples_split= 3, min_samples_leaf= 3, max_leaf_nodes=80, max_depth=5, criterion= 'gini'* is observed. The ideal model will have the lowest log loss, hence the tuned parameters model are selected over the default model.

Figure 13 show how accuracy increases and log loss reduces with the increase in the number of steps. Like before, to test for over-fitting, generated data is run on the model, where model performance has an accuracy of 97.07% and log loss of 0.03.

4.1.3 Random Forest

The initial accuracy gotten from random forest model is 99.44% with the log loss as 0.03. On applying the hyper-tuned parameters, the accuracy remains at 99.44% but the log loss improves by reducing to 0.005. Figure 14 show how the number of steps relates with the accuracy and log loss. Generated data is run on the model and observed result shows the accuracy on the new test set is 98.9% with a log loss of 0.01. It can be concluded the model has not been over-fitted.

From the above observations, the random forest is the winning model and the research generates weights for *5, 10, 20, 50, 100* steps initially. Later more weights are generated for *1000, 2000, 5000 and 10000 steps.* When the winning model is added to the robot control and particle dispersion for a fixed wheel on the robot is carried out, it is observed

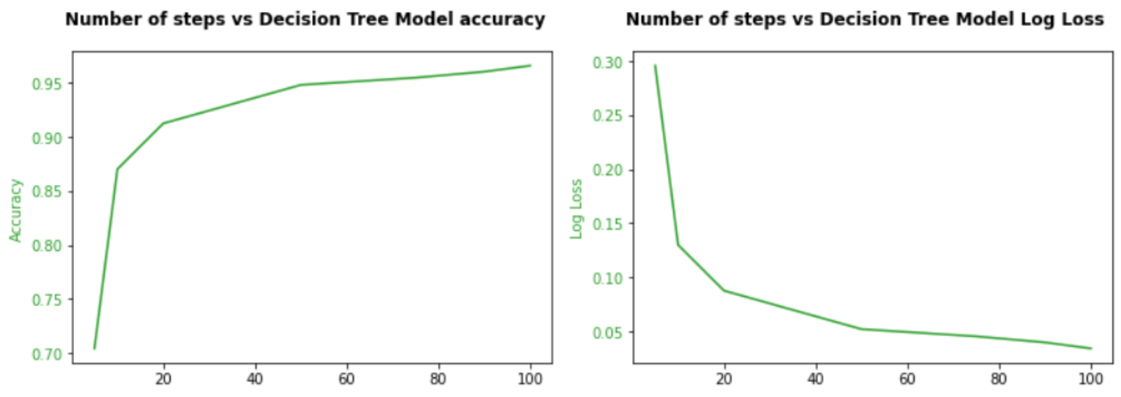

Figure 13: Number of steps vs hyper tuned Decision Tree accuracy and log loss

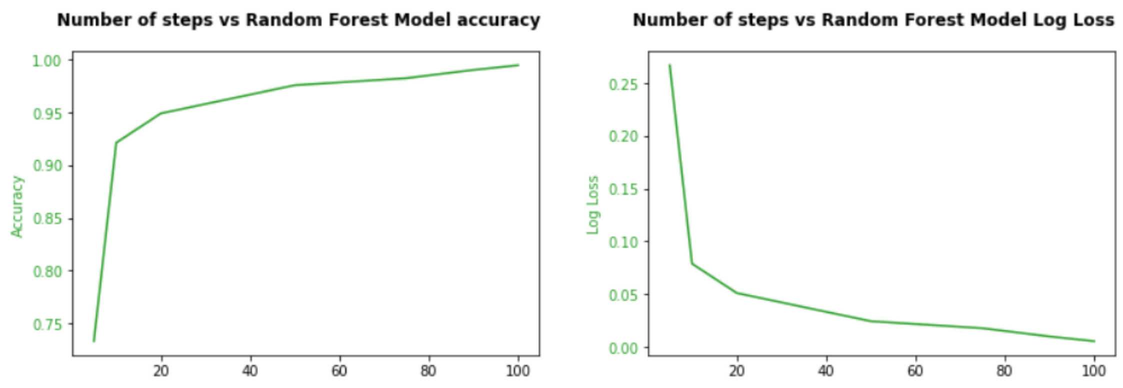

Figure 14: Number of steps vs hyper tuned Random Forest accuracy and log loss

that the model's prediction about the type of wheel it has is correct. The research has been able to successfully detect the type of wheel on the robot. In the next section, the use of this prediction by the robot is evaluated

5 Evaluation

The research objective is two fold. To predict as accurately as possible the motion model of a robot and to solve the dead reckoning issue. From the classifier analysis, random forest is selected as the prediction model. The evaluation used to select random forest is shown in Table 1. It has the least log loss.

To evaluate the success of the proposal, a look into how the robot performs before applying any motion model and after it implements a motion model based on the prediction from the classifier is considered. The expectation is that before applying a specific motion model, the particles will move in a random or omni-wheel direction when there is no landmark in sight, however on employing a specific motion model, like a fixed wheel for example, the particles will have to move in a straight line *(with some noise)* when there is no landmark and this gives a good estimate of the robot position at all times.

Figure 15 shows the graph of average distance between the robot and its particles before (red) and after (blue) applying a fixed-wheel motion model. The distance is calculated using the euclidean distance known as L2-norm while the angular difference is derived using cosine similarity. The diagram looks similar because the change reduction is 5% and is not easily visible. How then is the change evaluated?

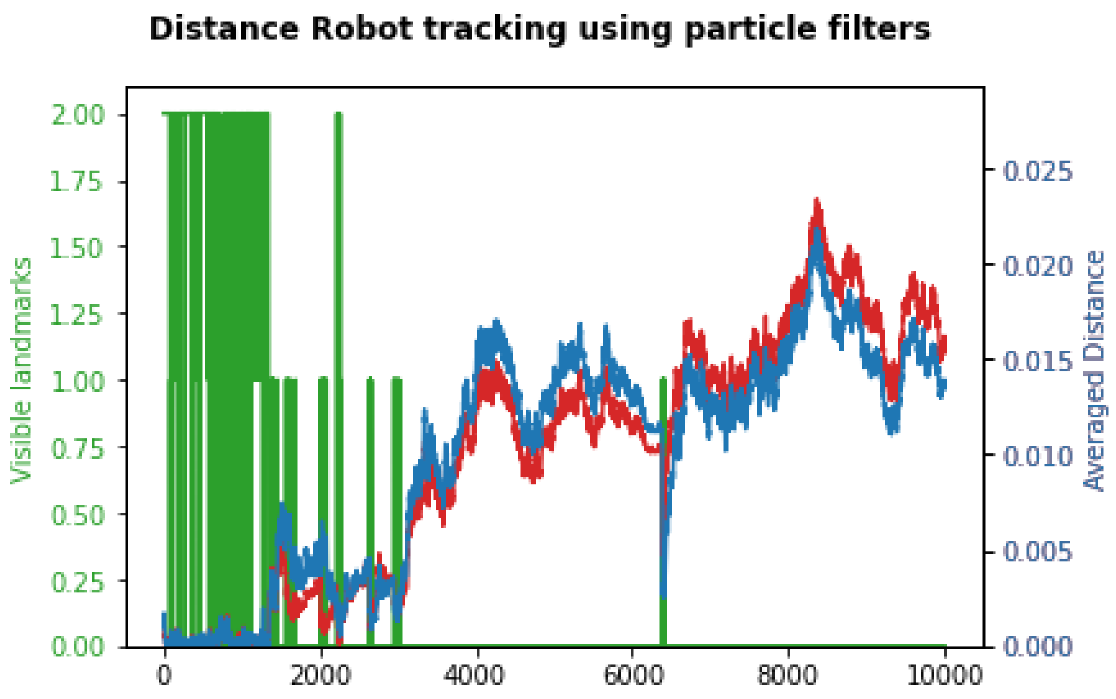

Figure 15: Particle tracker before(red) and after(blue) applying motion model

The fixed wheel motion constraint is used for this example. The average distance recorded between the robot and its particles is 378.71 before any motion constraint is added. Some empirical fine-tuning was carried out to find the optimal constraint for the x, y, and θ values. To simulate real world properly, random numbers are generated from a uniform distribution and added to the odometer input for every step taken by the robot. The motion model constraint is added to the respective axis by the noise we add. This is done based on the wheel constraint, the noise is weighted and used to promote forward motion. When we multiply the uniform random number we get for noise with the following constraint values (x_w: 0.25, y_w:0, θ_w: 1.5), a lower average of 358.95 is observed. With this, a 5% reduction in the average distance between the particles and the robot is observed. The constraint limits the motion of the particles while still allowing some random movement for the particles to resample correctly when they need to. This points to some success since the particles are constrained by the motion model for a fixed wheel. On further research, the following table is created.

Table 2 shows the results gotten between the average distance between the particles and the known robot position for a specific number of robot steps with selected x,y and theta constraints. The column *'Distance before'* refers to the distance between the particles and the robot before the motion constraint is added and the column *'Distance after'* refers to the distance between the particles and the robot after the motion constraint is added. Two values for the constraint on X is shown in the table. Empirical tuning of the constraint values for x,y and theta can be tested to find values for which the average distance after applying the motion constraint will be reduced. Further re-

Robot steps	**Distance before**	**Distance after**	**X**	**Y**	**Theta**
100	378.71	358.95	0.25	0	1.5
100	16.54	14.47	1.25	0	1
200	57.35	77.41	1.25	0	1
500	254.45	273.81	1.25	0	1
1000	522	472.36	1.25	0	1
2000	582.11	525.1	1.25	0	1
5000	1563.84	1421.14	1.25	0	1

Table 2: Model performance based on number of robot steps

search can be carried out to find the optimal number. In this research, it is discovered that with more robot steps taken, the motion constraint added on the noise is able to reduce the average distance between the particles and the robot. In Figure 21 seen in the appendix, we see graphs of robot steps. The x-axis shows the number of robot steps (which is the input robot step multiplied by 10), the left y-axis has the number of landmarks encountered while the right y-axis has the average distance for the particles and the robot for each unit robot step taken. It is noticed that when no landmark is seen by the robot, the particles diverge around the environment. However, according to the average distance calculated and shown in Table 2, the motion constraints added was able to constrain the movement of the particles and reduce the distance between the particles and the robot.

6 Ethics

The data used in this research is generated within the simulated environments. This research does not violate morality or known laws.

7 Conclusion

In this research, a way for a robot using particle filters to constrain the motion of the particles by applying a wheel motion constraint to the particles is proposed. This allows the robot's estimate error when there is no landmark to reduce because the particles can only move in a certain manner based on the robots movement. It is observed that the average distance between the robot and its particles before applying a motion model and the average distance after applying a motion model can be reduced. We used a fixed motion model and we see that the average distance reduces when a motion model is applied.

For future works, more can be done with the wheel prediction gotten from the random forest classifier. Specifically, particles that do not satisfy that wheel motion can be eliminated immediately if resample particles satisfy the range of values that is possible for the predicted wheel motion. This will lead to faster resampling and help the robot localize faster and better in unknown environments.

8 Acknowledgement

The guidance of William Clifford has been helpful in this research. I acknowledge the support of friends and family. A very special thank you to the library and Data Analytics department of the school of computing, National College of Ireland for their support.

This research has also been possible by leveraging on the initial simulation developed from a coding assignment given by John McDonald in Maynooth University. The particle filter adjustments, model building and adaptive motion constraints was done by the author of this research.

References

Adachi, H., Koyachi, N., Arai, T., Shimiza, A. & Nogami, Y. (1999), Mechanism and control of a leg-wheel hybrid mobile robot, *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)', Vol. 3, pp. 1792–1797 vol.3.

Aqel, M. O. A., Marhaban, M. H., Saripan, M. I. & Ismail, N. B. (2016), 'Review of visual odometry: types, approaches, challenges, and applications', *SpringerPlus* **5**(1), 1897.
URL: *https://doi.org/10.1186/s40064-016-3573-7*

Botta, A., Cavallone, P., Tagliavini, L., Carbonari, L., Visconte, C. & Quaglia, G. (2021), 'An estimator for the kinematic behaviour of a mobile robot subject to large lateral slip', *Applied Sciences* **11**, 1594.

Burgard, W., Derr, A., Fox, D. & Cremers, A. (1998), Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach, *in* 'Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)', Vol. 2, pp. 730–735 vol.2.

da Silva, S. P. P., Marinho, L. B., Almeida, J. S. & Rebouças Filho, P. P. (2018), Robot localization using classification with reject option from structural co-occurrence matrix in omnidirectional images, *in* '2018 7th Brazilian Conference on Intelligent Systems (BRACIS)', pp. 456–461.

Eder, M., Reip, M. & Steinbauer-Wagner, G. (2021), 'Creating a robot localization monitor using particle filter and machine learning approaches', *Applied Intelligence* .

Foroughi, F., Chen, Z. & Wang, J. (2021), 'A cnn-based system for mobile robot navigation in indoor environments via visual localization with a small dataset', *World Electric Vehicle Journal* **12**(3).
URL: *https://www.mdpi.com/2032-6653/12/3/134*

Fox, D. (2001), 'Kld-sampling: Adaptive particle filters and mobile robot localization'.

Hao, L., Li, T., Villarrubia, G., Sun, S. & Bajo, J. (2015), 'An adaptive particle filter for indoor robot localization', *Advances in Intelligent Systems and Computing* **376**, 45–55.

Hidalgo Carrió, J. & Cordes, F. (2012), Kinematics modeling of a hybrid wheeled-leg planetary rover, *in* ‘International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-2012), September 4-6, Turin, Italy’.

Jung, C. & Chung, W. (2011), ‘Calibration of kinematic parameters for two wheel differential mobile robots by using experimental heading errors’, *International Journal of Advanced Robotic Systems* **8**(5), 68.

Karkus, P., Hsu, D. & Lee, W. S. (2018), ‘Particle filter networks: End-to-end probabilistic localization from visual observations’, *CoRR* **abs/1805.08975**.
URL: *http://arxiv.org/abs/1805.08975*

Klančar, G., Zdešar, A., Blažič, S. & Škrjanc, I. (2017), Chapter 5 - sensors used in mobile systems, *in* G. Klančar, A. Zdešar, S. Blažič & I. Škrjanc, eds, ‘Wheeled Mobile Robotics’, Butterworth-Heinemann, pp. 207–288.
URL: *https://www.sciencedirect.com/science/article/pii/B9780128042045000056*

Muir, P. F. & Neuman, C. P. (1987), ‘Kinematic modeling of wheeled mobile robots’, *Journal of Robotic Systems* **4**(2), 281–340.

Roberts, E. (1999), ‘Robotics: A brief history’.
URL: *https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html*

Roland, S. & Illah, R. N. (2004), *Introduction to Autonomous Mobile Robots*, The MIT Press, London.

Rubio, F., Valero, F. & Llopis-Albert, C. (2019), ‘A review of mobile robots: Concepts, methods, theoretical framework, and applications’, *International Journal of Advanced Robotic Systems* **16**(2), 1729881419839596.
URL: *https://doi.org/10.1177/1729881419839596*

Sabzevari, R. & Scaramuzza, D. (2016), ‘Multi-body motion estimation from monocular vehicle-mounted cameras’, *IEEE Transactions on Robotics* **32**(3), 638–651.

Servières, M., Renaudin, V., Dupuis, A. & Antigny, N. (2021), ‘Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking’, *Journal of Sensors* **2021**, 2054828.
URL: *https://doi.org/10.1155/2021/2054828*

Thrun, S., Fox, D., Burgard, W. & Dellaert, F. (2001), ‘Robust monte carlo localization for mobile robots’, *Artificial Intelligence* **128**, 99–141.

Wang, C., Liu, X., Yang, X., Hu, F., Jiang, A. & Yang, C. (2018), ‘Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy’, *Applied Sciences* **8**, 231.

Youssef, A., Al-Subaie, N., El-Sheimy, N. & Elhabiby, M. (2021), ‘Accelerometer-based wheel odometer for kinematics determination’, *Sensors* **21**, 1327.

Zhang, H., Wen, J., Liu, Y., Luo, W. & Xiong, N. (2020), ‘Mobile robot localization based on gradient propagation particle filter network’, *IEEE Access* **8**, 188475–188487.

9 Appendix

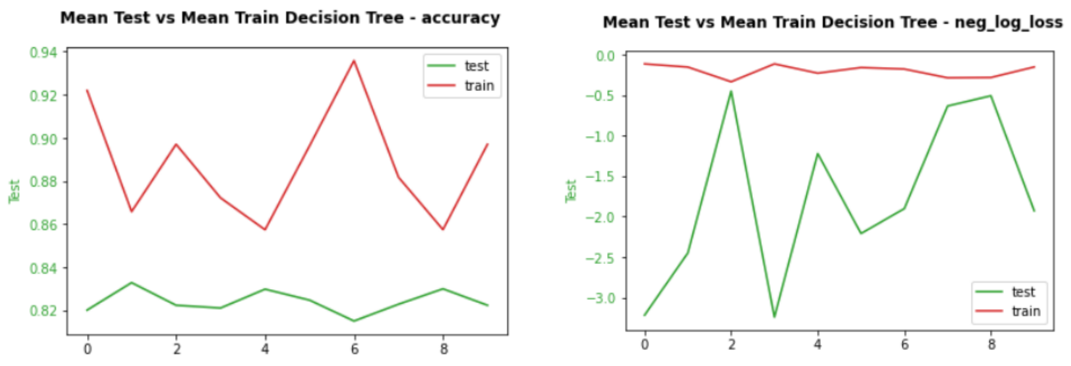

Figure 16: Hyperparameter tuning for random forest

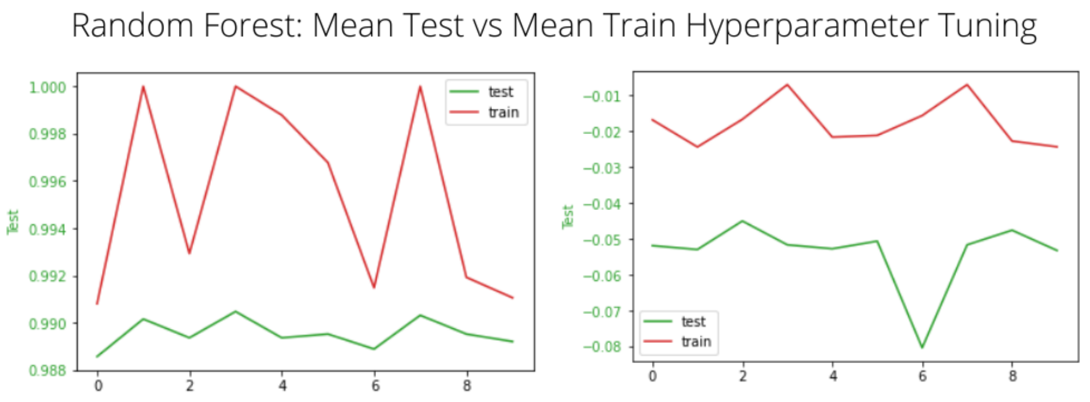

Figure 17: Hyperparameter tuning for random forest

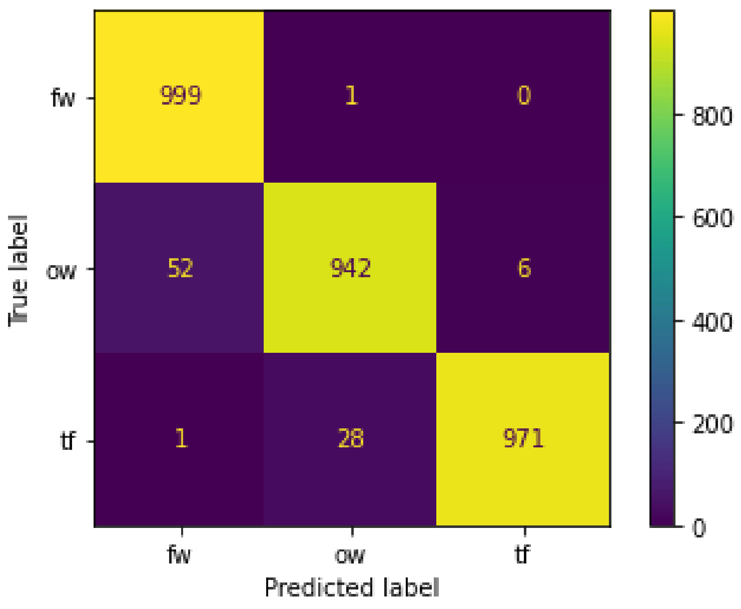

Figure 18: Test for decision tree overfitting

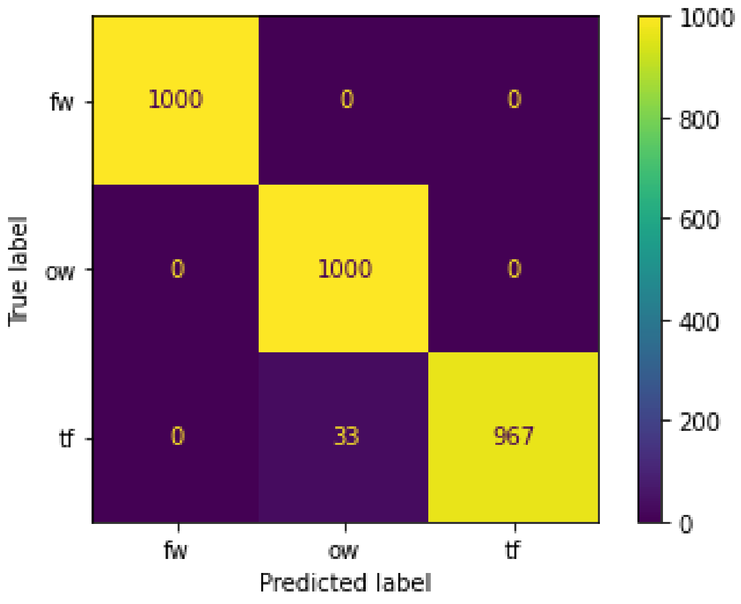

Figure 19: Test for random forest overfitting

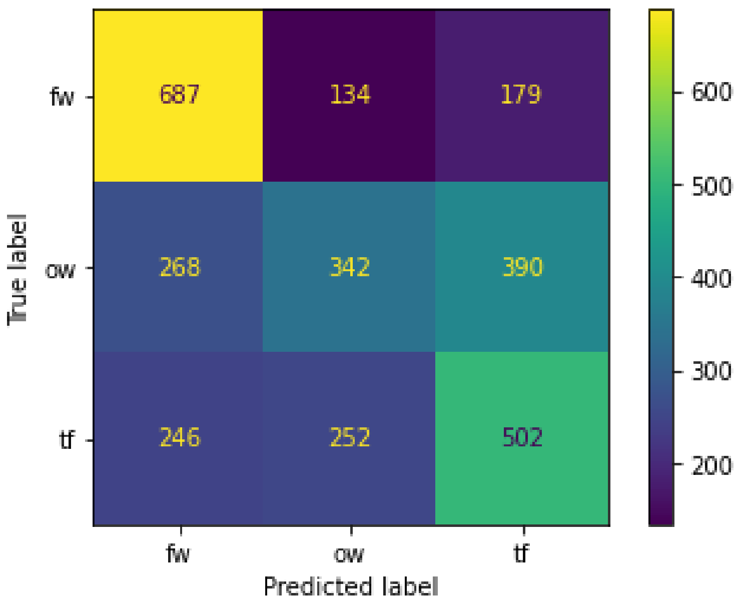

Figure 20: Test for logistic regression overfitting

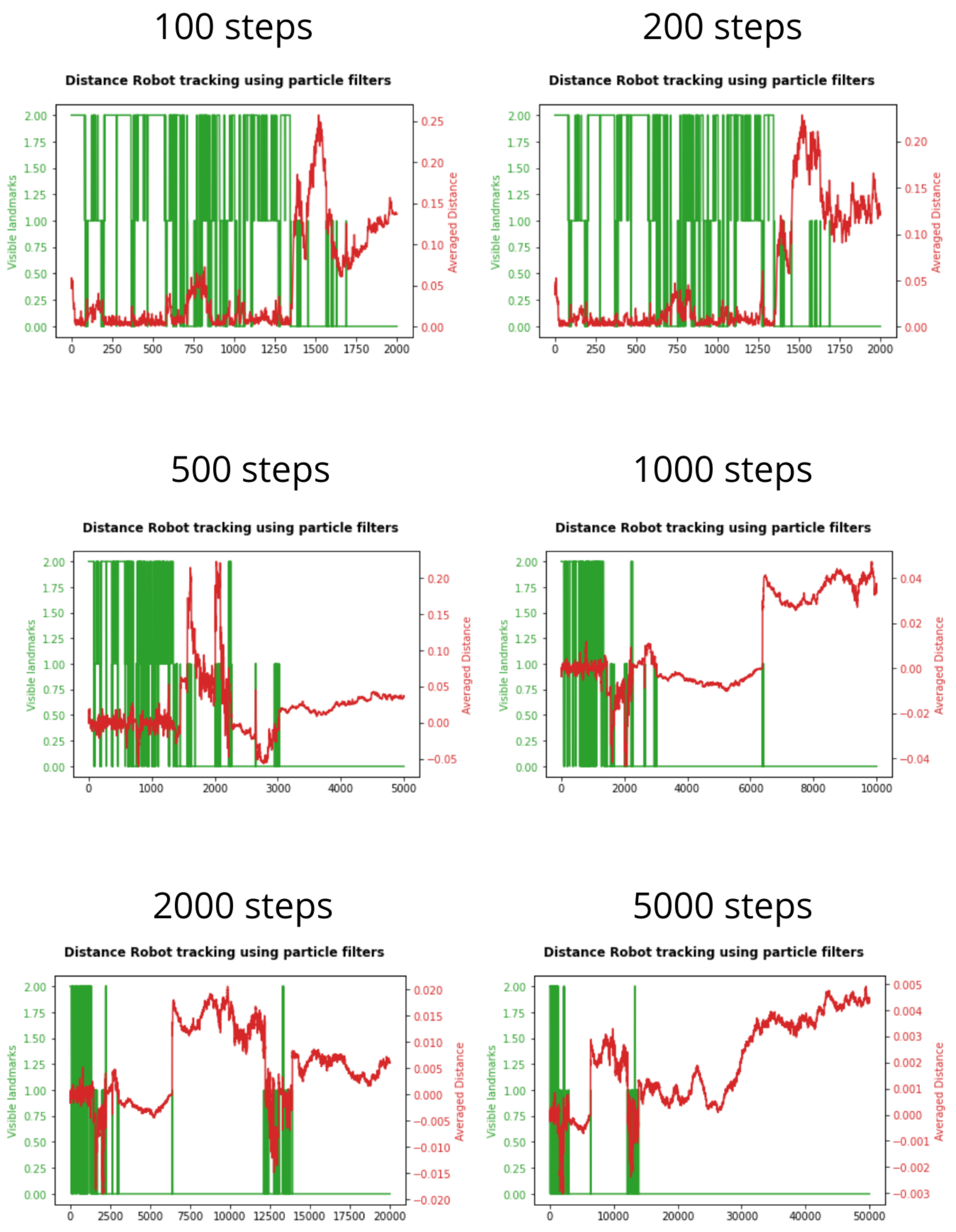

Figure 21: Robot steps vs Landmarks vs Average distance between particles and robot