Toward Navigating Complex Terrains Using A Biomimetic Whisker Sensor Array

By

Mohammed Salman

Department of Aerospace Engineering
University of Bristol

Supervisor:
Dr. Martin Pearson

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in Robotics in the Faculty of Engineering.

September 2018

Word count: 63,084

Abstract

This thesis proposes a parsimonious approach to localization, mapping and object recognition for a pseudo-mobile robot equipped with a biomimetic array of tactile whiskers to autonomously interact, explore and represent a real-world environment. Tactile whisker sensors enable the robotic platform to perceive unique environmental properties and can operate in extreme conditions that preclude the use of conventional sensors, however, such sensors are disadvantaged by their limited range and sample sparsity. To address the sparsity, the information contained in each contact should be fully exploited, whilst the limited range of the array can be addressed through appropriate movement and placement of the whiskers and the array.

An existing Simultaneous Localization and Mapping (SLAM) algorithm called RatSLAM was adopted as the basis for the inference of location and demonstrated as suitable for correcting odometry errors using whisker tactile sensing. The adoption of a closed loop contact induced whisker placement strategy, directly inspired by rat whisking behavior, improved the performance of the algorithm in further reducing odometry error. The fidelity of object shape reconstruction through the forward kinematic projection of whisker contact locations was analyzed and a number of machine learning approaches compared to assess their efficacy at discerning radial distance to contact and thus improve object shape reconstruction. A support vector regression technique was found to reliably improve estimates of radial distance to contact along the whisker shaft following natural, unconstrained whisker contacts. A framework for combining the 3D pose estimation from RatSLAM with a 6D pose estimation system suitable for object recognition is proposed with the 6D system implemented and demonstrated correctly identifying household objects through tactile whisker exploration. The adoption of whisker array placement strategies inspired by cutaneous-tactile research improved the robustness of object identification and two regional search strategies were investigated for the purpose of reducing the time taken to correctly classify objects.

Dedication and acknowledgments

Thank you Dr. Martin Pearson who has gone above an beyond in terms of supporting me throughout this undertaking.

I would also like to thank Gareth Griffith for his enthusiasm, kindness and all the hard work he has put in supporting my work.

Thank you Jason Welsby for all the work you put in with the whiskers, without you this thesis would not be possible either.

Last but not least I would like to thank my friends and family for all the love and support they have given me in life. I would also like to thank my Father for all his efforts in developing my love in technology, math and science.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .....  ..... DATE: 3/1/2020

Table of Contents

Chapter 1

Introduction



Figure 1.1: Depending on their size, environment and needs, animals such as elephants, bees and sharks all have unique senses that help them achieve their goals. Elephants have been shown to be able to sense ground vibrations that enable them to communicate with one another across large distances (Mortimer et al., 2018). Honey bees are known to sense magnetic fields surrounding flowers that communicate their level of nectar (Liang et al., 2016). Hammerhead sharks are known to sense electromagnetic fields that enable them to detect hidden prey (Kajiura & Holland, 2002).

Nature has shown us that no single sensory modality is perfect. The environmental conditions of an animal's habitat will largely determine which sensory modality is the most effective at detecting behaviorally relevant cues such as threats or rewards. Often an animal will use multiple sensory systems in parallel to gain an ecological advantage or, in many situations switch from relying on information from one sense to that of another. The study of ecology has shown us that animals adapt to their environment by developing sensory organs that exploit the nature of their surroundings for survival.

From our own lives we have experienced moments such as waking up in the middle of the night to a pitch black room forcing us to blindly find our way to the light switch

using our hands. We adapt to the situation by exploiting our sense of touch and balance, searching with our hands outstretched and sweeping them along our path, until we reach a wall. Our hand moves quickly towards where we expect the light switch, only for it to hit a corner wall. We track back, making large sweeps along the height of the wall, eventually feeling the raised edges of the switch panel and finally flicking the switch. Now our ability to navigate becomes much easier since our sense of vision grants us a longer range and we may perceive our environment with greater detail. We can choose shorter and direct paths towards our desired goal, an ability that could not have been done so easily with only our sense of touch.

Analyzing such a scenario we can see how beautifully versatile our bodies are in dealing with environmental changes and the temporary loss of vision. We can see how we re-purpose our sense of touch to a proximity sensor that allows us to navigate well enough to complete our task of switching the light on. We can also appreciate the way we try to correct our estimation of where our hand is relative to the light switch once we hit the corner wall. Hitting the corner wall made us realize that we were further along the wall than we previously approximated, making us back track and make larger sweeps along the height to increase our chances of hitting our target.

Robotics research has tried to replicate these navigational abilities ever since its inception, drawing inspiration from nature as well as from human ingenuity. Algorithms have been designed with the intention of mimicking our navigational capabilities and how we determine our location by observing landmarks around us. Further, our remarkable ability to maintain a map like representation of a new environment, which can be used for working out not only shorter paths to our desired goal but also the one that requires the least amount of effort (Bird & Burgess, 2008). Some of these algorithms are bio-inspired and derive their inspiration from neuroscience research on the navigational mechanisms of rats and other animals. Other algorithms are more mathematical and programmatic in design, dealing with precise distances and measurements as opposed to fuzzy terms like ‘near’ and ‘far’ (Cadena et al., 2016).

Research has also been progressing towards improving our ability in identifying objects using artificial vision sensors like cameras. Methods have been designed to estimate with great precision the distances that an agent has moved based on the changes in observations from these sensors. In combination, the coupling of object recognition and sense of motion has been integrated into the navigational algorithms to develop robotic systems that can negotiate new areas (Pillai & Leonard, 2015). Once familiar with their environment these systems can navigate efficiently and precisely, all while correcting for over and under assumptions in how far they have moved, as well as misidentifying land-

marks. These systems are also adept at being able to correct their perceived location within their surroundings once they observe familiar landmarks, such as the corner wall in our light switch search scenario.

Obtaining information about its surroundings, a robot has to rely on multiple sensors since, as seen in nature, no single sensing method is ideally suited for use in all scenarios. For these reasons the area of research concerned with combining the data from multiple sensors has been highly active (J.-H. Kim, Starr, & Lattimer, 2015). Different sensors each have varying operating ranges that include temperature, altitude, pressure, radiation and humidity. Operation outside of these limits can cause damage which is highly likely in extreme environments like that experienced deep underwater, characterized by high pressures and low light levels, or in proximity to fires and smoke, characterized by high temperatures and particulate matter causing low visibility. If robots were to be designed for operation in the latter case, careful consideration must be made by selecting sensors that are not effected by excessive air particulate matter or high temperatures, and can continue to provide reliable measurements needed for environmental perception and navigation (dos Santos, 2013).

Current robotic platforms that are designed for operating in smoky and high temperature spaces rely on electromagnetic based sensors like radar and thermal imaging. These sensors are expensive yet are advantageous due to their long range and fidelity. In combination they are able to extract geometrical and thermal properties about objects in the environment. One sensing modality that has been relatively neglected up until recent years is tactile sensing, which can allow for the extraction of other rich features such as texture and vibration frequency (Diamond & Arabzadeh, 2013).

Cutaneous-tactile sensors, which are inspired from the sense of touch available to us through our finger and hands, typically have a small operational area and are more useful for detecting fine spatial details of a surface (Liu, Wu, Sun, & Guo, 2017). Whisker-tactile sensing, however, has an extended operational range and has been known to be used for navigation in rats, mice and seals. As an array of whiskers, the sensing system can cover a large surface area that allows for the extraction of geometric shape, texture and vibration of objects. Additionally they can act as proximity sensors and, depending on the method of construction, durable against impact. The working principle of artificial whisker systems is fairly simple and has the potential of being constructed with technology that is resistant against harsh conditions. Whisker sensing is one modality that has been undervalued in the robotics community as it has the potential of being a versatile complementary sensor for robotic perception and navigation in extreme environments.

The work is therefore motivated by the aspiration towards the design of a whiskered system that is able to navigate in a similar fashion to rats and mice. Drawing inspiration from animal behaviour as well as the plethora of research carried out in the fields of robotic vision, navigation and tactile sensing, it will be shown that whisker-tactile sensing is worthy of consideration as one of sensory systems implemented on exploratory robotic systems.

The following list includes publications that are a product of this thesis:

1. Salman and Pearson (2016). Advancing whisker based navigation through the implementation of bio-inspired whisking strategies. In *IEEE International Conference on Robotics and Biomimetics (ROBIO 2016).*

2. Salman and Pearson (2018). Whisker-RatSLAM applied to 6d object identification and spatial localisation. In V. Vouloutsi et al. (Eds.), *Biomimetic and Biohybrid Systems* (pp. 403–414). Cham: Springer International Publishing.

The list of novel contribution that this work has produced includes:

1. Adapt RatSLAM's front end to work with a whisker-tactile sensor array.

2. Defining a RatSLAM specific performance metric that aids in tuning feature matching thresholds as well as gauge the algorithm's confidence matching observations' similarity.

3. Produce a support vector regression based model to estimate radial-distance without the need for specifying whisker length.

4. A whisker object-recognition system that is able to learn and recall 3-dimensional objects.

Chapter 2

Background

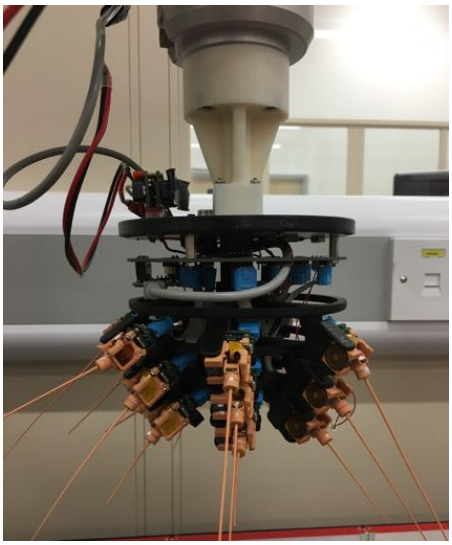


Figure 2.1: The design of the whisker-tactile sensory array that is used in this work takes inspiration from several rodents including Etruscan shrews, mice and rats (Pearson et al., 2011). One of the major goals of this research is to develop the perceptual capabilities of the artificial whisker-array to match that of animals.

The work that supports this thesis focuses on the the design of a localization and mapping, and object recognition system that operates with a whiskered-sensor array, the predominate sensing modality of this work's robotic platform. This chapter begins with a discussion of localization and mapping algorithms that are relevant to this work as well as a general description about how they work. The robotic platform's hardware is then discussed, with a particular focus on the whisker-tactile array. The final section

talks about the challenges of whisker-tactile sensing; observation of rat behavior and artificial whisker-tactile sensing related research is discussed.

2.1 Localization and mapping

This section describes the group of localization and mapping algorithms that are most relevant to this work's robotic platform, and brief explanation of their operational principles is given. A more in depth analysis of the chosen algorithm, RatSLAM, is given in order to familiarize the reader with its architecture that would aid in understanding our modifications to the default algorithm that is available online. Our work uses a C++ and Matlab implementation of RatSLAM called OpenRatSLAM. The C++ version supports ROS (Ball, 2018a) while the Matlab version (Ball, 2018b) is mainly used for offline testing. We stress that our search for an appropriate localization and mapping algorithm was done with the intention of finding a proven solution and modifying it to our needs. This work was done with the aim of integrating a whisker-tactile sensory system with a navigating mobile robot platform and contributing to the improvement of whisker-tactile based navigation, not the design of a state-of-the-art localization and mapping algorithm. We therefore include a review of appropriate solutions found in literature and clarify our choice.

2.1.1 SLAM

The localization and mapping algorithms are commonly referred to as Simultaneous Localization and Mapping algorithms or SLAM for short (Durrant-Whyte & Bailey, 2006). These group of algorithms deal with the problem of determining a robots location within an unknown space and do so by processing sensory information that inform it of the robots motion as well as its external perception (Thrun, Burgard, & Fox, 2005). Simultaneously the algorithms are tasked with mapping the robot's space, which can then be used for path planning and to inform the user of the robots environmental layout.

The need for SLAM comes from the fact that there will always be a requirement for mobile robotic platforms to navigate within spaces that lack the infrastructure or reception needed to implement exact localization systems such as that offered by Global Positioning Systems or proximity sensor networks. Typically mobile robots, as well as animals, rely on their sense of motion for estimating their current position, a process known as dead-reckoning or path integration. Dead reckoning consists of integrating

one's velocity over the period of time in which the motion occurred, adding this displacement value to the initial position in order to calculate the current position. Since self-motion sensors, be they artificial (Lozano-Perez, 2012) or biological (Etienne, Maurer, & Séguinot, 1996), are susceptible to noise, the velocity values used in the integration process are corrupted, leading to an accumulation of error and the inevitable drift of estimated pose from the actual pose.

In robotics this problem is known as SLAM and the solution to it has been the subject of research since the foundation of mobile robotics. SLAM solutions, or in robotic applications SLAM algorithms, are generally known to use path integration for estimating location, while at the same time using external perception sensors, such as cameras, for recognizing previously visited locations and correcting the estimated robot trajectory. The consequence of a more accurate trajectory is a more accurate distribution of observed landmarks and, therefore, an improvement in map fidelity. The recognition of a previously visited location is known as loop-closure and is a fundamental requirement for SLAM algorithms since without it the system will not be able to infer the amount of error built up from the path integration process.

The different subsystems of the SLAM algorithm can be divided into two, the front-end, which is responsible for the processing of internal and external perceptions and the back-end, which is responsible for processing the probabilities regarding the robots state (such as its pose) and map based on the information input from the front-end (Cadena et al., 2016).

The design of the front-end is very much specific to the robotic platform it was intended for since it would be responsible for processing the sensory data, which is therefore specific to the types of sensors used on it. Its functions include the extraction of appropriate features from the external perception sensors, such as cameras, and identifying any matches with previous observations i.e. landmark recognition. The front-end could also be responsible for processing the self-motion sensors, such as wheel encoders or inertial motion units, in order to derive an estimate for the change in position i.e. odometry (Thrun et al., 2005).

The back-end acquires the processed information in order to infer the robots current state as well as its map using the concept of probabilities i.e. it measures the belief for a given robot state and map layout conditioned on a set of previous measurements and control inputs. A general SLAM back-end formulation is described mathematically in

Equation 2.1, which is based on Bayesian inference.

$$p(x_t, m|z_{1:t}, u_{1:t}) = \eta p(z_t|x_t, m) \int p(x_t|x_{t-1}, u_{1:t}) p(x_{t-1}, m|z_{1:t-1}, u_{1:t-1}) dx_{t-1} \quad (2.1)$$

The equation describes that the posterior probability of a robot state x_t at the current time t and a map m, given all sensory observations $z_{1:t}$ and control inputs $u_{1:t}$ from initialization to the current time, may be calculated from product of two terms. For the sake of brevity the left hand term may be reduced to a shorter notation that is stated in Equation 2.2, and describes the belief in a particular outcome.

$$bel(x_t, m) = p(x_t, m|z_{1:t}, u_{1:t}) \quad (2.2)$$

Using Equation 2.1, Equation 2.2 may be reduced to:

$$bel(x_t, m) = \eta p(z_t|x_t, m) \int p(x_t|x_{t-1}, u_{1:t}) bel(x_{t-1}, m) dx_{t-1} \quad (2.3)$$

Thus, the belief $bel(x_t, m)$ is calculated by first predicting the state of the robot using path integration. The integration involves the product of the motion model, which describes the likelihood of a past control input leading to the current state ($p(x_t|x_{t-1}, u_{1:t})$), and the prior belief ($bel(x_{t-1}, m)$) (Thrun et al., 2005).

This prediction is then corrected based on the measurement model or a measure describing the belief in a particular measurement, or sensor observation, arising from the current robot state and map combination ($p(z_t|x_t, m)$). Finally since probability values lie between 0 and 1 the normalization term η is included to ensure that the values for the posterior are between the desired range. Thus, a belief measure for a specific robot state in a particular map can be obtained by first predicting the state based on the system's motion model and then correcting it based on the observations made by the system's sensors. As stated previously, Equation 2.1 is only one type of formulation and we shall describe later on how some SLAM algorithms modify this function in order to improve their performance. Following the calculation of posteriors for all states, the combination returning the highest belief value is selected to be the current most likely estimate (Thrun et al., 2005).

So far the description of the formulation has been focused mainly on the localization portion of the SLAM problem. Once the most likely estimate of the trajectory has been executed by the robot, the belief in a particular map can be generated using the sensor measurement model (Thrun et al., 2005). There are typically two predominant variants of mapping approaches used by conventional SLAM algorithms, these are occupancy

grid maps (Thrun, 2003) and topological maps (Thrun, 1998). The two methods are sometimes combined to form a hybrid map to take advantage of both their strengths while reducing the impact of their weaknesses (Tomatis, Nourbakhsh, Arras, & Siegwart, 2001).

Occupancy grid maps segment the environment into a grid like structure whose dimensions vary depending on the desired application. Operation of a robot on a 2D space will typically require a 2D grid map where each individual cell would contain a particular belief in it being an occupied or otherwise, free space. Depending on the desired resolution, the number of cells will increase quadraticaly for a 2D grid and cubicly for a 3D grid. Since the belief of each cell needs to be calculated, a 3D grid for a large volume of space would require a great deal of computational effort. This disadvantage is counterbalanced by the increase in precision that occupancy grid maps bring to the system since they allows for more precise path planning as well as the construction of a map with higher fidelity relative to topological maps (Thrun et al., 2005).

Topological maps instead resemble a graph like structure whose nodes represent particular landmarks and their connections, or edges, describe the relative spatial positioning or required transition needed to move the robot from one node to the other (Tomatis et al., 2001). The advantage of such a map is that there is no need to store any information on the free space between each landmark, which saves on storage requirements as well as computational effort. Of course this advantage is counterbalanced by the reduction in path planning precision since only the relative placement of each landmark is known and the robot would need to make sure that a large enough perception field is available for it to move from one node to another without getting lost (Thrun, 1998).

Hybrid maps combine both the pre-mentioned map structures by assigning regions within the map a particular structure, be it grid like structure or a topological layout. If a room requires precise path planning then a occupancy grid map like structure would be assigned to it where as the transition between different rooms such as that within a hallway would be better represented by a topological structure if the robot only needs to know where to head towards in order to gain access to its desired destination (Tomatis et al., 2001).

Selecting an appropriate mapping approach must take into account the size of the intended exploration environment, the needs of the user, the types of sensors and actuators that the robot would utilize as well as the computational limitations of the system. The same may be said about the chosen SLAM algorithm since, currently, there is no one-size-fits-all algorithm and the overall system specification must be taken into ac-

count before a selection is made. A particularly important aspect of this work's robotic platform are the limitations of the external perception sensor i.e. the whiskered-array. Unlike visual or range based sensors, whisker-arrays are very limited in terms of range, resolution and sampling rate. State-of-the-art SLAM methods, especially those that map 3-dimensional space, are typically not designed for whisker-tactile sensors and instead rely on those with inverse qualities such as that offered by cameras and Lidar sensors (Cadena et al., 2016).

With regards to running a SLAM algorithm for navigation on a whiskered-robot platform, research work is very sparse. Two particularly relevant pieces of work include the use of particle filter based SLAM algorithms with occupancy grid mapping (C. Fox, Evans, Pearson, & Prescott, 2012; Pearson et al., 2013). The whiskered-robots were both set to explore a small planar area ($6.25m^2$ and $9m^2$ for (C. Fox et al., 2012) and (Pearson et al., 2013) respectively) and the results indicated that only the work of Pearson et. al performed a successful set of loop closures. The main differences between the two pieces of work was the number of whisker modules attached to the robot platform as well as the array's morphology with Pearson et. al using 18 active whiskers that are arranged in a way that mimics that of a shrew, and Fox et. al using 4 passive whiskers that are arranged in two v-shaped pairs. Since the robotic platform is similar to that in (Pearson et al., 2013) a particle filter approach would be sensible.

Particle filters are a subset of non-paramteric Bayesian inference based solution for estimating the robot's state and map (Gustafsson et al., 2002), and unlike paramteric Guassian based filters like the Kalman and Extended Kalman filter, are not limited to describing the posterior probability of states as a uni-modal Gaussian distribution (Thrun et al., 2005), thus allowing for the consideration of sensors that exhibit non linear errors (Aulinas, Petillot, Salvi, & Lladó, 2008). The filter operates by sampling a unique set of random variables from the posterior distribution that represents the belief in robot state and map, which is defined in Equation 2.3. Each particle would move according to the value drawn from the control model and assigns a weighting according to the value drawn from the measurement model. The goal is to converge towards the true state of the robot and map once the robot has gathered enough evidence, which is done through exploration and observation. Alternative parametric based Bayesian filters include the Extended Kalman Filter (EKF), which has been known to demonstrate superior performance in certain cases (Cadena et al., 2016). Solutions such as FastSLAM integrate both filters to leverage the versatility of particle filters and the speed of EKFs (Montemerlo, Thrun, Koller, Wegbreit, et al., 2002), producing a capable SLAM solution that could potentially be used for this work's robotic system.

2.1.1.1 Kalman Filter

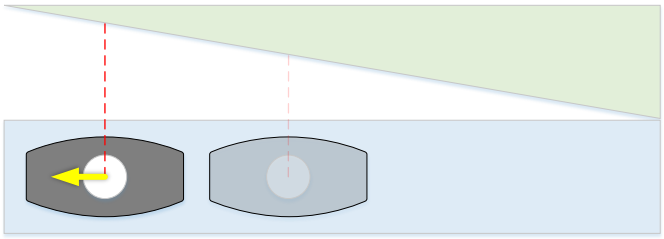


Figure 2.2: An illustration of a toy example where the state of a system, which in this case is a boat's position and velocity $\hat{\boldsymbol{x}}_k$, is estimated using a combination of kinematic equations based on previous state $\boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1}$. The boat's ability to accelerate is taken into consideration by the factor $\boldsymbol{B}_k\vec{\boldsymbol{u}}_k$, where $\vec{\boldsymbol{u}}_k$ represents the external influence on the system i.e. acceleration, which is illustrated by the yellow vector. In order to improve the estimate of system state, a distance measuring sensor is used to infer position and velocity, It can be seen that with a sloping shoreline, there exists a unique distance that defines the boat's position. To consider noise within the sensor device, measurements are defined as a Gaussian probability distribution with a mean of $\vec{z}_k$ and a covariance of $\boldsymbol{R}_k$.

Given a toy example, which is illustrated in Figure 2.2, a boat with a single degree of freedom moves in the left hand direction, along the shoreline. Given a distance measuring sensor, the rider needs to know its current position x_k and velocity $\dot{x}_k$. The vector in Equation 2.4 represents its current state. If the position and velocity of the boat are correlated, their covariance matrix $\mathbf{P}_k$ in Equation 2.5 would a non-identity matrix. The combination of mean and covariance imply a Gaussian distribution, which are the ideal types of errors for which the Kalman filter is designed to correct against, however, Kalman filters have also been shown to correct for other types of error distributions (Kalman, 1960). For the sake of simplicity, this example assumes that all uncertainty can be modeled by a Gaussian distribution.

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{P}_k = \begin{bmatrix} \Sigma_{xx} & \Sigma_{x\dot{x}} \\ \Sigma_{\dot{x}x} & \Sigma_{\dot{x}\dot{x}} \end{bmatrix} \tag{2.5}$$

Taking into consideration $\hat{\boldsymbol{x}}_{k-1}$ the previous state, part of the current state $\hat{\boldsymbol{x}}_k$ can be estimated using kinematic equations, as shown in Equation 2.6.

$$\begin{aligned} x_k &= x_{k-1} + \Delta t \dot{x}_{k-1} \\ \dot{x}_k &= \qquad\qquad \dot{x}_{k-1} \end{aligned} \tag{2.6}$$

In matrix form the predicted state vector would take the form of Equation 2.7.

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\boldsymbol{x}}_{k-1}$$

$$\hat{\boldsymbol{x}}_k = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1} \tag{2.7}$$

The matrix $\boldsymbol{F}_k$ is thus used in part to predict the current state based on the influence of the system's previous state. To update the covariance, the matrix identity relating the covariance of a random vector to that of a random vector times a constant one (which is stated in Equation 2.8 (Petersen, Pedersen, et al., 2008)) is used to derive the updated state covariance matrix $\mathbf{P}_k$. Thus, the covariance matrix for the current state would be equal to Equation 2.9.

$$\begin{aligned} Cov(x) &= \Sigma \\ Cov(Ax) &= A\Sigma A^T \end{aligned} \tag{2.8}$$

$$\mathbf{P}_k = \boldsymbol{F}_k \mathbf{P}_{k-1} \boldsymbol{F}_k^T \tag{2.9}$$

So far the influence of the system's previous state is only considered. If the boat in this example were to have a motor, this control input's influence would also need to considered. The motor's operation would result in a change in acceleration, which affects position and velocity according to the Equation 2.10. The random control input matrix is defined by the variable $\vec{\boldsymbol{u}}_k$ and the constant matrix relating the input to the boat's state is $\mathbf{B}_k$. The current state vector in terms of the previous state and current control input is defined in Equation 2.11.

$$\begin{aligned} x_k &= x_{k-1} + \Delta t \dot{x} + \frac{1}{2} \ddot{x}_k \Delta t^2 \\ \dot{x}_k &= \qquad\qquad \dot{x}_{k-1} + \ddot{\vec{x}}_k \Delta t \end{aligned} \tag{2.10}$$

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1} + \begin{bmatrix}\frac{\Delta t^2}{2}\\ \Delta t\end{bmatrix}\ddot{\vec{x}}_k \\ &= \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}_k\vec{\boldsymbol{u}}_k \end{aligned} \tag{2.11}$$

In order to consider the uncertainty brought on by external influence such as wind and drag the covariance matrix would need to be adjusted accordingly. This additional uncertainty is represented by another covariance matrix that is added to Equation 2.9 and results in the updated state covariance being equal to Equation 2.12.

$$\mathbf{P}_k = \boldsymbol{F}_k\mathbf{P}_{k-1}\boldsymbol{F}_k^T + \boldsymbol{Q}_k \tag{2.12}$$

So the current state vector and its corresponding covariance matrix represents the systems predicted state. To further improve the estimate, sensor measurements that can be used to infer system state can be included so that a best estimate can be calculated. In the case of the boat example, a measurement of distance to the shoreline can be used to infer position, and its derivative can be used to infer velocity. To map the relationship between sensor measurement and system state a matrix $\boldsymbol{H}_k$ is defined in Equation 2.13. The equation states that the sensor measurements can be predicted given the multiplication of the constant matrix $\boldsymbol{H}_k$ with the current system state $\hat{\boldsymbol{x}}_k$. Equation 2.13 also includes the covariance of the predicted sensor readings and is derived using the same matrix identity in Equation 2.8.

$$\vec{\mu}_{predicted} = \boldsymbol{H}_k\hat{\boldsymbol{x}}_k \tag{2.13}$$

$$\boldsymbol{\Sigma}_{predicted} = \boldsymbol{H}_k\boldsymbol{P}_k\boldsymbol{H}_k^T \tag{2.14}$$

This step is equivalent to saying that given the boat's state $\hat{\boldsymbol{x}}_k$, the predicted sensor readings would have probability distribution with a mean of $\vec{\mu}_{predicted}$ and a covariance of $\boldsymbol{\Sigma}_{predicted}$.

The Kalman filter continues to obtain a best estimate of system state by taking into account the actual sensor readings $\vec{z}_k$. Given a Gaussian distributed error for the sensing system, the readings would thus have a mean of $\vec{z}_k$ and a covariance of $\boldsymbol{R}_k$.

The best estimate of system state would finally be equal to the mean of the distribution resulting from the intersection of the predicted sensor readings' distribution and the observed readings' distribution. The final best estimate mean and covariance is defined in Equation 2.15, where $\boldsymbol{K}$ is the Kalman gain.

$$\begin{aligned}\hat{\boldsymbol{x}}_k^{'} &= \hat{\boldsymbol{x}}_k + \boldsymbol{K}^{'}(\vec{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k) \\ \boldsymbol{P}_k^{'} &= \boldsymbol{P}_k - \boldsymbol{K}^{'}\boldsymbol{H}_k\boldsymbol{P}_k \\ \boldsymbol{K}^{'} &= \boldsymbol{P}_k\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_k\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}\end{aligned} \tag{2.15}$$

From equation 2.11 it can be seen that the state matrix consists of a set of linear equations. In the event that a state vector does not behave linearly and the state and sensing systems are non-linear like that shown in Equation 2.16, one type of approach would be to implement an Extended Kalman Filter. EKF uses Taylor Series expansion to linearise the non-linear functions $g()$ and $h()$ in Equation 2.16 (Thrun et al., 2005).

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= g(\vec{\boldsymbol{u}}_k, \hat{\boldsymbol{x}}_{k-1}) \\ \mathbf{P}_k &= \boldsymbol{G}_k\mathbf{P}_{k-1}\boldsymbol{G}_k^T \\ \vec{z}_k &= h(\hat{\boldsymbol{x}}_k) + \boldsymbol{R}_k\end{aligned} \tag{2.16}$$

Where $\boldsymbol{G}_k = \frac{\delta g(\vec{\boldsymbol{u}}_k, \hat{\boldsymbol{x}}_{k-1})}{\delta \hat{\boldsymbol{x}}_{k-1}}$ and $\boldsymbol{H}_k = \frac{\delta h(\hat{\boldsymbol{x}}_k)}{\delta \hat{\boldsymbol{x}}_k}$ (Thrun et al., 2005).

2.1.1.2 Particle Filter

The Kalman Filter and EKF are both optimal for systems that exhibit Gaussian noise and uncertainty. For certain cases were a system under consideration is not easily modelled by Gaussian distributions, a Particle Filter might provide a better sollution. Particle Filters represent probability distribution in the form of discrete samples that are drawn from a particular probability distribution and such a characteristic is advantageous since a broader range of distributions can be approximated in comparison to Gaussian distributions for which filters like the Kalman Filter and EKF are limited to.

The posterior belief of a particular system state is represented by the relative weighting of each particle within the set $\boldsymbol{\chi}_k$, where $\boldsymbol{\chi}_k := \hat{\boldsymbol{x}}_k^{[1]}, \hat{\boldsymbol{x}}_k^{[2]}, ...\hat{\boldsymbol{x}}_k^{[M]}$ and M is the total number of particle samples that the Particle Filter is set up to use. Each particle is drawn from a probability distribution i.e. $\hat{\boldsymbol{x}}_k^{[m]} \sim p(\hat{\boldsymbol{x}}_k|\vec{\boldsymbol{u}}_k, \hat{\boldsymbol{x}}_{k-1}^{[m]})$ and their weighting would be equal to the probability that a system's observed sensor reading $\vec{z}_k$ matches the estimated sensor reading i.e. $w_k^{[m]} = p(\vec{z}_k|\hat{\boldsymbol{x}}_k^{[m]})$ (Thrun et al., 2005).

2.1.2 RatSLAM

A relatively unique SLAM solution that does not explicitly use Bayesian probabilities and instead draws inspiration from biology to solve the problem of SLAM is called RatSLAM

(Arleo & Gerstner, 2000; Sünderhauf & Protzel, 2010). RatSLAM is also unique relative to previous SLAM methods in that it does not require the specification of a motion model nor that of a measurement model. It instead only requires an odometry input that describes the velocity of the sensor-array and a method of measuring similarity of observations for the purpose of place-recognition.

The algorithm has been shown to perform loop closure over large terrains in real-time running on multiple robotic platforms with modest hardware (Ball et al., 2013; Milford & Wyeth, 2008). The lack of requirement for a motion and measurement model would reduce the complexity of setting up the algorithm therefore making it easier to transfer onto other platforms should the need arise. Such advantages make RatSLAM a very attractive choice particularly since it has been ported to the Robot Operating System and is available in open source (Ball et al., 2013). Literature has also shown that RatSLAM can be modified so as to consider movement through higher dimensions (Zaffari, dos Santos, Duarte, d. A. Fernandes, & d. C. Botelho, 2016).

For these reasons RatSLAM is chosen as the main SLAM algorithm, noting that although there are other algorithms available that are just as capable, RatSLAM offers an alternative to traditional algorithms and would therefore be interesting to investigate, particularly any added benefit that its neural network based architecture can provide in combination with neuromorphic hardware (Wang et al., 2017).

RatSLAM is a bio-inspired SLAM algorithm that mimics aspects of the spatial navigation mechanisms of the mammalian brain, specifically the hippocampal formation (Arleo & Gerstner, 2000; O'Keefe & Dostrovsky, 1971). The algorithm however comes short of producing a faithful representation of the models explained in the literature due to the author's aim of developing a high performance robotic SLAM algorithm, giving precedence over a more biologically plausible one.

2.1.2.1 Algorithm

The algorithm consists of a continuous attractor network model arranged into a 3 dimensional manifold of 'pose cells' that functionally represent the behaviour of 'grid cells' observed in entorhinal cortex of a rat (Moser, Kropff, & Moser, 2008). A manifold in this case is a continuous topological space that maps its locations to a Euclidean space; an example of a 1 dimensional manifold is a circle that represents locations on a line. A manifold structure is used as it provides the means to represent a large and continuous environment with a discrete set of finite elements. In line with the requirement first suggested by (O'Keefe & Dostrovsky, 1971), inputs that provide allothetic (external)

and idiothetic (internal) cues are fed to the network so as to be able to build a coherent cognitive map of the environment. RatSLAM was originally designed to obtain external cues via visual sensors and therefore contains elements within its framework that assume for such a decision, such as its definition of visual cells that refer to the nodes containing observed environmental features. However, it must be noted that the algorithm's external cues may be sourced from any form of sensory modality given that the observed features contain enough detail that allow for separating distinct locations from one another. In order to remain consistent with the the author's (Milford & Wyeth, 2008) description the terminology will remain the same.

2.1.2.1.1 1-Dimensional Example For the sake of brevity, the authors of RatSLAM give a good description of the algorithms operation by focusing on a 1 dimensional continuous attractor network (CAN) of HD cells as opposed to the 3 dimensional pose-cell grid (Milford & Wyeth, 2008). The network is visualized by unfolding the circular manifold into a line containing a series of interconnected head direction cells, which due to the unfolding, now have wrapping connections between cells on the edge of the line as seen in Figure 2.3.

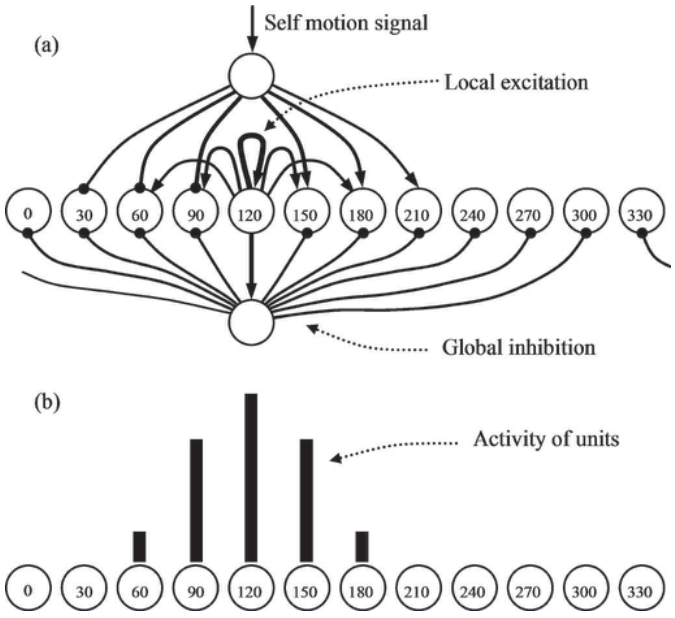


Figure 2.3: (a) Excitatory (arrows), inhibitory (round) and self-motions for a continuous attractor network representation of head direction cells. (b) A stable activity packet centered at 120°. This figure was obtained with permission from the work of Milford et. al (Milford & Wyeth, 2008)

The figure illustrates an instantaneous moment where a self-motion signal, denoting a positive angular velocity, is given as input to a current heading of 120 degrees. The self-motion node is connected to the HD cell's via inhibitory and excitatory synapses that allow for the shifting of cell activity in the appropriate direction, this process is known as path integration. Each HD cell incorporates connections that excite itself as well as, at a lower intensity, neighboring cells. In combination with global inhibition, the connections serve to maintain a stable peak of activity centered on the head direction. The overall process may be seen as a mechanism for integrating self-motion data, hence the term path integration. The excitation of HD cells via self-motion signals will be referred to as local excitation.

The term for estimating one's position through path integration is called dead reckoning. Dead reckoning, however, is susceptible to drift, which is an outcome of integrating noisy odometry data, leading to a drift of perceived position from the actual position. Unfortunately eliminating noise is not possible due to the innumerable sources of disturbances that affect the accuracy of self-motion signals, including but not limited to, wheel slippage and limited sensor resolution. This issue calls for a method that is capable of estimating the drift associated error and correcting for it. The correction mechanism that RatSLAM implements is derived from the systems inherent filtering that occurs as the influences of self-motion and external cues compete for HD cell packet dominance. The figure shows a single HD cell packet, which contains HD cells 60 through to 180 degrees with a peak centered on 120 degrees. The distribution resulting from local excitation is of a Gaussian type. Unlike most other SLAM algorithms, the result of path integration does not correspond to the built up uncertainty. The parameters of the Gaussian distribution resulting from the self-motion signal is fixed in RatSLAM and therefore is not a representation of accumulated uncertainty.

To enable the influence of spatial related cells by visual cues, local-view cells (LV) are defined. LV cells each contain a template of the observed scene and a link to the centroid of the active HD cell packet. In RatSLAM, a feature is extracted from the template, which is then used to compare similarities against other scenes. RatSLAM defines a template as a cropped region in an image that is pertinent in classifying scenes. For example, an image from the dashboard of a car may be split into an upper and lower region. The upper will contain the sky that lacks identifiable features, which is in contrast to the lower region that is rich with details and thus abundant in identifiable features. In the case of RatSLAM, the feature that is extracted from the template is its scan-line profile intensity, a vector that contains the summed pixel intensities of the template's columns.

The uniqueness, or similarity, of LV features is determined by calculating the sum of absolute differences between the feature vectors. A value lower than a user-defined threshold would classify the features as similar, where as a value exceeding the threshold would result in the creation of a new LV cell. In an effort to link a spatial location with an observation, a one-shot learning method is used to strengthen the connections between co-activated LV and HD cells. The future activation of an LV cell would therefore result in the associated HD cells being activated in parallel. Alternatively, the cessation of LV cell activation would result in the decay of the HD cell's activity. The emergence of a dominant HD cell packet will therefore only occur given the ordered and consistent activation of LV cells. The emergence of a dominant packet will shift the perceived location from the currently held one, to the one associated with the activated LV cell. Such a shift is referred to as a re-localization. This process describes the interplay between internal and external cues that form the drift correction mechanism.

2.1.2.1.2 Operation For a robot operating in a conventional planar environment RatSLAM requires 3 spatial variables for vertical y', horizontal x', and angular θ' position. The manifold consists of pose cells (PC), each of which represents a unique pose and encode both a discrete surface position and heading. The structure is thus referred to as the pose cell grid. Similar to the drift correcting mechanism described in the example, the pose cells operate within a competitive attractive network where internal and external cues influence pose cell packet activity and vie for dominance, the centroid of the dominant pose cell packet represents the systems best estimate of the agent's current pose. Equation 2.17 defines the change in pose cell activity according to the activity level of a LV cell. The learnt connections between PC and LV cells are defined by a connectivity matrix β, which is calculated according to equation 2.18 and results in higher weights being assigned to highly activated PC and LV pairs. The pose code P is defined as a matrix that contains activity levels of all the cells within the pose cell grid while the view code V is defined as a vector that contains the activity levels of LV cells.

$$\Delta P_{x',y',\theta'} = \frac{\delta}{\eta_{act}} \sum_i \beta_{i,x',y',\theta'} V_i \tag{2.17}$$

$$\beta^{t+1}_{i,x',y',\theta'} = \max(\beta^t_{i,x',y',\theta'}, \lambda V_i P_{x',y',\theta'}) \tag{2.18}$$

It must be noted that this is not observed to be the mechanism employed in the code provided by the authors of RatSLAM and the connections between LV and PC cells is simply represented as the calculated location of the maximally active pose cell during

the time in which the LV cell was first generated. This simplification did not seem to degrade the performance of the system and it continued to perform correct loop closure on the several data sets (Ball & Milford, 2015) that were used in their publications (Ball et al., 2013; Ball, Heath, Wyeth, & Wiles, 2010). The authors did mention that they sacrificed biological correctness in favor of computational efficiency and this may be one of the areas in which they focused on (Milford & Wyeth, 2008).

The path integration mechanism is adapted for the pose cell grid by expanding the variance of the excitation pattern from the 1 dimensional Gaussian distribution to a 3 dimensional one, its value calculated according to equation 2.19. For all equations, the values (a, b, c) each represent distance between elements in the pose cell grid coordinates (x', y', θ').

$$\varepsilon_{a,b,c} = \exp^{-\left(-a^2+b^2\right)/k_p} \exp^{-c^2/k_d} \tag{2.19}$$

The resulting change in pose cell activity due to the self-motion signal is defined in equation 2.20. In an effort to speed up the computation involved, activity related to path-integration is shifted in the desired direction by copying the packet and placing it in the required position. Global inhibition is defined in equation 2.21. To address the wrap around connectivity of elements at the borders, equation 2.22 defines how their distance values are to be derived. The values denoted by the variable n refer to the size of the specified dimension and the variables i, j and k refer to the coordinate of the cell in the excitation or inhibition matrix.

$$\sum_{i=0}^{n_{x'}-1} \sum_{j=0}^{n_{y'}-1} \sum_{k=0}^{n_{\theta'}-1} P_{i,j,k} \varepsilon_{a,b,c} \tag{2.20}$$

$$\Delta P_{x',y',\theta'} = \sum_{i=0}^{n_{x'}} \sum_{j=0}^{n_{y'}} \sum_{k=0}^{n_{\theta'}} P_{i,j,k} \psi_{a,b,c} - \varphi \tag{2.21}$$

$$\begin{aligned} a &= (x' - i)(\mathrm{mod} n_{x'}) \\ b &= (y' - j)(\mathrm{mod} n_{y'}) \\ c &= (\theta' - k)(\mathrm{mod} n_{\theta'}) \end{aligned} \tag{2.22}$$

In order to recall past poses along with their associated observations, an experience map (EM) is constructed in order to store these links. Analogous to the *cognitive map* that is described in the biology section, the EM consists of experience nodes that are linked together, forming a topological map. An experience node is a structure that

contains information regarding the pair of associated of pose and local view cells. In addition to these associations, experience nodes contain their relative position to adjacent experiences within the EM. Experience at index i is defined in equation 2.23 and contains the coordinates of the highest active pose cell P and identification V of the LV cell that were active during the creation of the experience. The third element $\mathbf{p}_i$ is a vector that contains the position of e_i relative to e_{i-1} as defined by the odometry.

$$e_i = \{P^i, V^i, \mathbf{p}^i\} \tag{2.23}$$

The creation of a new experience is ensued by a sufficient change in either pose or observed scene. Equation 2.24 formulates this condition by defining metric S that encodes experience similarity. A similarity value exceeding the threshold S_{max} would create a new experience while one lower would result in the re-localization to a past experience. A re-localization, or loop-closure, would result in the adjustment of all experiences' positions within the EM in order to correct for the perceived error that was accumulated during the time before any localization occurred. This correction mechanism is referred to as graph relaxation and works by spreading the accumulated error across all nodes in an attempt to reduce the overall error in estimated robot trajectory. By reducing the error in robot trajectory the positioning of the observed landmarks would also be closer to the ground truth and a more coherent topological map may be constructed. The adjustment to the experience position is defined by equation 2.25 where α (optimally set by the authors to 0.5) is the rate of graph relaxation.

$$S = \mu_p|P^i - P| + \mu_v|V^i - V| \tag{2.24}$$

$$\Delta\mathbf{p}^i = \alpha \left[\sum_{j=1}^{N_f} (\mathbf{p}^j - \mathbf{p}^i - \Delta\mathbf{p}^{ij}) + \sum_{k=1}^{N_t} (\mathbf{p}^k - \mathbf{p}^i - \Delta\mathbf{p}^{ki}) \right] \tag{2.25}$$

RatSLAM's mechanisms are inspired from theoretical models related to operations within the hippocampus. The pose cell grid can be compared to the layers of conjunctive grid cell networks that also exhibit the wrap around behavior with each of their layers tuned to a particular heading (Burak & Fiete, 2009).

As mentioned previously, the original design of the RatSLAM algorithm took into account the use of a camera as the main sensor for obtaining externel cues. We instead would like to modify the algorithm in order for it to operate with this work's whisker-tactile sensor array. Before discussing the specifics of the modification, a description of the the sensory system is given.

2.2 Hardware

The complete list of hardware used in this work includes: the whisker-tactile array that is mounted on the end of a UR10 or UR5 arm, and two computers that each control the whisker-tactile array and the Universal-Robots arm. The whisker-tactile array is connected to a lenovo x201 laptop via a USB port, and is loaded with Ubuntu 14.04 along with the drivers needed to communicate with the array's custom FPGA, which is responsible for motor control and sensory data retrieval. The Universal-Robots arm is operated via a separate PC that uses the ROS framework for communication, which includes arm control and pose retrieval.

2.2.1 Robotic Arm Manipulator

The work carried out in this thesis uses two Universal-Robots arms for different experiments: a UR10 arm for the work in Chapter 3 and the UR5 for all remaining chapters. The main difference is their payload capacity and reach, which is higher for the UR10 arm (1.3m vs 0.85m). Precision of the arms with regards to pose repeatability was the same for both models ($\pm$0.1 mm) (Universal-Robots, 2018a, 2018b).

Both arms were controlled via the Robot Operating System (ROS), specifically ROS Indigo on a PC running Ubuntu version 14.04. A motion planning framework called MoveIt was used for all Inverse Kinematic related tasks. Universal-Robots provides all the drivers and arm description files needed to interface the manipulators with ROS and MoveIt (*Universal Robots ROS*, 2018). All the proceeding work did not involve the need for any specific velocity control and instead limited the motion planning trajectories to low joint speeds for precautionary reasons (0.1 rad/s for all joints). MoveIt also provided the necessary support for collision avoidance and other desirable restrictions such as the limitation of trajectories to a certain range of joint values.

Collision avoidance works by defining a virtual environment that MoveIt uses to make sure that any trajectories it generates do not cause the arm to come into contact with any object. The arm's virtual model can be modified to include the end-effector i.e. the whisker-tactile array. To avoid colliding the whisker-array with any obstacle, while still allowing for the whisker-shafts to make contact, a mesh was modeled in order to serve as the virtual end-effector. The model is shown in Figure 2.4, where a surrounding mesh is added to ensure the arm is not put in a position that prevents whiskers from being able to fully retract.

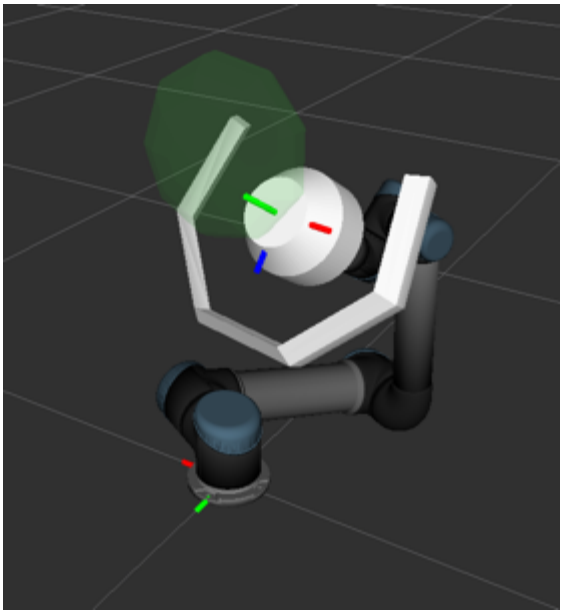


Figure 2.4: Using the MoveIt package from generating the arm's trajectories, a virtual environment is defined, one that best represents the real robot's space in order for the planner to avoid collisions. Modifications are made to the arm's virtual model to include a region of space that is to be avoided by the planner (shown in white). The model includes a U-shaped mesh that represents the extreme locations of the whiskers during their sweeping motion, which prevents the arm from being in a position that stops the whiskers from reaching their fully retracted state. The green elispodal object is an example of an environmental obstacle that the planner has to consider when planning a movement.

2.2.2 Whisker-tactile Sensor Array

The physical whisker-sensor array consists of 18 individually actuated whisker modules that are arranged in a manner that mimics the whisker-morphology of a shrew (Mitchinson, Pearson, Pipe, & Prescott, 2012a). More specifically, the whiskers are arranged into 3 concentric circles of 6 whiskers where each whisker measures approximately 60 mm, 100 mm and 160 mm from rostral to caudal.

The nature of all tactile-sensors is that they need to make contact with the surface that is to be examined and therefore need to be manipulated across the surface at a close distance. To facilitate this requirement the array is mounted to the end of a Universal-Robots robotic arm that operates with 6 degrees of freedom. The arm also includes position sensors that can return the exact position of the sensor-array relative to the arm's base thereby allowing us to use forward kinematics for contact localization and

error calculations based on the ground truth measurements. For the work in Chapter 3 a UR10 arm with a reach of 1.3 m is used, and for the proceeding Chapters a UR5 arm with a reach of 0.85 m is used. Aside from the difference in reach and maximum payload weight, the arms had similar specification with respect to their degrees of freedom and joint-position sensors' resolution and sampling rate. The whisker-array was mounted to the end of the arm by a custom built adapter, which is shown in Figure 2.5.

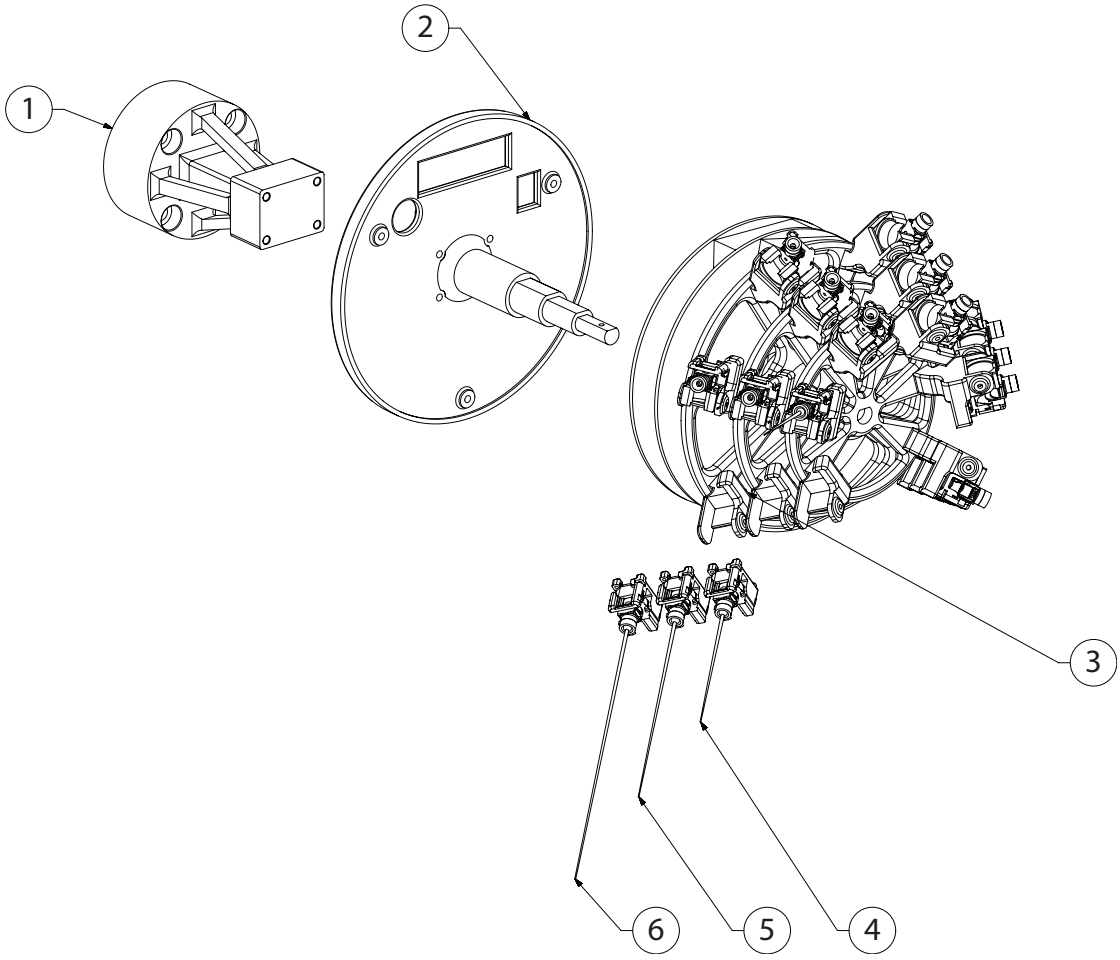


Figure 2.5: The whisker-array is mounted onto the robotic arm via a 3D printed adaptor (1). The adapter is connected to the whisker-array base (2) that is used to hold onto the whisker-modules' support structure (3). The structure supports 18 individually actuated whisker modules, which are arranged in 3 concentric circles, with each circle containing 6 whiskers. In a rostral to caudal direction the length of the whiskers increase. The approximate length of each whisker varies from 60 mm (5), 100 mm (6) and 160 mm (7). The model used to generate this image was obtained with permission from work of Pearson et al. (2013).

The whisker-array has been used in several pieces of work including (Mitchinson et al., 2012a; Mitchinson, Pearson, Pipe, & Prescott, 2012b; Pearson et al., 2013) with a detailed technical specification of the individual whisker modules described in the work of (Sullivan et al., 2012).

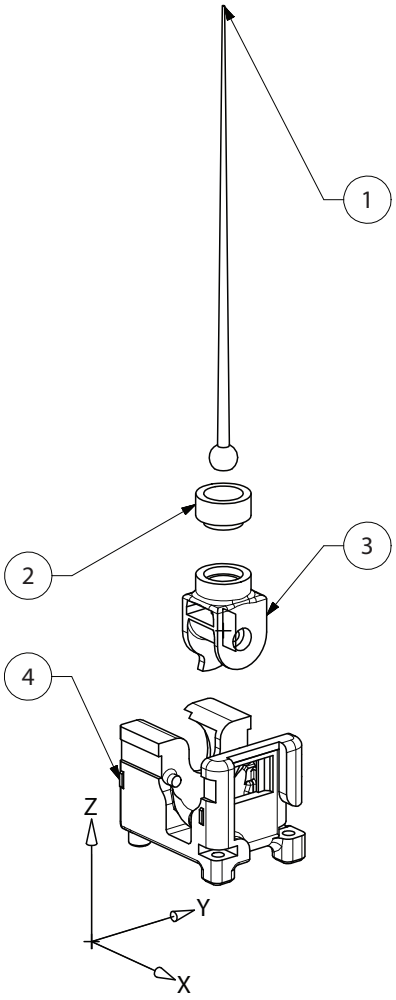


Figure 2.6: The whisker module consists of several components that include a whisker shaft (1), artificial follicle (2), follicle-motor adaptor (3) and the motor casing (4). The deflection sensor is placed within the follicle-motor adaptor and is able to detect movement of the shaft's base in two orthogonal directions. The whisker-angle sensor is placed on the side of the motor casing and measures the rotation of the follicle-motor adaptor. The model used to generate this image was obtained with permission from work of Pearson et al. (2013).

Each whisker module, which is illustrated in Figure 2.6 on page 24, consists of a

3D printed linearly tapered whisker shaft. The whisker shafts were constructed from a material called NanoCure RC25 on an Envisiontec rapid prototyping 3D printer. The diameter at the base of each whisker (irrespective of length) was 2 mm, with a linear taper toward a tip diameter of 0.6 mm. The whisker shaft is held in place within an artificial follicle using a flexible polyurethane mold rubber, which allows for a limited range of movement that approximately places the pivot point around the center of the whisker's bulbous base. A contact along the length of the whisker shaft will therefore translate to a small movement of the whisker's base. The base of the whisker holds a magnet that is used along with a Melexis MLX90333 Hall-effect based sensor, which is able to measure changes in the magnet's position along two orthogonal planar directions; the design of the whisker allows us to detect deflection of the whisker shaft by observing the displacement of its base, this is illustrated in Figure 2.7. This mechanism attempts to mimic the phenomenon of how rats measure shaft deflection at the base of their whiskers (Diamond & Arabzadeh, 2013).

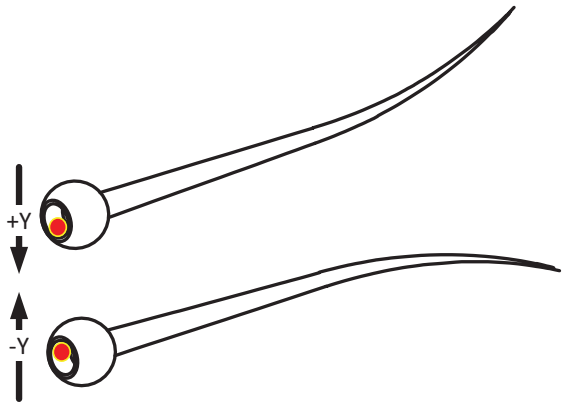


Figure 2.7: The figure illustrates how the whisker deflection is translated to a motion at the whisker base. The red circle highlights the position of a magnet attached to the base of the whisker, which is detectable by a hall effect sensor. The deflection signal is therefore a representation of this motion. The 2D hall effect sensor picks up motion in the$x$(horizontal) direction as well, however, this is omitted from the figure for brevity. The model used to generate this image was obtained with permission from work of Pearson et al. (2013).

The artificial follicle in turn is fixed to the shaft of a brushless DC motor that is driven by a custom built micro-controller PCB (Sullivan et al., 2012) that allows for a whisking rate of up to 10Hz, equivalent to the dominant whisking frequency exhibited in rats (Carvell & Simons, 1990). On the side opposite to where the motor shaft is fixed to the follicle, another magnet sensor pair (Melexis MLX90316) measures the whisker's angle

of rotation. The combination of the sensor, signal processing software, and whisking controller, results in a maximum sensory sampling rate of 2kHz (Sullivan et al., 2012). It is therefore possible to measure the whiskers deflection at its base in two orthogonal directions and its angle relative to base of the whisker module. Figure 2.8 illustrates the rotational limits of a whisker module. The whisking range from a fully retracted to protracted position spans 100 degrees, with the initial, fully retracted position starting at 40 degrees off the base of the module.

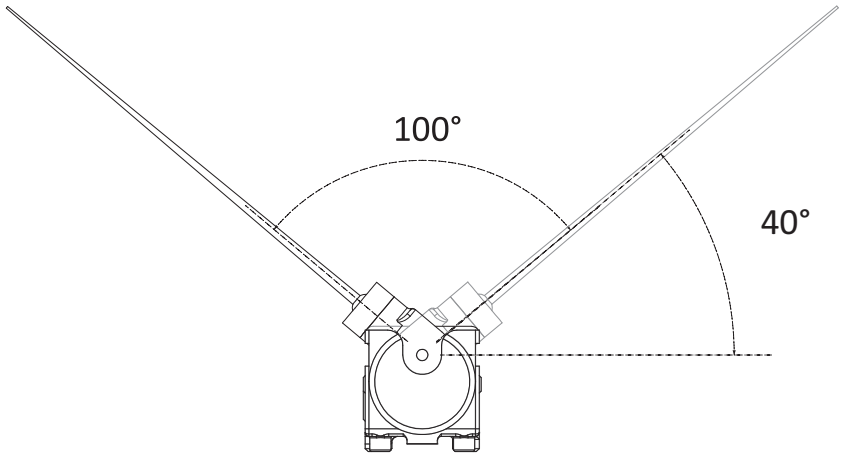


Figure 2.8: The figure illustrates how the whisker angle is measured. The initial angle starting from the fully retracted state is 40 degrees and ends at the fully protracted state at angle 140 degrees. The angle of the whisker shaft is measured relative to a plane that is parallel to the module's base. The model used to generate this image was obtained with permission from work of Pearson et al. (2013).

Figure 2.9 shows a readout for the deflection and angular sensors when the contact is made close to the protraction limit. The time of contact, which is marked by the vertical red line, can be estimated by first determining a peak within the Y-Deflection signal using several signal processing tools, described in detail in Chapter 3. The Y-Deflection signal is used in particular since it is parallel to the sweeping plane, and therefore the dominant deflection value during contact. Once the Y-Deflection peak is obtained, the time that the signal began to accelerate can be worked out to establish the time of contact. The time of contact consequently allows for the whisker contact-angle to be estimated, and other contact-signal features that allow for the extraction of geometric features such as surface slope and the distance of the contact point along the whisker shaft. The details regarding the estimation of these geometric features are described in greater detail in Chapters 4 and Section 5.2.

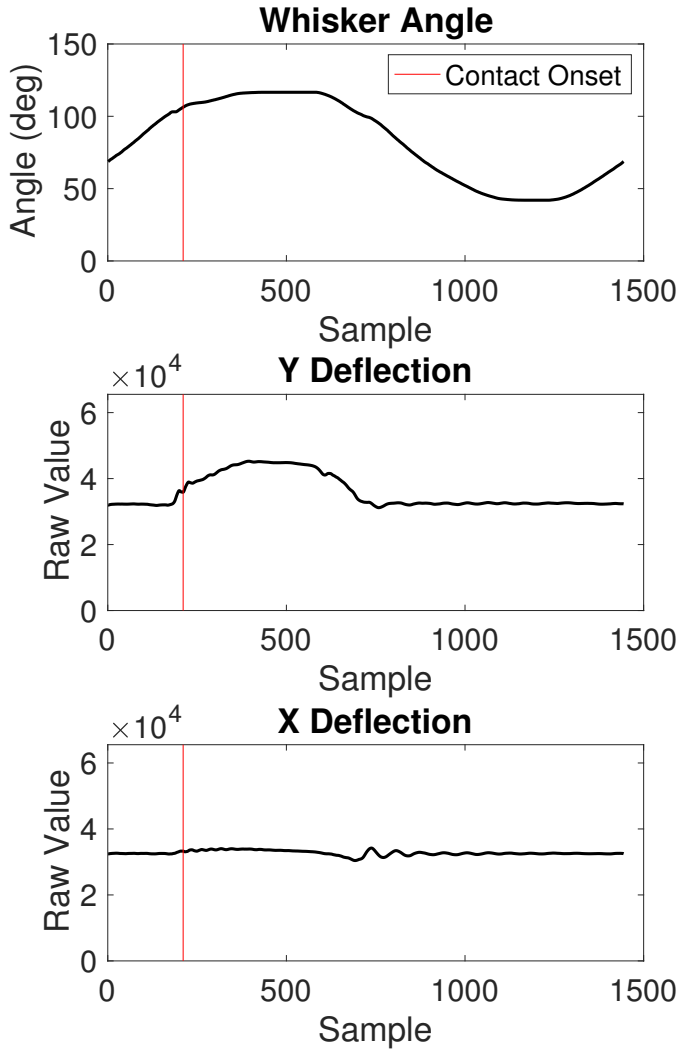


Figure 2.9: Sensor readings during a contact made close to the whisker's protraction limit. A vertical red line indicates the estimated time of contact, which is derived by finding the time at which the Y deflection signal begins to accelerate. The whiskers are whisking in an open-loop and no whisking control strategy is implemented. The whisking motion is relativly perpendicular to the surface of the contact obstacle since the$x$deflection is flat compared to the Y deflection. As discussed in the review, the surface of a contact may be infered by finding the ratio of the Y-X deflection signals during time of contact.

The whiskers are driven in a sinusoidal manner with a cosine function set with an amplitude of 100 degrees and a frequency of 1Hz, just as shown in Figure 2.9 on page 27. This cyclic motion is referred to as whisking and is a behavior observed in animals, such as rats, mice and shrews (Prescott, Mitchinson, & Grant, 2011). It must be noted that a sinusoidal function is a simplification of the rats whisking motion since they are observed to whisk in a more complex fashion where the frequency is actively varied depending on their current task (Diamond & Arabzadeh, 2013). Rats also exhibit asymmetrical whisking profiles when making contact with objects, maintaining a longer time in contact with a surface, and retracting at a slower velocity (Grant, Mitchinson, Fox, & Prescott, 2009). A sinusoidal function with a fixed frequency is used since it is less taxing on the motors and reduces the number of variables that have to be considered when trying to improve the sensor array's perception capabilities.

This work's aim, with regards to the whisker sensor-array, is to maximize the qualitative and quantitative information that can be gathered from the environment so that the mobile robot can navigate effectively. Whisker-tactile sensing is a particularly challenging sensory modality to use for navigation since it does not share a similar operational range, resolution or sampling rate afforded to more conventional navigational sensors like cameras. Environmental features thus need to be rich in detail so that we are able to better segment different regions of space. This work therefore sets out to improve the perceptual abilities of the whisker-tactile sensor array by using tools learned from other fields that include image recognition, signal processing, cutaneous-tactile sensing, machine learning and through the mimicry of rat whisking behaviour.

2.2.2.1 Hall Effect Sensor

The Melexis MLX90316 sensors in the whisker modules measure position by taking advantage of the Hall effect (Melexis, n.d.). The Hall effect is the observation of a potential difference across a current carrying electrical conductor when a magnetic field is applied (Ramsden, 2011). This effect was first observed by Edwin Hall and was used by him to determine what the sign of the predominate charge carriers are for a particular material (Ramsden, 2011).

The potential difference observed for a given material under such conditions is referred to as the Hall voltage and is defined by the Equation 2.31 (Ramsden, 2011). The equation can be derived by observing that the charge carriers, when in a state of equilibrium, have a balance of electrical field and magnetic field induced forces. Thus, the equation of forces on a charged particle would be equal to Equation 2.26, where e is the

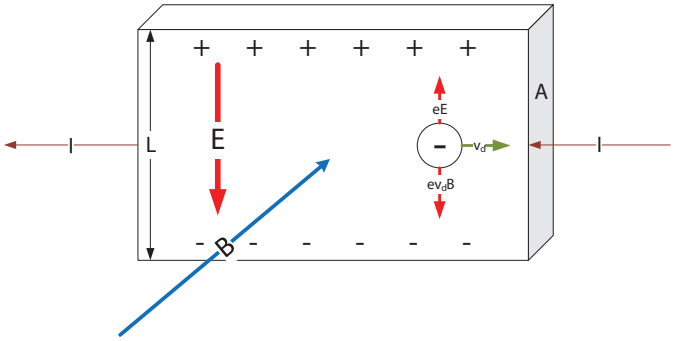


Figure 2.10: An electric conductor with a length L and a cross section area of A is identified to have predominantly negative charge carriers using the Hall effect. The Hall effect is an observation that an electric potential (called the Hall voltage) is built across a conductors length when a magnetic field B is introduced and is is perpendicular to its electrical current direction I. The charge carriers under a state of equilibrium experience a balanced pair of forces, one induced from the electric field E and is equivalent to eE where e is the magnitude of electric charge, and the second force from the magnetic field and is equivalent to ev_dB, where v_d is the charge's average drift velocity. The final Hall voltage in terms of magnetic field strength is defined in Equation 2.32, which illustrates that the Hall voltage is proportional to the magnetic field strength and average particle drift velocity. When the charge carrier is of positive polarity the drift velocity will be of an opposite sign and the measured Hall voltage will be of the same sign. Thus, a materials predominant charge carrier polarity may be deduced by measuring the Hall voltage and observing its sign.

magnitude of electric charge, E is the magnitude of electric field, and B is the magnitude of magnetic field (Ramsden, 2011).

$$eE = ev_dB \tag{2.26}$$

Equation 2.26 can be rearranged for v_d to give Equation 2.27.

$$v_d = \frac{E}{B} \tag{2.27}$$

The symbol v_d refers to the drift velocity, which is a measurement of average charge particle velocity within the conductor. The drift velocity is proportional to current and an equation relating the two together is shown in Equation 2.28. Where n is the number of charged particles per volume. Combining Equation 2.28 with Equation 2.27, Equation 2.29 may be obtained (Ramsden, 2011).

$$I = nev_dA \tag{2.28}$$

$$I = ne(\frac{E}{B})A \tag{2.29}$$

Finally, to derive the Equation for Hall voltage, the definition of electric field strength, which is the capacity of moving one unit charge per unit length i.e. Equation 2.30, and Equation 2.29, can be combined to give Equation 2.31 (Ramsden, 2011).

$$E = \frac{V}{l} \tag{2.30}$$

$$V = \frac{IBl}{neA} \tag{2.31}$$

This equation holds true for the cuboid geometry shown in Figure 2.10. Note that the figure illustrates the case where the charge carrier has a negative polarity. In the case where the B and I stay constant, a positively charged carrier would move in the opposite direction, resulting in a negative v_d value. Combining Equations 2.26 and 2.30 to give Equation 2.32, it can be seen that a negative drift velocity would result in a negative Hall voltage (Ramsden, 2011).

$$V = Blv_d \tag{2.32}$$

Since the distance between a magnetic device is inversely proportional to the magnetic field strength felt by the conductor (Zangwill, 2013), a linear distance sensor utilizing the Hall effect can be designed by noting the Hall voltage for varying distances. For rotary position sensors such as the Melexis MLX90316, three conductors or magentic flux components, are used to determine either linear motion, rotary or joystick type motion (Melexis, n.d.).

2.3 Sensing

Touch or tactile sensing is a relatively new modality that has been implemented on current robotic platforms, predominantly on humanoid robots that are needed for grasping and object manipulation tasks (Dahiya, Metta, Valle, & Sandini, 2010). Tactile sensors that are implemented on robotic hands are referred to as cutaneous-tactile sensors since they attempt to mimic receptors found under the skin. These artificial sensors attempt to extract the same information that our skin receptors can, which include object shape, deformity, temperature and texture. There are additional receptors that are responsible for detecting pain and some attempts have been made to replicate them artificially (Xu

et al., 2015), however, these type of sensors are more state-of-the-art and are uncommon on commercial robotic hands/platforms.

The following work instead focuses on whisker-tactile sensing and the implementation of artificial whisker-tactile sensors for the purpose of object recognition and navigation. As such the following section includes review of research dealing with the behaviour and abilities of rats, including the morphology of their whiskers, and observed capabilities in texture, shape and vibrational identification (Diamond & Arabzadeh, 2013). Subsequently, the section continues to describe advances in artificial whisker-sensing and how challenges regarding contact localization, texture and object identification have been addressed.

Rats major set of active whiskers may be found on their snouts with approximately 30 mirrored on each side. Rat whiskers are divided into two sets, Macrovibrissae and Microvibrissae. Macrovibrissae are the larger set of whiskers that are motile, and are mirrored on both sides of the rodent's snout. Each side consists of approximitly 30 whiskers with the rostral (nearer to the nose) whiskers being the shortest, and those in the caudal (nearer to the tail) direction, the longest. It has been suggested that this arrangement allows for the rat to cover a concave and convex shape during its whisking cycle (Towal, Quist, Gopal, Solomon, & Hartmann, 2011), which increases the variety of shapes that it can come into contact with. A whisk is defined as the cyclic movement of an individual or group of whiskers from a retracted (caudal) to a protracted (rostral) state. Whisking has been shown to be greatly beneficial for texture identification (Diamond & Arabzadeh, 2013) and general improvement in information gathering (Mitchinson & Prescott, 2013a).

For example, rats are known to be able to discern with great accuracy and precision the texture and shape of contacted surfaces (Diamond & Arabzadeh, 2013). Research has been carried out with regards to understanding the mechanisms that rats use to achieve such discernibility in identifying small variations in surface textures and object shapes (Brecht et al., 2011; Diamond & Arabzadeh, 2013; Lucianna, Albarracin, Vrech, Farfan, & Felice, 2016; Pammer et al., 2013; Towal et al., 2011). Rats are also known perceive these changes under varying conditions of whisker approach and contact speed relative to the surface (Huet, Rudnicki, & Hartmann, 2017), as well as changes in whisker-shape after experiencing whisker breakages (Zuo, Perkon, & Diamond, 2011).

Research has focused on determining the sensitivity of the neurons responsible for mechano-reception (Arabzadeh, Panzeri, & Diamond, 2006), the effect that whisker shape has on differentiating contact distances along the length of the shaft (Ahn & Kim, 2017; Lucianna et al., 2016), the effect of whisker-morphology on the range of

shapes that can be contacted (Towal et al., 2011) and the effect that whisking-control strategies have on the perfromance of whiskers in being able to operate in an effecient manner (Berg & Kleinfeld, 2003; Grant et al., 2009; Mitchinson & Prescott, 2013a).

2.3.1 Whisker Control

Observational studies (Grant et al., 2009) have shown that, following contact, rats control their whiskers to exhibit three unique strategies; Contact Induced Asymmetry (CIA), Rapid Cessation of Protraction (RCP) and Spread Reduction (SR).

CIA is described as the state in which the mean position of the right and left sided whiskers are not symmetrical along the center line of the rat's snout. The mean position refers to the average angular position of the whiskers during its whisking motion, thus group of whiskers retracting (caudal) and protracting (rostral) from -50 degrees to 50 degrees will have a mean position of 0 degrees. If the left sided whiskers where whisking instead between -50 to 0 degrees while the right sided whiskers ranged between 0 and 50 degrees, the mean positions of the left and right sided whiskers would be -25 and 25 degrees respectively. Their mean positions will, therefore, be non-symmetrical along the snout of the rat. This is typically observed when an obstacle is contacted while the head is at an angle to its surface, thus causing he whiskers to adapt their positioning so that as many contacts can be made while the head is in its current orientation.

RCP is the observation that rats stop protracting their whiskers following contact, specifically at the site in which the contact was made.

SR describes the observation that the range between protraction and retraction is reduced, on both sides, following the detection of contact. This change would cause all whiskers to make contact at a relatively similar time since their relative angular displacement will be reduced. It has been suggested that rat's whisker control exhibit Spread Reduction in order to maximize contact with an obstacle while and doing so in a manner that minimizes excessive bending of the whiskers and spreads out the contact time over a longer period (Grant et al., 2009). It can be observed from the high speed recordings found in the supplementary materials of (Mitchinson & Prescott, 2013b) that rats maintain their whiskers in an evenly distributed manner when targets are absent, and whisk at a range that is less than their maximum protraction and retraction limits. Once a contact is detected, the whiskers' mean position is shifted in the rostral direction and the whisking range is concentrated to an area closer to the protraction limit.

Together these responses have been summarized in a simple strategy referred to as Minimum Impingement Maximum Contact (MIMC) (Mitchinson & Prescott, 2013b).

The strategy of MIMC has been proposed to maximize the quality and quantity (number of contact points) of information gathered from the whisker array (Mitchinson & Prescott, 2013b; Pearson et al., 2011).

It must be noted that although the figures illustrate these observations in the overhead direction, rats exhibit this whisker-control behavior in both vertical and horizontal planes (Grant et al., 2009). This is possible since rats have many whisker controlling muscles that afford them a high degree of freedom (Towal et al., 2011) unlike this work's artificial whiskers that are limited to a single degree of freedom.

2.3.2 Whisker-Contact Localization

Maximizing the number of contacts is but one important aspect of improving the perceptual ability of whisker-tactile sensors. The other aspect is the quality of the extracted tactile features that include geometrical and textural properties of the contacted surface (Diamond & Arabzadeh, 2013). One large area of interest in whisker-tactile sensing is understanding the mechanism that rats use for estimating the radial-distance to contact, particularly for robotic applications since it would greatly improve the precision of contact-localization and thus benefit any object recognition tasks.

The radial-distance of contact refers to the point along the whisker-shaft, relative to its base, to where contact with a surface is made. The task is non-trivial since there are no receptors along the shaft and the estimation of radial-distance needs to be inferred based on the sensory feedback of the mechano-receptors located at the base of the whisker, inside the follicle.

Rats have been observed to accurately and precisely determine the radial distance of contact (Pammer et al., 2013). The authors of (Pammer et al., 2013) suggest that rats could potentially employ one or more of three strategies when estimating contact radial distance. Two of these strategies operate on the principle of triangulation that uses two depth varying whisker contacts. The third being the use of deflection magnitudes at the whisker follicle, which is based on the assumption that a whisker acts similar to a fixed cantilever beam. Several studies have focused on the latter strategy as rats have exhibited radial distance discrimination with a single whisker (Pammer et al., 2013). Studies such as (Birdwell et al., 2007; D. Kim & Möller, 2007; Pammer et al., 2013) have assumed the knowledge of the whisker's material properties thus allowing them to use the Bernoulli–Euler equation to derive an equation relating radial-distance to contact.

The Bernoulli–Euler equation describes the deflection experienced by a beam of a particular material and shape, given a set of boundary conditions and applied force. For

(D. Kim & Möller, 2007) the whisker is modelled as a non-tapered cylindrical beam with one fixed end and one free end, while (Birdwell et al., 2007; Pammer et al., 2013) further consider the effects of tapering and include it in their radial-distance estimation models. (Birdwell et al., 2007) describes the process of deriving the radial-distance estimation model from the initial assumption of a straight cylindrical cantilever beam and for small whisker deflection angles ($< 14°$). Under these assumptions the curvature of the beam can be described by Equation 2.33 where curvature measures the extent at which a linear object deviates from a line and is equal to: $\kappa = 1/R$. R is the radius of the curvature and is illustrated in Figure 2.11 for a bent beam.

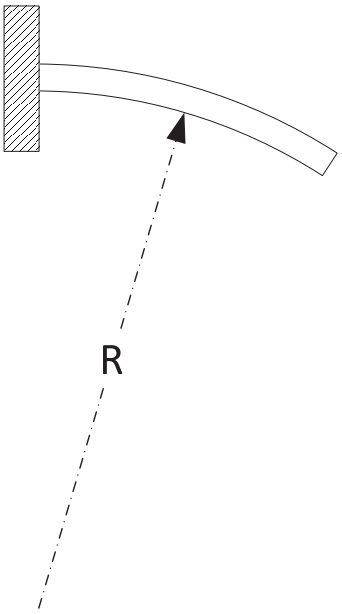
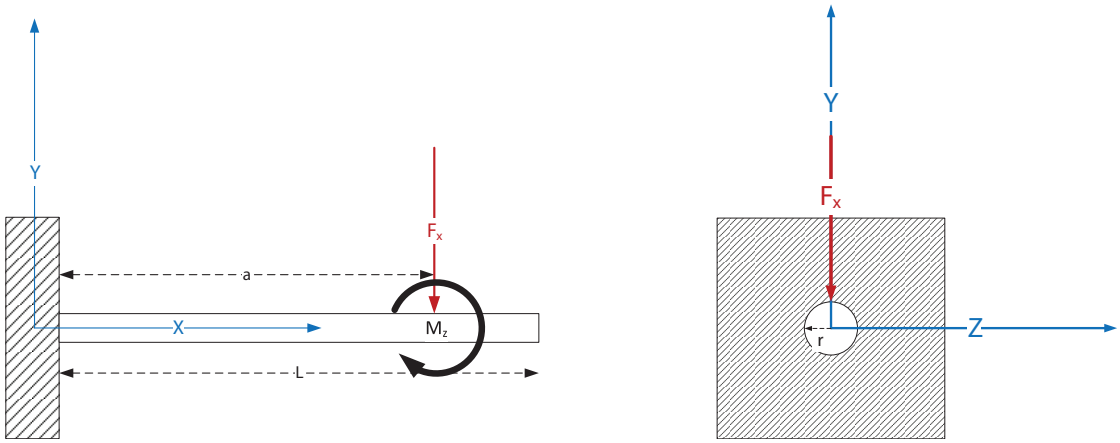


Figure 2.11: The curvature κ is equal to the inverse of the radius R of the curvature.

$$\kappa(x) = \frac{d^2y}{dx^2} = \frac{M(x)}{EI} \tag{2.33}$$

Curvature for a one fixed end one free end cantilever beam can be calculated given its E, which is the Young's modulus and I, the second moment of area or moment of inertia (Gere, 2004). Young's modulus is a value describing how much a material deforms given an application of force applied along its longitudinal axis. The second moment of inertia describes the distribution of material about a particular axis (Gere, 2004). Given the example cylindrical whisker shown in Figure 2.12, where the applied force is parallel to the x-axis and results in a moment about the z-axis, the second moment of area needed for the calculation of curvature would also need to be derived along the z axis. For a

cylindrical cantilever beam, the second moment of area is $I_z = \frac{\pi}{2}r^4$, where r is the radius of the cylinder.



(a) (b)

Figure 2.12: The curvature of a whisker due to a bending moment can be approximated by modelling it as a cantilever beam with a cylindrical shape that is fixed on one end and free on the other. The figure highlights the symbols used in the text and what they represent, including the length of the whisker L, the distance from the base of the whisker a at which a force F_x is applied. The subscript in F_x denotes the axis along which the force acts along, which in this case is the x-axis. The moment $M(x)_z$ is a rotating force that is experiences at a particular location x along the beam's length, and its direction is about the z-axis. The point a at which the linear force F_x acts, the moment can be calculated as $M(a)_z = F_x \times a$. Since this Moment acts along the z-axis, the resulting compressive and tensile forces will act parallel to it. To determine the extent at which the beam would deform, the moment of inertia I_z about the z-axis would need to be calculated. Looking at Figure 2.12b it can be seen that the moment of inertia for this beam about the z-axis is that of a circle, which can be worked out to $I_z = \frac{\pi}{2}r^4$ (Birdwell et al., 2007; Gere, 2004).

If the force is applied at a distance a from the base of the whisker, the moment at that point would be equal to $M(a) = F \times a$, and in general the moment experienced at any point along the whisker is given by equation 2.34. Given a straight beam, the curvature is naught beyond point a due to a lack of turning moment which can be seen in 2.34.

$$M(x) = \begin{cases} F(a-x), & 0 \leq x \leq a \\ 0, & a \leq x \leq L \end{cases} \tag{2.34}$$

By integrating Equation 2.33 twice to derive an expression for $y(x)$ which is the

vertical displacement of the beam at a horizontal point x, and using the boundary conditions $\frac{dy}{dx} = 0$ when $x = 0$ and $y(x) = 0$ when $x = 0$ since at point $x = 0$ the beam is fixed, (Birdwell et al., 2007) obtains Equation 2.35.

$$y(x) = \begin{cases} \frac{F}{6EI}(3x^2a - x^3), & x \leq a \\ \frac{F}{6EI}(3a^2x - a^3), & x \geq a \end{cases} \tag{2.35}$$

To consider the effects of a tapered whisker (Birdwell et al., 2007) substitutes the second moment of area for a circle $I = \frac{\pi r^4}{4}$ into that of a tapered whisker. The tapered whisker is modeled as a linearly tapered cone and its radius varies with respect to horizontal distance according to the following equation: $r(x) = r_{base}(1 - \frac{x}{L})$. The second moment of area for a tapered whisker is thus found to be equal to Equation 2.36.

$$I = \frac{\pi}{4}(\frac{r_{base}}{L})^4(L - x)^4 \tag{2.36}$$

Using the same procedure for deriving Equation 2.35, the vertical displacement of a cone shaped cantilever beam is found to be Equation 2.37.

$$y(x) = \begin{cases} \frac{2FLx^2}{3E\pi r_{base}^4}(\frac{3La - Lx - 2ax}{(L-x)^2}), & x \leq a \\ \frac{2FLa^2}{3E\pi r_{base}^4}(\frac{3Lx - La - 2ax}{(L-a)^2}), & x \geq a \end{cases} \tag{2.37}$$

Substituting Equation 2.36 and 2.34 into Equation 2.33 (Birdwell et al., 2007) shows that under the assumption of small angle deflections an expression relating radial-distance to contact, shown in Equation 2.38, can be derived and found to be a function of rate of change of turning moment with respect to the rate of change of whisker deflection angle.

$$d = \frac{3EI_{base}L_{BT}}{3EI_{base} + \frac{dM}{d\theta}L_{BT}} \tag{2.38}$$

Note that L_{BT} is the length of the whisker from its base to its tip when straight and I_{base} is the second moment of area of the whisker's base, which is a circle and $I = \frac{\pi r^4}{4}$.

Authors of (Evans et al., 2013) on the other hand use a feature based classifier to relate deflection magnitude and period to a radial-distance estimate. Figure 2.13 shows the deflection response when radial distance and contact speed are varied. Unlike the equation based methodology, the classifier does not need explicit knowledge of the whisker parameters, such as second moment of area or Young's modulus (Evans et al., 2013). The second moment of area describes the spread of an objects shape about

a specific axis and the Young's modulus describes how much a material is prone to displacement given a set amount of applied force.

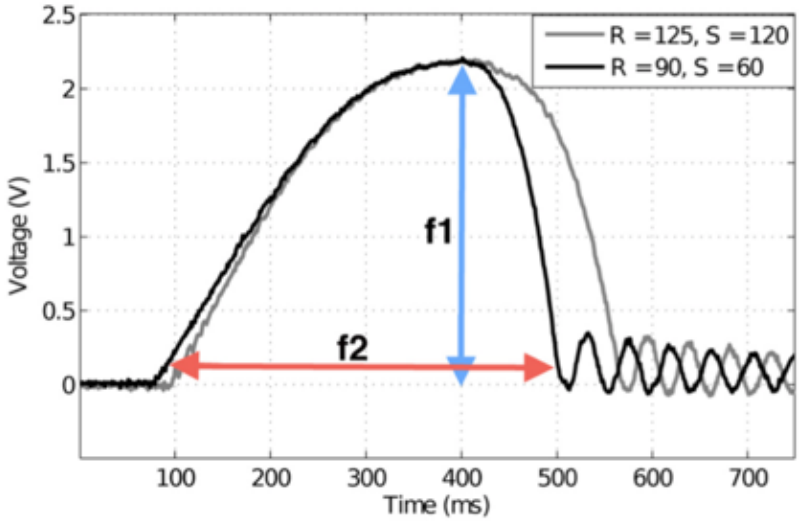


Figure 2.13: Magnitude of deflection, or force, has been used previously as a discriminator of radial distance to contact. Here the two traces are at different radial distances (R), measured in millimeters, but create the same magnitude of deflection. Speed (S) measured in millimeters per second. Colored arrows indicate how the extracted features for classification are measured. Peak deflection magnitude (f1) and contact duration (f2) are used to discriminate radial distance to contact and contact speed, respectively. Figure copied from (Evans et al., 2013) with permission.

The latest research has shown that this could be achieved by measuring the axial force and bending moments experienced at the base of the whisker at any time during contact (Huet et al., 2017). The simulation results indicated that a neural network trained to map the three forces to radial distance achieves an error of no more than 1.5% of the whisker length which is slightly lower than previous attempts at radial-distance estimation 1.65% (Evans et al., 2013). Unfortunately the work of (Huet et al., 2017) cannot be applied to this work's hardware set up as the sensory system lacks the capability of directly measuring the axial force experienced at the base of the whisker.

Similar to the work of (Huet et al., 2017) and (Evans et al., 2013), this work's whiskers were also designed with a linear taper, as it has been shown that for increasing values of radial-distance to contact, a tapered whisker would deflect to a greater degree than a non-tapered whisker. This degree of change would result in a higher variance of forces experienced at the base of the whisker for a unique radial-distance value and thus allow for better discenerability (Ahn & Kim, 2017; Pammer et al., 2013; Williams & Kramer,

2010).

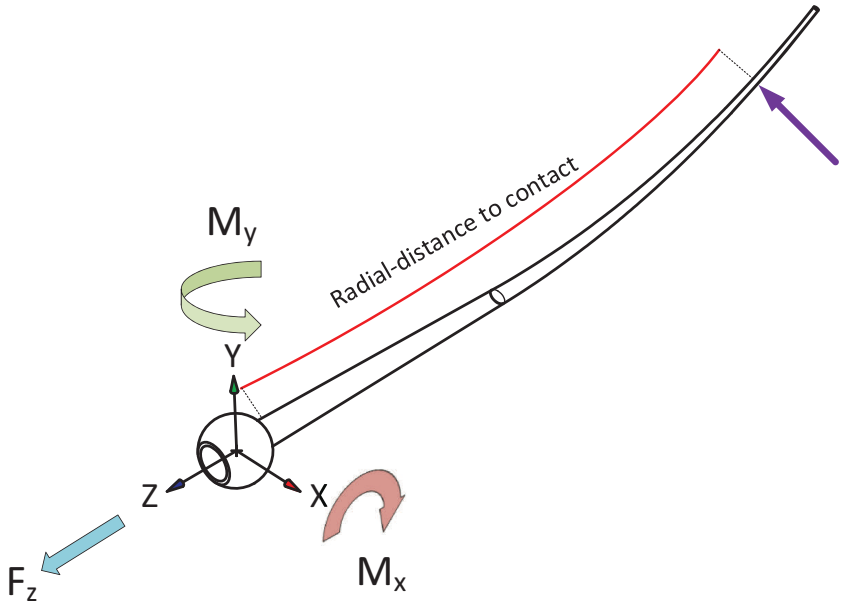


Figure 2.14: When a whisker makes contact and is bent in a particular direction, there is an induced set of forces at its base. The figure illustrates a whisker that experiences a deflection in a predominately Y direction. The contact is illustrated by the arrow near the tip of the whisker and illustrates the direction of the force produced by the contact. The point along the whisker at which the contact was made can be estimated using the values of the linear force acting along the$x$axis, F_x, the moment acting around the$x$axis, M_x, and the moment acting around the Y axis, M_y. This distance is referred to as the radial-distance to contact and is measured from the base of the whisker-shaft to the point of contact along the arc of the whisker-shaft. The coordinate frame illustrated in this figure is specific to this work's system and sensory set up and its orientation is different from that in (Huet et al., 2017). The model used to generate this image was obtained with permission from work of Pearson et al. (2013).

The two pieces of work are, however, different in terms of the sensory data that they use as well as the sampling rate that is possible. The work of (Huet et al., 2017) for example requires sensory data that includes measurement of the axial force (F_z) and bending moment at the whisker's base (M_x and M_y), which are described more clearly in Figure 2.14. The two values can then be used to train a regression model to map to an appropriate radial-distance value that is extractable at any point during the contact and is therefore mainly limited to the sampling rate of the sensory system. The work of (Evans et al., 2013) instead presents a regression model that maps he amplitude and period of the deflection signal (requiring sensors that measure F_y and

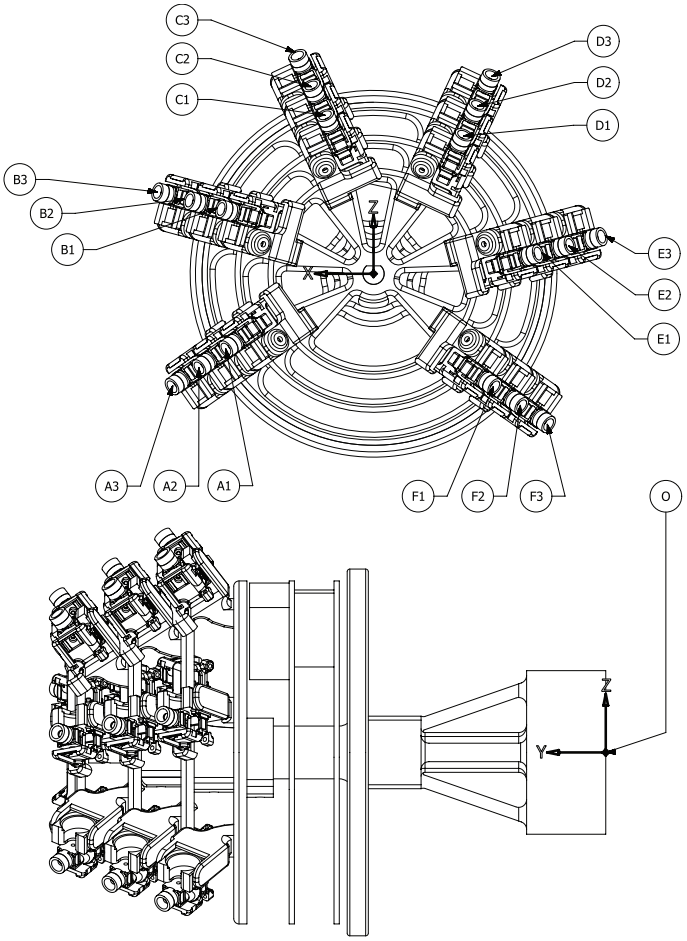
F_x) experienced after a single whisking cycle to an appropriate radial-distance value and is therefore restricted to extracting radial-distance estimates at a rate lower than the whisking frequency. Both the regression models require the training to take place for a whisker with a unique shape and material property. Chapter 4 includes a description of the work carried out to design a regression model that aims to generalize to whiskers of varying length while operating over a range of varying whisker dynamics.

The final whisker-contact position may be worked out by using forward kinematics of the arm and sensor-array. Figure 2.15 on page 40 illustrates the positioning of each whisker module in 3D space and contains the exact coordinates of each whisker at their respective pivot points. The whisker contact angle can be determined by observing the angle of the whisker at the estimated time of contact, which as explained previously, is the time where the whisker-deflection signal begins to accelerate. Having knowledge of the whisker-angle and sensor array pose, the contact position can be narrowed to lying somewhere along the length of the whisker. Using an appropriate radial-distance estimation method the contact position estimate can be narrowed down to a more precise location in space. Having a better estimation of contact position would allow for a higher quality reconstruction of shape, which would in turn improve the performance of object recognition related tasks.

2.3.3 Object Recognition

Object recognition based studies using whiskered robotic platforms, in comparison to cutenous-tactile sensing platforms, have been relatively lacking. One study (Russell & Wijaya, 2005) involved using passive whiskers for determining the contours of an object. Features such as lines and corners were extracted based on the sequence of contact points which were used to infer the shape of the object, based on a database of known objects. A mobile robot with multiple active whiskers is used in (D. Kim & Möller, 2007) to determine object shape features including lateral and vertical shape and very elegantly, slope of an objects surface. The authors have suggested that a surface's slope may be inferred from the slope of the vertical and horizontal deflection sensors, indicating a linear relationship between sensory data slope and that of the contacted surface, as can be seen in Figure 2.16.

The work of (D. Kim & Möller, 2007) presents some interesting methods for inferring surface features, which is particularly useful since it provides an idea of what data is relevant to designing an effective classifier for object identification.



Label	X (mm)	Y (mm)	Z (mm)
O	0	0	0
A1	48.752	201.860	-23.455
A2	58.967	181.903	-30.607
A3	69.181	161.946	-37.760
B1	49.305	201.860	22.270
B2	61.350	181.903	25.497
B3	73.394	161.946	28.725
C1	14.633	201.860	52.084
C2	19.903	181.903	63.386
C3	25.173	161.946	74.687
D1	-30.493	201.860	44.689
D2	-35.763	181.903	55.990
D3	-41.033	161.946	67.291
E1	-53.834	201.860	5.366
E2	-65.879	181.903	8.594
E3	-77.924	161.946	11.821
F1	-38.715	201.860	-37.790
F2	-48.929	181.903	-44.942
F3	-59.144	161.946	-52.095

Figure 2.15: To calculate the position of a contact point at a specific whisker, forward kinematics is used. This figure illustrates the position of each whisker relative to the array-arm mounting adapter (O). The measurement is made from the adapter to the pivot point from which the whisker rotates. The model used to generate this image was obtained with permission from work of Pearson et al. (2013).

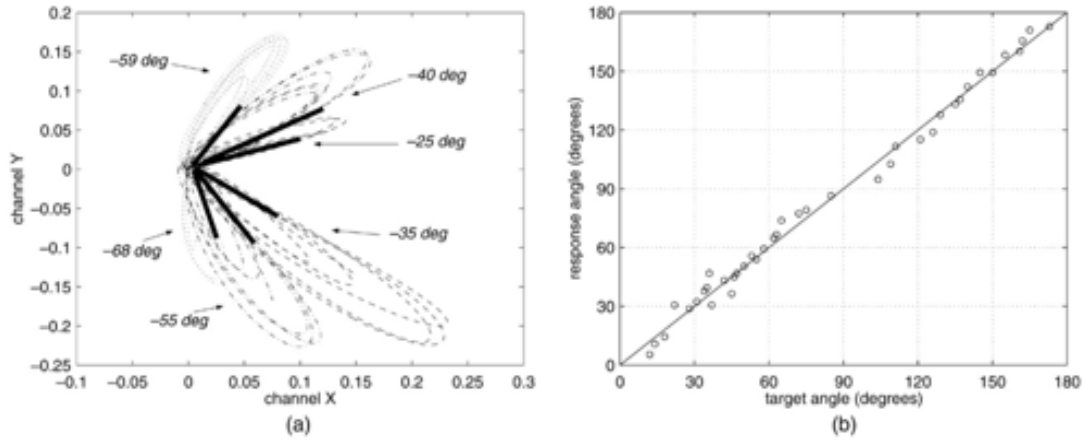


Figure 2.16: X–Y plot of deflection signals (a) examples of X–Y plot over two channels with several slope tests (thick lines: estimated slope) (b) estimation of slope with X–Y channels (deg: real data, solid line: theory) figure copied from (D. Kim & Möller, 2007) with permission.

2.3.4 Texture Identification

In addition to geometrical features, whiskers are also able to extract textural information from a contacted surface. Much research has been made in the area of texture identification using whiskers (Diamond & Arabzadeh, 2013; Fend, Bovet, Yokoi, & Pfeifer, 2003; Jadhav & Feldman, 2010; Lottem & Azouz, 2009; Zuo et al., 2011), however, all studies have yet to conclude the mechanism by which texture invariance can be achieved (Diamond & Arabzadeh, 2013). Texture invariance being the ability to re-identify a texture irrespective of the contact condition, that is irrespective of contact speed or angle. The problem arises from the fact that contact signature properties greatly vary with contact conditions (Diamond & Arabzadeh, 2013). Contact signatures include the temporal profile of whisker motion and whisker deflection velocity and spectral composition.

Some progress has been made with regards to designing texture classifiers that exhibit good performance, such as in (C. W. Fox, Mitchinson, Pearson, Pipe, & Prescott, 2009). The authors of the paper have developed a 6-dimensional classifier that is able to differentiate 4 textures, at varying contact conditions, with a performance of $72 \pm 3\%$ correct classifications. When identifying just two, rough and smooth, textures the classifier performed much better at a correct classification of $91 \pm 2\%$.

The 6 dimensional classifier described in (C. W. Fox et al., 2009) would serve as a good starting point for implementing texture identification on this project's system.

However, in order to avoid over complicating the investigation this work focuses on extracting high quality geometrical based features to improve our system's place recognition precision and leaves texture identification as part of future work. It is speculated that the combination of geometrical and textural features for characterizing different surface regions would serve to reduce the system's likelihood of making false matches.

2.4 Exploration Strategy

Successful navigation hinges upon a good perception of the environment, which is why the initial work focuses on improving the work-life of the whiskers and the precision of their contact-localization estimates. Extending the work-life of whiskers would allow us to maintain a high number of contacts during whisker-exploration, while improving contact-localization precision would allow us to better discriminate different regions across the terrain. Another important factor in navigation is the exploration strategy. Given the eventual desire for autonomous navigation that efficiently maps a terrain, Active Curios Exploration (ACE) would serve as a good strategy (Gordon, Fonio, & Ahissar, 2014).

ACE describes the behavior of exploring an environment by driving the attention of the agent towards areas of high novelty. Prior to implementing ACE, a model is needed to describe what influences the attention of an agent, as well as the appropriate response based on the location of the target. A study that implements such an attention seeking model is described in (Mitchinson & Prescott, 2013b). The study includes the design and simulation of a computational model that attempts to explain the observed behavior in rats when whisking. In order to implement ACE the model would only need to be adapted by including excitation from areas that have yet to be explored. An unexplored area can be indicated by an open border, the location of which can serve as a point of excitation that will compete for the agents attention with other ego-centric points of excitation.

2.5 Movement Through Higher Configuration Space

We have previously eluded to the limitations of whisker tactile sensing over electromagnetic sensing such as sampling density, resolution, sample rate and range. Tactile sensors, be they cutaneous or whiskered, must therefore be actively moved across the surface of an object to sample it effectively. This will typically demand movement outside of a simple 2D planar space, however, RatSLAM does not currently accommodate higher

dimensional state estimates. The work of (Pearson et al., 2013) also does not work in higher dimensions and instead reduces the contact points obtained by the whisker-array to a 2D projection and carries out localization in a 2D plane.

In regards to this work, the desire is to sample as many surface regions as possible, which would require manipulation of the sensor array in higher spatial dimensions. By sampling more surface regions the system would better suited to perceive higher level features such as 3-dimensional shape.

In preparation for exploring a more complex environment the RatSLAM algorithm would at least need to be adapted to consider changes in height. There have been numerous studies trying to determine how an animal's hippocampus interprets positioning in a 3D world. One study (Yartsev & Ulanovsky, 2013) involving bats suggests place cells being mapped to a volumetric space, thus each place cell is associated with a unique location in 3D space. Unlike bats that move in 3D space, rats' movements are constrained to the floor. Studies regarding rats have shown that their navigation related cells (place, grid and head direction cells) are most likely mapped to a 2D plane that lies on the environmental surface (floor) (Hayman, Casali, Wilson, & Jeffery, 2015; Jeffery, Jovalekic, Verriotis, & Hayman, 2013) but instead include multiple mapped plannar spaces that are diffrentiated according to a non-metric term. These multi-planar maps are called bi-coded maps and have been suggested by Jeffery et. al as a likely method in which rats and other 3-dimensional animals use to map their environment (Jeffery et al., 2013).

Thus far a description of the robotic platform has been given, along with the potential strategies that are inspired from both neuroscience and robotics research that can help improve the perception capabilities of the whisker-tactile sensor array. The proceeding chapters describe the implementation of whisker control strategies that improve the longevity of a whisker's life and the precision of contact localization estimates. Further, work regarding the implementation of a regression model for improving radial-distance estimates in a more practical robotic setting is presented. Combining the progress up to this point, a novel whisker object-recognition system will be presented, one which is able to extract the higher level environmental features including object shape. The system's operation is possible on account of it's ability to accommodate movement through higher-degrees of space, which is similar to how 3-dimensional dwelling animals (Yartsev & Ulanovsky, 2013) are proposed to operate.

To operate in 3-dimensional space, our work takes advantage of appropriate feature descriptors such as the point feature histogram to identify specific regions on a 3d surface. Further, we utilize quaternions for the purpose of applying 3-dimensional rotations.

2.5.1 Working in 3-dimensions

2.5.1.1 Point Feature Histogram

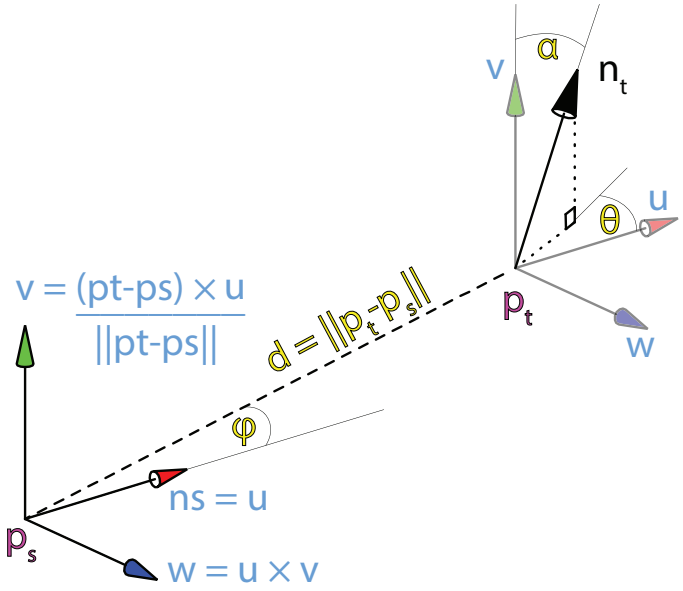


Figure 2.17: The point feature histogram is a function of the 4 features that describe the relative orientation and translation of one points coordinate frame (p_s) and the normal vector of its paired point (p_t). The features are the α, θ and ϕ angles, as well as the relative distance between the two points d.

PFH is a statistical representation of a point cloud distribution i.e. a set of points distributed in 3-dimensional space, which can be used to measure similarity in manner that is robust to noise, occlusion and point cloud resolution (Wahl, Hillenbrand, & Hirzinger, 2003). The authors of (Guo et al., 2016) perform evaluations of several 3D feature descriptors and point out the benefits that the fast point feature histogram (FPFH) has in terms of being ideal and computationally efficient for data-sets that include a low number of contact points. FPFH is an optimized variation of the point feature histogram (PFH) and is intended to operate on data sets that include higher resolution point clouds (Rusu, Blodow, & Beetz, 2009), much higher than this work's 18 contact points. For this reason PFH was opted for so as to retain as much of the original algorithms discriminating ability (Rusu et al., 2009).

A point feature histogram (PFH) is a statistical signature describing the geometrical distribution of contact points relative to each other in the form of a 1-dimensional vector. PFH is commonly used as a 3D feature descriptor that identifies common 3D

locations between misaligned point cloud data-sets (Rusu, Marton, Blodow, & Beetz, 2008), however, in the work of (Wahl et al., 2003) it has been shown that PFH performs well in object recognition tasks as well.

The point feature histogram generation algorithm is based on the work of (Rusu et al., 2008), with few modification that are pointed out in the text below.

The algorithm initial step includes iterating through all the points within the point cloud and assigns each one a normal vector that is determined based on the distribution of neighbouring points. Thus, for point p_i, any point falling within a spherical volume defined by the radius r_α, belongs to the set Pk_α. The normal vector n_i is derived by calculating the first principal component of the points in Pk_α. The principal component is calculated by obtaining the eigenvalues and eigenvectors of the covariance matrix of the 3D points. The eigenvector corresponding to the largest eigenvalue, and thus first principal component, is selected as the direction of the normal vector of point p_i.

It is observed that the contact points generated from the whisker-array following a single whisk-sample are sparse (maximum of 18 points) and are generally spread out due to the morphology of the whisker-array. Given these characteristics, it would be difficult to ensure that a minimum of 3 points (which is the minimum number of points required to define a plane and thus be able to extract a normal vector) are located within the Pk_α set for a particular point p_i. The selection criteria is therefore modified to being the three closest points as opposed to all points that fall within the volume defined by r_α.

When calculating the eigenvectors of a given matrix, the sign of the normal vector is ambiguous. Reminding the reader that the standard definition of the eigenvalues and eigenvectors for a square matrix A is:

$$A\mathbf{v} = \lambda\mathbf{v} \tag{2.39}$$

Where $\mathbf{v}$ is an eigenvector and λ is an eigenvalue. Thus, the solution still holds whether sign of the eigenvector $\mathbf{v}$ is positive or negative. The ambiguity of the normal vector's sign presents a problem when comparing sets of points from the same region.

Taking this into consideration, measures need to be taken so as to ensure that the normal vectors are orientated in a consistent manner. This is done by exploiting the fact that each sensor-array point cloud sample is extracted from a single point of view i.e. the estimated contact point locations are all relative to the end-effector's coordinate frame. In this work, the viewpoint is the coordinate of the origin $(0, 0, 0)$. If the normal vector is pointed away from the point of origin, the sign is flipped, thus maintaining a consistent

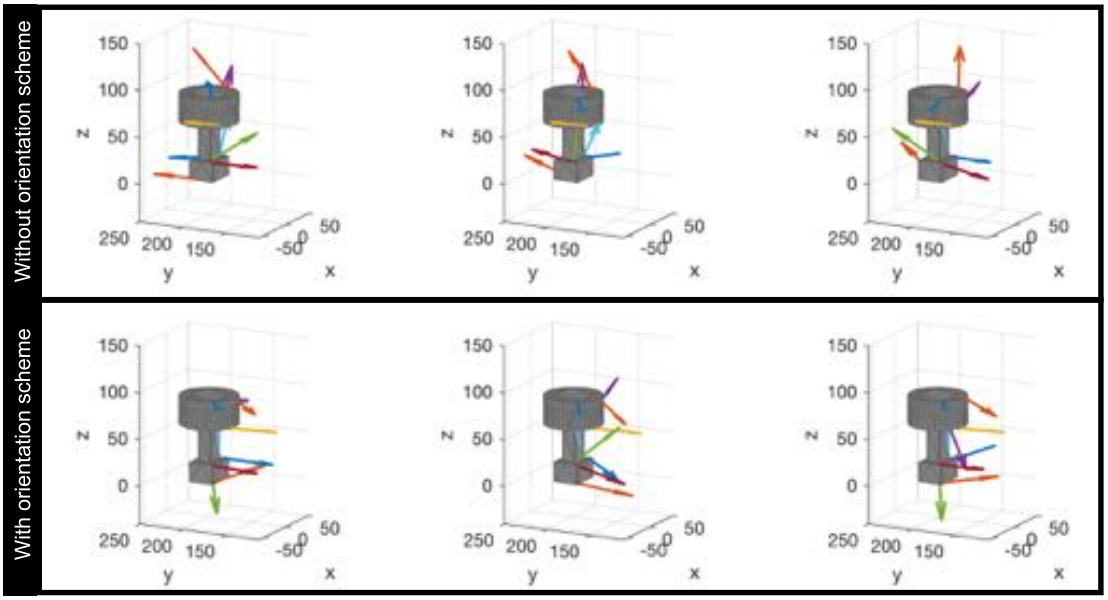
normal vector orientation for all the whisk-samples. This condition is summarized in Equation 2.40.

$$\frac{(v - p_i) \cdot n_i}{\|v - p_i\|} < 0, \text{then} \quad n_i = -n_i \tag{2.40}$$

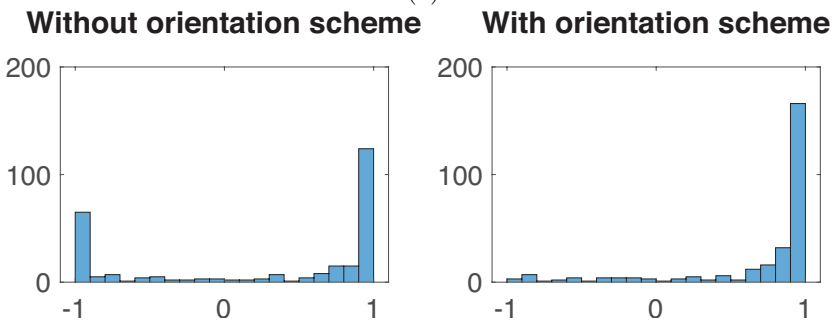
Figure 2.18a visualizes the effect of the normal vector orientation scheme, where the top row illustrates the arrangement of normal vectors without any orientation scheme and the bottom row illustrating the arrangement of normal vectors following the suggested orientation scheme. For the sake of clarity, the normal vectors are processed for a fraction of the points on the right half of the 3D model. Each column in the figure represents the state of the normal vectors after adding some noise to the positions of the mesh vertices, in imitation of how whisk-samples would be subject to noise every time a point cloud is generated. It can be seen that in comparison to the first column, the sans orientation scheme results in normal vectors changing their signs; the blue, red and green vectors change signs even though the relative distribution of mesh vertices has not changed significantly. When the orientation scheme is used, the same vectors keep the same sign and are more similar to the normals in the first column in comparison to the sans orientation scheme scenario. This advantage is highlighted even more clearly in Figure 2.18b where the histogram shows the values obtained when calculating the dot product of the normal vectors from all the mesh's vertices in the first column with that of the remaining columns. A dot product closer to 1 indicates high similarity where as a dot product closer to -1 indicates least similarity. It can be seen that the orientation scheme makes it more likely to observe a similarly orientated normal vector for a given surface region when the positioning of the vertices are corrupted by noise.

Once all normal vectors have been assigned, the algorithm iterates through all unique combinations of neighbouring point pairs. The set of neighbours P_{k_β}, is determined based on those points that fall within a spherical volume defined by the radius r_β surrounding a specific point p_i. The value k_β is the total number of points within the volume, including the subject point p_i. The combinations of point pairs p_{j_1} and p_{j_2} are bound by the conditions: $j_1 < k_\beta$, $j_1 \neq j_2$ and $j_2 < j_1$.

For each point pairs p_{j_1} and p_{j_2}, the algorithm assigns one of them to be the source point p_s. The source point will then be defined a Darboux frame from which its paired point, now dubbed the target point p_t, would have its normal vector n_t measured against.



(a)



(b)

Figure 2.18: The consequence of implementing a consistent normal vector orientation scheme. Figure 2.18a includes a visualization of normal vectors for when the normal vector orientation scheme is either used (bottom row) or not (top row). For the sake of brevity only a few normal vectors for each specific mesh vertex is shown. The first column shows the normal vectors belonging to a specific set of mesh vertices before any position noise is introduced, while the following columns show the changes in the calculated normal vectors for the same set of vertices. Figure 2.18b plots two histograms that quantitatively show that the normal vectors generated for the same 3D model albeit with noise added to the positions of its vertices, are more likely to be similar when using the orientation scheme. When a vector is closer in similarity and value of 1 is returned while a vector that is maximally dissimilar i.e its sign is flipped, the value returned is -1. It can be seen that without the normal vector orientation scheme the normal vectors that are calculated following the introduction of noise to the mesh vertices are less likely to be similar (they are therefore less positive). When the normal orientation scheme is used, it is more likely to obtain a normal vector that is more similar to the original normal vectors before noise is introduced to the mesh vertices.

The condition under which the source point is selected is defined in Equation 2.41.

$$\begin{aligned} \text{if} \quad & (n_{j_1} \cdot (p_{j_2} - p_{j_1})) \leq (n_{j_2} \cdot (p_{j_1} - p_{j_2})) \\ \textit{then} \quad & p_s = p_{j_1}, \quad n_s = n_{j_1} \quad p_t = p_{j_2}, \quad n_t = n_{j_2} \\ \textit{else} \quad & p_s = p_{j_2}, \quad n_s = n_{j_2} \quad p_t = p_{j_1}, \quad n_t = n_{j_1} \end{aligned} \tag{2.41}$$

The frame can then be defined according to Equation 2.42, which includes how each of the 3 orthogonal unit vectors that define the pose of the frame can be worked out. Following the frames definition, the four point feature histogram features for the pair of points can be worked out. The four features are illustrated in Figure 2.17 on page 44 and include 3 angular values (α, θ, ϕ) describing the orientation of the target point's normal vector n_t against the source's Darboux Frame (u, v, w), and one linear distance δ between the points. Equation 2.43 includes the formulas used to calculate these four features, noting that $\|\|_2$ refers to the Euclidean norm.

$$\begin{aligned} u &= n_s \\ v &= \frac{(p_t - p_s) \times u}{\delta} \\ w &= u \times v \end{aligned} \tag{2.42}$$

$$\begin{aligned} \alpha &= v \cdot n_t \\ \theta &= atan2(w \cdot n_t, u \cdot n_t) \\ \phi &= u \cdot \frac{(p_t - p_s)}{d} \\ \delta &= \|p_t - p_s\|_2 \end{aligned} \tag{2.43}$$

The values are then stored within a histogram whose dimensions are determined by the user. δ corresponds to the number of divisions each feature should be discretized to. For example, given a scenario where there is 1 feature, its value normalized to its max-min range will vary between 0 and 1, and given a value of $d = 3$, values between $0 - \frac{1}{3}$ will be represented with an index of 1, $\frac{1}{3} - \frac{2}{3}$ with an index of 2, and $\frac{2}{3} - 1$ with an index of 3. Thus if there were 4 features, like is the case with the point feature histogram, it would require d^4 bins to represent all possible combinations of 4 features, each with a division of d.

The bin index that is to be incremented can be worked out according to Equation 2.44. The the scalar value f_i refers to that of one of the four features $(\alpha, \theta, \phi, \delta)$ and

$f_{i_{max}}$ and $f_{i_{min}}$ are their maximum and minimum possible values. The $[\cdot]$ operator refers to an integer operation where contained value is rounded up to the nearest whole value.

$$idx = \sum_{i=0}^{i\leq 3} \left[\frac{f_i \cdot (d-1)}{f_{i_{max}} - f_{i_{min}}} \right] \cdot d^i \tag{2.44}$$

Thus, given a histogram for each P_{k_β} set of points, with d^4 bins, the bin number corresponding to idx is incremented by 1. Once all increment operations are complete, each of the bins is normalized according to the total number of unique pair combinations, which is equal to $(k_\beta \cdot \frac{(k_\beta+1)}{2})$.

For this work the P_{k_β} radius is set to $r_\beta = 0.3\ m$, which would result in all whisker-contacts being considered. One histogram is therefore generated for each whisk-sample.

To measure the similarity of one point feature histogram with that of another, a chi-squared divergence measure is used and is calculated according to the equation 2.45 (Hetzel, Leibe, Levi, & Schiele, 2001).

$$\chi^2(\mathbf{q}, \mathbf{v}) = \sum_i \frac{(q_i - v_i)^2}{(q_i + v_i)} \tag{2.45}$$

Where $\mathbf{q}$ and $\mathbf{v}$ are the pair of histogram vectors that are to be compared, and q_i and v_i are their respective i^{th} element.

2.5.1.2 Quaternions

Quaternions are a form of mathematical notation that is useful for representing 3-dimensional orientations as it provides means by which rotational operations can be performed without succumbing to gimbal lock (Jazar, 2010). Gimbal lock is used to describe a scenario where rotation applied along a specific axis is rendered useless due to its alignment with another axis, which is a common problem associated with alternative methods of rotational operations such as with Euler angles (Jazar, 2010).

One method of deriving the quaternion describing the orientation of a body is by equation 2.46. The vector $\boldsymbol{\theta}$ is a 4-dimensional axis-angle vector. An axis angle vector is another method of representing orientation, albeit a more intuitive one. However, although easier to visualize, axis angle representations lack quaternion's qualities that allow for faster calculations of rotational operations as well as being more suited for applying small rotations (Jazar, 2010).

An axis angle again may either be represented as a 4-dimensional vector, or a 3-dimensional one. For the purpose of this explanation the more intuitive 4-dimensional representation is used and an axis-angle vector $\boldsymbol{\theta} = [e_x, e_y, e_z, \theta]$, where the first three

elements represent the axis along which a point is rotated by θ radians. To convert the axis-angle to a 3-dimensional vector the vector is reduced by multiplying the magnitude of rotation θ to the unit vector consisting of the rotation axis, thus giving $[e_x, e_y, e_z, \theta] = [\theta e_x, \theta e_y, \theta e_z]$

$$\boldsymbol{q} = \begin{bmatrix} e_x sin(\theta/2) \\ e_y sin(\theta/2) \\ e_z sin(\theta/2) \\ cos(\theta/2) \end{bmatrix} \tag{2.46}$$

In order to rotate a body represented by a current orientation of $\mathbf{q} = [q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}]$ by a rotation represented by $\mathbf{p} = [p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}]$ a quaternion multiplication operation is used to give $\mathbf{r} = \mathbf{q} \times \mathbf{p} = [r_0 + r_1\mathbf{i} + r_2\mathbf{j} + r_3\mathbf{k}]$. Where the elements of $\mathbf{r}$ are derived using the equation 2.47. Note that in this example the $\times$ represents a quaternion multiplication and not a cross product (Jazar, 2010).

$$\begin{bmatrix} r_0 = q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ r_1 = (p_0q_1 + p_1q_0 - p_2q_3 + p_3q_2) \\ r_2 = (p_0q_2 + q_0p_2 + p_1q_3 - q_1p_3) \\ r_3 = (p_0q_3 - p_1q_2 + q_0p_3 + p_2q_1) \end{bmatrix} \tag{2.47}$$

2.5.1.3 Transformation matrices

Homogeneous transformation matrices are used for translational and rotational operations throughout this work. Transformation matrices that operate in a 3-dimensional environment take the form of a 4×4 matrix. One way to visualize transformation matrices is to use a moving frame that describes a particular pose for a given body. The term pose includes a reference to both the translational and rotational state of a body. Given a fixed world frame (small frame shown in Figure 2.19, a point's pose can be described by a transformation matrix with the form shown in Equation 2.48 and visualized in Figure 2.19a.

$$T^1_{ob} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.48}$$

The subscript of the transformation matrix specifies the frame from which the current one is being referenced from. In this case, the B refers to the current body's frame, and

O the origin frame. T_{OB} may be read as the transformation matrix describing the pose of the body in relation to the origin's frame. In a transformation matrix the three rows in the last column represent the x, y, z coordinates of the frame, while the 3×3 sub-matrix from columns 1 to 3 and rows 1 to 3, represent the orientation of the frame.

For example, when the initial body frame T^1_{OB} is rotated by 45° about the x-axis, a rotation matrix specific to a rotation about the x-axis, defined in Equation 2.49, is multiplied with the initial body frame T^1_{OB} so as to obtain the new transformation matrix T^2_{OB}. The resulting frame is shown in Figure 2.19b. To translate the T^2_{OB} frame along the z-axis, the z-axis specific translation matrix defined in Equation 2.54 is multiplied with T^2_{OB} to obtain T^3_{OB}. This final frame is shown in Figure 2.19c and the operations to reach this final form is shown in Equation 2.55.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.49}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.50}$$

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.51}$$

$$Tr_x(d) = \begin{bmatrix} 0 & 0 & 0 & d \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.52}$$

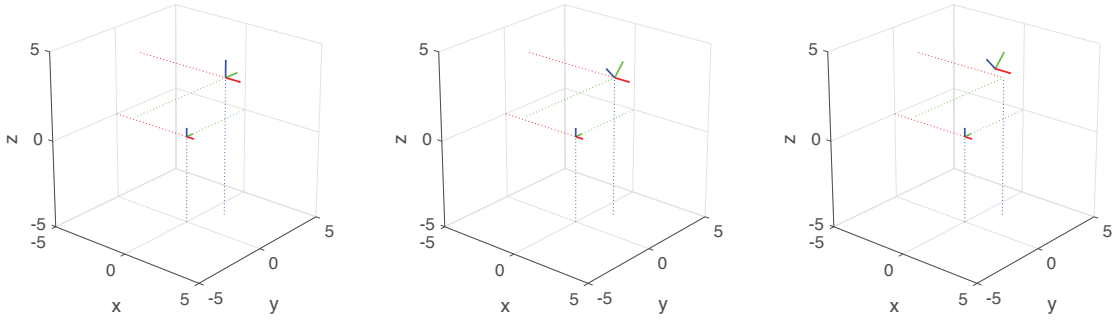
$$Tr_y(d) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.53}$$

$$Tr_z(d) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.54}$$

$$\begin{aligned} T^2_{OB} &= T^1_{OB} R_x(45°) \\ T^3_{OB} &= T^2_{OB} Tr_z(1) \end{aligned} \tag{2.55}$$

To return to a previous transformation, or perform a reverse operation, the inverse of the transformation matrix is multiplied with the desired frame. Equation 2.56 shows the operations needed to return from T^3_{OB} to T^1_{OB}.

$$\begin{aligned} T^2_{OB} &= T^3_{OB} Tr_z(1)^{-1} \\ T^1_{OB} &= T^2_{OB} R_x(45°)^{-1} \end{aligned} \tag{2.56}$$



(a) Initial body pose T^1_{OB}

(b) T^1_{OB} rotated 45° about x-axis results in T^2_{OB}

(c) T^2_{OB} translated 1 unit along z-axis to result in T^3_{OB}

Figure 2.19: Illustrations of a body's pose, which represented by a coordinate frame, undergoing transformation through the multiplication of appropriate transformation matrices.

Rotation matrices specific to other axis are included in Equations 2.50 and 2.51 which represent rotations in the y and z axis respectively. Translations along the x and y axis are included in Equations 2.52 and 2.53 (Craig, 2009).

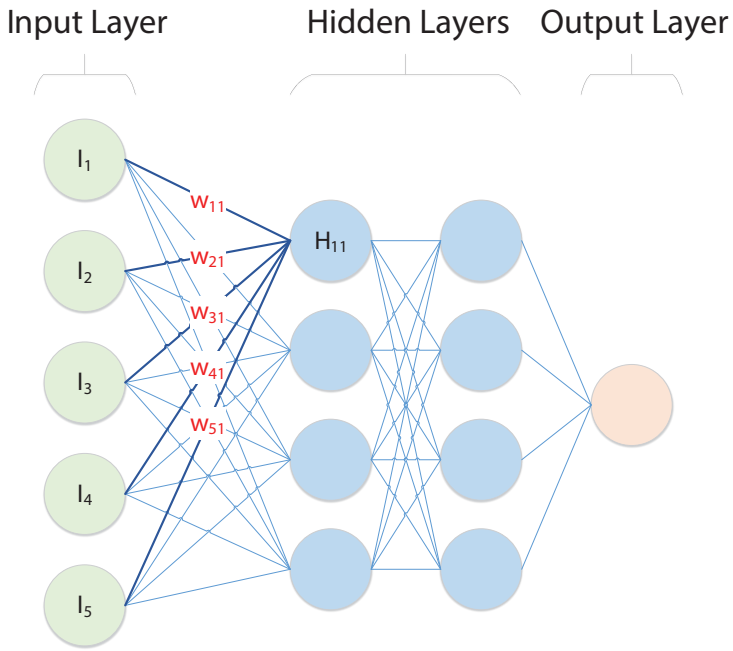


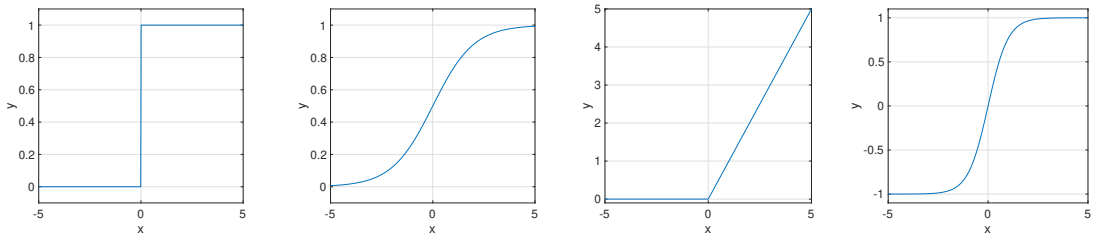
Figure 2.20: An example of a multilayer perceptron with a 5 input 1 output network. In this particular example the sum of the weighted input from each input node $w_i I_i$ defines whether a particular hidden layer node H is activated. In the case of node $H_1 1$ and a binary threshold activation function, its state will be active given $\sum_{i=1}^{5} w_i I_i > 0$.

2.6 Regression techniques

2.6.1 Multilayer Perceptron

The multilayer perceptron refers to uni-directional networks made up of artificial neurons. MLP's tend to be limited to feedforward networks, which means that the data that is input is passed on forward from the input layer, through the hidden layer and finally to the output layer (Negnevitsky & Intelligence, 2005). Figure 2.20 shows an example of such a network and includes 5 input neurons, two hidden layers with 4 neurons in each layer, and finally a single output neuron.

Each neuron consists of its set of input connections, activation function, and output. Considering the first hidden layers neuron H_{11}, its output would be a function of the weighted sum of all inputs. For a simple perceptron where a binary function is used the output would be equal to 1 provided a positive sum of weighted inputs, and 0 for a negative one. This calculation is summarized in Equation 2.57. Alternative activation functions include sigmoid, tanh and ReLu, all of which are plotted in Figure 2.21.



(a) Binary Threshold (b) Sigmoid (c) ReLu (d) TanH

Figure 2.21: Examples of different activation functions. Given an input whose value corresponds to a particular x-axis value, a node's activity will be equal to the corresponding y-axis value.

$$H_{11} = \sum_{i=1}^{5} w_i I_i = \begin{cases} 1 & \text{if } H_{11} > 0 \\ 0 & otherwise \end{cases} \tag{2.57}$$

MLP's are trained by presenting the network with a series of training data sets where the error for each training sample is used to adjust the weights of the network connections. Thus, given a loss function that measures error of the network i.e. the error between the target output t and the network output y, an optimization problem can be formulated such that the function $L(t, y)$ is minimized for a given set of weights $w_{i...n}$.

For a simple one-input-one-output example, a gradient descent algorithm can be used for adjusting the weights of the network to learn the underlying function from the training data. The algorithm calculates the derivative of the loss function for a given pair of weights and identifying, from the gradient, the direction in which the value would need to be adjusted so that a decrease in error can be achieved. Given many training samples the system should converge to a solution and the system would have learnt the underlying function.

For more complex networks, efficient back-propagation algorithm variations have been developed including those that use Newtons method for optimization and Levenberg-Marquardt method. Each variation have their strengths and weaknesses and for our work we limited the training algorithm to the Levenberg-Marquardt method using Bayesian regularization since research has shown that it is better suited for smaller data sets due to its consideration for the prevention of over-fitting (Zhang, Xu, & Zhou, 2010).

One of the major advantage of a neural network for the purpose of regression is its ability to approximate any smooth function given a finite number of hidden neurons (Murphy, 2012).

2.6.2 Support Vector Regression

Support vector machine based regression, or support vector regression (SVR) for short, is a form of supervised learning where a line of best fit is derived to describe the underlying function relating the input data x to the output y. The function is obtained by minimizing the cost function described in Equation 2.58 while subject to the boundary conditions described in Equation 2.59.

$$minimize\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(\xi_i^* + \xi_i) \tag{2.58}$$

$$subject\ to \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \epsilon + \xi_i^* \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \epsilon + \xi_i \end{cases} \tag{2.59}$$

In a simplified example where an input sample consists of a single dimension, Equation 2.58 includes a variable w that refers to the weight of a given input x_i, which defines the gradient of the line used to approximate the underlying function $f(x) = y$ in figure 2.22. The variable C is a penalty parameter that is used to adjust the influence of support vectors i.e. the points that exists outside of the margin. The margin is defined by the parameter ϵ and is included to avoid over-fitting the function to the data. A lower penalty parameter will result in a smoother curve while a higher penalty parameter will result in a sharper one.

The constrains defined in Equation 2.59 includes a dot product operator $\langle \cdot \rangle$ that is performed on the weight and input vectors. For an example such as the one shown in Figure 2.22 the dot product is used to approximate a linear function and intuitively, measures the similarity of the vector $\mathbf{w}$ and $\mathbf{x}_i$ in a feature space that assumes a linear relationship. To consider cases other than linear functions, support vector regression uses the 'kernel trick', which is a method that allows for the measurement of vector similarity in other feature spaces without the need for explicitly mapping the input to said feature space.

When a support vector machine is tasked with classifying (as opposed to regression) non-linearly separable data sets such as that shown in 2.23a, the data set would need to be mapped onto a feature space of higher dimensions. For example, by taking the inputs x and y, a third dimension z can be calculated according to $z = x^2 + y^2$, which would result in the data being linearly seperable. Figure 2.23b highlights in green the linear hyperplane that is capable of separating the two distinct classes. In the case of support vector regression the same logic holds true and a linear function defining the hyperplane can be derived that best fits otherwise non-linear data.

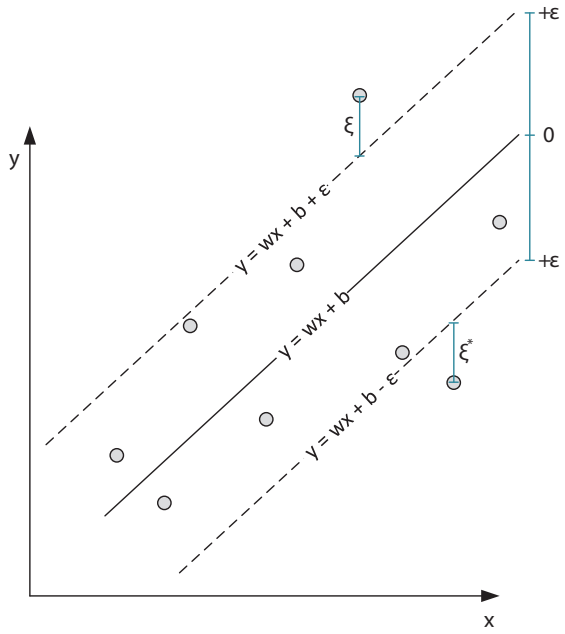
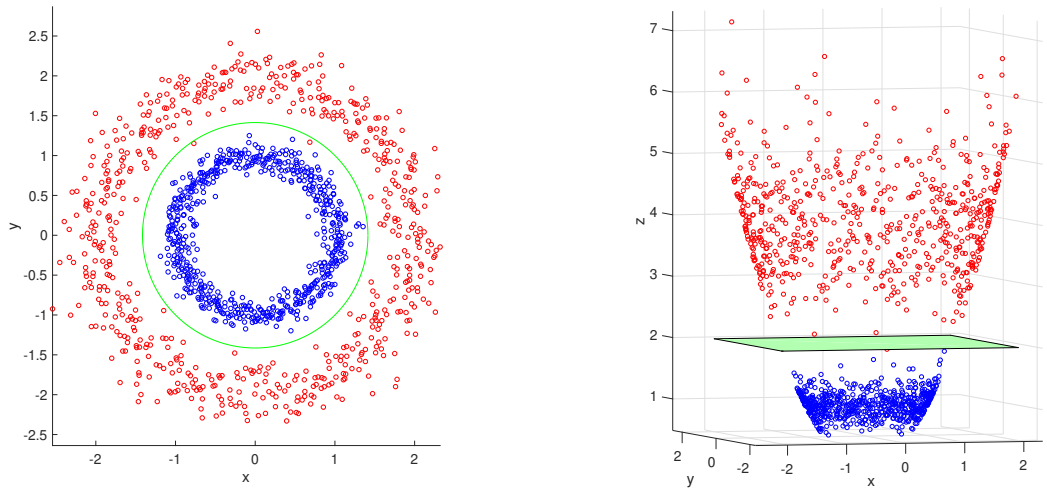


Figure 2.22: The illustration shows the different variables involved when carrying out support vector regression on a set of data that is assumed to be linear. The margin ϵ defines the boundary at which points do not influence the shape of the function. Points closest to the boundary are referred to as support vectors and are directly responsible for the shape of the estimated function. The aim of SVR is to reduce the distance between the function's margins and the support vectors i.e. the distance represented by ξ.

2.7 Principal Component Analysis

Principal component analysis is the procedure for analysing the contribution that different independent variables have on the variance of the overall data. Given a toy example (illustrated in Figure 2.24) where the data consists of three column vectors $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, each generated using a normal distribution with a mean of 0 and a standard deviation of 0.5, 0.3 and 0.9 respectively, the PCA of the data $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$ indicates the higher contribution of variables $\mathbf{x}_1$ and $\mathbf{x}_3$ to the overall data variation. This contribution is visualized by the component vectors, color coded in red, green and blue, which are respectively in increasing order of Eigenvalues.

Each corresponding Eigenvector represents the direction of a specific component vector and the Eigenvalue its magnitude. The importance of each dimension i.e. $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, can then be determined by analysing the principal component, which is the Eigenvector with the largest Eigenvalue. The principal component vector will consist of n-number of elements, where n corresponds to the number of dimensions in the data. In our toy



(a) (b)

Figure 2.23: An illustration of how a non-linear data set can be linearly separated by a hyper-plane given the addition of another dimension. In this example the extra dimension is defined as $z = x^2 + y^2$. In the case of support vector machines and regressions, this explicit mapping is not required an instead a kernel is used to measure similarity in this different dimension.

example, the first element corresponds to the first dimension $\mathbf{x}_1$, the second to $\mathbf{x}_2$, and so on. The principal component in this example is highlighted by the blue vector, and the largest element is the third element, which corresponds to the the third dimension $\mathbf{x}_3$, followed by the first element which is $\mathbf{x}_1$, and finally the second element which corresponds to $\mathbf{x}_2$.

2.8 Robot Operating System

The Robot Operating System (ROS) is a framework that provides standardized messages and protocols that hardware and software can use to communicate with one another.

Under this framework there exists a master node that is responsible for the overall organization of other nodes and their respective processes (Quigley et al., 2009). Each node typically belongs to a unique system such as a sensory system. This sensory system can, for example, publish its readings at a specific frequency, such as the Ultrasonic sensor specific node shown in Figure 2.25. The publishing process involves transmitting a message to a unique topic, which has an associated message type and publishing

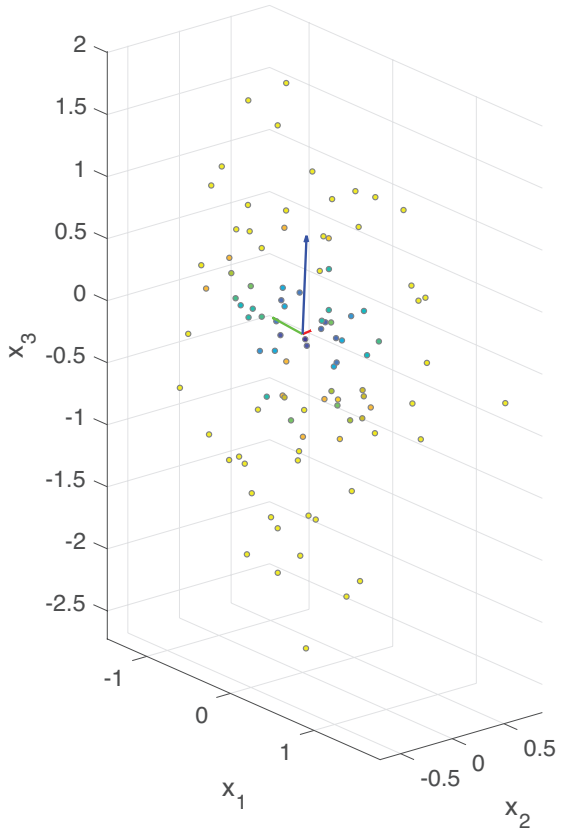


Figure 2.24: Example of principal component analysis being carried out on a 3-dimensional set of data $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$. The component vectors are determined by calculating the Eigenvector and Eigenvalues of the covariance of the data $\mathbf{x}$, where an Eigenvector determines the component vector's direction and the Eigenvalue its magnitude. The component with the largest Eigenvalue is called the principal component vector. By analyzing the elements of the principal component vector the dimension with the largest absolute value would indicate the dimension that contributes the most to the data's variance. In this example the data is generated with a zero mean normal distribution with a standard deviation of 0.5, 0.3 and 0.9 respectively for dimensions $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. The principal component from this example is found to be [0.1337, -0.0634, 0.9890]. Taking the absolute values, it can be seen that the third dimension is the largest contributor, followed by the first then second.

frequency. Other nodes can subscribe to said topics and obtain readings at the rate specified by the topic, which in Figure 2.25 would be the SLAM specific node. Alternative communication methods include broadcasting services. For example, given a service broadcasting node associated with an imaging subsystem, a user can obtain an image by making a request via the broadcast service. Services differ with respect to publishers

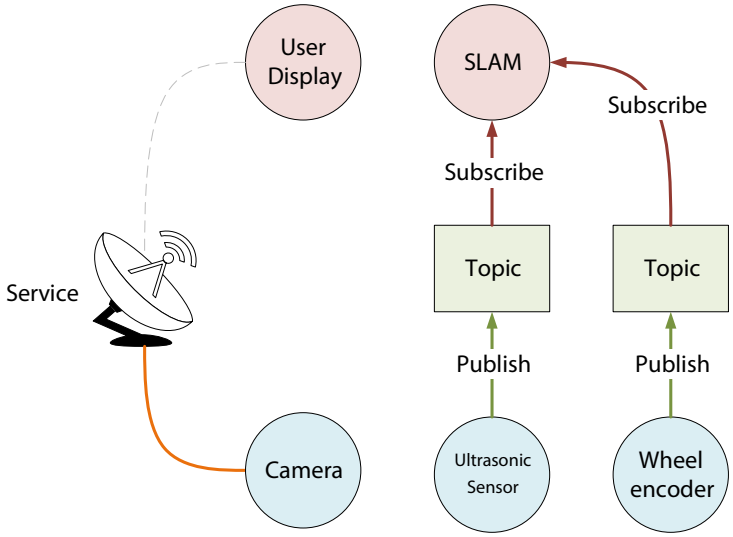


Figure 2.25: An example showing an overview of a simple ROS network. The figure shows how an Ultrasonic sensor publishes its readings to a specific topic from which other nodes, like the SLAM specific node, may subscribe to. In the event that a user wants to take a picture of the robots current view, they may request an image from the camera specific node, which is broadcasting a service for capturing and returning images.

in that the parent node only performs the necessary work when requested, instead of at regular intervals.

To aid in debugging, ROS provides a manner for which an entire session can be emulated and played back in either real time or otherwise at varying speeds. To store this playback data, *.bag* files are used.

2.8.1 MoveIt

MoveIt is a motion planning framework that works with ROS and provides functions related to the operation of arm manipulators. Arm manufacturers are able to provide support for their specific manipulators so that MoveIt is able to perform inverse kinematics, motion planning and environment set up and collision detection (Chitta, Sucan, & Cousins, 2012).

Chapter 3

RatSLAM Navigation Using A Whisker-Sensor Array

The following chapter is based on previously published work (Salman & Pearson, 2016).

This chapter focuses on the work carried out to investigate whether the whisker-sensor array in combination with RatSLAM is capable of solving a SLAM problem. A novel RatSLAM specific metric is described and used to measure the suitability of the place-recognition image matching threshold, along with how confident the algorithm is with each observation. Furthermore, the chapter includes an analysis of RatSLAM's performance when using a more accurate estimate for whisker-angle at contact and when using an RCP whisker control strategy.

3.1 Method

To evaluate the capability of the RatSLAM whisker-array system, a terrain that visually had enough variation across its surface was constructed and used to facilitate place-recognition. The terrain was constructed on a $1.35 \times 1m$ re-configurable maze, which was composed of several plastic walls and posts. The maze, along with the whisker-array and UR10 arm platform, are shown in Figure 3.1.

The arm was set to move to a set of hard-coded positions on a planar surface (with the resulting path shown in Figure 3.3), while the whiskers were in a continuous state of whisking. In addition to the sensory data described in 3.1.3, odometry of the end-effector and the generated whisker-tactile images were also logged for the duration of each run so that RatSLAM could be tested offline and its parameters tuned quickly.

Since this work uses the ROS version of OpenRatSLAM, the algorithm requires two

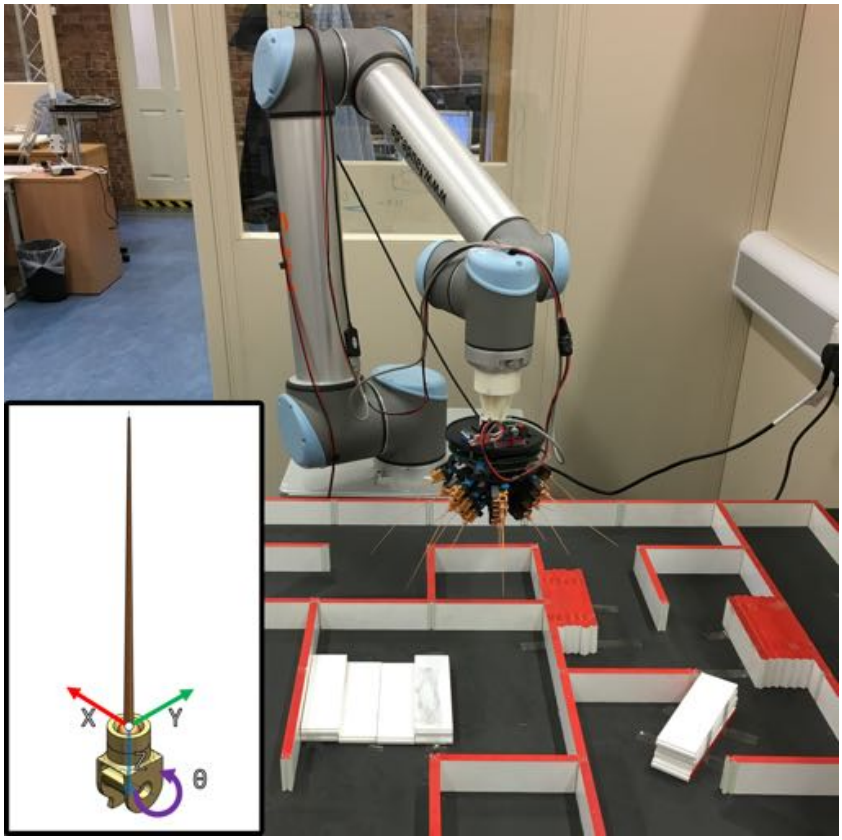


Figure 3.1: Whisker-array mounted on a UR10 Universal-Robots arm and positioned above the maze used to evaluate the RatSLAM algorithm. The inset focuses on an individual whisker and highlights the directions of measured whisker-deflection (x, y) as well as the pivot point around which the whisker-angle (θ) was measured. The deflection in the$z$direction is not measurable and its inclusion in the image is to clarify the orientation of the$x$and y component vectors.

inputs: An image of the current observation and an odometry reading describing the angular yaw velocity and the linear translational velocity of the sensor-array.

3.1.1 Tactile Image

The tactile image was generated by first defining an image that had each pixel represent a specific whisker on the sensor-array. Figure 3.2 shows the mapping of whisker to pixel. The 8-bit image was in grey scale and its pixel values ranged between 0 and 255.

The intensity of each pixel was proportional to the approximate vertical height of any obstacle detected using a measure of whisker angle, θ, at point of contact and the forward kinematics of the specific whisker.

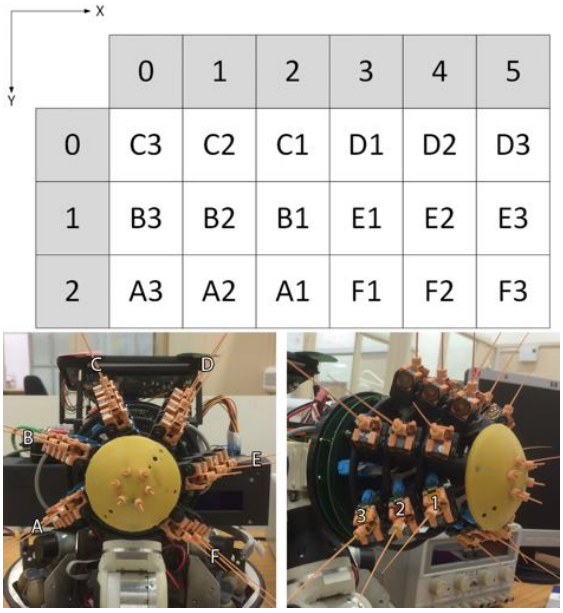


Figure 3.2: The tactile image is composed of 18 pixels that are arranged in a similar distribution to that of the whiskers in the sensor-array. Each pixel corresponds to a specific whisker, which can be identified according to the label. Each of the six collumns are identified using letters (A-F) and the three rows using numbers.

The angle of contact was derived at the end of each whisk cycle using two approaches; the first, which is called *max_angle*, was taken as the maximum whisker angle observed throughout the cycle; the second, known as *contact_angle*, is the angle measured at the time when the magnitude of the first derivative of the whisker deflection crosses a threshold.

The angle of the whisker at that time in the whisk cycle is then passed through the forward kinematics to determine depth. *Max_angle* is found to be robust to sensor noise, however, it did reduce the frequency of positive wall detection as the whisker must remain in contact with the wall for the duration of the whisk cycle. *Contact_angle* generated a larger number of contact measurements, but, was susceptible to sensor noise and therefore spurious observations.

3.1.2 Odometry

The odometry and tactile image were both published at a rate of 1.5Hz. OpenRatSLAM requires the odometry message to include: the velocity of the yaw angle, which would effect the estimated heading value, followed by the linear translational velocity. The

odometry needs to be considered in that order since the algorithm carries out path integration by rotating and then translating.

Due to the low sampling frequency of this work's system, path integration using the odometry input resulted in the accumulation of error. The errors have been characterized by comparing the linear and angular displacements to the ground truth displacement. For linear displacement, the errors were found to be strictly positive with a log normal distribution best fitting the data $Lognormal(-4.3, 0.45)$. The angular displacement error was found to follow a Normal distribution $\mathcal{N}(8.39e^{-4}, 0.3)$.

Without any correction, the estimated pose of the robot based on pure path integration will drift from the true value as shown in Figure 3.3. Thus, to improve the accuracy of the estimated pose and map, RatSLAM is used to minimize the accumulated errors.

3.1.3 Data Collection

The whisker-array's sensor data was sorted in a series of *.csv* (comma seperated values) files, with each file capturing the data from a single whisk cycle. A single file 4 columns with each column belonging to the angular sensor data, deflection in the x-direction and deflection in the y-direction respectively. The last column does not contain any relevant information and exists as a placeholder for possible system modifications that includes adding extra sensors.

The sensory data from the arm included the pose of the end-effector i.e. the whisker-array. The pose was published to custom ROS topic at a rate of 1Hz in the form of a ROS message of type *geometry_msgs/PoseStamped.msg*. The message held the systems current time, a 3-dimensional Cartesian coordinate, an orientation that is represented by a quaternion vector and included the latest whisker data *.csv* name that was created at the time of generating the message. *.bag* files were used to log ROS messages and tune the RatSLAM parameters offline.

3.2 Experimental Setup

Since RatSLAM assumes that the sensor array moves within a planar environment, the sensor-array's movement is constraint to a plane set 77 mm above the maze. The chosen height prevents the whiskers from contacting the floor and limits contact to the protruding walls that were distributed across the terrain. Rotation is also limited to a single degree of freedom and the whisker-array can only rotate about its major axis.

Table 3.1: The OpenRatSLAM parameter values selected for our work in Chapter 3. These values were obtained during the tuning session and are fixed for subsequent runs.

Parameter	**Value**
vt_shift_match	0
vt_step_match	1
vt_active_decay	1
template_x_size	6
template_y_size	3
pc_dim_xy	30
pc_cell_x_size	0.015
exp_delta_pc_threshold	1
exp_loops	10
exp_initial_em_deg	180

The whisker-array was then translated around a maze, following this work's predetermined path, which is illustrated in Figure 3.3 on page 68. The whiskers were set to whisk continuously at a rate of 1Hz at all times. Each run included 3 complete clockwise circuits of the path, 3 complete anti-clockwise circuits and a subsequent clockwise and anti-clockwise circuit.

The first run was used to tune OpenRatSLAM's parameters, specifically those related to the measurement model.

3.2.1 OpenRatSLAM Parameters

Before evaluating any performance results, OpenRatSLAM needs to be set up for use with this work's robotic system, which includes taking into consideration the platform's velocity range. The parameter *pc_cell_x_size* scales the translational velocity and needs to be set such that path integration causes a shift of one cell per iteration (Ball et al., 2013). For this work's system, the value is specified in Table 3.1. Other modified parameters include the dimensions of the pose grid cell *pc_dim_xy*, which was kept large so as to reduce the likelihood of false re-localizations occurring due to pose-cells representing too many regions in the environment (Ball et al., 2013). The Local View cell decay value *vt_active_decay* and the threshold defining the change in pose needed to result in a new experience *exp_delta_pc_threshold* were not modified and were kept the same as that of the *iRat* dataset (Ball et al., 2013).

The tuning process mainly dealt with two parameters, which are the *visual tem-*

plate matching threshold and the *pose cell inject energy*. The *visual template matching threshold* parameter refers to the threshold used to define whether two separate visual templates are similar. The similarity measure was modified to include all pixels when calculating the sum of absolute differences, as opposed to the sum of the image columns used in (Ball, 2018a). The formula of the modified similarity measure is shown in Equation 3.1, where A and B are the two images being compared, and a and b are their respective pixel from row i and column j.

$$S(A, B) = \sum_{ij} \sqrt{|a_{ij} - b_{ij}|} \tag{3.1}$$

Since the tactile-image has a significantly lower number of pixels as opposed to a typical camera based image, down-sampling is not needed; this work's visual template is equivalent to the raw tactile-image that is fed into RatSLAM. The similarity measure is used to compare against the threshold and provided that it is lower, the templates are considered to be a matching pair.

When a visual template is matched, the activity in the associated pose cell is increased. The *pose cell inject energy* parameter affects the degree in which the activity is increased by. A balance between the matching threshold and injection energy is needed to make sure that occasional false matches do not cause to much influence while still allowing for occasional correct matches to cause regular re-localization.

3.2.2 Whisker Control

An additional aim of this chapter is to evaluate the benefit that a bio-inspired whisking strategy brings to the navigational performance of a whisker-sensing mobile platform. Rapid cessation of protraction is one of the three behaviors observed in rats (Mitchinson & Prescott, 2013a); it involves the cessation of whisker protraction when an unexpected contact is made, followed by a retraction. The behavior suggests an attempt to protect the integrity of the whiskers, which is a desirable objective. Given the fragility of this work's artificial whiskers, the implementation of RCP is necessary. Further, the current work seeks to investigate whether RCP brings any additional benefit to improving the navigational performance of the system and include two extra variable parameters for the set of runs: Open-loop whisking and RCP whisking.

Open-loop consisted of a simple trajectory tracking of each whisker angle following a sinusoidal 1Hz pattern of fixed magnitude that moved the whiskers throughout their full range of motion.

RCP mode received the same desired trajectory, however, the actual angle of each whisker could also be perturbed by local feedback from the whisker deflection sensors themselves. The magnitude of the perturbation was defined as the absolute average deflection, in the y direction, that exceeded the contact threshold.

The contact threshold was calculated at time of calibration as $\pm 20x$ the standard deviation of the recorded noise from the mean of the deflection data during *free-whisking*, i.e., whilst the whiskers were whisking at 1Hz in absence of any obstacles. An increase in perturbation resulted in a decrease of whisking amplitude, while keeping the initial whisking angle at the start of each cycle constant. Keeping the initial whisk angle constant is desirable as it meant the mean whisking angle kept the whiskers in a region that was higher than the tallest obstacle, thereby reducing the chances of any whiskers getting trapped within the maze's walls.

An additional variable parameter is the method of estimating whisker contact-angle, which varies between *max_angle* and *contact_angle*. There are thus two variable parameters, the method of contact-angle estimation and the whisker control strategy, with each parameter having two distinct states. When varying these parameters, the tactile-image changes, which in turn requires the OpenRatSLAM parameters that are discussed in section 3.2.1 to be re-tuned. Given the empirical derivation of the parameter values, 3 additional repetitions are made so that a statistical analysis can be performed.

To better tune OpenRatSLAM's parameters, as well as analyze the benefit that each of the different whisker control and angle estimation brings to the performance of the SLAM algorithm, a novel RatSLAM specific metric called the Experience Metric (ExM) is designed.

3.2.3 Performance Metrics

Two metrics were used to quantitatively assess the performance of the RatSLAM algorithm. The *Experience Metric* (ExM) is novel and is designed specifically for use with the RatSLAM algorithm while the second, *Energy Metric* (EM), is derived from (Kummerle et al., 2009), and is a more general metric for evaluating SLAM algorithms.

A benchmark visual data set was taken from an online repository (Ball & Milford, 2015) (and described here (Ball et al., 2010)), to serve as a sanity check for discussion of the more general performance of whisker based SLAM. The *iRat* data set was selected because it was derived from a similarly sized environment to the maze used here, as well as containing ground truth pose data.

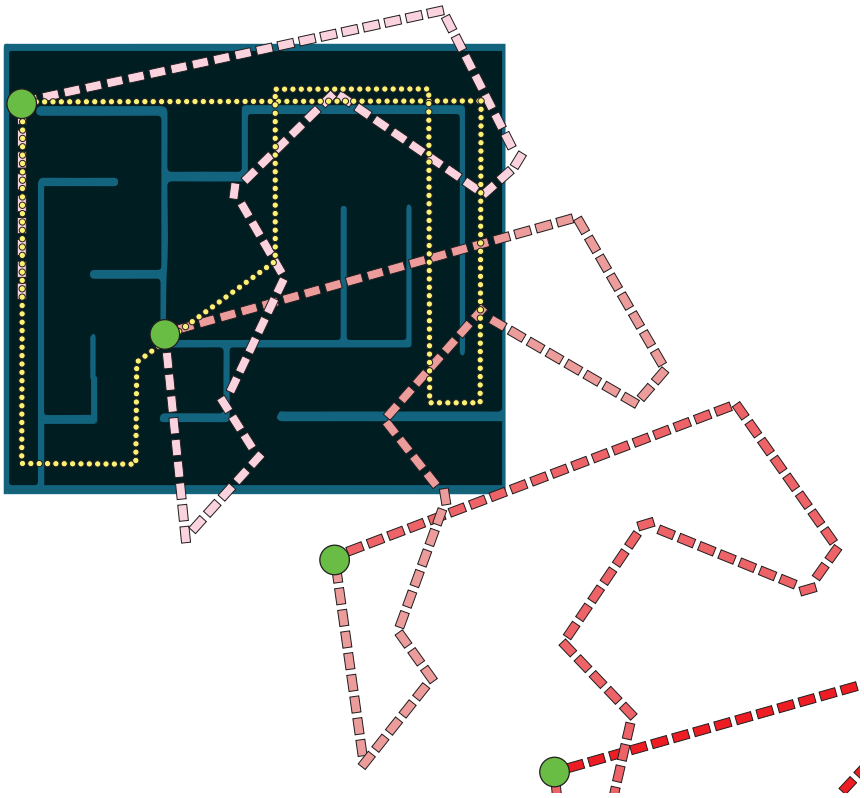


Figure 3.3: Actual path taken by the whisker module (circle markers) and the path derived from integrated odometry data (rectangular markers), the latter of which was cropped to only include the first three clockwise loops. Each loop increases in color saturation in order to show their progression in time additionally a large circle marks the start/end of a loop. The image highlights the effects that accumulating small odometry error has on the estimation of robot pose and thus a need for an appropriate error filtering mechanism that SLAM is known for solving.

3.2.3.1 Experience Metric (ExM)

At each RatSLAM iteration, there exists an active experience that represents the system's current belief in pose and location within the experience map. When a loop closure or re-localization occurs, the active experience is replaced with the experience that represents the previous pose and observation. Figure 3.4 plots these changes for the *iRat* data set.

In order to assess the correctness of these re-localizations, the range of experiences that represent the initial loop must first be defined. Thus, when a future re-localization occurs, and the active experience is set to one of the experiences in the initial loop,

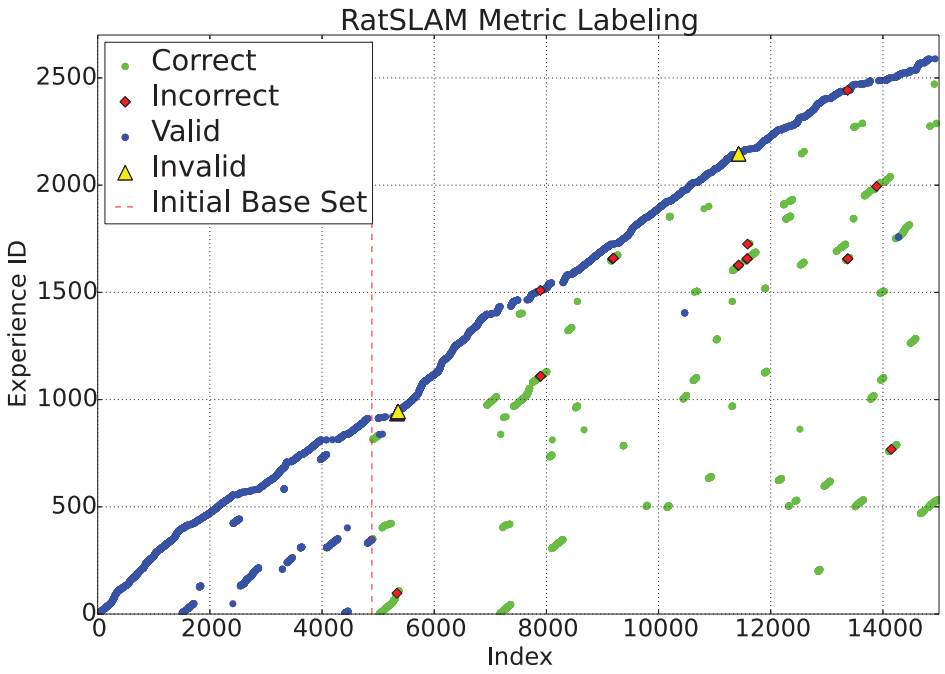


Figure 3.4: RatSLAM Metric Labeling. The image shows how RatSLAM associates the agents current position and observations with an experience ID. An experience ID that is visited more than once signifies a re-localization and is considered *correct* or *incorrect* based on the specifications of the user regarding their desired accuracy in pose. Novel experiences following an *incorrect* re-localization are deemed *invalid* as their accuracy can't be validated until a *correct* re-localization occurs.

the re-localization's correctness can be determined by comparing the current true pose (actual pose) with that of the true pose at the time at which the experience was first generated (perceived pose).

In Figure 3.4, the experiences to the left of the vertical line are a part of the initial loop. A re-localization is therefore considered correct provided the difference between its perceived and actual position and angle are below a certain threshold.

To consider the scenario where a re-localization occurs to an experience outside of the initial loop set, a set of *valid* and *invalid* experiences are defined. *Valid* experiences are those that proceed a *correct* experience and their positions are not corrupted by previous false re-localizations. *Invalid* experiences are, however, corrupted by a preceding *incorrect* experience and any re-localizations to these set of experiences would be deemed as *incorrect*.

The Experience Metric returns two values, the Average Rate of Re-localization

(ARR) and the Average Rate of Correct Re-localization (ARCR).

ARR is defined as the average number of re-localizations over the total number of experiences not including the original base set. The value is influenced greatly by the *pose cell inject energy* parameter since it affects the magnitude that a matching observation has on the activity of pose-cells. A perfect score of 1 indicates that all proceeding experiences in the subsequent loops originated from the intial base set, and that the sensor-array is repeating the first loop perfectly.

ARCR is defined as the average number of correct re-localizations over the total number of re-localizations. A correct re-localization is one where the absolute difference between perceived and actual pose do not exceed $0.1m$ and $25°$. A perfect score of 1 means that the algorithm is able to correctly localize at every iteration following the initial loop.

The two values, ARR and ARCR, can be used to tune the OpenRatSLAM paramters described in 3.2.1. Exceeding the optimal *pose cell inject energy* would result in a high ARR but lower ARCR, while a low *pose cell inject energy* would results in a high ARCR but lower ARR. The latter is true because a low *pose cell inject energy* would reduce the influence that a matching observation has on the belief of the system, and would therefore require a longer series of matching observations before a re-localization can occur; the system would spend more time (low ARR) confirming its place and making sure it has gathered sufficient evidence before it changes its current belief which would be more likely correct (high ARCR). The opposite is true for when the *pose cell inject energy* is high and any matching observation would result in a re-localization.

Given a fixed *pose cell inject energy* value, the reduction of the *visual template matching threshold* would result in lower ARR and higher ARCR. When the threshold is decreased, an observations needs to be more similar before a match occurs, thus increasing the likelihood of a correct re-localization (high ARCR). However, at the same time, since the requirements are more stringent, the likelihood of observing such similar matches is hindered by noise and other factors that reduce the likelihood of a re-localization from even occurring (low ARR).

The pseudo-code shown in Algorithm 1 on page 77 describes the steps taken to calculate the Experience Metric.

3.2.3.2 Energy Metric (EM)

The Energy metric was derived from (Kummerle et al., 2009). The authors measured the performance of a SLAM algorithm by defining the energy that it takes to transform the

trajectory of the agent according to the SLAM algorithm to, ideally, the true trajectory of the agent.

The Energy Metric is defined by Equation 3.2 where N is the number of relative relations (an experience point in the RatSLAM experience map and its corresponding sample point from the set of collected pose data). The variable $\delta_{i,j}$ is defined in Equation 3.3 and is the relative transformation from node x_i to node x_j. The functions $trans(\cdot)$ and $rot(\cdot)$ refer to translation and rotation respectively. The Energy Metric indicates a good performance by returning a low value, with zero being a run that resulted in no error at all.

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j} trans\left(\delta_{i,j} \ominus \delta^*_{i,j}\right)^2 + rot\left(\delta_{i,j} \ominus \delta^*_{i,j}\right)^2 \tag{3.2}$$

$$\delta_{i,j} = x_j \ominus x_i \tag{3.3}$$

3.3 Results

3.3.1 Performance Metric Evaluation

The Experience Metric was validated using the *iRat* data set and the Energy Metric for comparison. The data set was replicated and modified by"kidnapping" the agent, i.e., skipping it forward to a future position, thereby creating two data sets for comparison, the latter set should result in a reduction of performance to confirm the correctness of the metric.

The results of this test is shown in Figure 3.5, which by the decrease in ARR and increase in the Energy Metric, indicates the expected drop in performance. The decrease in ARR may be attributed to the fact that the chain of "visual" scenes that leads to a re-localization was disrupted by the kidnapping and thus temporarily prevented any re-localization. In the case of the Energy Metric, RatSLAM was penalized for not detecting the kidnapping immediately, which led to the increase in error. ARCR remained relatively constant, which indicated that the RatSLAM parameters were appropriately tuned to correctly associate "visual" templates.

With confirmation of a valid metric behavior, it was then used for a quantitative evaluation of the impact on performance of RatSLAM through the adoption of the different whisker contact-angle estimation methods and whisker motion control schemes.

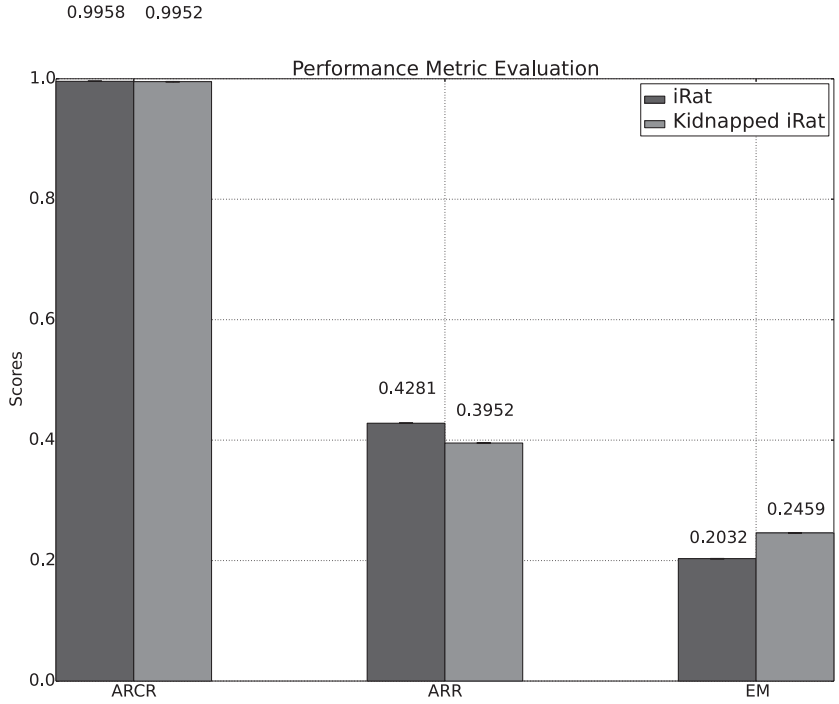


Figure 3.5: Performance Metric Validation. The results show that for a manipulated data-set designed to reduce the performance of the RatSLAM algorithm, this work's novel RatSLAM specific performance metric, the Experience Metric, does indeed show a reduction in performance via a reduction in the frequency of re-localizations (ARR). This reduction of performance is also observed by the increase in the general SLAM performance metric, the Energy Metric (EM) (Kummerle et al., 2009). The Average rate of correct re-localization (ARCR) indicates that the algorithm was able to maintain its re-localization accuracy.

3.3.2 Vanilla Whisker-RatSLAM Performance

Using the simple open-loop mode of whisker control and the *max_angle* scheme for whisker tactile to image transform, the RatSLAM-Whisker system was proven capable of accommodating the relatively sparse sensory information from the whisker "tactile images" by demonstrating loop closure and expressing only a limited number of *incorrect* re-localizations, as indicated by the transition from the erroneous path in Figure 3.3 to the corrected path in Figure 3.6.

When faced with changes to path direction, it was observed that for all data sets, experiences created when driven in an anti-clockwise direction were not associated with their clockwise counterparts, instead they were treated as unique locations. By alternating the path direction multiple times, the separate loops from the clockwise and

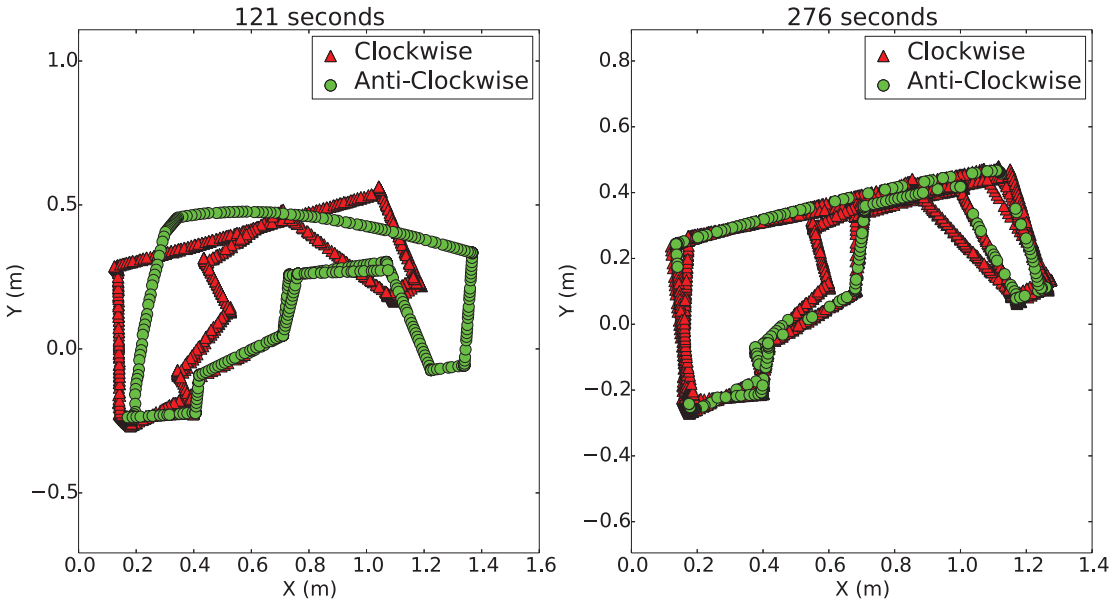


Figure 3.6: Plots showing trajectory estimates derived from whisker-RatSLAM following multiple runs of alternate robot loop directions (clockwise in red and anticlockwise in green). The **Left** plot emphasizes how the different directions of travel generate different estimates of path trajectory. However, this difference reduces in **Right** panel following repeated iterations of loop closure in both directions. This was due to links being made between points of similarity in the maze leading to re-localization through shared experiences, which was particularly apparent at the turning points in the trajectory

anticlockwise runs began to overlap as the agent observed similar tactile images when rotating at corners. This scenario of overlap can be seen in Figure 3.6.

3.3.3 Effect of Whisker Control Strategy

The following set of results compare the performance of the robot when its whiskers are controlled using the *open-loop* and *RCP* schemes described earlier.

Figure 3.7 shows that the adoption of RCP whisker control improves the performance of the whiskered robot by making frequent *correct* re-localizations as indicated by the increase in ARR values from 0.1765 to 0.2257. ARCR remained constant for both *open-loop* and RCP. The improvement in performance through the adoption of RCP based whisker control was also observed using the Energy Metric that resulted in a decreased average measure of 0.1568 from 0.1916 across all data sets.

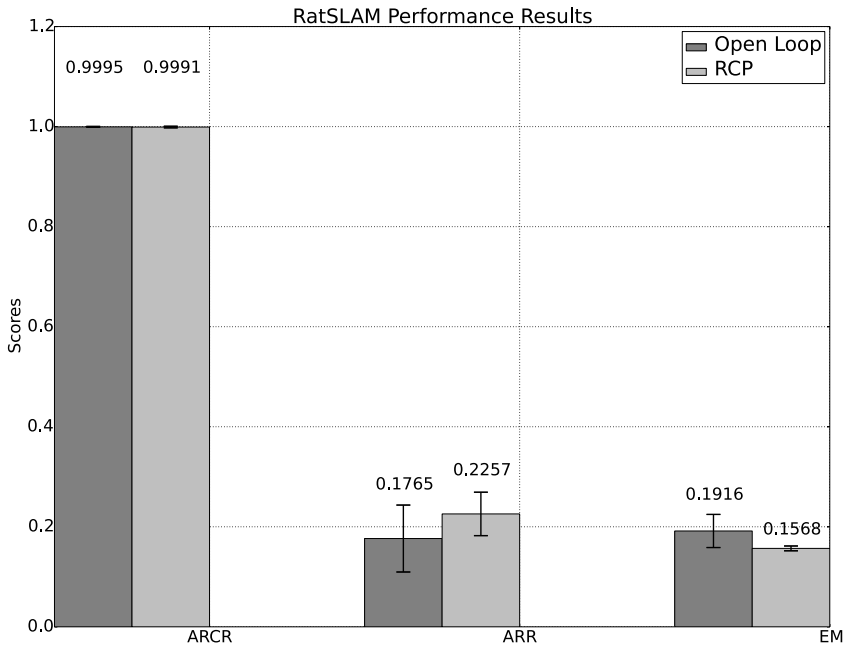


Figure 3.7: RatSLAM Performance comparison between *open-loop* and RCP whisker control. Increasing ARCR and ARR and decreasing EM indicate improvement in performance.

3.3.4 Effect of Whisker-Contact Angle Estimation Strategy

The previous runs were processed such that the whisker image was constructed based on the *max_angle* method, the following results are however the results of the second method *contact_angle*.

The Experience Metric shows a clear improvement in performance with increasing ARR values for both *open-loop* and *RCP* variations as shown in Figure 3.8. *Open-loop's* ARR value is also observed to increase following the use of *contact_angle*. EM performance shows slight improvement in the case of *open-loop*, unlike *RCP*, which shows a very small reduction in performance. The visual template matching threshold was increased during the use of *contact_angle*, which suggested a reduction in 'image' ambiguity.

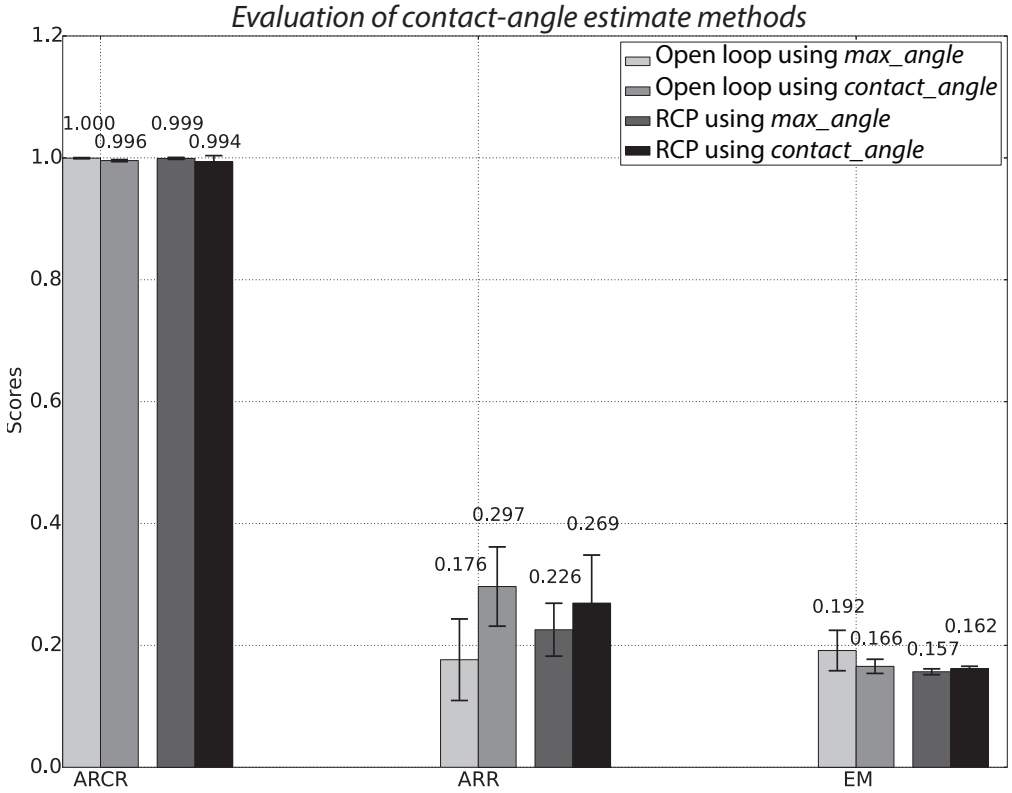


Figure 3.8: Results illustrating the benifit of implementing rapid cessation of protraction and a *contact_angle* approach for estimating contact-angle. Increasing average rate of re-localization (ARR) and decreasing energy metric (EM) indicate an improvement in performance. A relatively stable average rate of correct re-localization (ARCR) shows that the increase in re-localization rates are not causing a decrease in re-localization accuracy.

3.4 Discussion

In summary, this chapter highlights three accomplishments; first, the fusion of active whisker tactile data with the vision based RatSLAM by transforming contact height detected by the whiskers into pixel intensity; secondly, the introduction of a RatSLAM specific performance evaluation algorithm, the validity of which has been confirmed using the Energy Metric from (Kummerle et al., 2009); and third, an empirical evaluation of a biomimetic whisker control strategy.

The ultimate ambition of this is work is to develop a system that is capable of building a map of its environment and maintaining an accurate estimate of its location through whisker based touch using minimal computational resources. Through the

adoption of the RatSLAM algorithm, it has been demonstrated that this algorithm has the potential for further investigation as a substrate for efficient tactile mapping and localization. Further, it is now possible to empirically evaluate the change in performance of RatSLAM, in response to different sensory placement strategies and pre-processing schemes, by measuring the dynamics of experience association within the algorithm. Theoretically, if RatSLAM were to map the environment completely it would create no further experiences, instead associating each new visual/tactile image with a previous experience and therefore re-localizing confidently. This condition would be indicated by the Experience Metric returning an ARR value of 1 and would be visualized in Figure 3.4 as an absence of new experiences following the establishment of the initial base set (as indicated by the red dotted line). In addition if the algorithm were performing perfectly all re-localizations would be deemed correct and therefore the Experience Metric's ARCR would also be 1. This ability to decompose the performance of the RatSLAM algorithm highlights the advantage of using the Experience Metric over the more generic Energy Metric.

3.4.1 Summary

This chapter presented results confirming that RatSLAM, in combination with this work's whiskered-tactile sensory array, is capable of performing SLAM on a planar surface. Implementation of Rapid Cessation of Protraction whisker control in combination with the *contact_angle* for identifying the angle of whisker contact resulted in the best localization performance since it showed a reduction in tactile 'image' ambiguity.

The experiment's maze surface is more crowded than that of a office floor and therefore would not be representative the intended work environment. In order to navigate within an office like environment where the floor is of a uniform texture and characteristics, there is a need for identifying higher level features such as object shape.

To better characterize the shape of an object the whisker-sensors' perceptual abilities need to be improved, particularly its precision in estimating the location of contact. Chapter 2 described how contact localization accuracy is dependent on the whisker's contact angle and radial-distance estimates. Having realized an effective solution for estimating whisker-angle at time of contact, the next step is to obtain an appropriate solution for estimating radial-distance to contact.

Algorithm 1: Experience Metric

function ExperienceMetric $(\mathbf{E}, \mathbf{P}, i_{base}, \delta_p, \delta_\theta)$;
Input : Experience Log $\mathbf{E} = (e_1 e_2 \dots e_n)$
Ground Truth Log $\mathbf{P} = (\mathbf{p}_\theta, \mathbf{p}_x, \mathbf{p}_y)$
i_{base} = final index of base set in $\mathbf{E}$
δ_p = position error threshold
δ_θ = angular error threshold
Output: Average rate of correct relocalization ψ_c
Average rate of relocalization ψ_r
$\mathbf{C} = unique(\mathbf{E}[1 : i_{base}])$; // Remove duplicates
$\eta_c, \eta = 0$;
$r_c = True$;
for $i = i_{base} + 1$ *to* n **do**
  $e_{i-1} = \mathbf{E}[i-1]$;
  $e_i = \mathbf{E}[i]$;
  $\mathbf{E}_{past} = (e_1 \dots e_{i-1})$;
  $\kappa_{r1} = e_i \in \mathbf{E}_{past}$;
  $\kappa_{r2} = (e_i = e_{i-1}) \wedge (e_{i-1} \notin \mathbf{R})$;
  if κ_{r1} **and not** κ_{r2} **then**
    $\mathbf{R}[\eta] = e_i$; // Relocalization
    η++;
  end
  if $e_i \in \boldsymbol{R}$ **then**
    $p_{true} = (\mathbf{p}_x[i], \mathbf{p}_y[i])$; // Ground Truth
    $\theta_{true} = (\mathbf{p}_\theta[i])$;
    i_{past} = for e in $\mathbf{E}_{past}[j = 1 \dots i-1]$ return last j where $e = e_i$; // Perceived
    $p_{perceived} = (\mathbf{p}_x[i_{past}], \mathbf{p}_y[i_{past}])$;
    $\theta_{perceived} = (\mathbf{p}_\theta[i_{past}])$;
    $\epsilon_p = norm(p_{true} - p_{perceived})$; // Euclidean
    $\epsilon_\theta = abs(\theta_{true} - \theta_{perceived})$; // Absolute
    $\kappa_{c1} = (e_i \in \mathbf{C}) \wedge ((\epsilon_p < \delta_p) \wedge (\epsilon_\theta < \delta_\theta))$;
    $\kappa_{c2} = (e_i = e_{i-1}) \wedge (e_{i-1} \in \mathbf{R}_c)$;
    if κ_{c1} **and** κ_{c2} **then**
      $\mathbf{R}_c[\eta_c] = e_i$; // Correct relocalization
      η_c++;
      $r_c = True$;
    else
      $r_c = False$;
    end
  end
  if r_c **then**
    if $e_{i-1} \notin \mathbf{C}$ **then**
      append $\mathbf{C}$ with e_{i-1}
    end
  end
end
$\eta_{exp} = length(\mathbf{E}) - i_{base}$;
$\psi_r = \eta / \eta_{exp}$; // ARR
$\psi_c = \eta_c / \eta$; // ARCR

Chapter 4

Object Shape Reconstruction

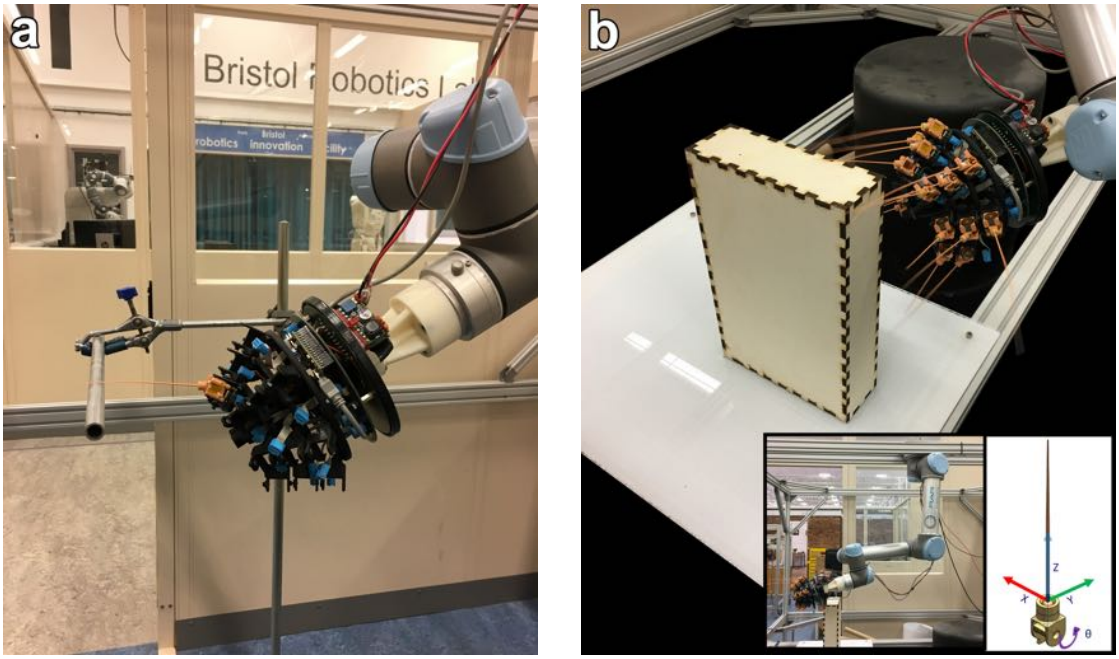
The work included in this chapter addresses the problem of constructing an internal 3D model of an object encountered by an active array of artificial whisker sensors mounted as the end-effector of a robotic arm.

One of the goals of this work is to identify unique tactile landmarks that can be used to create sparse topological maps. Operating in such environments implies a high likelihood of making uncontrolled contacts with objects resulting in damage and loss of whiskers. Therefore, it is desirable that any object discrimination algorithm is robust to variation in whisker shape and collision dynamics. Toward this end, the performance of 3 regression techniques, taken from standard machine learning literature, are applied to active whisker 3D object reconstruction and are compared.

This work's stated method of estimating the 3D point of contact is a two step process, which involves estimating pose of the whisker shaft at the onset of contact, followed by the estimation of radial-distance along the whisker shaft to the point of contact. The focus of this chapter is, therefore, to improve the the accuracy of the estimated radial-distance to contact. In addition to improved accuracy, the aim of this work seeks to design a method that reduces the work involved when replacing broken whiskers on the sensory array, which is a reasonably common problem. Whisker replacements cannot always be guaranteed to preserve the dimensions of the whisker that's being replaced, and would therefore vary in length.

It was therefore decided to explore a suitable regression model that can map a whisker's sensory signals' features to a radial-distance measurement while generalizing for different whisker dimensions. The work would be limited to using a single material for the construction of the whiskers and would assume that the ratio of tip and base diameters is constant.

4.1 Method



(a) (b)

Figure 4.1: Experimental apparatus. Figure 4.1a includes a photograph of the rod and single whisker module that were used to gather a training data set of whisker point contacts at known radial distances. The arm was moved parallel to the whisker length, pausing at different radial distances whilst a range of whisking frequencies were applied and the deflection data collected. Figure 4.1b includes a photograph of the Box shaped object and full whisker array during the collection of the validation data-set. The **left inset** shows a full view of the UR-5 arm mounted to the aluminum scaffold. The **right inset** includes a diagram of an individual whisker module highlighting the reference frames that defines the whisker deflection vector $\mathbf{x}$ and $\mathbf{y}$, and whisker rotation $\boldsymbol{\theta}$. The $\mathbf{z}$ direction in the image is only for clarifying the frame's orientation and is not available as a sensor reading.

To collect the training and validation data sets, the whisker-array, which consists of 18 whisker modules, is mounted onto the end of a Universal-Robots UR5 arm. The arm is described in greater detail on page 21 and the whisker-array and individual modules are described on page 22.

Fixed at the base of each whisker shaft is a magnet, which is positioned above a 2D Hall-effect sensor to measure deflections in the orthogonal axes referred to as x and y (see inset of figure 4.1b). A small brush-less DC motor built into the housing enables whisking

through $\pm 50^{\circ}$ in 1 degree of freedom referred to as θ. The whisk angle is controlled and monitored by an embedded 16-bit micro-controller which relays the sensory information (x, y, θ) back to the remote data capture computer at 0.5 millisecond intervals. A more in depth description of the whisker module can be found in (Sullivan et al., 2012). The array was mounted as the end-effector of a Universal Robotics UR-5 arm as shown in Figure 4.1. This was mounted onto an aluminum scaffold to demarcate the workspace with each actuator controlled and coordinated using the ROS execution framework.

4.1.1 Whisking pattern

The whisking pattern of each whisker was controlled using using a global sinusoidal pattern individually modulated by an analogue of RCP to reduce excessive stress on the whiskers and constrain the range in sensory response of deflections (Salman & Pearson, 2016). RCP was implemented by triggering an early retraction of the whisker in the event that a threshold in its deflection magnitude was crossed.

Since one of the signal features described below includes the amplitude of the deflection signal, it is logical to assume that the inclusion of an RCP strategy would serve to saturate the signal's amplitude. To reduce the likelihood of this saturation occurring, the threshold was set to the maximum detectable deflection value. The threshold thus prevented extreme bending, while still allowing for the amplitude to vary in most scenarios. It is noted that even without RCP the contact amplitudes would still saturate due to the limitation of the sensor. However, with RCP causing an immediate retraction regardless of whether the whisker reached a fully protracted state the chances of whiskers breaking are reduced.

4.1.2 Data collection

To derive the approximations of the functions mapping whisker sensor features to radial-distance, the performance of three standard regression techniques are investigated: Polynomial Regression, Neural Networks and Support Vector Regression. To investigate which of these techniques performs best for our application, appropriate training and validation data-sets were collected.

4.1.2.1 Training set

The training set data was collected by whisking whiskers of different lengths into a rod set to make contact at various radial distances (see Figure 4.1a). The set of whiskers used to obtain this training data included only a subset of whiskers since it was decided to

leave the remaining whiskers with unique lengths as a means of validating the generality of the models. The whisker lengths used in the training set were: 65 mm, 78 mm, 87 mm, 88 mm, 94 mm, 107 mm, 113 mm and 137 mm. The collection involved the horizontal displacement of a whisker, fixed on the end of the UR-5 arm, from its base to its tip at 20 equally spaced intervals starting from 25mm from the base of the whisker-shaft. Each run consisted of whisking between 0.2 and 2.5Hz at each radial distance.

4.1.2.2 Validation set

To assess the performance of each technique in a mobile setting, a data-set that consisted of the whiskers exploring a box shaped object was recorded (see Figure 4.1b). The trajectory of the end-effector was set by manually positioning the arm in desired poses and hard coding these as way-points. The way-points were selected such that contact occurrences ranged from tip to anywhere along the length of the whiskers. Since contact speed has been shown to be an important contact feature in classification of radial distance, the data-set needed to include samples in which the end-effector was held still, as any motion would be translated to the whiskers and therefore influence the response of the deflection signal. The arm was therefore held at each way point for one minute before continuing on to the next pose. The length of each mounted whisker was measured for use as one of the input variables to the estimators. The number of unique whisker lengths outnumbered those used for the training portion of the investigation, to allow an evaluation of the generality of each estimator. The set of unique whisker lengths included in the validation set and not in the training set were: 50 mm, 64 mm, 80 mm, 95 mm, 101 mm, 127 mm, 128 mm, 150 mm and 154 mm.

4.1.2.3 Ground truth and error calculation

To assess the quality of reconstruction from each regression technique, a set of simulated ground truth contact locations were derived. This required the calibration of the arm's position with respect to the work-space so that a virtual object could be placed in the same position as the physical one. Calibration was carried out by mounting a short straight whisker at the midpoint of the end-effector, followed by positioning the whisker in the center and on the surface of the workspace center point, thus obtaining the desired reference point. An algorithm was developed to simulate what would be the anticipated contact position given the assumptions of a movable and rigid whisker.

The algorithm operates by using the known position of the end-effector in world frame to locate the pose of each whisker base using forward kinematics. Each whisker

is then swept from its retraction to protraction limits at intervals of 1 deg within the world frame. Given the intersection of the swept vectors and faces of the box model, a list of intersection points can be generated. Figure 4.2a highlights a moment in time where a single whisker is swept through its entire range of motion and points of intersect are filtered such that the point closest to the retraction limit and the whisker base is selected. Since the whisking trajectory initiates in retraction, the first occurrence of a contact would be from that direction.

The radial-distance error is therefore defined as:

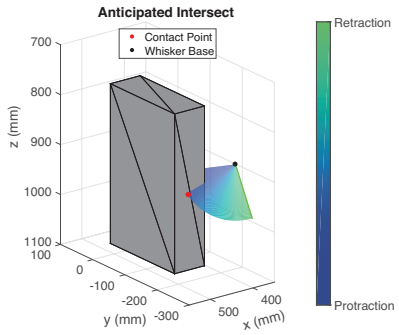
$$\delta_r = \frac{(d_m - d_{gt})}{d_{wl}} \tag{4.1}$$

Where d_m is the radial-distance estimate according to the specific regression model, d_{gt} is the radial-distance value according to the simulated ground truth method, and d_{wl} as the length of the whisker whose error is being measured.

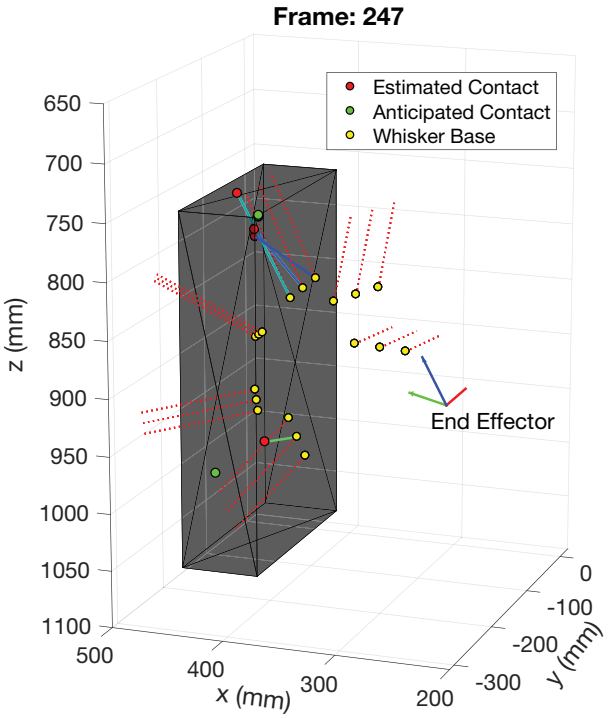
The whiskers in simulation are assumed to be rigid and straight, which is not the case for the physical whiskers. This assumption leads to variations in the estimated whisker angle at moment of contact between the simulated ground truth and that of the whisker-contact feature extraction methods described in section 4.1.2.4. The variation of contact angle would affect the simulated ground truth 3D contact location and in turn, the radial-distance. The histogram of the contact angle errors is plotted and shown in Figure 4.3 on page 86, which illustrates the distribution of angle differences between the simulation derived contact angles and those using the whisker-contact feature extraction method described in the following section. The plot indicates that the median angle error is -16.7°.

Still, variations in contact angle would suggest that either the simulated ground truth or the feature extraction method are not representative of the real ground truth, and, since there is no definitive way of solving this ambiguity, additional analysis of the models' prediction performance is included. These analyses, which are the distribution of estimated points to model distances, do not rely on the simulated contact approach and therefore should provide a better overall comparison. The distributions would illustrate the fidelity of object reconstruction.

It has been mentioned that the feature extraction method includes an estimate for whisker-angle at the point of contact. The following section continues to describe this estimation procedure, along with details about the extraction of other signal features that are to be used as inputs to the proposed regression models.



(a) Illustration of Anticipated Intersect algorithm.



(b) A comparison between anticipated an estimated whisker contacts.

Figure 4.2: Given the position of a whisker, it is swept from its retraction to its protraction limit, which is illustrated in Figure 4.2a. With the assumption of a straight and rigid whisker, the contact point is selected from the list of intersect points that are nearest to the retraction limit and whisker base. The assumption results in the variation of contact angle, which in turn, might result in small or large discrepancies in contact location depending on the radial distance. An example of this discrepancy may be observed in 4.2b by noting the pair of contact points within the 950-1000 range in the$z$direction, both of which are estimated and anticipated contact locations for the same whisker.

4.1.2.4 Extraction of Whisker-Contact Features

Using a moving average filter, the deflection and angular sensor data for each whisk-sample was smoothed and further processed to extract the following contact features; deflection velocity, deflection acceleration, angular velocity, angular acceleration, contact period and contact amplitude.

Contact period and amplitude have been used by (Evans et al., 2013) to estimate radial-distance to contact and are therefore also included in the set of potential input variables.

Birdwell et. al specifies how angular velocity and rate of change of moment are needed to work out radial-distance to contact. Since the deflection is proportional to the moment (Evans et al., 2013), the rate of change of deflection (deflection velocity) should be proportional to the rate of change of moment. The angular velocity and deflection velocity are therefore included in the set of potential input variables.

Deflection acceleration, angular acceleration were also included in the set of potential inputs so as to provide the regression model with a complete description of the whisker's state during contact.

To confirm what variables are indeed important for estimating radial-distance to contact, principal component analysis was carried out in order to determine which variables contribute the most variation in the data, and thus, are most informative in estimating radial-distance. Further, with the intention of developing a regression model that does require whisker dimension measurements, multiple regression models are trained using a variety of unique contact-feature combinations in order to discern the best performing set of inputs

The contact-features were extracted by processing the deflection and angular sensors' signal following a complete whisk-cycle. A contact, under ideal conditions, generates a single prominent peak in the y component of the deflection signal as shown in Figure 4.4 on page 86. To identify these peaks the Matlab function *findpeaks* (*findpeaks*, 2017) was used. Ideal conditions are those where the contact is made without any slippages occurring, and the whisker maintains contact with an obstacle through out its protraction phase. Figure 4.5 on page 87 illustrates other types of signals that include whiskers slipping off an obstacle as it continues to protract, and when the whisker-array moves while a whisker is in contact with an obstacle.

4.1.2.4.1 Principal Component Analysis

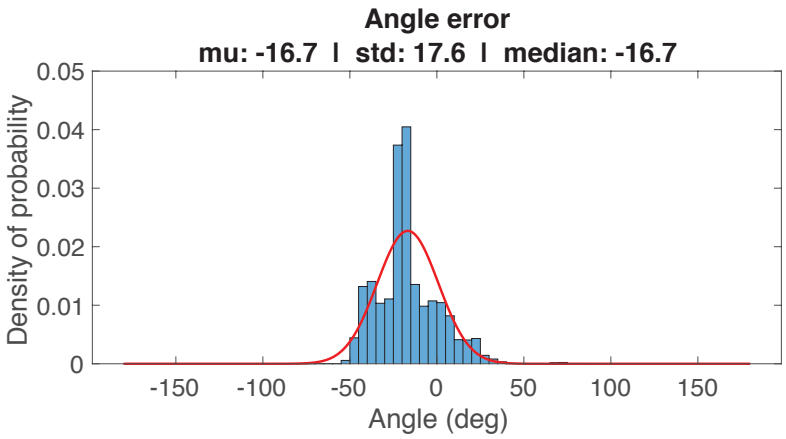


Figure 4.3: Distribution illustrating the difference between our simulated ground truth contact-angle estimates and those derived using our contact-feature extraction methods detailed in section 4.1.2.4.

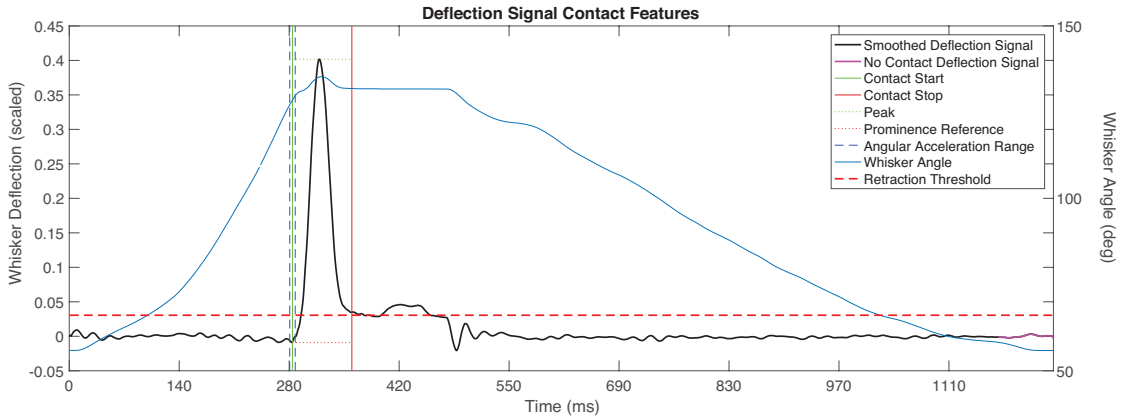
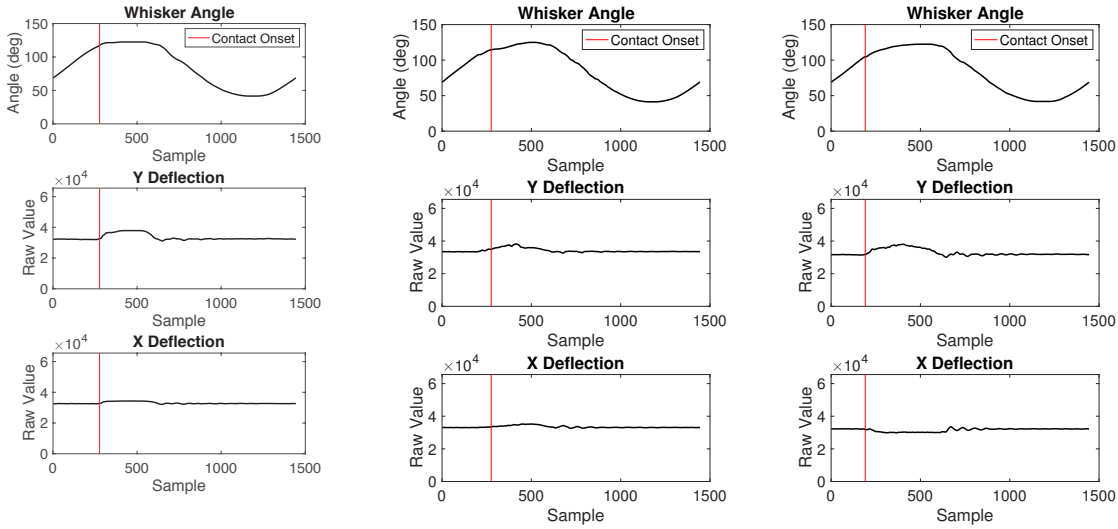


Figure 4.4: Illustration of a deflection signal during contact under ideal conditions where slipping does not occur, and contact occurs during the protraction phase of the whisk-cycle. A deflection value in the upper half suggests a deflection in the positive direction of the y-axis while a value in the lower half suggests a deflection in the negative direction of the y-axis, the values of the deflection signal are scaled such that limits are between 1 and -1. The whisker angle sensor data is represented in degrees and its axis is shown on the right hand side.



(a) Ideal contact (b) Contact Slip (c) Head retraction

Figure 4.5: Figures illustrating the whisker sensory signals for different contact scenarios. Each figure contains the sensory data from a single whisk-cycle i.e. a whisker moving from a fully retracted position to a fully protracted position and back. The top row includes the angular sensor's data, the middle the Y-deflection sensor's data and the bottom the X-deflection sensor's data.**4.5a** highlights a scenario where the whisker makes contact with an obstacle and results in an ideal contact that includes continuous contact with an obstacle during its protraction phase. **4.5b** highlights a scenario where the whisker makes contact with an obstacle and slips before completing its protraction phase. **4.5c** highlights a scenario where the whisker makes contact with an obstacle throughout its protraction phase while the whisker-array retreats from the obstacle. It can be seen that by working out the time in which contact occurs, via the method described in section 4.1.2.4, the whisker angle at time of contact can also be inferred by looking at the point of intersect of the contact onset line (vertical red line) and the angular sensor's plot line.

Principal Component Analysis was performed on the training data set to determine which set of contact features are most relevant in explaining the variance in the data, and therefore, necessary function variables. Due to the difference in feature units as well as their respective magnitudes varying by several orders of magnitude, the data is run through a whitening pre-processing step. Whitening includes normalization of the feature values such that their standard deviation is equal to 1. Figure 4.6 plots the coefficients of the 3 highest principal component vectors to illustrate the relative importance of each contact feature.

The figure highlights the low contribution of angular acceleration and whisker length

in explaining the variance in the collected data. It is therefore decided to proceed with the regression models' analyses with the input variables being fixed to: contact amplitude and period, deflection velocity and acceleration and angular velocity. The potential gain in estimation accuracy brought on by including the whisker length in the set of input variables is also investigated.

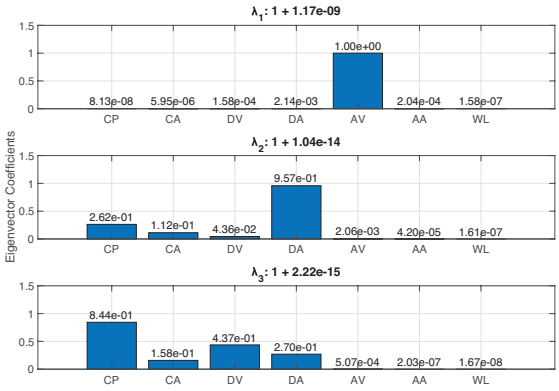


Figure 4.6: Principal component analysis on the learning data-set. The figure illustrates the relative importance of each variable in explaining the variability in the Training data-set that includes varying radial distance to contact. The plots illustrate the three vectors with the highest eigen values i.e. those that most affect the variation in radial-distance value. The coefficients represent the contact features, listed as follows from left to right: Contact Period (**CP**), Contact Amplitude (**CA**), Deflection Velocity (**DV**), Deflection Acceleration (**DA**), Angular Velocity (**AV**), Angular Acceleration (**AA**) and Whisker Length (**WL**). Since the units of the variables are different, as well as having significant variance in their range of values, the data was taken through a pre-processing step called whitening. The first three principal components, following the whitening process, suggest that CP, CA, DV, DA and AV to be major contributors to the variance seen in the data. Thus if dimensionality reduction were to be carried out, these would be the more relevant variables that should be kept in order to minimize precision loss.

4.1.3 Regression techniques

The three regression techniques for radial distance estimation that have been evaluated using these data sets were; neural networks, support vector machines and polynomial regression. Polynomial regression was specifically chosen so as to compare against the work of Evans et al. (Evans et al., 2013). The neural network approach, specifically the Multi-Layer Perceptron (MLP) model, was chosen for to its generalised ability in pattern recognition and classification problems (Cheng & Titterington, 1994). Finally, Support

Vector Regression (SVR) was selected due to its well established performance in pattern classification problems (Karamizadeh, Abdullah, Halimi, Shayan, & Javad Rajabi, 2014) and the smaller number of hyper-parameters compared to MLPs (*fitrsvm*, 2017).

To measure the performance of the trained models, K-fold cross-validation with K set to 10 was used. Polynomial regression was implemented using the Matlab function *polyfitn* (*polyfitn*, 2017). To determine the appropriate order of the polynomial model the bounds of the search space was set to include orders between 2 and 6. Using K-fold cross-validation to obtain the averaged validation mean square error, a 3rd order polynomial regression was found to be optimal.

For the neural network model, optimization of the network's structure was carried out by searching for the number of hidden layers and hidden nodes that resulted in the best performance. Using the following equation $N_h = (N_{in} + \sqrt{N_p})/L$ to set an upper limit on the number of hidden nodes, as described by (Gnana Sheela & Deepa, 2013). Further, the optimization search was limited by setting a boundary on the number of hidden layers to 2 to reduce the computational cost of the search. The optimum value was found to be 2 layers with 58 hidden nodes in each layer.

The final regression technique was the regression variant of Support Vector Machines. The optimization and training was performed using the Matlab function *fitrsvm* (*fitrsvm*, 2017) which included the support to search through linear, polynomial and Gaussian kernel functions as well as their specific parameters. The results of the optimization indicated that a Gaussian Kernel performed the best for these data sets.

4.2 Results

4.2.1 Validation of Regression Models

The results include a total of 12 regression models and Table 4.1 includes their respective labels and specific input variables.

Each of the regression models were trained using the training data sets described in section 4.1.2.1 and then validated on the validation set. Two sets of results are included: The first being the distributions of the point to model distances i.e. the distance between the model estimated contact-location and the nearest face of the object model (shown in Figure 4.8 on page 92). The second set of results include the radial-distance error, where the error is defined as the difference between the radial-distance estimate and the simulated anticipated radial-distance value (shown in Figure 4.9 on page 93). The radial-distance error was normalized against the actual whisker length to give a value

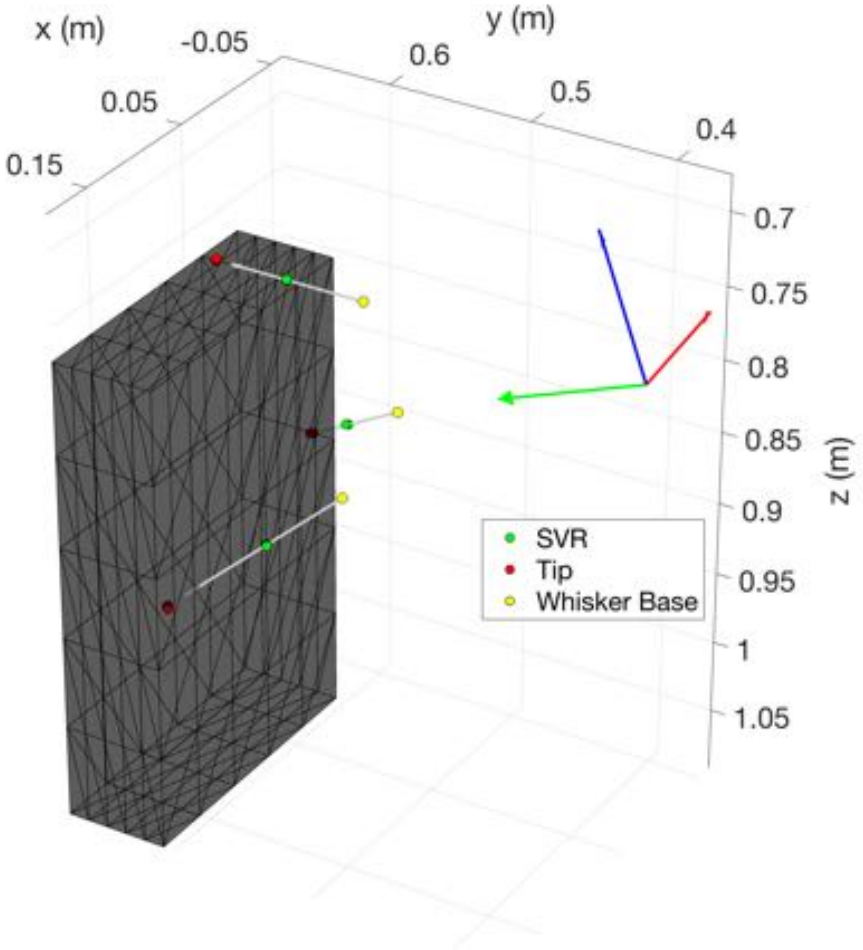


Figure 4.7: Contact point estimates derived from tip assumption versus a trained Support Vector Regression (SVR) model (id: *svr*9) superimposed over the validation object model. This figure includes contacts collected across 8 sequential whisk-samples, where in these samples 3 whiskers made contact with the object. It can be seen that the radial estimation using the SVR model is more accurate than the tip assumption, although it is not perfect as shown by the whisker in the most positive x-direction where the SVR estimate underestimates the radial-distance to contact and is further away from the face model than the tip based estimate.

Table 4.1: Regression models' labels and input variables. The inputs variables are contact period (CP), contact amplitude (CA), deflection velocity (DV), deflection acceleration (DA), angular velocity (AV) and whisker length (WL).

Label	**Features**
nn1	CP-CA-DV-DA-AV-WL
nn2	CP-CA-DV-DA-AV
nn3	CP-CA-DV-DA-WL
nn4	CP-CA-DV-DA
poly5	CP-CA-DV-DA-AV-WL
poly6	CP-CA-DV-DA-AV
poly7	CP-CA-DV-DA-WL
poly8	CP-CA-DV-DA
svr9	CP-CA-DV-DA-AV-WL
svr10	CP-CA-DV-DA-AV
svr11	CP-CA-DV-DA-WL
svr12	CP-CA-DV-DA

between -1 (underestimating) and 1 (overestimating).

Two additional cases have also been included for comparing the performance of each regression model against: a crude tip assumption where the radial-distance is always equal to the whisker length, and a uniformly distributed random radial-distance estimation method that is bound between 0 and the whisker length. Due to the morphology of the whiskers, the shape of the object, and arm motion, the data sets include a higher occurrence of tip-contacts. To minimize the effect of this bias, contacts where the simulated ground truth occurs $\leq 0.7 d_{wl}$ are considered.

Looking at Figure 4.8 on page 92 it can be seen that the models $svr9$ and $svr10$ performed the best, as indicated by their low median and standard deviation values, which would suggest that the contact points reconstructed the object with greatest accuracy and precision in comparison to other models. A two-sample Kolmogorov-Smirnov test is performed to quantitatively assess whether the distributions were similar to the tip based assumption, or otherwise to the random distribution. The results indicated that within a 1% significance level neither distribution was similar to the tip or random distribution. Reminding the reader that $svr9$ and $svr10$'s input variables were CP-CA-DV-DA-AV-WL and CP-CA-DV-DA-AV respectively, the two distribution were similar at a 5% significance level. This suggests that a similar radial-distance estimation performance level can be achieved regardless of whether whisker length is included or not in the set of model inputs.

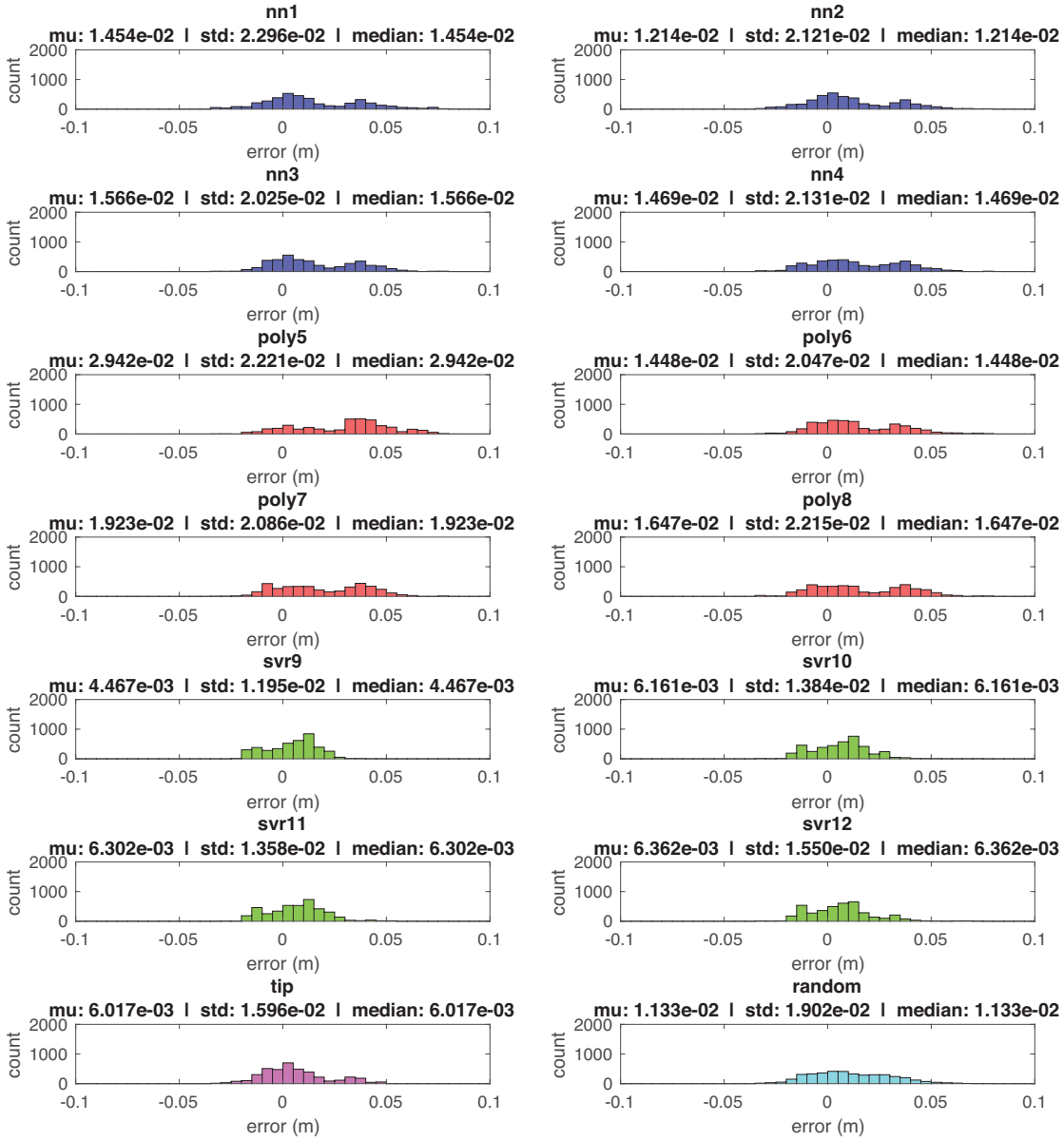


Figure 4.8: Histograms detailing the distribution of contact-point to model face distances. A tight distribution centered on zero would suggest that the model performs well with respect object reconstruction fidelity. It can be seen that based on the lowest median value followed by the lowest standard deviation, the two best performing models are svr9 and svr10. A two-sample Kolmogorov-Smirnov test performed on the models indicated that using a 1% significance level the distributions are not similar to the tip or random distributions. At a 5% significance level they are, however, similar to one another. Since the difference between svr9 and svr10 is that svr10 does not include whisker length as an input variable, their being similar would suggest that leaving out whisker length would not cause significant deterioration in the model's performance.

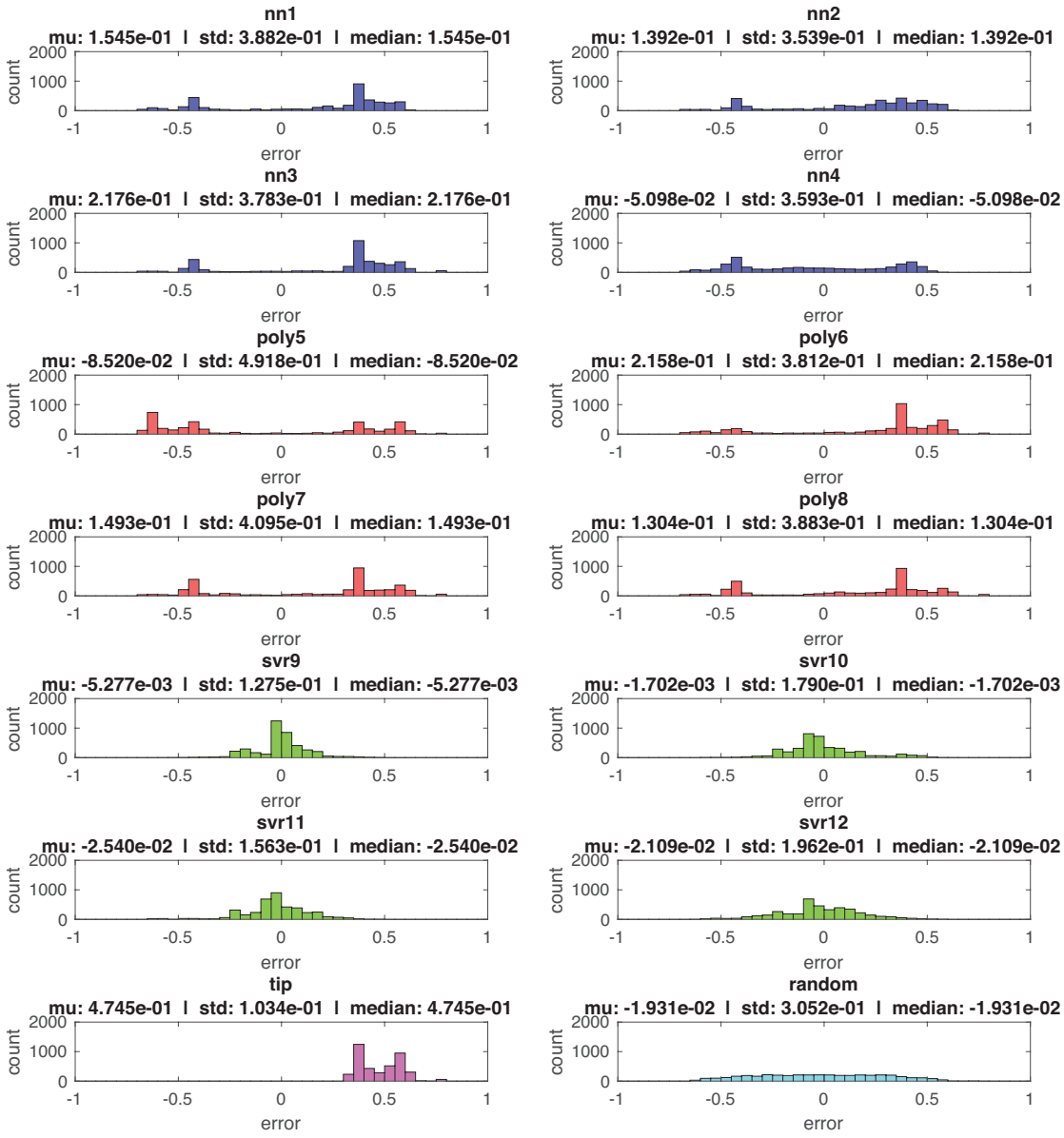


Figure 4.9: Histogram of the radial-distance errors, where the error is defined as the whisker-length normalized difference between the model-estimated radial-distance and the simulated ground-truth radial-distance. A tight distribution would suggest good model percision where are one that is centred on zero would suggest good accuracy. The comibination of a tight and zero centred distribution is desired as this would suggest both good accuracy and percision. It can be seen the the svr models perfrom the best as indicated by their low standard deviation and median that is close zero.

The superiority of the two support vector regression models are further highlighted by their radial-distance error distributions that again display the best performance as noted by their combined low values of median and standard deviation. In comparison to the tip-based radial-estimation method, the $svr9$ and $svr10$ models are able to better predict the radial-distance according to the simulated ground truth. A two-sample Kolmogorov-Smirnov test was also carried out to compare the similarity of the two models' distributions against the tip, random and each others' distributions ($svr9$ vs. $svr10$ and vice versa). It was found that at a 1% significance level, neither model shared any similarity with the tip, random or with each other. Since $svr9$ and $svr10$ are not similar, and with $svr9$ (which includes whisker length as an input variable) having a lower median error and tighter distribution, it would suggest that including whisker length would result in a better performing model according to the radial-distance error distributions. This better performance is however very moderate as the difference in median values is less than an order of magnitude, which is significantly less than the performance loss seen in models $svr11$ and $svr12$.

Since the radial-distance error is measured against a simulated ground truth, which itself contains some error (as shown in Figure 4.3), a higher weight could be placed on the results of the fidelity distributions, which suggests that with the set of features being 'CP-CA-DV-DA-AV', the benefit of including whisker length (WL) is negligible, particularly when attempting to faithfully reconstruct an object's shape with whisker-sensors.

Reminding the reader that $svr9$ and $svr11$ contain whisker length, and their features are 'CP-CA-DV-DA-AV-WL' and 'CP-CA-DV-DA-WL' respectively, while svr10 and svr12 do not contain whisker length and their features are 'CP-CA-DV-DA-AV' and 'CP-CA-DV-DA' respectively. Looking at Figure 4.8 it can be seen that if the desire is to maintain the same performance as the best SVR model ($svr9$), while reducing the number of features, the available options are omitting either angular velocity or whisker length. Since angular velocity is effortless to include, relative to the manual measurement of whisker length every time a replacement is needed, it would be best to include AV in the set of model inputs. It is thus concluded that given the hardware and initial goals of simplifying the whisker replacement process, the best option is to use a support vector regression model that maps 'CP-CA-DV-DA-AV' ($svr10$) to radial-distance.

4.2.2 Comparison with state-of-the-art

Looking back at Chapter 2, the state-of-the-art approach that most resembles our radial-distance machine learning approach and hardware is that of Evans et al. (2013). Here we compare the performance of our newly proposed support vector regression model and input features 'CP-CA-DV-DA' (svr12), with those proposed in Evans et al. (2013), which use contact amplitude and contact period, referred to as $f1$ and $f2$ respectively in their text, and a 5th order polynomial model. Our comparison omits AV from the set of independent variables since the data set that we obtained from Evans et al. (2013) does not include whisker angle sensor values. Omitting AV would result in our radial-estimation model being less than optimal, however, if the model does indicate better performance than that in Evans et al. (2013) it can safely be said that our proposed model and feature combination would result in better performance than that proposed in Evans et al. (2013).

The data set obtained from Evans et al. (2013) is specifically for the X-Y positioning robot, with the whisker material being of Acrylonitrile butadiene styrene (ABS) material, as opposed to our whisker's NanoCure RC25 material. The whisker deflection sensors on the whisking module in Evans et al. (2013) is similar to our Melexis MLX90333 Hall effect sensors. The X-Y positioning robot simulates whisker contact by driving a rod linearly into a fixed whisker, retracting once a deflection threshold is reached. The data set included 101 different radial distance ranging from 80-180 mm, 26 speeds ranging from 36-216 mm/s and each combination being repeated 4 times, bringing the total number of samples to $101 \times 26 \times 4 = 10504$. The data set included one whisker of length 185 mm, and 2 mm to 0.5 mm diameter from the base to the tip. Our learning data set included multiple whiskers with lengths 65, 78, 87, 88, 94, 107, 113 and 137 mm, each tapering from 2 mm to 0.6 mm.

Model training and testing was carried out by randomly splitting the contact data set into 70 and 30% respectively. The set of input features $f1$ and $f2$ are labelled as 'Evans features' and 'CP-CA-DV-DA' as 'Our Features'. The box plot allows for a visual detection of median difference at a significance level of 5% and if the box's notches do not overlap, the medians (illustrated by horizontal red lines) can be said to be different with a 95% confidence level.

To observe the effectiveness of the solution proposed in Evans et al. (2013), Figure 4.10 plots the distribution of errors when training and testing a particular model and input feature combination on a whisker with a specific length. For every case, it can clearly be seen that the SVR models result in significantly lower errors with respect to

the polynomial models. The effect of feature input is more subtle, and in most cases our proposed set of feature inputs result in lower errors. Highlighting the results from the Evans et al. (2013) data set, it can be seen that the best performance is resulted from our combination of proposed regression type and input features, with the median of the polynomial and 'Evans features' combination being 3.3651% and the median of the SVR and 'Our features' combination being 2.7671 %.

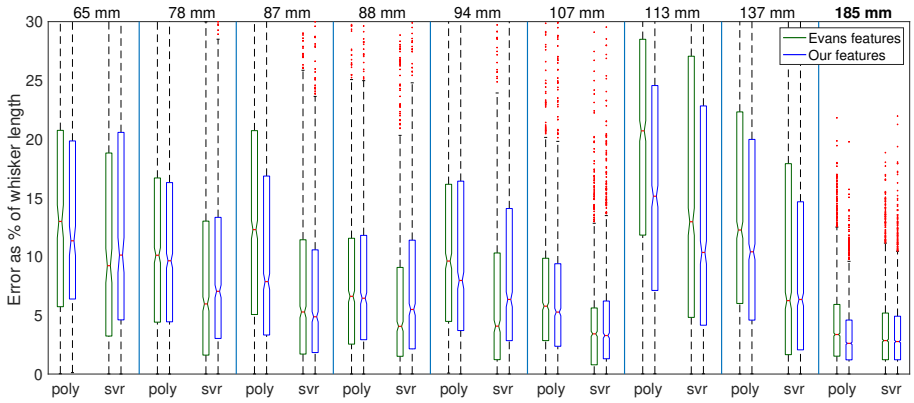
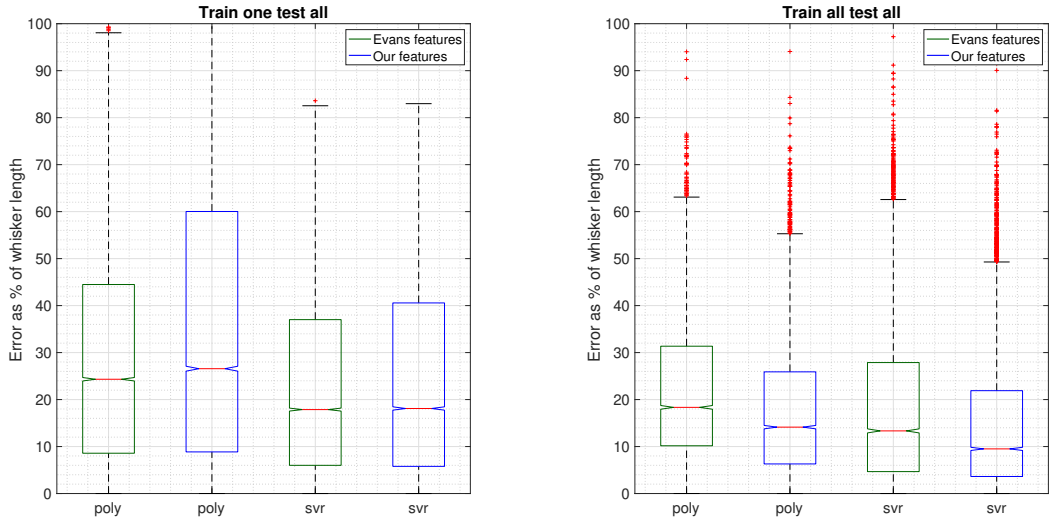


Figure 4.10: Box plots showing the distribution of radial-distance estimation error (as a percentage of whisker length) for each model and input feature combination, when trained and tested on a whisker with a specific whisker length. Note that the y-axis is capped to 30% of the whisker length so as to better discern the characteristics of the distributions. The results of the data set from Evans et al. (2013) is highlighted at furthest right of the Figure, which is the 185 mm whisker.

The polynomial model that is produced in the work of Evans et al. (2013) is trained on a whisker of a unique length, and unlike our work, does not try to generalize the model to work with whiskers of varying lengths. We continue this comparing of solutions by replicate the methodology described in Evans et al. (2013) and train a 5th order polynomial model on a whisker with a unique length, and observes its accuracy in radial-distance estimation for contacts occurring on the same whisker, and those occurring on whiskers with different length. To compare the effectiveness of the proposed solution in Evans et al. (2013) with that of ours, the polynomial model is trained using their $f1$ and $f2$ contact features as input, as well as, our 'CP-CA-DV-DA' feature input. Furthermore, we investigate the effect that regression model type has on the radial-distance estimation accuracy.

Figure 4.11 thus shows two scenarios: Figure 4.11a shows the absolute error, as a percentage of whisker length, when the models are trained on a whisker of a unique length

(in this case 107 mm) and is tested on whiskers of varying length. Figure 4.11a, however, trains and tests the models on whiskers of varying lengths. It can clearly be seen that training needs to be done on whiskers of varying lengths so as to enable generalization to varying lengths. Furthermore, the figures clearly show a significant improvement in accuracy when using the SVR model to carry out radial-distance estimation while maintaining whisker-length generalizability, which is indicated by the lower median of absolute errors. Another observation is the improvement that our set of feature inputs has on the performance of the estimation model when trained and tested on whiskers of varying lengths.



(a) Training on one whisker, testing on whiskers of varying lengths.

(b) Training and testing on whiskers of varying lengths.

Figure 4.11: Box plots showing the distribution of radial-distance estimation error (as a percentage of whisker length) for two scenarios that attempt to highlight the generalizability of the proposed solutions, from our work and that of Evans et al. (2013), in regards to whisker length. Figure 4.11a focuses on the scenario where a model is trained on a unique whisker (of length 107 mm in this case) and is set with radial-distance estimation on whiskers of varying lengths, while figure 4.11b is both trained and tested on whiskers of varying lengths. It can clearly be seen that the SVR models perform better than the polynomial models, as well as the significant improvement in whisker length generalization when the model is trained on whiskers of varying lengths.

In conclusion these results clearly show an improvement of radial-distance estimation accuracy brought on by employing a support vector regression model with our proposed

set of input features.

4.3 Discussion

The overall motivation for this work is to evaluate and develop algorithms to enable a whiskered robotic platform to navigate more effectively through complex, cluttered environments. The two principle challenges to overcome are somewhat in conflict with each other; firstly, the range and sampling frequency of the whisker sensor array is limited to the length of the whiskers and the rate at which the array is whisked; and secondly, interpreting the world as a rich metric map of occupancy derived through whisker contacts is memory intensive, thereby, limiting the size of the state space that can be represented and explored using bounded computational resources.

Addressing the first challenge, if whiskers do not make contact during their whisk cycle the only information that can be derived is that the space through which those whiskers traveled was unoccupied. Therefore, when the platform encounters an object, as observed in the behavior of whiskered mammals, it should attempt to maximize the number of whisker contacts. Further, the information content from each contact should be fully exploited to maximize likelihood of correctly identifying the spatial region in which the whiskered platform is located.

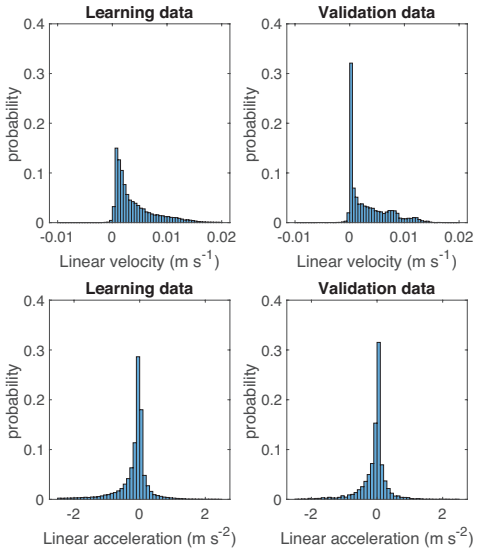
This leads into the second challenge of how to represent that spatial region efficiently such that a robot can use the information to navigate effectively through a large state space. One approach is to adopt a topo-metric representation of the global space (Bazeille & Filliat, 2011), in other words, rich metric representations of small regions of space connected topologically through sparsely represented regions.

To realize such a scheme, the problem of identifying unique *landmarks* through the local 3D reconstruction of objects encountered using a point-cloud of contact points has been focused on. Toward this end, it has been demonstrated that the fidelity of such a reconstruction can be improved by training a regression algorithm to accommodate variation in whisker morphology and contact dynamics in determining the radial distance to contact. This is highlighted in the fidelity distribution plots in Figure 4.8 on page 92 where the validation data-set, which included whiskers not included in the training data-set, resulted in the support vector regression models outperforming the crude tip-based assumption, a uniformly random distribution, neural network models and polynomial regression models. Similarly, these fidelity distributions also include contacts that occurred under a broad range of whisker dynamics as is clear from Figure 4.12 on page

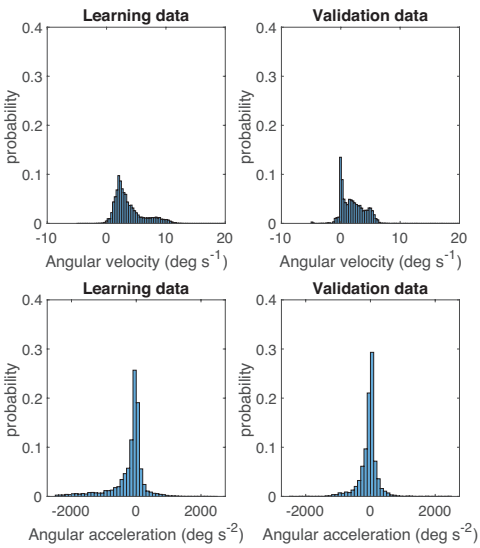
100, which plots the range of linear and angular velocities and accelerations measured at the points of whisker contact.

The limitation of the support vector regression model is that it does not account for the movement of the whisker-array. Movement of the whisker-array would significantly alter the sensory features extracted at the time of contact and would therefore affect the radial-distance estimate. The current approach is to fix the whisker-array at a particular pose whenever a sample is to be taken. Additionally, a whole whisk-cycle needs to occur before any estimate can be carried out, and the sampling frequency is therefore mainly limited by the array's whisking-frequency. This is unlike the proposed method in the work of Huet et. al (Huet et al., 2017). Figure 4.12 on page 100 also illustrate how the range of whisker contact dynamics did not greatly differ between the training and validation phases, it therefore cannot be said that the model generalizes to whisker dynamics beyond those experiences in the training phase.

Nevertheless, it has been shown that, given this work's hardware configuration, the system is able to reconstruct the shapes of contacted objects to a higher accuracy and precision than the tip-assumption that was used in Chapter 3.



(a) Histogram illustrating the probability of a particular linear velocity or acceleration occuring at the time of contact within either the training or validation data set.



(b) Histogram illustrating the probability of a particular angular velocity or acceleration occuring at the time of contact within either the training or validation data set.

Figure 4.12: A comparison between the distributions of linear and angular velocity and acceleration during time of contact for the Validation and Training data-sets. It can be seen that the validation data set does not include any significant number of contacts that resulted in whisker dynamics beyond the range of that experienced in the training data set. However, both phases include occurrences in which whisker contact was made under a range of whisker dynamics, which would suggest that the SVR model is able to estimate radial distance given the range of whisker dynamics shown in the above plots. The linear components are dependent on the radial distance to contact and thus, to illustrate that the work did vary the whisking frequency during the different phases of data collection, the histograms detailing the distribution of angular velocities and accelerations within each data set are included.

Chapter 5

WhiskerRatSLAM

This chapter is split into two section: the first describes the WhiskerRatSLAM architecture, and the second section describes how well the algorithm performs with respect to 6D localization precision, and object recognition. The second section is based on previously published work (Salman & Pearson, 2018).

WhiskerRatSLAM's process is very similar to RatSLAM in that it maintains the concept of pose cells for determining belief in a particular pose, as well as the generation of a topological like map that includes a distribution of *experience* nodes that each encapsulate a unique set of observations and pose state. The major differences, however, are the increase in dimensions that the algorithm has to consider, and the type of features that are used for the purpose of feature recognition.

The second major difference between WhiskerRatSLAM and RatSLAM, is that the latter assumes for a single visual based feature describing a single observation. WhiskerRatSLAM on the other hand uses two whisker-sensor features for describing a single observation. As such, the mechanism of processing sensory data as well as evaluating feature similarity must be modified.

5.1 Algorithm Architecture

Reminding the reader that WhiskerRatSLAM is intended to be used as a 3D object recognition algorithm, and with the intention of using it in conjunction with the planar RatSLAM algorithm, like OpenRatSLAM (Ball et al., 2013), for navigation. The purpose of WhiskerRatSLAM is thus, to provide the lower level navigational algorithm with higher level feature identification, which in this case is object recognition. Figure 5.1 provides a general guide as to how these algorithms are to be fused together in order to

create a more robust navigational architecture for a whiskered robotic platform.

With the whisker-array mounted to the end of a 6 degrees of freedom arm manipulator, the nature of the sensory platform requires it to be manipulated across the surface of an object in order to extract contact features — WhiskerRatSLAM would need to operate in 6-dimensions so as to be able to consider the observations made by the whisker-array from all possible poses. This would therefore require modifications to be made to the RatSLAM's original pose cell, experience map and experience node structures.

WhiskerRatSLAM's process can be outlined clearly by looking at its main function, which is shown in Algorithm 2.

Algorithm 2: WhiskerRatSLAM function

1 function wratslam $\{\mathbf{P}_i, \mathbf{E}_i, \mathbf{F}_i, \boldsymbol{o}_i\}$;
Input : Set of pose cells for current iteration $\mathbf{P} = p_{x,y,z,\alpha,\beta,\gamma} \in \mathbb{R}^{m\times m\times m\times k\times k\times k}$
Set of experiences for current iteration $\mathbf{E}_i = \{e_1 \dots e_z\}$
Set of features for current iteration
$\mathbf{F}_i = \{\mathbf{f}_1 \dots \mathbf{f}_n\} = \{[pfh_1, sda_1] \dots [pfh_n, sda_n]\}$
Odometry input for current iteration $\boldsymbol{o}_i = [\boldsymbol{v}, \boldsymbol{\omega}, \Delta t]$
Output: Set of pose cells for next iteration $\mathbf{P}_{i+1}$
Set of experiences for next iteration $\mathbf{E}_{i+1}$
2 $matched\_idx = match\_features(\mathbf{F}_i)$;
3 $\boldsymbol{f}_{matched\_idx} = \mathbf{F}_i(matched\_idx)$;
4 $\mathbf{P} = feature\_excitation(\mathbf{P}_i, \mathbf{F}_i)$;
5 $\mathbf{P}_{i+1} = path\_integration(\mathbf{P}, \boldsymbol{o}_i)$;
6 $best\_pose\_cell\_location = get\_best\_pose\_cell\_location(\mathbf{P}_{i+1})$;
7 $\mathbf{E}_{i+1} = process\_object\_map(\mathbf{P}_{i+1}, \mathbf{E}_i, \mathbf{F}_i, \boldsymbol{o}_i, \boldsymbol{f}_{matched\_idx})$;
8 **return** $\{\boldsymbol{P}_{i+1}, \boldsymbol{E}_{i+1}\}$

The Front End section covers how the set of features obtained from the whisker-array are processed as well as how they are compared with features of similar form, so as to determine their similarity. The Back End section continues to describe how the grid of pose cells are effected by the recognition of features as well as by the direction and intensity of motion described by the odometry. The section also includes the description of the process used for calculating the *best pose cell* location, which is the averaged location of the highest active pose cell within the grid of pose cells i.e. the algorithms best estimate for pose. The final part of the Back End section describes the 6-dimensional object map that is generated from the mapping process, including the description of the complex experience nodes and the methods used to process their pose within the map.

The following description focuses on the differences of WhiskerRatSLAM, in com-

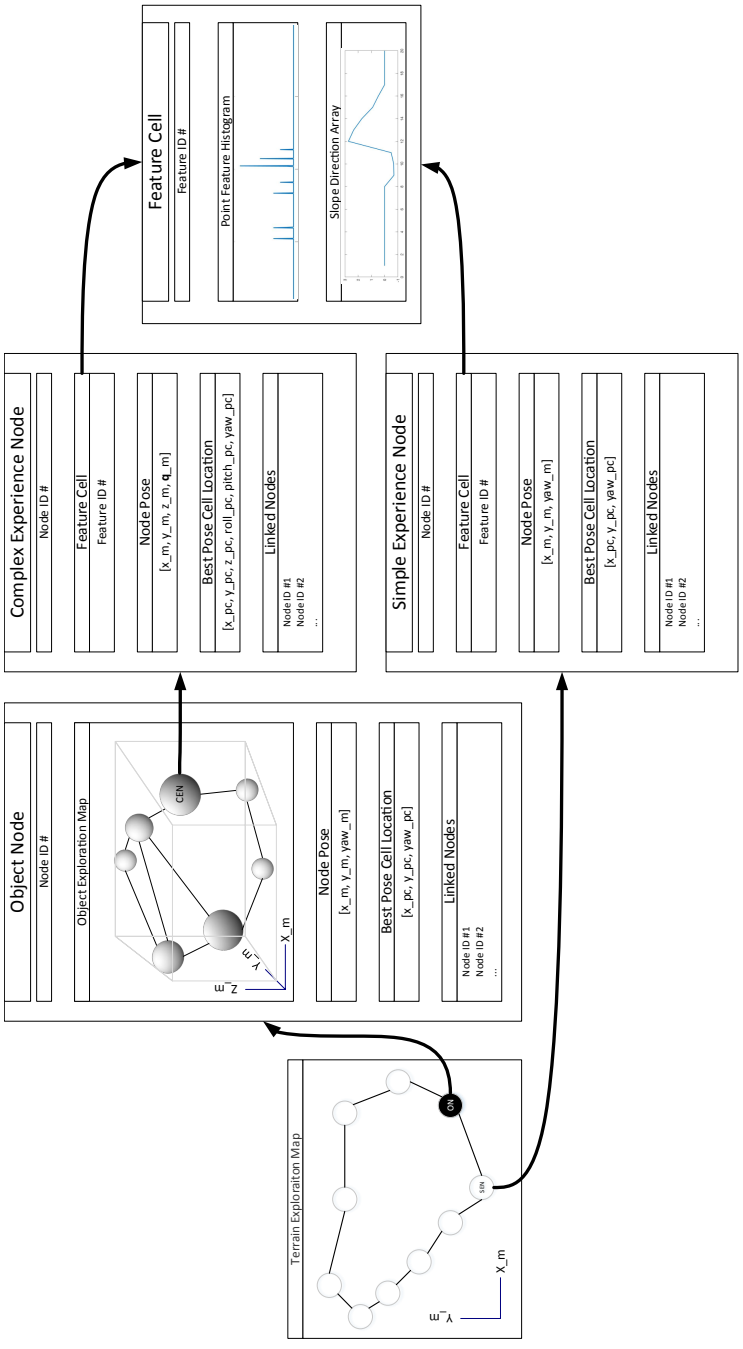


Figure 5.1: Proposed navigational framework that combines RatSLAM for planar navigation and Whisker-RatSLAM for object recognition to result in a more robust landmark based SLAM algorithm for whisker-tactile sensing. The following work focuses on Whisker-RatSLAM and how an object may be identified during future encounters by first characterizing it's surface's features by generating an *object map*. Characterization involves the mapping of unique geometrical features within a *feature cell* to each region on an objects surface. The *feature cell* includes two vectors: *Point feature histogram* and *slope distribution array*, which describe the contact location and surface slope respectively at a particular whisker-array pose.

parison to RatSLAM (Milford & Wyeth, 2008) , and details the modifications made to the OpenRatSLAM algorithm (Ball, 2018b) so that it may be used with the whisker-tactile sensor array. This modification includes the changes made to the front end of OpenRatSLAM, which is the subsystem responsible for the processing of sensory data and external cues.

5.1.1 Front End

With its current implementation (Ball, 2018b), OpenRatSLAM relies on images captured by an on board set of cameras. A visual template represents the observation at each given iteration, and is generated by first pre-processing the raw image.

OpenRatSLAM includes pre-processing the input image by converting it into grey scale, cropping the resulting image to a region that is most informative, flattening the image to a vector by summing up each column in the cropped image, and finally, normalization of the resulting array by its sum. The vector that results from these pre-processing steps is thus defined as a visual template. Visual templates are compared with one another by calculating the sum of absolute differences, where a user-defined threshold is set to determine whether one visual template matches another. To consider slight variations in the pose of the camera, the vectors are shifted in the positive and negative directions and compared at each instance with one another. For example, if an image taken again from a slightly more right position, it would be better matched with the correct visual template once an appropriate offset is applied and the values align more precisely.

5.1.1.1 Features of Whisker Perception

In the case of WhiskerRatSLAM the set of external features are the: contact locations and the surface slope at the point of contact. The contact locations estimated for each whisker in the sensor-array, following a single whisk-cycle, is collected and represented as a point cloud. The point cloud is reduced to a 3D feature descriptor called a point feature histogram (PFH) that is in the form of a 1-dimensional vector. The estimated slope, or orientation, of the contacted surface is again collected for each whisker to produce an overall slope distribution that is experienced across the whole whisker-array. This distribution of slope values across the whole array is represented in the form of a 1-dimensional vector called a slope distribution array (SDA).

The quality of the contact localization and slope estimates are all dependent on the accuracy of the contact time estimate. Contact time allows for the inference of whisker

angle at moment of contact and thus, the azimuth and altitude of the contact location. Slope of the contacted surface is also dependent on contact time accuracy as its value is derived using the gradient of the x-y deflection signals during the initial contact period.

5.1.1.1.1 Contact Time

Contact time is estimated by running the pre-processed y-deflection signal through the *findpeaks* (*findpeaks*, 2017) Matlab function, so as to determine the location and characteristics of any prominent peaks. Pre-processing includes running all whisker sensory signals through a moving average filter with a span of 100 samples. The y-deflection was used specifically because it was the direction parallel to the motion of the whisker, and thus had the largest variation in magnitude given the occurrence of a contact.

Running the signal through the *findpeaks* function, the important set of variables needed to result in a good detection rate are: the minimum peak width (100 samples in our work), minimum peak prominence (500 count in our work, where count is the unit describing the value output from the sensor's analogue to digital converter), and the limit of there being 1 peak within a single whisk-sample.

Using the location of the peak, the time of contact (contact onset) is determined by back tracking towards the instance closest to the peak in which the derivative of the signal changes sign. A similar approach was used for determining the time at which the contact was lost (contact offset), which included finding the first instance after the contact peak that resulted in first derivative of the signal changing sign. The final check to confirm the occurrence of a contact includes checking whether the distance between the peak's height and the mean of the free-whisking deflection signal is above a set threshold. The free-whisking deflection signal is selected as the final 100 samples of the deflection signal where no contact would likely occur.

Given the calculation of contact time, the azimuth and altitude of the contact location can be estimated. The following section continues to describe how a more precise estimate of contact location is calculated.

5.1.1.1.2 Contact Localization

Contact location is determined using three steps:

- Obtaining the transformation matrix relating the base of the robotic arm to the base of the whisker that is under consideration

- Determining the angle of the whisker at contact onset

- Estimating the distance from the base of the whisker, to the contact point along the whisker shaft (radial-distance to contact)

The 4×4 homogeneous transformation matrix needed to determine the location of whisker i's shaft's base W_i, relative to the robotic arm's base B is T_{BW_i}. The matrix can be derived using the following equation: $T_{BW_i} = T_{BE}T_{EW_i}$, where T_{EW_i} describes the transformation matrix relative the frame of the end-effector to the whisker shaft's base. The CAD model of the whisker-array, illustrated in Figure 2.15, is used to derive the values of T_{EW_i} for each whisker. To derive T_{BE}, the ROS package MoveIt uses forward kinematics to determine the current pose of the end effector, relative to the arm's base, by taking into consideration the dimensions of the arm's links and joint positions.

Once T_{BW_i} is calculated, the coordinate frame needs to be rotated such that it reflects the angle of the whisker shaft's base at time of contact. The angle is estimated by extracting the whisker angle sensor value at contact onset. In this work's case, the frame represented by T_{BW_i} needs to be rotated about the z axis $R_z(\theta)$, with $\theta = 50°$ placing it in its maximum protraction limit and $\theta = -50°$ in its maximum retraction limit. The Rotation matrix is defined as a 4×4 matrix with the form:

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

The pose of the whisker shaft's base at time of contact t_c is thus $T^{t_c}_{BW_i} = T_{BW_i}R_z(\theta^{t_c})$.

The length of the whisker shaft projects along the y-axis given this work's coordinate frame set up. Given a whisker-tip radial-distance estimation method, the contact location would be the Cartesian coordinate found in the matrix $T^{t_c}_{BWc_i}$, where the subscript Wc_i refers to the contact point along the shaft of whisker W_i. The matrix is calculated according to the following equation: $T^{t_c}_{BWc_i} = T^{t_c}_{BW_i}T_y(r)$, where r is the radial-distance value and the translation matrix is:

$$T_y(r) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

$T^{t_c}_{BWc_i}$ thus refers to the position of the contact point detected by whisker i. This process is repeated for all the whiskers within the array to come up with a set of 3-

dimensional points that would ideally characterize the shape of the given region on the objects surface.

The set of contact points need to be compared to a database of point sets, each of which represent a particular surface region. Taking into consideration the noisy nature of the sensory system, the scarcity of contact points, and the need for a computationally efficient algorithm, research concluded that a point feature histogram (Rusu et al., 2008) would be a suitable 3-dimensional feature descriptor for this work's whisker-array contact data. The method under which a PFH is generated based on a collection of 3-dimensional data points is described in Chapter 2.

The second geometrical feature that is extracted from each whisker-contact is the slope of the contacted surface, which is inspired from the work of (D. Kim & Möller, 2007), and is used in combination with the point feature histogram to characterize the geometrical distribution of a whisked surface region.

5.1.1.1.3 Contacted Surface Slope Kim describes in their work (D. Kim & Möller, 2007) that the slope of a contacted surface may be inferred by the gradient of the function describing the change of the y-deflection signal against the x-deflection signal, during the initial contact period.

Figure 5.2 on page 108 is included to better demonstrate the method of inferring contacted surface slope. The figure includes the sensory data extracted from a single whisker where a contact was made. The deflection signal along the sensor's x-axis (x-deflection) is plotted against the deflection signal along the sensor's y-axis (y-deflection) signal, and is shown in the bottom plot. The plot highlights the region that is considered as the initial contact period. The initial contact period is defined as the time between the estimated time of contact and the time at which the y-deflection peaks. A line is fit to these points and resulting gradient is extracted, which is then used as a measure of surface slope. This process is completed for each whisker in the array and values are concatenated into an array that is referred to as the Slope Distribution Array (SDA). The SDA is made up of 18 elements, which corresponds to the 18 whiskers on the sensor-array. For each whisk-cycle a single SDA is produced.

To measure similarity against other SDAs a normalized mean square error is used, which is defined in equation 5.3. The vectors $\mathbf{x}$ and $\mathbf{y}$ are the pair of SDA vectors being compared and $\mathbf{0}$ is a zero vector with the same dimensions as $\mathbf{x}$ and $\mathbf{y}$. MSE refers to the mean square error and is defined in Equation 5.4. The operator $\|\|_2$ refers to the

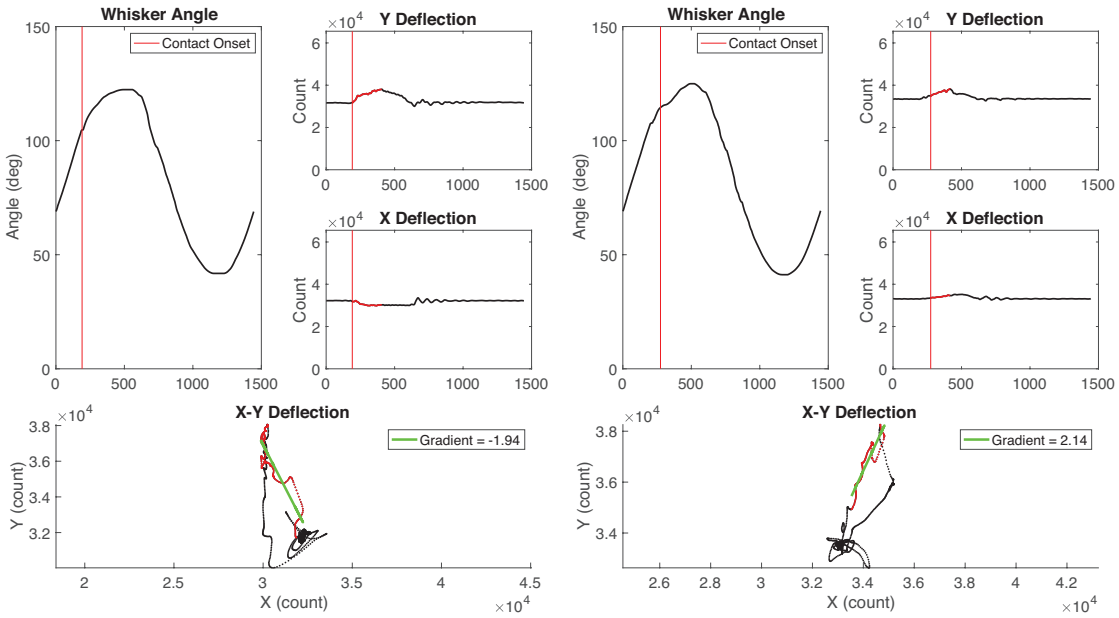


Figure 5.2: The figure includes 2 different moments in which a particular whisker made contact with a surface of different curvature. The angular data of each whisker is included through out the whisk-cycle, along with the whisker's X and Y deflection signals. All three signals are plotted against the sensor sample number. The bottom figure, however, includes a plot of the x-deflection signal against the y-deflection signal for the entire whisk-cycle. The region that is considered as the initial contact period is highlighted in red, which shows that by considering that range of points in the X-Y plot, the gradient value can be inferred from the line fit to the data (highlighted in green). According to (D. Kim & Möller, 2007), the calculated gradient is proportional to the surface orientation. The initial contact period is defined as the time between the estimated time of contact and the time at which the y-deflection peaks. The gradient value that is calculated will be included in the whisker-array's slope distribution array (SDA), which is a vector whose elements corresponds to each whisker's detected surface slope. The SDA in combination with the point feature histogram will thus make up the two feature vectors that are used to characterize the shape of a particular surface region covered by the whisker-array.

Euclidean norm, n is the number of elements for vectors $\mathbf{x}$ and $\mathbf{y}$.

$$NMSE(\mathbf{x}, \mathbf{y}) = \frac{MSE(\mathbf{x}, \mathbf{y})}{MSE(\mathbf{x}, \mathbf{0})} \tag{5.3}$$

$$MSE(\mathbf{x}, \mathbf{y}) = \frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{n} \tag{5.4}$$

5.1.1.1.4 Region similarity Each surface region o_i is represented by a point feature histogram $\mathbf{f}_i$ and a slope distribution array $\mathbf{s}_i$. To compare one region against another, Equation 2.45 is used to measure similarities of point feature histograms and Equation 5.3 to measure similarities of slope distribution arrays.

At each iteration of the WhiskerRatSLAM algorithm, the current region o_n of the latest observation needs to be compared against all previous observations $o_{1...(n-1)}$ that are considered unique and part of the previously mapped objects set.

The similarity measures for the current observation against previous observations are concatenated with each row belonging to a unique pair of regions. Iterating from 1 to $n-1$, each row would be in the form shown in Equation 5.5.

$$\textbf{Similarity}(o_n, o_i) = [\chi^2(\mathbf{f}_n, \mathbf{f}_i) \ , \ NMSE(\mathbf{s}_n, \mathbf{s}_i)] \tag{5.5}$$

The final 2-dimensional matrix, where each row compares the currently observed region o_n with a stored region observation o_i, is thresholded element-wise to determine which rows should be considered as the potential set of matching observations. Thus a row is considered if the following condition is met:

$$\text{if } \chi^2(\mathbf{f}_n, \mathbf{f}_i) < T_{pfh} \ \& \ NMSE(\mathbf{s}_n, \mathbf{s}_i) < T_{sda}$$

Where the symbol & refers to a logical And operation. The points feature histogram threshold T_{pfh} and slope distribution array threshold T_{sda} are both unique and their values were obtained through manual tuning. From these set of potential matching observations, the PFH and SDA similarity measures are summed so as to reduce the similarity vector to a scalar value:

$$SUM(\textbf{Similarity}(o_n, o_i)) = \chi^2(\mathbf{f}_n, \mathbf{f}_i) + NMSE(\mathbf{s}_n, \mathbf{s}_i)$$

The equation results in single value for each row, and the observation o_i belonging to the minimum value of $SUM(\textbf{Similarity}(o_n, o_i))$ would be considered as the matching observation.

Thus far, the front-end portion of WhiskerRatSLAM that is responsible for the processing of external cues has been described. The following section will discuss the methods used to process internal cues i.e the 6 degrees of freedom odometry.

5.1.1.1.5 Odometry Reminding the reader that one of RatSLAM's main components is its grid of pose cells, which is a 3-dimensional manifold in which the algorithm's belief in robot pose is distributed. The activity of the cells are influenced by the activity of visual cells (cells that are activated when an observation specific to one that they each represent is seen), and by the self-motion of the robot i.e. odometry.

The grid of pose cells in WhiskerRatSLAM is represented as a 6-dimensional manifold; 3 dimensions representing the position in Cartesian coordinates (x, y, z) and 3 dimensions representing orientation using Euler angles (α, β, γ).

Orientations in 3D space can be defined using different notations that include quaternions, rotation matrices and rotation vector representations (Diebel, 2006). The system represents orientation within the grid of pose cells using Euler representation so as to take advantage of the fact that it defines a particular orientation using 3 values as well as being intuitive to interpret. The latter property is desirable particularly for debugging purposes as it allowed us to better visualize whether the dynamics of this work's grid of pose cells is operating correctly.

Other representations such as the rotation vector, quaternion and rotation matrices require 3, 4 and 9 values respectively in order to define orientation. The rotation vector, although requiring only 3 values, is not intuitive in 3 dimensional form and was therefore not used to represent orientation in this work's pose cells. Rotation vectors are defined as:

$$\boldsymbol{\theta} = \theta \mathbf{e} \tag{5.6}$$

Where θ is the angle rotated about and axis defined by the unit vector $\mathbf{e}$. When expanded to an axis-angle representation, the rotation becomes more intuitive, however, the advantage of a 3 dimensional representation is lost. For these reasons orientation in the pose cells is represented using Euler angles.

For processing odometry input either the axis-angle or rotation matrix representations may be used as they are not susceptible to gimbal lock, which is a problem for Euler angle representations. Gimbal lock describes a scenario where the orientation of an object is such that any transformation using the Euler angle conventions would result in the loss of a degree of freedom and thus prevent the target orientation from being reached (Diebel, 2006). The rotational matrix representation is therefore used to

perform rotational transformations and the axis-angle representation to represent the rotational velocity of the end effector.

The odometry input $\boldsymbol{o}$ is a vector with the following form:

$$\boldsymbol{o} = [\mathbf{v}, \boldsymbol{\omega}, \Delta t] \tag{5.7}$$

Where $\mathbf{v} = [\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}, \frac{\Delta z}{\Delta t}]$ is the translational velocity and $\boldsymbol{\omega} = \omega\mathbf{e}$ is the rotation vector describing the angular velocity ω about an axis $\mathbf{e}$.

Thus far, a description of how the odometry is processed has been given and its form shown. Furthermore, a description regarding how the whisker-array 'observations' are processed into a pair of vectors that describe the distribution of the contact points (PFH), as well as the orientation of the contacted surface experienced across the whole array (SDA), has been given. Each of the two features are compared against other previously observed features using to their own specific metrics, thus providing a mechanism for measuring observation novelty.

The following section describes the back-end of the WhiskerRatSLAM algorithm that includes a description of the pose cell dynamics and experience map construction.

5.1.2 Back End

The back-end of the WhiskerRatSLAM algorithm is responsible for processing the external and internal cues of this work's robotic system. It expects an input of processed sensory features that it may use for landmark recognition as well as self motion signals in the form of odometry input so as to be able to affect its belief in robot state and map.

5.1.2.1 Grid of Pose Cells

As described in Chapter 2, RatSLAM's belief in robot pose is influenced by the activity in the grid of pose cells. The distribution of pose cell activity may be shifted, increased and decreased, according to three sources of influence: path integration, feature recognition and global inhibition mechanisms.

5.1.2.2 Feature recognition

Given the identification of a similar set of observed features, a process that is described in Section 5.1.1, the underlying RatSLAM algorithm calls for the input of additional activity to the pose cell associated with the matched features. Each unique set of features are associated with the system's current *best pose cell* location. The process for calculating *best pose cell* is detailed on page 127, however, it essentially is the average

location of the highest activated packet of activity within the grid of pose cells. Thus, when a feature $\mathbf{f} = [pfh, sda]$, belonging to the latest observation, is matched to one of those collected from previous iterations, the *best pose cell* location associated with the matched feature would be used to locate the pose cell that requires an increase in activity level.

The precise value of the activity level increase is a function of matching occurrences. If the particular feature is being matched to at a high rate per iteration, the injected activity value will decrease. If instead the rate of matching occurrences are low, the injected activity value will increase. The purpose of this mechanism is to prevent a single feature from gaining too much influence over a re-localization event (Ball, 2018b). The mechanism instead requires the coordinated effort across multiple features, as opposed to a single one, for influencing the distribution of activity in the grid of pose cells.

The formula for the activation level is obtained from the work of (Ball et al., 2013) and (Ball, 2018b), and is stated in Equation 5.8.

$$\Omega = \zeta * 1/30 * (30 - e^{(1.2 * f_{decay})}); \tag{5.8}$$

Where Ω is the activation that is added to the associated pose cell $p_{x,y,z,\alpha,\beta,\gamma}$, ζ is the maximum activation level that can be added to a pose cell and is defined by the user at run-time, and f_{decay} is the decay value that is a function of matching occurrences.

The decay value is decreased at every iteration, down to 0, so that Ω increases towards its limit ζ. Given a matching observation, the term is increased only when a match occurs. The value f_{decay} is therefore unique for each feature within the set of features $\mathbf{F}_i$.

The value of the maximum activation level ζ can be interpreted as a level of confidence associated with external observations. A higher value would therefore increase the likelihood of a re-localization occurring given fewer repeated observations, while a lower one would require more repeated observations before a re-localization can occur. In this work a value of $\zeta = 0.3$ was obtained empirically.

Given the pose cell coordinate, which is determined from the *best pose cell* location, the pose cell's activity is adjusted according to Equation 5.9.

$$p_{x,y,z,\alpha,\beta,\gamma} = p_{x,y,z,\alpha,\beta,\gamma} + \Omega \tag{5.9}$$

Thus far, the process of pose cell excitation due to feature recognition has been described. It included a description of how the activation of a specific pose cell, which is associated with the recognized feature, is varied according to the rate at which the

feature is being recognized. The function *feature_excitation* is described in Algorithm 2 on page 102. Following its execution, the proceeding step includes the path integration process that involves the shifting of pose cell activity in a manner that represents the direction and intensity of the odometry motion.

5.1.2.3 Path Integration

The process of path integration can be described by returning to the 1-dimensional RatSLAM example shown in Figure 2.3 on page 16. When a particular pose cell contains any level of activity, such as the cell belonging to the 120° head direction, it elicits activity in its neighbours that is proportional to a function of its own activity and the shape of the excitation distribution. This behavior is referred to as local excitation and is the source of the Gaussian shaped packet of activity centered around each active pose cell in Figure 2.3. For the 1-dimensional example, the grid of pose cells instead is represented as a vector of head direction cells that represent the orientation γ as opposed to this work's 6-dimensional pose cells, which represents $(x, y, z, \alpha, \beta, \gamma)$.

5.1.2.3.1 Local Excitation To implement local excitation to the head direction cells computationally, the excitation connections are defined in the form of a weight matrix. For the 1-dimensional case, this would be a vector whose elements are valued according to the probability density function of a univariate normal distribution. Figure 2.3 on page 16 is recreated using the algorithm found in OpenRatSLAM (Ball, 2018b) in order to familiarize the reader with the mechanisms used in the grid of pose cells.

By noting the number of excitation connections shown in Figure 2.3, the length of the excitation vector can be worked out to being equal to 5. Assuming that the Figure illustrates the state of the head direction cells after the first application of the excitation, all head direction cells are set with an activity of 0 while the activity for the 120° cell is set to 1 (activity levels in the head direction cells are normalized according to the maximum activity, thus, the values range between 0 and 1). The state of the head direction cell activation levels after the application of local excitation according to is shown in Figure 5.3a on page 115.

To process the pose cells so that their activities reflect the effect of the excitation connections, Algorithm 3 is used.

The process is analogous to image convolution where the excitation matrix is akin to a kernel and the grid of pose cells an image. To consider the case where the excitation matrix is centered on a pose cell that results in it over extending beyond the limits of the head direction cells vector, the wrap around function $g()$ is used. Reminding the

Algorithm 3: Local Excitation Function

```
function LocalExcitation (HDCells, ExcitationVector)
        Output: ExcitedHeadDirectionCells
ExcitedHDCells = zeros(size(HDCells)) for i = 1 to len(HDCells) do
    ExcitedHDCells(g(i)) = ExcitedHDCells(g(i)) + HDCells(i). * ExcitationVector
    ;        // ".*" is element wise multiplication.  [Vector = Vector + Scalar.*Vector]
end
return ExcitedHDCells
```

reader that the head direction cells, or grid of pose cells in the higher dimension case, act as a manifold and any influences that extend beyond their borders would result in cells on the opposite side being affected. The wrap around function $g()$ therefore takes this manifold structure into account and provides a way for directing the values of over extended values to their appropriate cells on the other side. Figure 5.3 illustrates two cases where local excitation is applied onto a vector of head direction cells when the initial activation is made about the 120° cell, and another about the 0° cell that more clearly shows how the wrap around function directs activity to the appropriate cells.

To maintain this Gaussian shaped packet of activity, given no additional excitation from the path integration or landmark recognition components, a combination of local and global inhibition is required in order to prevent the over saturation of the neighbouring cells, following subsequent algorithm iterations.

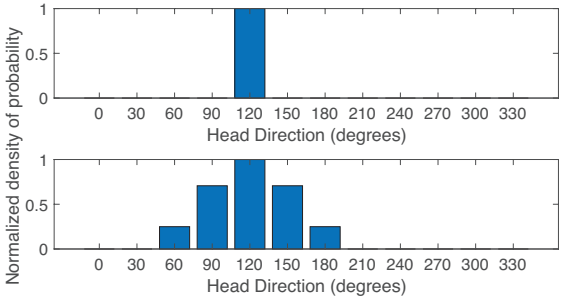
5.1.2.3.2 Local and Global Inhibition

Local inhibition is implemented in a similar manner as local excitation, however, the final vector describing the head direction cells is the difference between the local excitation output and the local inhibition output. The pseudo-code of the local inhibition function is included in Algorithm 4.

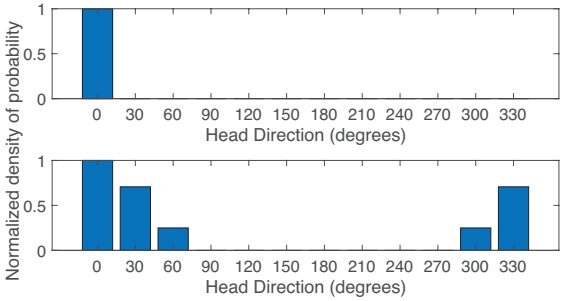
Algorithm 4: Local Inhibition Function

```
function LocalInhibition (ExcitedHDCells, InhibitionVector)
        Output: InhibitedHeadDirectionCells
Inhibition = zeros(size(ExcitedHDCells)) for i = 1 to len(ExcitedHDCells) do
    Inhibition(g(i)) = Inhibition(g(i)) + ExcitedHDCells(i). * InhibitionVector ;
      // ".*" is element wise multiplication.  [Vector = Vector + Scalar.*Vector]
end
InhibitedHDCells = ExcitedHDCells − Inhibition
return InhibitedHDCells
```

The global inhibition function is the second inhibition component that makes sure



(a) Two successive frames showing how the activity levels of head direction cells change when the excitatory connections elicit activation in cells neighbouring the initially activated head direction cell of 120°.



(b) Two successive frames showing how the activity levels of head direction cells change when the excitatory connections elicit activation in cells neighbouring the initially activated head direction cell of 0°. Note the wrapping effect around the 1-dimensional manifold structure.

Figure 5.3: This figure illustrates the change in head direction cells' activation levels following the application of the local excitation function. The local excitation function mimics the behavior that is expected from excitatory connections protruding from a cell to itself and its surrounding neighbours as shown in Figure 2.3 on page 16. Two examples are shown in which the initial head direction cell that is active is varied. Figure 5.3a recreates the state of the head direction cells shown in Figure 2.3 and initiates activity in the 120° cell, while Figure 5.3b the 0° cell. The latter is shown to highlight the effect of the wrap around function $g()$ that is responsible for redirecting activity to the appropriate cells on the opposite sides of the vector. The same principle applies to this work's 6-dimensional case.

cells that are no longer receiving any activity input, decay and cease to be active after time. The function is parameterized by a threshold value that specifies the minimum cell activity level needed before it is turned off. The threshold also defines how much activity is subtracted from all cells during every iteration. The pseudo code for the global inhibition function is shown in Algorithm 5. Global inhibition is applied after local excitation and inhibition, according to (Ball, 2018b). The value for the global inhibition threshold has been kept equal to that used in Ball et. al's algorithm, which is $2e^{-5}$.

Algorithm 5: Global Inhibition Function

```
1 function GlobalInhibition (InhibitedHDCells, GinhibThreshold)
          Output: FinalHDCells
2  FinalHDCells = zeros(size(InhibitedHDCells))
   AboveThresholdIdx = InhibitedHDCells >= GinhibThreshold
   FinalHDCells(AboveThresholdIdx) =
   InhibitedHDCells(AboveThresholdIdx) − GinhibThreshold
3 return FinalHDCells
```

In the case of this work's 6-dimensional grid of pose cells, the local excitation/inhibition is implemented using a 6-dimensional matrix that also takes form of a Gaussian distribution. The excitation and inhibition is applied in a similar manner to that described in Algorithm's 3 and 4 respectively.

In order to maintain a stable packet of activity, that is, an activity that does not lead to the saturation of all pose cells as time progressed, manual tuning needed to be carried out so as to determine the values for the two matrices' dimensions and distribution variances. This process was carried out on a grid of pose cells that was cut off from any path integration or landmark recognition influence.

This stable packet behavior is illustrated in Figure 5.4 on page 118, where a 6-dimensional grid of pose cells with a centered activity about the grid's midpoint is plotted. The 3 dimensions of orientation are segmented into $5 \times 5 \times 5$ individual blocks of $5 \times 5 \times 5$ grids representing translation. Pose cells within each individual block are fixed with respect to their (α, β, γ) values, however, the cells vary with respect to their (x, y, z) values. The image highlights the state of the grid of pose cells a few samples after initialization, which is the time needed for the system to stabilize with respect to the pose cells' activity levels - at this point the activity is centered about the initialization point $x : 3|y : 3|z : 3|\alpha : 3|\beta : 3|\gamma : 3$.

The excitation matrix is set to one with a length of 5 for each of the 6-dimensions and with a covariance matrix being an identity matrix and each dimension's variance equal

to 1. The inhibition matrix is similar in length, however it has a diagonal covariance of 2s. The values of each element within the matrix are calculated using Equation 5.10, which is derived from the standard equation for the probability density function of a multivariate Gaussian distribution (Bishop, 2006).

$$w_{x,y,z,\alpha,\beta,\gamma} = \frac{1}{\sqrt{(2\pi\sigma^2)^6}} \cdot \exp \frac{-(x-\mu)^2-(y-\mu)^2-(z-\mu)^2-(\alpha-\mu)^2-(\beta-\mu)^2-(\gamma-\mu)^2}{2(\sigma^2)^6} \tag{5.10}$$

It has been mentioned previously that the dimensions of the grid of pose cells and the excitation and inhibition matrices are dependent. Thus, any change in dimensions of the grid of pose cells would require re-tuning of the excitation and inhibition matrices' parameters. For the remainder of this work these values are fixed to the values quoted in the above text.

Following the selection of the excitation and inhibition parameters, as well as the grid of pose cell dimensions, the following section describes how pose cell activity is moved about to reflect the odometry input i.e. carry out path integration.

5.1.2.3.3 Path Integration As shown in Figure 5.4 on page 118, there are a discrete set of voxels that contain a 3-dimensional grid of pose cells. Within each voxel, the set of pose cells $P_{\alpha,\beta,\gamma}$ have a fixed set of unique orientation values (α, β, γ), while their positioning within the 3-dimensional grid determines their respective (x, y, z) values.

Path integration implemented by the RatSLAM algorithm does not involve integration of velocity over time, the work of Milford et. al (Milford & Wyeth, 2008) and Ball et. al (Ball et al., 2010) all specify that the shifting of pose cell activity is instead proportional to the magnitude of velocity.

The underlying RatSLAM algorithm assumes no noise for the motion of the robot and instead directly shifts the belief of the pose in the direction described by the odometry (Sünderhauf & Protzel, 2010) and that of a pose cell's property. This shifting of pose cell activity is split into two steps: one that involves the translational component of the odometry, followed by a shift specific to the rotational component (Ball, 2018b).

5.1.2.3.3.1 Translational shift The translational shift is implemented by first iterating through each of the unique set of orientation value (α, β, γ) found in the grid of pose cells. Given a unique combination, for example $(\alpha', \beta', \gamma')$ i.e a specific voxel shown in Figure 5.4 and focused on in Figure 5.5, the translational velocity vector $\boldsymbol{v}$ is rotated so that it reflects the motion of a robot with an initial orientation equal to $(\alpha', \beta', \gamma')$.

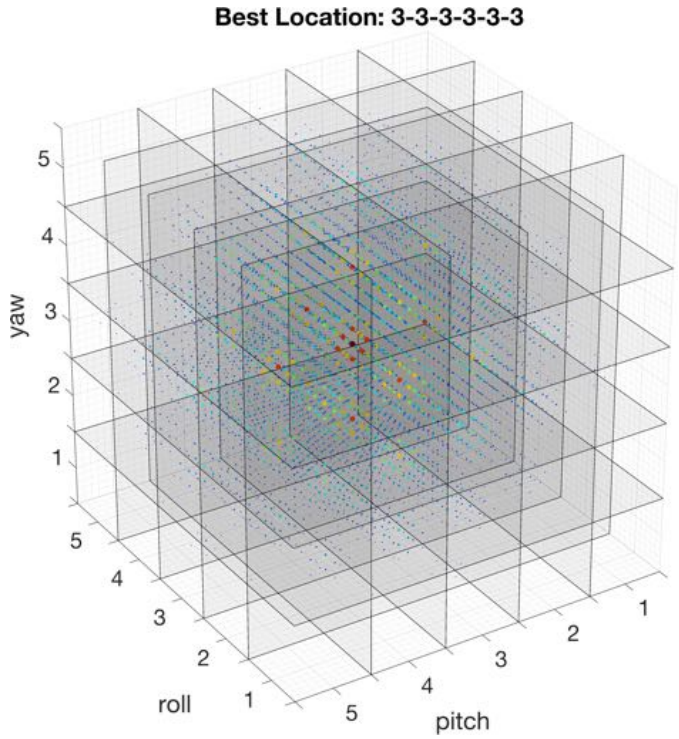


Figure 5.4: The figure illustrates a stable packet of activity centered around an arbitrary point in this work's 6-dimensional grid of pose cells. The grid's dimensions are defined to have a length of 5 for the (x, y, z) dimensions and a length of 5 for the (α, β, γ) dimensions. Each voxel that is specific to a unique combination of roll, pitch and yaw values ((α, β, γ) respectively) is segmented using planes. Thus, each unique combination of roll, pitch and yaw has its own 3-dimensional grid of cells that represent a unique combination of x, y, z.

Given a length of 5 for each of the 6-dimensions of the grid of pose cells, each step in a particular rotation dimension would represent a change of $\frac{360^\circ}{5} = 72^\circ$. Thus, for the set of pose cells $P_{\alpha',\beta',\gamma'}$, their corresponding velocity vector would be equal to:

$$\boldsymbol{v}^{\boldsymbol{T}}_{\boldsymbol{\alpha'},\boldsymbol{\beta'},\boldsymbol{\gamma'}} = \boldsymbol{R}_{\boldsymbol{\alpha'},\boldsymbol{\beta'},\boldsymbol{\gamma'}}\boldsymbol{v}^{\boldsymbol{T}}$$

Where the the superscript T refers to the transpose of the row vector $\boldsymbol{v}$ so that it results in a column vector. The matrix $\boldsymbol{R}_{\boldsymbol{\alpha'},\boldsymbol{\beta'},\boldsymbol{\gamma'}}$ is the rotation matrix representing the Euler orientation defined by the values $(\alpha', \beta', \gamma')$.

The translation is applied to the 3-dimensional grid of cells by considering it in 3 separate planar views $(x, y), (x, z)$ *and* (y, z). An example of a planar view is illustrated in Figure 5.6, it's noted that this is not a 2-dimensional grid but a 3-dimensional one, and

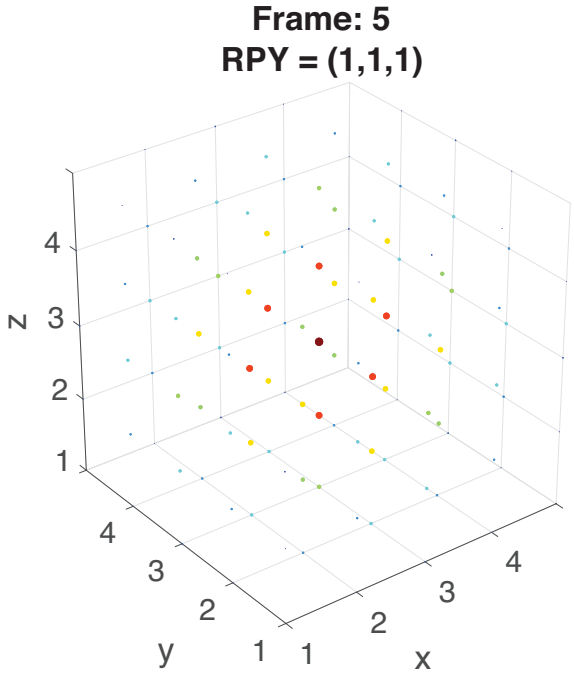


Figure 5.5: The figure illustrates a stable packet of activity centered around an arbitrary point in a set of pose cells that have a fixed orientation value i.e the set $P_{\alpha',\beta',\gamma'}$ where $\alpha' = 1,\ \beta' = 1,\ \gamma = 1'$. Given a length of 5 for each of the (α, β, γ) dimensions, the set of pose cells on display would represent an orientation of 0° for each of the roll, pitch and yaw Euler angles. An additional increment in one of the orientation dimensions would include a step of $360° \div 5 = 72°$.

the rear elements cannot be observed since the view is orthogonal. This process is done to simplify the calculations needed to shift the activity in a 3-dimensional environment by breaking down the process into 3 steps where the 3-dimensional vector's components are broken down into 3 2-dimensional vectors.

Since the pose cells are represented as a matrix, shifting of the cells is limited to 8 different directions: North, North East, East, South East, South, South West and West (N, NE, E, SE, S, SW, W). To reflect more subtle variations in direction and velocity, the activation values of the cells are spread out in a direction and intensity that is proportional to the angle of the vector and its magnitude.

This is more clearly illustrated in Figure 5.6, which includes a $3 \times 3 \times 3$ matrix that is viewed from the (x, y) plane, and a single active cell $p_{1,2,2}$ that is highlighted in black. On the right hand side of the Figure 5.6, the state of the grid of pose cells after the translation has been applied is illustrated, and the cells adjacent to $p_{1,2,2}$, and within

the direction of the translation vector, are injected with activity. In this example the velocity vector is $\boldsymbol{v} = [-0.5, 1]$ and the velocity magnitude is 0.3. Since the chosen velocity vector has higher $+Y$ component and a lower $-X$ component, the upper cell (relative to $p_{1,2,2}$) is activated more so than the left cell. The direction in which activation is spread is therefore proportional to the direction of velocity vector while the intensity of the spread is instead proportional to the magnitude of the velocity.

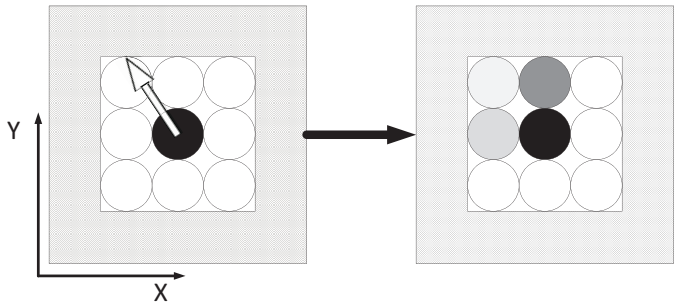


Figure 5.6: This figure presents a toy example where we illustrate, from a planar view, the shifting process due to translation in a 3-dimensional matrix. The 3-dimensional velocity vector is split into two components relevant to the current view, which is used to determine which direction the activity should be re-positioned. Given a NW vector $\boldsymbol{v} = [-0.5, 1]$, the activity of each cell is moved one step up and one step to the left. The magnitude of the activity that is re-positioned is determined by the weight matrix. This weighted shifting method that takes into account the finer details of the vector's angle as well as it's magnitude.

More specifically, the final matrix, which is shown on the right hand side, is obtained by summing 4 matrices that is referred to as the weighted shift matrices. Each matrix represents a specific direction and magnitude of the activation spread.

The weighted shift matrices are derived using a set of geometric equations that are dependent on the quadrant of the velocity vector. Given the example's velocity vector $\boldsymbol{v} = [-0.5, 1]$, its angle relative to the horizontal axis given the current view (x, y) is $\theta = acos([-0.5, 1]^T \cdot [1, 0]) \approx 2 \approx 117°$. This places the vector in the second quadrant according to Figure 5.7.

Given the NW direction of the velocity vector, there are three degrees of freedom from the original 8 shifting directions, that can be used to spread out the cells' activity. These three directions are North, West and North West. Additionally, the magnitude of the activity left in the same position can be specified. Thus, there are four shift directions that require weights. Figure 5.10 illustrates a guide to how the relative weighting of each of these 4 components can be calculated given a velocity vector in the 2^{nd} quadrant.

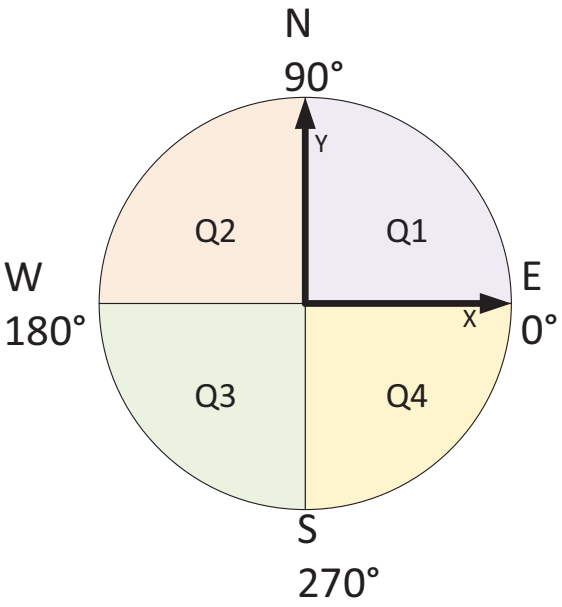


Figure 5.7: The weight matrices that define how much activity is spread and what direction it should be spread in can be calculated according to a set of geometric equations. These geometric equations vary depending on the quadrant that the velocity vector is in. Depending on the angle of the vector, or otherwise, the signs of individual vector elements, the quadrant can be determined. Given this example's vector $\boldsymbol{v} = [-0.5, 1]$, the quadrant would be 2.

The weighting related to the current position is the gray shaded box in the bottom right sector (BR). The N component in this case is referred to as the top right sector (TR), the NW the top left section (TL) and the W component the bottom left sector (BL). The weights for each of the 4 shift components are defined by the area of their rectangular sectors, which in turn is defined by the angle of the vector and its magnitude. These 4 scalar weight values $(w_{tl}, w_{tr}, w_{bl}, w_{br})$ are then multiplied their respective shifted matrices to produce 4 individual matrices as shown in the middle of Figure 5.8 on page 122.

To produce the 4 set of shifted matrices, the axis associated with a particular dimension in the matrix needs to be considered. For this example's $3 \times 3 \times 3$ matrix, the first, second and third dimensions are (x, y, z) respectively. Given a plane view of (x, y) and a shift direction of NW, the matrix would be shifted one step backwards in the first dimension (x), one forward in the second dimension (y) and zero in the third dimension (z). For a N shift direction the matrix would be shifted zero steps in the first and third dimensions, while for the second dimension it would be shifted forwards one step. The

remain component does not require any shifting and the original matrix is used.

Once the shifted matrices are generated, their elements are each multiplied by their specific weights. Looking at Figure 5.10 on page 124, it can be seen that for the N shifted matrix, the weight belonging to the top right sector should be used. For the NW shifted matrix, the top left sector's weight. For the W shifted matrix the bottom left sector's weight and finally, for the remain component, the bottom right sector's weight should be used.

The weighted shift matrices are then summed together and following the consideration of any wrap around requirements, the final $3 \times 3 \times 3$ matrix for the (x, y) plane of view is obtained.

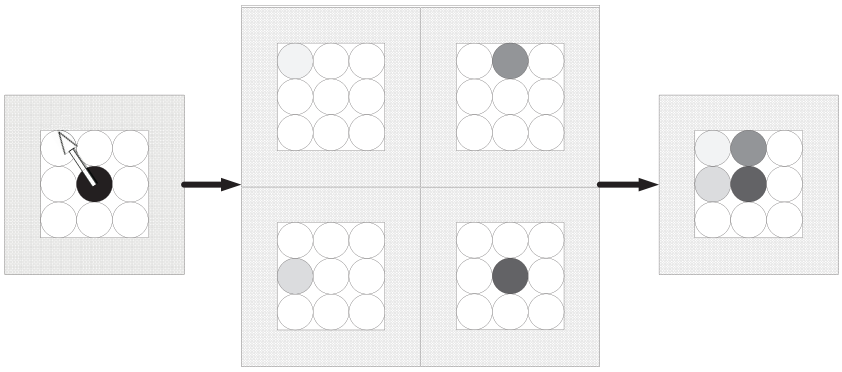
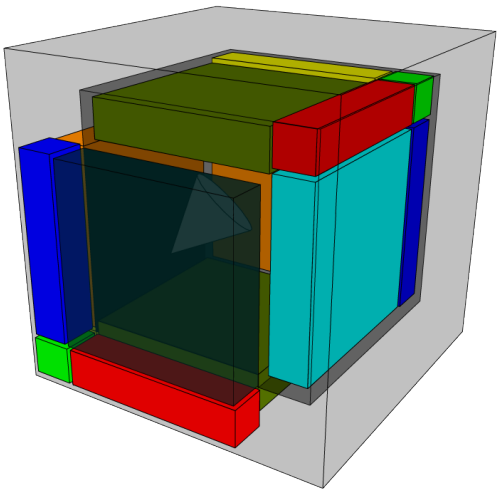


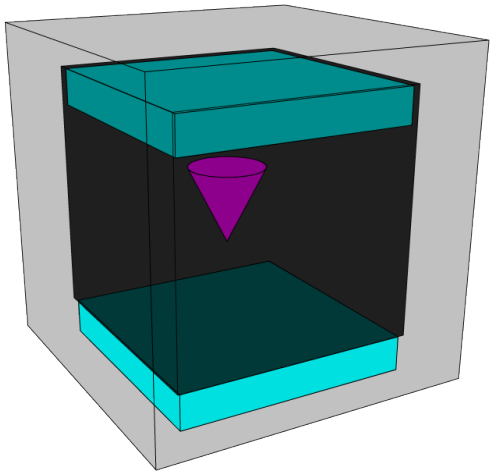
Figure 5.8: Given a NW direction of motion that places the velocity vector in the 2^{nd} quadrant, there are 3 primary directions in which the activity in the matrix may be shifted towards: North, West and North West. A fourth component is also included so as to be able to define how much of the activity should be kept in the current position. The weightings of each of these 4 components can be defined according to the areas of their corresponding sectors. The current position can be represented by the bottom right (BR) square sector and the remaining 3 directions, N, NW and W, using the top right (TR), top left (TL) and bottom left (BL) sectors respectively. Using the equations specific to the 2^{nd} quadrant shown in Figure 5.10 on page 124, the areas may be calculated and thus each of the 4 component's weights. The 4 matrices in the middle of the figure represent the shifted original image multiplied by their relevant weights. The final matrix is shown on the right hand side and is a sum of these 4 matrices.

Shifting involves moving an elements' activity from its current position, to that of its neighbour in the desired direction. To consider the case where the activity is shifted towards the outside of the matrix, a wrap around function is used to redirect the activity to the appropriate elements on the opposite side of the matrix. This process is illustrated in Figure 5.9 on page 123.

Thus far, the procedure for a single plane of view (x, y) and a velocity vector in the



(a) Wrapping activity when the velocity vector has more than one non-zero component and therefore results in activity extending towards corners.



(b) Wrapping activity when the vector's direction is towards one of the cuboid's side.

Figure 5.9: In the case of the elements that end up outside of the matrix, the wrap around function redirects them to the appropriate positions on the opposite sides of the matrix. For the typical case where the velocity vector has more than one non-zero component, and the activity is directed towards one of the matrices corners, the activity is wrapped around in manner illustrated in Figure **5.9a**. The figure shows two large cubes; the largest cube is the temporary matrix used for holding the values of the shifted activity from the smaller cube. The grid of pose cells are therefore represented by the small cube, and the wrap around function needs to re-arrange the activity that has been pushed out into the temporary matrix back into this original matrix. The cuboids that are to be re-arranged given the direction of motion are each highlighted in a specific color. Those that are shown in the temporary matrix, indicate the elements that are to be re-arranged back into their corresponding locations in the original smaller matrix. Their correspondence is indicated by their colors and it can be seen that they are mirrored along the plane perpendicular to the direction of motion. The length of the shortest side of each cuboid is a single element, while the remaining dimensions are defined such that completely cover the three sides of the smaller matrix. The direction of motion used for this example is indicated by the direction that the cone structure is pointing towards. The simplest case is when the velocity vector has one non-zero component and the activity is therefore directed towards one of the matrices sides, and non towards a corner. This case is highlighted in Figure **5.9b** where the direction of motion is towards the bottom of the matrix. Using the same logic, the method can be applied to all motions directed to any of the other 6 sides of the matrix.

second quadrant has been described. To consider other quadrants, the equations along with their respective weights sectors guide are included in Figure 5.10 on page 124. To consider different view planes the shifting of the matrix needs to take into account which dimension in the matrix belongs to which axis and shift appropriately. Thus, for the $3 \times 3 \times 3$ matrix, the first, second and third dimensions are (x, y, z) respectively. Given a plane view of (x, y) and a shift direction of NW, the matrix would be moved one step backwards in the first dimension (x), one forward in the second dimension (y) and zero in the third dimension (z). Given a plane view of (y, z) and the same shift direction, the matrix would be moved zero steps in the first dimension (x), one step backwards in the second dimension (y) and one forwards in the third dimension (z).

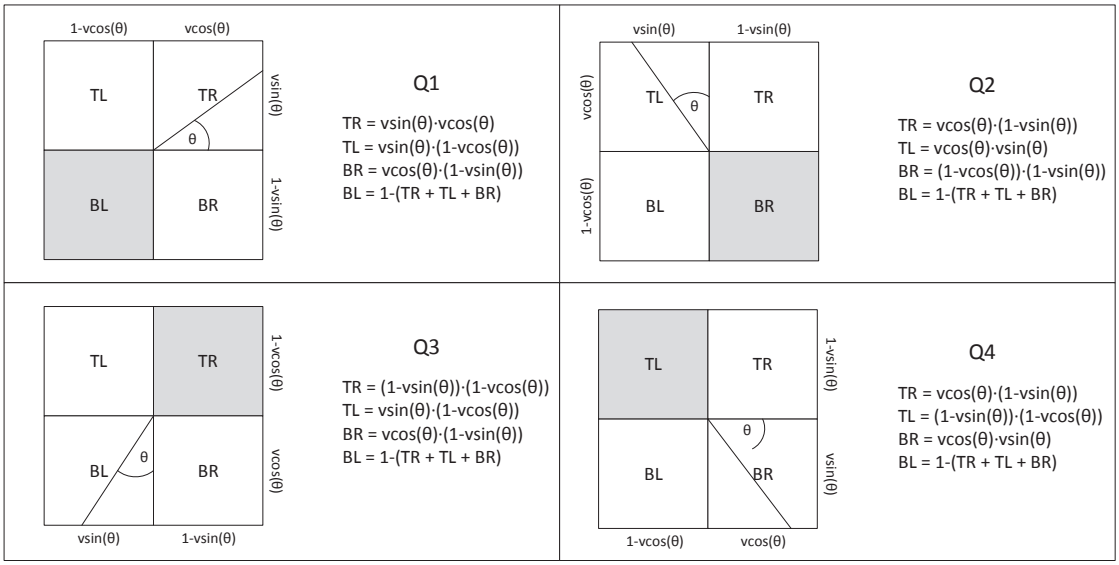


Figure 5.10: Depending on which quadrant the 2-dimensional velocity vector is in, the shift matrices' weights can be calculated according to the equations shown in this Figure. The illustrations on the left hand side show how the weights for the 4 shift components are equal to the area of their respective sectors. The darker shaded sector refers to the location that the motion is assumed to take place from. From example if the motion is in the 3^{rd} quadrant and therefore in the SW direction, 3 shifted matrices corresponding to a W, SW and S motion need to be calculated. The 4th shifted matrix would correspond to no motion and instead be used to determine how much activity should remain in it's original location. The weights that are then multiplied with the 4 shifted matrices are equal to the areas of their respective sectors. The W, SW and S shifted matrices would use the weights defined by the areas of sectors TL, BL and BR, while the remain component would use the weight defined by the area of sector TR.

Once the weighted shifted matrices have been summed together the set of pose cells

$P_{\alpha',\beta',\gamma'}$ that have undergone a translation may be obtained. The second step of the path integration process is to consider the rotational component of the odometry.

5.1.2.3.3.2 Rotational shift The pose cells represent orientation using the three Euler angles referred to as roll (α) pitch (β) and yaw (γ). By looking at Figure 5.4, it may be seen that a rotational shift corresponding to a particular angular motion can be achieved by shifting the matrix towards one of the cube's sides. Since the figure illustrates the 6-dimensional grid of pose cells as a 3-dimensional matrix, a single shift step in a particular rotational dimension would be akin to shifting the $25 \times 25 \times 25$ matrix 5 steps.

Given a length of 5 for each of the rotational dimensions, a single rotational shift corresponds to an angle change of $\delta_\gamma = \frac{360^\circ}{5} = 72^\circ$. Thus, if the odometry input included yaw angular velocity of 72°, the $25 \times 25 \times 25$ matrix would be shifted 5 steps in the positive yaw direction, which is illustrated in Figure 5.11.

To reflect more subtle variations in angular velocity, a similar weighted shift approach is applied, similar to that used in the translational shifting case. Figure 5.11 illustrates the change in activity values for each layer of pose cells that have the same yaw value. With an initial state shown on the left hand side, the 1^{st} yaw layer is fully active.

Following an input of $158^\circ s^{-1}$, the resultant matrix is shown on the right hand side, and is composed of a weighted sum of two shifted matrices. The two shifted matrices are obtained by finding the rounded down and rounded up ratio of the input angular velocity $158^\circ s^{-1}$ and the step value (72°). These two values are $floor(\frac{158}{72}) = 2$ and $ceil(\frac{158}{72}) = 3$ respectively, where the functions $floor$ and $ceil$ round down and up respectively. Two matrices that are each shifted 2 and 3 steps respectively are thus produced. The weight for the lower bound shifted matrix is equal to $1 - (\frac{\omega_\gamma}{\delta_\gamma}\%1)$ and $\frac{\omega_\gamma}{\delta_\gamma}\%1$ for the upper bound shifted matrix, where ω_γ is the angular velocity for the yaw component, δ_γ is the rotational step size in the yaw direction and the % is the modulo operator. Using these equations the weights can be calculated to being equal to 0.8 and 0.2 for the lower and upper bound shifted matrices respectively.

Multiplying each of the shifted matrices with their corresponding weight values, the next step includes summing them to produce the final matrix that reflects the example's given angular velocity input. In the case of additional angular velocity components, the same process is repeated, however, the orientation of the layers and therefore their direction of shift changes depending on which axis the specific angular component lies on. The final rotational matrix is then equal to the combination of all lower and upper

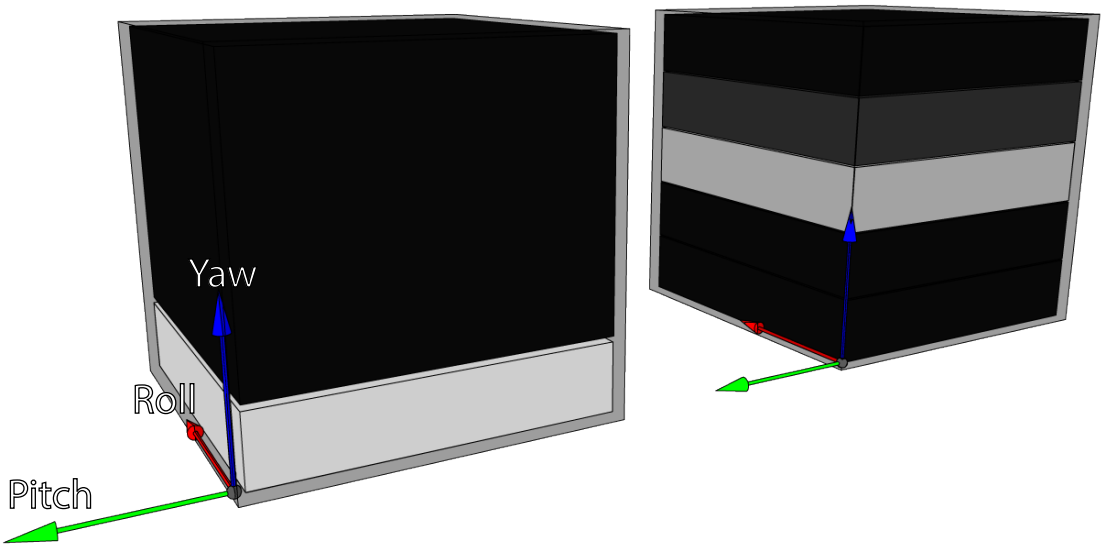


Figure 5.11: The following figure illustrates a scenario where the grid of pose cells receives an odometry input with an angular velocity of $\omega_\gamma = 158°s^{-1}$ for the yaw angle only. The initial state of the grid is shown on the left hand side with a single yaw layer being in a fully activated state. A single yaw layer includes all pose cells that have a fixed yaw value, which in this example is the first layer. The rotational shift is applied by determining the lower and upper bound shifts needed to enclose the full range of the input. The step size for both these lower and upper bound shifts can be determined from the lower and upper bound ratios of the angular velocity and the rotational shift step size. The rotational shift step size is determined by the length of the angular dimension, which in work is 5, thus, each rotation step is equivalent to $\delta_{gamma} = \frac{360°}{5} = 72°$. Given an angular velocity input of $158°s^{-1}$, the lower and upper bound ratios are $floor(\frac{158}{72}) = 2$ and $ceil(\frac{158}{72}) = 3$ respectively, where the functions $floor$ and $ceil$ round down and up respectively. The remainder of the ratio ($\frac{158}{72}\%1 = 0.2$), where % is a modulo operation, is used to calculate how much each of lower and upper bound shifts need to be weighted. Again, this is to reflect the more subtle changes in angular velocity that is less than a full rotation step. The weight for the lower bound shifted matrix is equal to $1-0.2 = 0.8$ and 0.2 for the upper bound shifted matrix. The final matrix is the combination of the weighted lower and upper bound matrices, which is illustrated on the right hand side.

bound weighed matrices for each Euler angle. The same wrap around function, described in the previous section detailing the process of a translational shift, applies here as well.

Thus far a description regarding the different processes involved in influencing the activity of the grid of pose cells has been given. In order for the algorithm to determine what it's current belief in pose should be, the average center of the activity surrounding the maximally active pose cell is calculated. This center location is referred to as the *best pose*.

5.1.2.3.4 Best Pose The distribution of activity around the grid of pose cells reflects the algorithm's distribution of belief in pose. Therefore, in order to determine which pose the algorithm most likely believes in, the highest active pose cell needs to be calculated. However, given WhiskerRatSLAM's path integration process and how pose cell activity is spread in a direction and intensity proportional to the motion direction and magnitude, the location of the highest active pose cell should take into account the activity of its neighbouring cells as well. Furthermore, considering that the grid of pose cells are a discrete representation of location and orientation, a population vector coding (Georgopoulos, Schwartz, & Kettner, 1986) approach is used to increase the accuracy of the estimated pose.

The population vector approach includes calculating the average direction of a vector from a linear combination of weighted vectors that each have a unique direction. The benefit of this approach is that a higher resolution can be represented given a set of unique discrete values.

An example of the population vector approach can be seen in Figure 5.12 where three head direction cells, out of a total set of 5, are each active to a different degree. To obtain the weighted sum of the headings, the angular values are converted to a 2-dimensional unit vector representation. Thus for headings 360°, 288°and 216°the unit vectors would be $[cos(360), sin(360)]$, $[cos(288), sin(288)]$ and $[cos(216), sin(216)]$ respectively. Given an activation level of 1 for the 360°head direction cell, 0.8 for 216°and 3.1 for 288°, the sum of the weighted vectors would be equal to:

$$1 \cdot \begin{bmatrix} cos(360) \\ sin(360) \end{bmatrix}^T + 3.1 \cdot \begin{bmatrix} cos(288) \\ sin(288) \end{bmatrix}^T + 0.8 \cdot \begin{bmatrix} cos(216) \\ sin(216) \end{bmatrix}^T = [1.311, -3.419]$$

The average angle can now be calculated using the atan2 function, where atan2 returns the inverse tangent using two variables and takes into consideration the quadrant

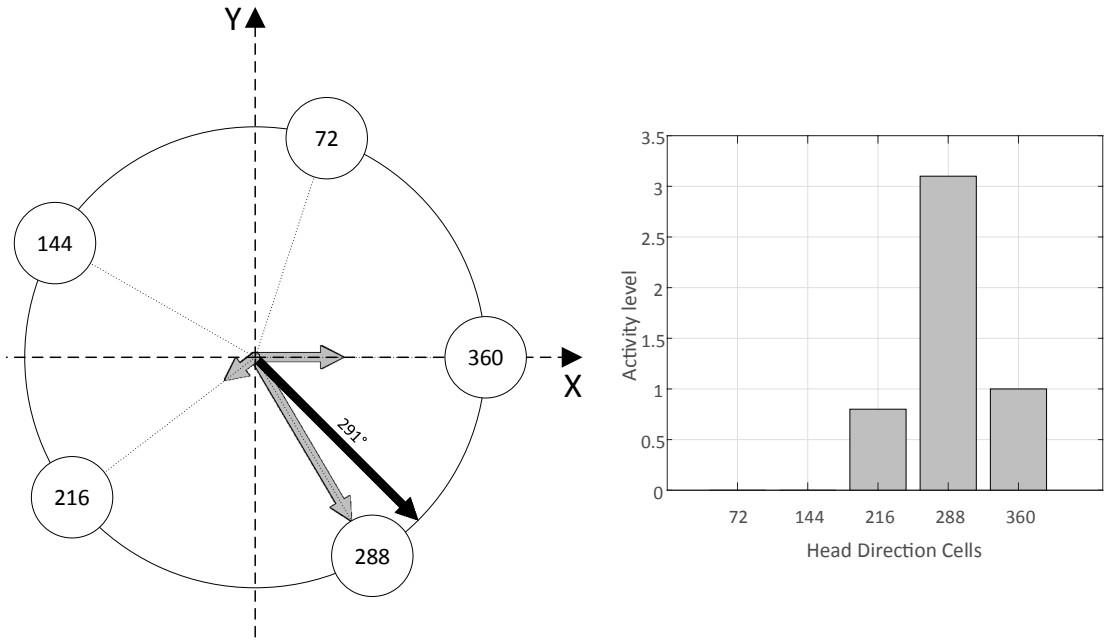


Figure 5.12: An example of a population vector approach for calculating the average heading from a set of discrete head direction cells. The benefit of this approach is that a more precise value can be derived by calculating the weighted sum of discrete values, where the weights can be interpreted as a vote or an indication of preference for a particular value. In this example the head direction cells of 216°, 288°and 360° each have an activity level of 0.8, 3.1 and 1 respectively while the remaining head direction cells are inactive, as indicated by the bar graph on the right hand side. The activity level may be interpreted as an indication of preference and its value is therefore used as the weights for each head direction. To derive the average angle represented by the weighted headings, the angles are represented in the form of a unit vector. The weighted sum of the vectors (where the individual vectors for each heading value is represented as a grey arrow on in the left figure) results in a vector (black arrow) whose direction is equal to the average angle. Using the atan2 function the average angle can be calculated from the weighted sum vector. Given a value that is outside the range of the head direction cells, a modulo operator can be used to wrap the value to the desirable range.

that the vector lies in thus being able to return an unambiguous angle value. The angle given the sum of the weighted vector is:

$$atan2(-3.419, 1.311) = -69.022^{\circ}$$

Using modulo operator, the above angle can be wrapped around the circular manifold defined for the head direction cells to produce:

$$-69.022 \ \% \ 360 = 290.978^{\circ}$$

Thus, by using the population vector method described above, an average heading value with a higher precision value than any of the individual cell's heading can be calculated.

A similar approach is used to calculate the *best pose* location from the maximally active pose cell and its neighbouring cells. Given a 6-dimensional matrix that includes only the maximally active pose cell along with its neighbouring cells, the matrix is summed along each dimension to produce a set of six vectors. Given the vector $\boldsymbol{\alpha}$ belonging to the roll (α) dimension, each element would represent a unique orientation value. The vector $\boldsymbol{\alpha}$ is defined as follows:

$$\boldsymbol{\alpha} = [\boldsymbol{\alpha_1 \alpha_2 ... \alpha_n}] \tag{5.11}$$

$$\boldsymbol{\alpha_i} = \sum_{x,y,z,\beta,\gamma} p_{x,y,z,\alpha_i,\beta,\gamma} \tag{5.12}$$

Where n is the length of the α dimension, and $p_{x,y,z,\alpha_i,\beta,\gamma}$ represents the value of a pose cell that is fixed in the α dimension to α_i.

In this work, the length of each dimension is $n = 5$, thus, elements 1 through to 5 represent a heading of 72°, 144°, 216°, 288°and 360°respectively. The value found in each element would represent the activation level of its corresponding heading. The average heading can then be determined using the same process described in the above example where the headings are converted to a 2-dimensional unit vector representation and their weighted sum is calculated. The resulting vector's angle can be calculated using the atan2 function and, using a modulo operator, the value can be wrapped around the dimension's 360° limit.

The same process can be applied to the translational dimensions, and each element in the sum vector similar to Equation 5.11, can be represented as an angle similar to the rotational dimensions. For both translational and rotational dimensions, the final step involves converting the wrapped average angle to an index value. Given a vector of length n, and a wrapped average angle of ϕ, the wrapped average element index would be: $\phi \cdot \frac{n}{360}$. Thus the location of the *best pose cell* can be determined using the coordinate specific to the grid of pose cells matrix.

5.1.2.4 Object Map

The RatSLAM algorithm generates a topological-like map referred to as an *experience* map, which is a graph consisting of edges and nodes that describe the connectivity and positioning of experiences. An *experience* is defined as a combination of external observations coupled with the believed pose at the time of the observation. The *experience* map is beneficial for both path planning as well as providing the user with a layout of the robot's environment. Since RatSLAM operates on a planar space, the nodes represent their pose using 3-dimensions that includes a 2-dimensional Cartesian coordinate and a single angular value describing its heading/yaw angle.

Unlike RatSLAM, the WhiskerRatSLAM algorithm operates in a 6-dimensional space and generates a 6-dimensional map. To set apart these differences WhiskerRatSLAM's version of an *experience* map is referred to as an object map. An object map shares the same function as an *experience* map and its only difference is that the *experience* nodes represent their pose in 6-dimensional space.

5.1.2.4.1 Complex Experience Nodes The object map consists of complex experience nodes, which are a structure for containing the pair of whisker-tactile features that are described in section 5.1.1, the location of the *best pose cell* that is described on page 127, its own pose in the object map space, and the set of experience nodes that may be reached from its current pose.

To specify the nodes pose in 6-dimensional space, 3 Cartesian coordinates are used for position and a quaternion vector is used for orientation. A quaternion representation is used, as opposed to a Euler angle representation such as that by the pose cells, since it is not susceptible to gimbal lock. The experience nodes' pose is defined as being relative to another node, and the process of defining new experiences would always call for rotational operations. Choosing a quaternion representation would prevent gimbal lock scenario from posing a problem during these calculations (Morais, Georgiev, & Sprößig, 2014).

5.1.2.4.1.1 Relative Pose A new experience is created given a sufficiently large change in *best pose cell* location, or otherwise, the observation of a unique set of features. Between the creation of a new experience, WhiskerRatSLAM integrates the velocity over time and stores the accumulated displacement for dimensions $(x, y, z, \boldsymbol{\theta})$. Thus, for each iteration the accumulated change in the x-dimension would be:

$$d_{x_i} = d_{x_{i-1}}(v_x \cdot \delta t) \tag{5.13}$$

Where v_x is the transnational velocity in the x-direction, δt is the time elapsed since the last odometry input was received, and the subscript i denotes the current iteration of the algorithm. A similar approach is taken for calculating the accumulated change in dimensions y and z.

In the case of calculating the change in orientation, angular velocity which is represented as a 3-dimensional rotational-vector (Diebel, 2006) in the odometry input (see Equation 5.7, is integrated over time and the change in orientation is represented as a quaternion. This resulting quaternion, which represents the change in orientation experienced since the last iteration, is accumulated until a new experience is created.

Reminding the reader that angular velocity is represented in the form of a rotational vector $\boldsymbol{\omega}$ and is defined in Equation 5.7. Thus for every iteration, the change in orientation is updated according to Algorithm 6.

Algorithm 6: Delta Quaternion

function deltaQ $(\boldsymbol{\omega}, \Delta t)$ **Output:** $\boldsymbol{\Delta q}$
  $\boldsymbol{\phi} = \boldsymbol{\omega} \times \Delta t \times 0.5$
  $\phi = norm(\boldsymbol{\phi})$
  if $\theta > 0$ **then**
    $p = sin(\phi)/\phi$
    $\boldsymbol{\Delta q} = [cos(\phi)\ p\boldsymbol{\phi}]$
  end
  else
    $\boldsymbol{\Delta q} = [1\ \boldsymbol{\phi}]$
  end
  return $\boldsymbol{\Delta q}$

Where the conversion from rotation vector notation to quaternion is obtained from (Diebel, 2006) and is shown in Equation 5.14.

$$\boldsymbol{\Delta q}_{\phi}(\boldsymbol{\phi}) = \boldsymbol{\Delta q}_{\Delta\theta}(\boldsymbol{\Delta\theta}) = \begin{bmatrix} cos(\frac{\Delta\theta}{2}) \\ \frac{\boldsymbol{\Delta\theta}}{\Delta\theta} sin(\frac{\Delta\theta}{2}) \end{bmatrix} \tag{5.14}$$

Thus, given Δq, the current orientation relative to the initial orientation $\boldsymbol{q}_0$ is:

$$\boldsymbol{q}_i = \boldsymbol{q}_{i-1} \times \boldsymbol{\Delta q} \tag{5.15}$$

Where the $\times$ operator refers to a quaternion multiplication, which is defined in (Stevens, Lewis, & Johnson, 2015).

When a new experience e_{i+1} is created, the accumulated change in orientation represents its orientation relative to the current iterations experience e_i, thus:

$$\boldsymbol{q}_{e_i \to e_{i+1}} = \boldsymbol{q}_i$$

To obtain the orientation of the new experience in the object map space $q_m^{e_{i+1}}$, the orientation of the current iterations experience $q_m^{e_i}$ is multiplied with $\boldsymbol{q}_i$:

$$\boldsymbol{q}_m^{e_{i+1}} = \boldsymbol{q}_m^{e_i} \times \boldsymbol{q}_i$$

The measurement of orientation in object map space is done so relative to the orientation of the first experience node. WhiskerRatSLAM initializes the orientation with the following values $\boldsymbol{q}_m^{e_1} = [q_0\ q_1\ q_2\ q_3] = [1\ 0\ 0\ 0]$, where the the elements are referring to the coefficients of the quaternion vector of the form shown in Equation 5.16 (Stevens et al., 2015).

$$\boldsymbol{q} = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3 \tag{5.16}$$

5.2 Localization and Object Recognition Performance

The following section describes the experimental setup used to gather both the physical and simulation based data-sets for analyzing the localization and object recognition performance of WhiskerRatSLAM, as well as the presentation of the experimental results and its discussion.

5.2.1 Experiments

Experiments were designed to assess the accuracy of WhiskerRatSLAM to localize across a variety of object shapes and to discriminate between objects. The data included a set taken using a physical whisker array as it explored the surface of a box shaped object (see panel A of Figure 5.13). Five simulated data sets were also generated using ROS/Gazebo for trajectory and odometry, and Matlab for calculating the positions of the whiskers' points of contact as well as their 2D deflection vectors. The simulation data sets incorporated additive noise in the odometry, contact position and deflection vectors, which were all derived from the statistics of the original physical data set. An overall description of the experimental set up is described further in Figure 5.13 on page 133. To validate the simulation, the virtual box object was explored using the same trajectory as used for exploring the physical box object. All the other simulated objects were explored using the same trajectory but different from that used to explore the box (see Figure 5.14 on page 134 for details).

The experimental parameters that were adjusted between runs were: the features used to characterize a surface region (PFH only, SDA only, and both PFH & SDA

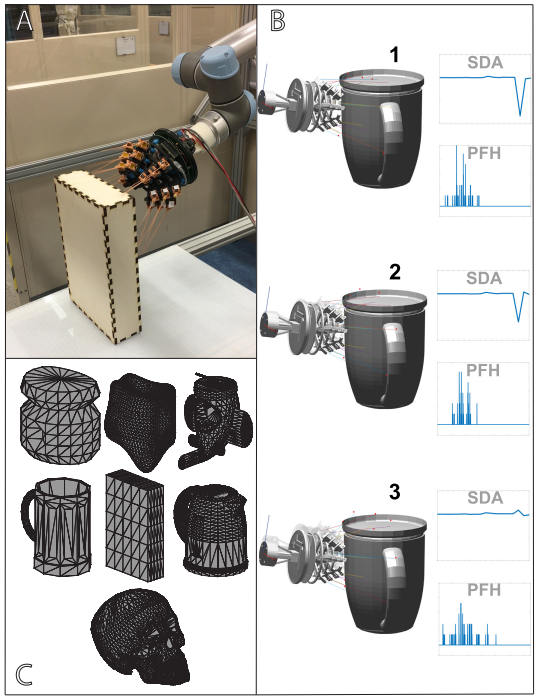
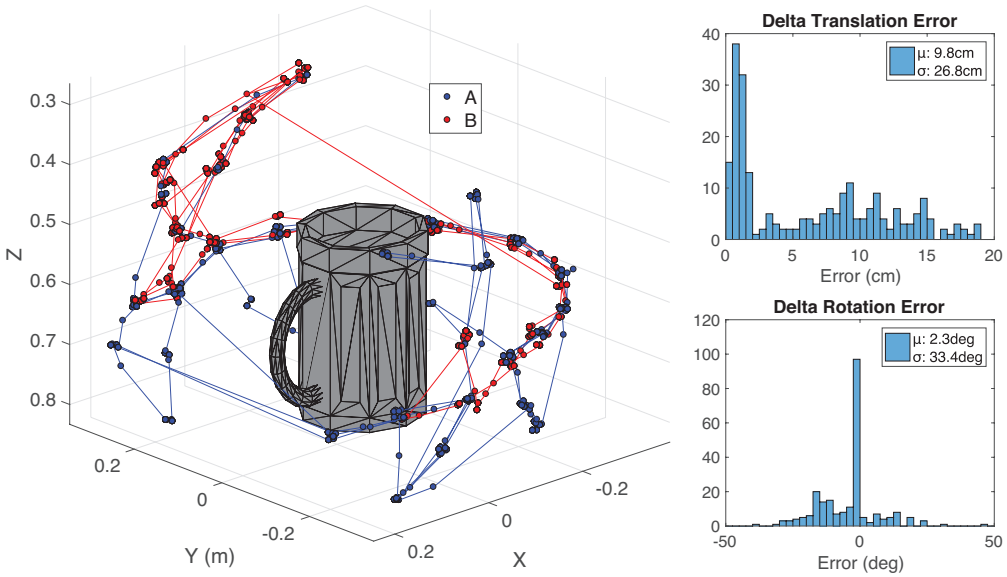


Figure 5.13: Experimental configuration for physical and simulated data set acquisition. **A**: The physical data set was acquired using a UR-5 arm to move an array of active tactile whiskers around the contours of a plywood box. The whisker-array consists of 18 whisker modules that are able to sense their whisk angle and 2D deflection forces at their base; a detailed description of the sensor array may be found in (Sullivan et al., 2012) and Chapter 2. The known location of the object was calibrated against the workspace of the UR-5 to serve as a measure of ground truth to whisker contact locations.
B: The odometry data was generated using ROS/Gazebo which simulated the motion of the UR-5 arm, while the simulated whisker deflection data were calculated using Matlab. The three images were taken from Gazebo at the end of consecutive whisks as the array moved in a downwards direction across the surface of the Kettle. The plots to the right of each image illustrate the two contact features that were used to identify a region. The top is the *slope distribution array* (SDA), a vector whose elements encode the slope of the surface that each whisker has made contact with, while the bottom is the *point feature histogram* (PFH) of the points of contacts detected during the whisk by the whole array.
C: Several simulated object models were used to test the robustness of the algorithm and to evaluate its ability to discriminate between objects. The average bounding box for all the objects is $20 \times 20 \times 35$ cm. From left to right the objects are named *Top*: Barrel, Blob, Plane. *Middle*: Mug, Box and Kettle. *Bottom*: Skull.

combined); the introduction of a feature update mechanism that averages matching feature vectors; the approach taken to determine point of whisker-contact (tip-assumption against *support vector regression* (SVR)). The SVR approach was trained to map whisker deflection characteristics and whisker length to a more precise radial distance estimate with an approximate accuracy of 25% relative to the whisker length (see figure 4.9 on page 93). Therefore, a total of 12 runs were processed for each object with each run consisting of a unique set of parameters that are summarized in Table 5.1.

Table 5.1: Set of unique experimental parameters.

Features	PFH	SDA	PFH + SDA
Feature update	ON	OFF	
Contact algorithm	Tip	SVR	



(a) (b)

Figure 5.14: Object exploration trajectories. Figure 5.14a plots of the two trajectories used to explore the various objects; *Trajectory A,* used for building the Object Maps, and *trajectory B,* a novel trajectory used to gather validation data sets to test the robustness of the system. Figure 5.14b includes the distribution of displacement errors between those obtained from the ground truth and integrated odometry measurements captured during a complete cycle of *trajectory A*.

5.2.2 Results

5.2.2.1 Localization

The localization accuracy was determined through the positional and rotational errors measured between the ground truth and re-localization estimates of the whisker-array pose. The positional error was calculated using a Euclidean distance while the rotational error was calculated as the minimum angle required to align the estimated and ground truth orientation in quaternion space following the method described in (Hodaň, Matas, & Obdržálek, 2016). The distribution of displacement errors between ground truth pose and integrated raw odometry during the complete exploration of an object are shown in Figure 5.14b on page 134. Specific to this thesis work, a SLAM algorithm is considered to operate correctly provided that it reduces errors in pose estimates to less than or equal to the minimum step size taken between observations; the step size is measured as the change in pose from one surface region that a sample is taken to the next. The step size used for objects other than the box were approximately 11 cm and 12°, while for the Box object the values were approximately 3 cm and 3°.

The aggregated results for all objects used in the experiments are shown in Figure 5.15 on page 136. The lowest errors in re-localization estimates were recorded when using SVR based radial distance estimation, both the PFH and SDA features combined for region matching, and not including the proposed feature merging method. With the exception of the Kettle object, all other objects returned localization errors that were lower than their associated step size. Analysis of the Kettle localization results revealed that the large errors were due to confusion brought on by symmetry. These erroneous localization estimates were eventually corrected following subsequent observations, highlighting a familiar problem for SLAM algorithms failing to localize in symmetrical environments.

Similarly, symmetrical objects such as the Barrel and Mug, did not suffer so markedly due to their position in space relative to their associated exploration trajectories. The trajectory around the kettle followed a loosely spheroid outline that shared the same geometric center as the kettle. This resulted in very similar observations from the whisker sensors at a range of array poses.

5.2.2.2 Object Identification

The second set of experiments focused on the use of WhiskerRatSLAM for object identification. The object map created by WhiskerRatSLAM is a topological map relating the features of each recorded surface region relative to one another in 6D space. The *object map* thus characterizes an object by its shape and can be used for identifying

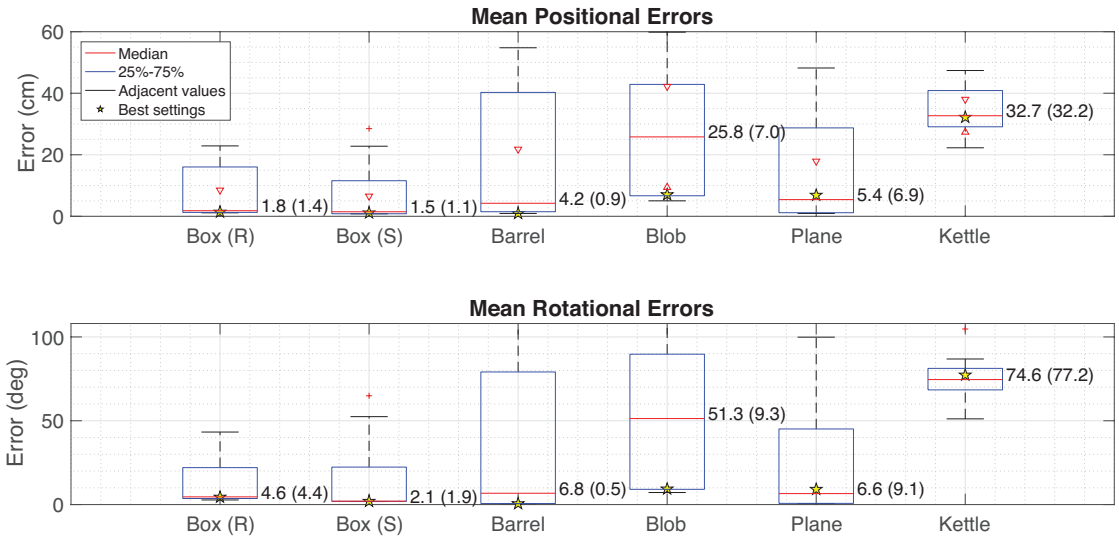


Figure 5.15: Box plot illustrating the distribution of localization errors (positional and rotational) for each object. Box(R) refers to the physical box data set, while Box(S) the simulation data set. The values within the brackets pertain to star shaped markers, which are the results of the runs using the best set of parameters: SVR based radial distance estimation, Unmerged contact features and Both (PFH and SDA combined) features for region matching. With regards to the notation of the box plot: the blue boxes mark the edges at which 25% or 75% of the data lies below while the adjacent values are used for marking the edges of where the majority of the data distribution lies. The exact formulation for calculating the position of these markers may be found in (Velleman & Hoaglin, 1981).

future encounters with the same object and for determining novel objects. The concept is analogous to treating each object as a room within a building, whereby the agent, in this case the whiskered robot, is switched on and left to explore a particular room with no prior knowledge of its path to that room. A re-localization to a previously explored room, or object in this work's case, would indicate that the agent has recognized which room it is currently in. Frequent re-localizations on subsequent samples by its whiskers would indicate an increased confidence of room/object identity.

Using this analogy, *experiences* from each *object map* were appended to construct an experience history that assumes sequential visits to multiple objects with no topological connection between them (as shown in Figure 5.16 on page 137). The whiskered robot is then presented with an unknown object and is set to explore its surface using either the same trajectory used for generating the *object maps*, *trajectory A*, or the novel trajectory, *trajectory B*, for testing (see Figure 5.14a on page 134 for details). Figure

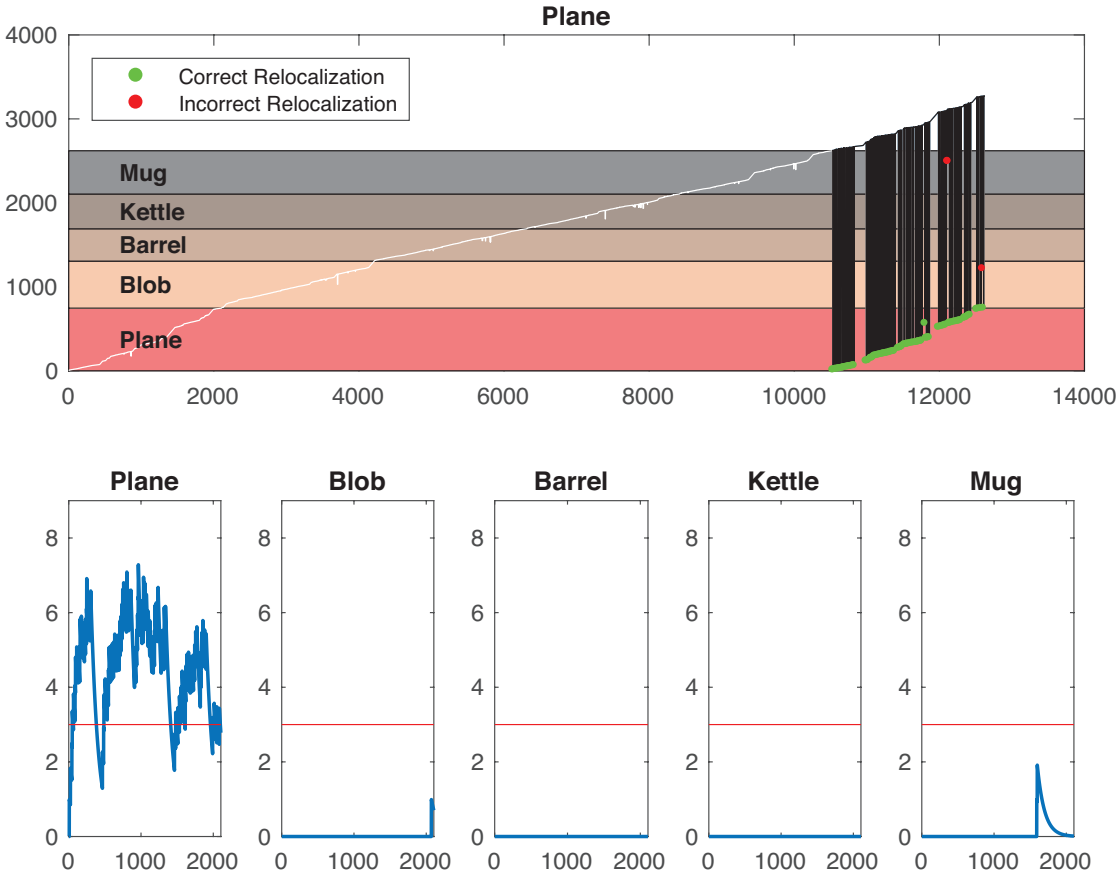


Figure 5.16: The response of Whisker-RatSLAM as the whisker-array is presented with the previously explored toy Plane object for the purpose of object identification. The **top** panel displays the history of the *experiences* generated through the exploration of all objects (white line). Each individual *experience* has an index (y-axis) and a sample number (x-axis) with the horizontal colored regions indicating that these *experiences* were generated whilst exploring the object named to the left. The transition between objects has been removed, i.e., the *experiences* representing each object have been sequentially appended in no particular order. The black line of *experiences* outside of the colored region represent new sample points as the whisker array explores an unknown object. The re-localizations that occur (indicated by green and red dots) indicate that a sample is very close to an existing *experience* which increases the belief in identity of the unknown object. The green markers indicate a re-localization to the correct object identity while the red markers indicate a re-localization to an incorrect object identity. The **lower** panel illustrates the time course of belief in each objects identity during the exploratory phase of the experiment, i.e., sample points > 10500 in the experience history plot. The dynamics of belief are governed by a simple leaky integrator which is injected with an impulse when a re-localization occurs and has a fixed decay constant. A user defined confidence threshold (horizontal red line) marks the desired level in belief required to confidently identify an object. The results in this image belong to the case where the exploration trajectories for the map generation and object identification portions are the same (*trajectory A*).

5.16 on page 137 details a single case in which a previously mapped object (Plane) is explored using *trajectory A* and how the rate of re-localization events (green/red dots) serves as a measure of confidence in classifying object identity.

The complete set of results from all cases are are shown in Figure 5.17 on page 140. For the Skull object, which has not been mapped and therefore does not feature in the experience history, the system exhibits very low confidence levels across all known object identities. It is also clear that when using *trajectory B* the confidence in object identity is much lower than using *trajectory A*. This is understandable since features would inevitably vary due to changes observed features and thus reduce the probability of experiencing consistent observations that lead to re-localizations. The level of activity is however correlated to the correct object identity. It can be seen that in some cases different *object maps* receive somewhat equal activity levels indicating ambiguity in object identity.

5.2.3 Discussion

Using both physical and simulation based experiments WhiskerRatSLAM is shown to successfully localize an array of mobile tactile whiskers in 6D space. Further, using the object maps that are generated during object exploration object identity can be confidently classified, including the recognition that an object is novel and not from a previously observed set.

The results illustrate that the algorithm can accommodate novel exploration trajectories for object identification but with significantly reduced confidence. A potential problem is that the set of features for each particular surface region are not pose invariant. Pose invariance refers to the property of being consistent regardless of a change in viewing orientation or distance. The PFH feature, as described earlier, is pose invariant, while SDA is not. The results have shown that both features are required to obtain a good localization accuracy and forgoing the SDA feature for the purpose of gaining the pose invariant property of PFH is not desirable.

To address this issue the next chapter will focus on implementing a low level controller that ensures a consistent whisker-array placement strategy. By limiting the distance and angle of the whisker-array relative to an object's surface, future encounters with a region should result in similar feature sets. Furthermore, ambiguity in object identity could be addressed by exploiting the topological properties of the object maps to determine which region from a set of ambiguous maps are least similar and therefore drive the whisker-array towards it. In the situation where a satisfactory level of confidence still can not be

reached then the algorithm can consider the object to be novel and, therefore, generate a new *object map*. Thus the next chapter will focus on improving the performance of object recognition.

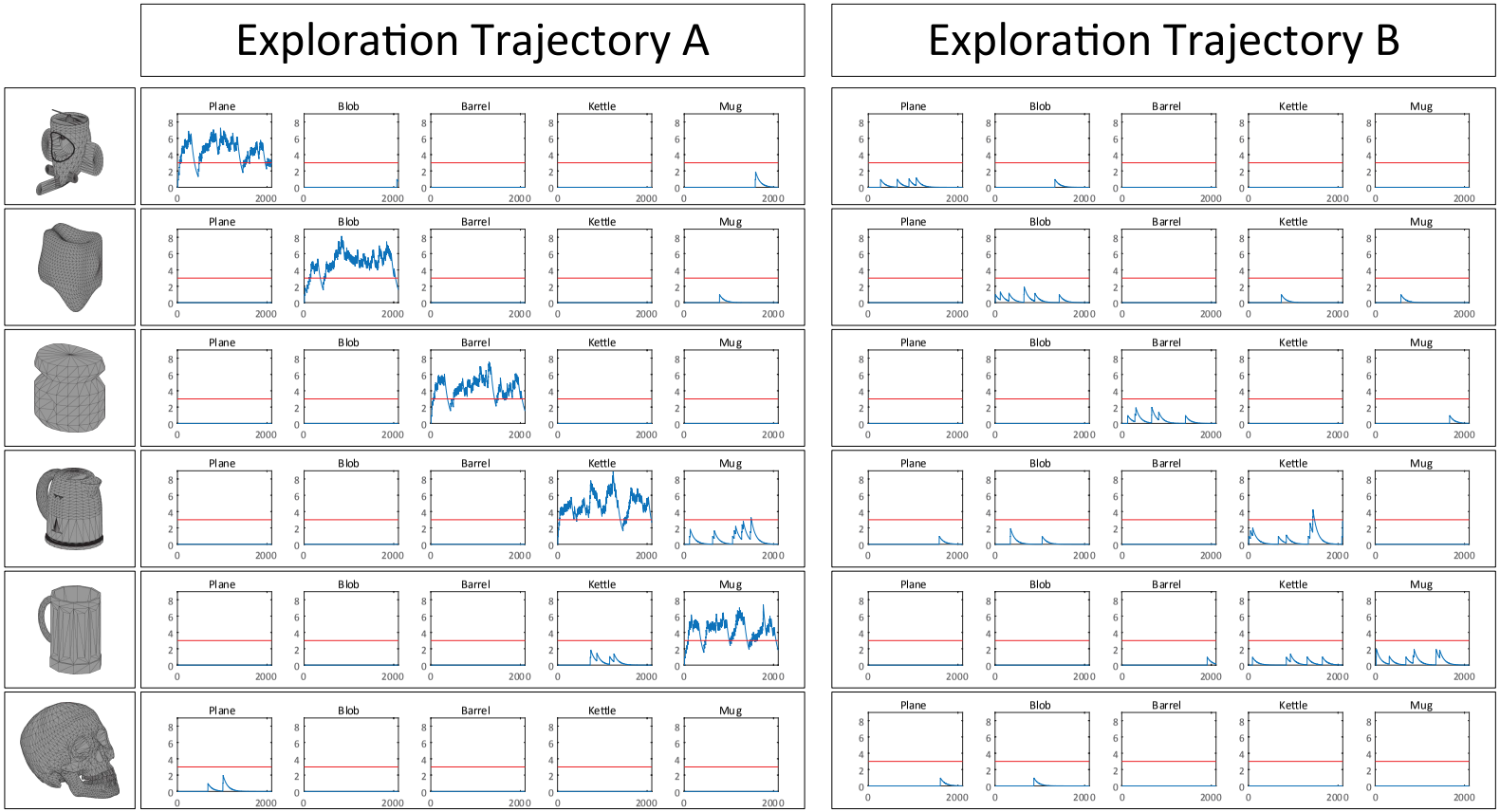


Figure 5.17: Time course of belief in object identity as the robot explores an unknown object (shown in the images to the left) following the original trajectory used for *object map* generation (A) or the novel trajectory (B). The five columns under each exploration trajectory represent the belief for each previously mapped object (Plane, Blob, Barrel, Kettle and Mug). The Skull object has not been mapped which reflects in the low level in belief in other object identities.

Chapter 6

Active Whisker-Array Exploration For Fast Shape Recognition

The previous chapter discussed the performance of WhiskerRatSLAM in 6-dimensional pose localization and object identification. The chapter ended on a discussion stating the need for a consistent whisker-array placement strategy such that the set of observed features for a given surface region is more likely to be similar when encountered again. This chapter focuses on determining the effectiveness of the surface-placement strategy, as well as the comparison of different search strategies for confirming object identity quickly.

Based on the workings of RatSLAM, which is the underlying architecture on which WhiskerRatSLAM is based on, the localization/object recognition can only occur once a series of consecutive surface features that are known to be in close proximity to each other, are observed. The cause of which is due to the the rise in pose cell activity that is associated with feature recognition and path integration. To ensure that the algorithm has a higher likelihood of recognizing features from the same region a low level controller for whisker-array placement was designed. The controller attempts to maintain a consistent whisker-array orientation and distance to an object's surface, thereby limiting the variation in observed features at a given region. This array-to-surface placement controller is inspired from the work of (Pezzementi, Plaku, Reyda, & Hager, 2011) where a cutaneus-tactile sensor is used to identify object shapes.

The cutaneus-tactile sensors used in the work of Pezzementi is similar to this work's whisker-tactile sensors in that they both have a limited range, field of view and both

require physical contact with a surface in order to perceive it. Representing their sensory data as a series of images, they observed that samples gathered from a specific region, albeit at varying sensor orientation, reduced the likelihood of recognition. To facilitate recognition, the authors generate rotation and intensity invariant local features for each image. In addition, the authors implement a surface contact controller that reduces the variation in observed features, which again is to facilitate feature recognition for previously observed regions.

A 3-dimensional equivalent to the rotation invariant transforms used in the work of Pezzementi is the point feature histogram, which is one of the features used by WhiskerRatSLAM for surface region characterization. One option for tackling the object recognition problem is to implement a similar bag-of-features approach as that in (Pezzementi et al., 2011). However, it is argued that the approach of exploiting the 6-dimensional localization capability of WhiskerRatSLAM would serve to reduce the time taken to reach an appropriate object identity confidence level, by targeting more distinctive regions.

The following chapter thus continues the work of Section 5.2 and uses the WhiskerRatSLAM architecture and the concept of re-localization in 6D space as a means of determining the algorithm's confidence in object identity. To improve the performance of the object identification system, a surface placement strategy is described, followed by two region search strategies that are used to confirm object identity or otherwise remove doubt in object identity. Furthermore, the physical and simulation set ups for object-map generation and object recognition experiments are described. The final section includes the results and discussion of the experiments' outcome.

6.1 Method

The first of the proposed methods for improving object identity performance using WhiskerRatSLAM is the implementation of a consistent whisker-array placement strategy, relative to the contacted surface.

6.1.1 Surface Placement

When approaching an object's surface with the whisker-array, the features that are generated would be dependent on the pose of the array relative to the surface. In an attempt to increase the rate of re-localization, limiting the pose of the array whenever it is in contact with a surface should reduce the variety in these features and thus facilitate feature recognition.

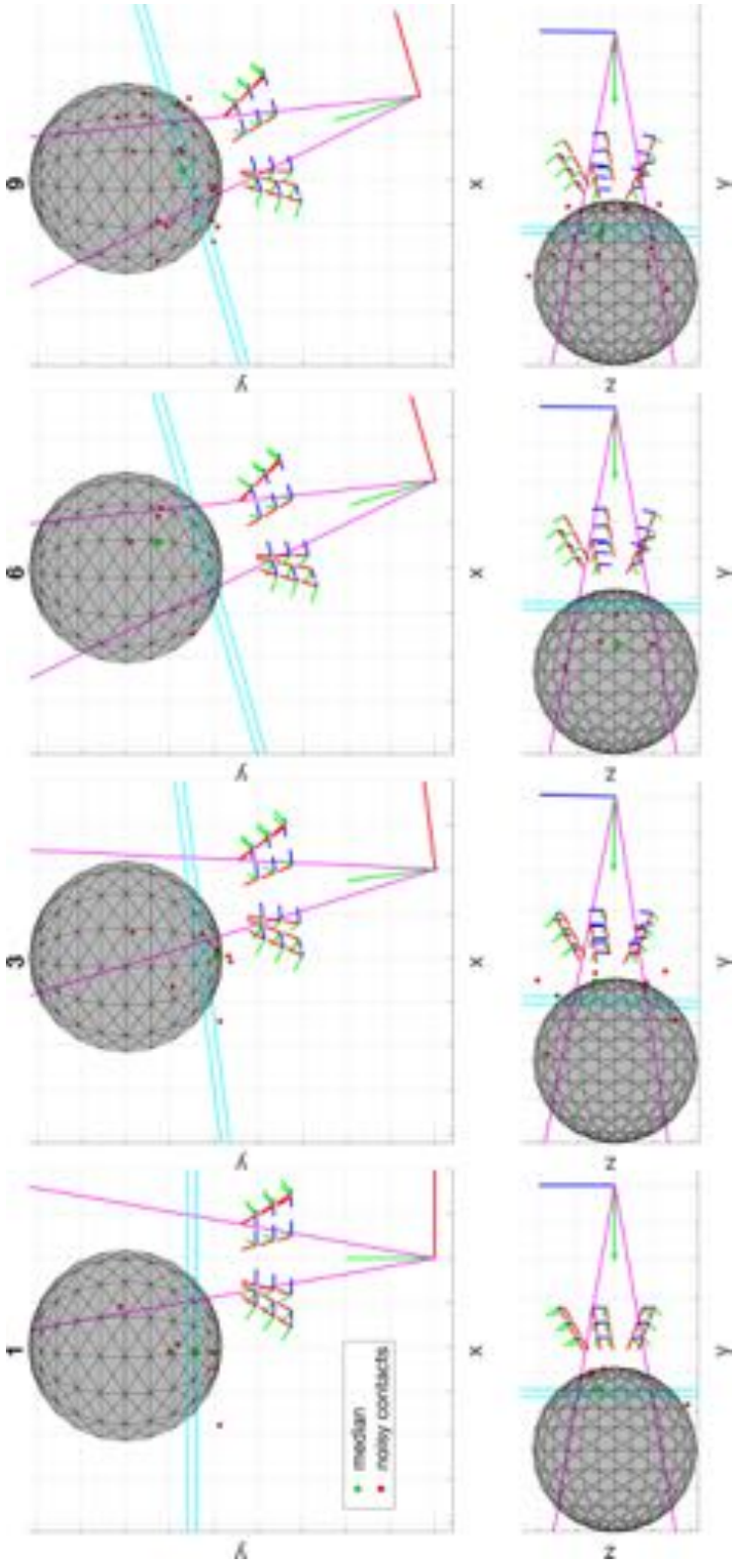


Figure 6.1: The sequence of frames highlights a surface placement process after an initial contact by the whisker-array with a spherical object. The surface placement's first attempts to orient the whisker-array such that the median contact location (green marker) is within a volume that illustrated by the cone outline (frames 1,3 and 6). The second step includes translation along the major axis of the cone such that the median distance to the points along the whisker-arrays y-component, are within a specified range that is illustrated by the pair of straight cyan lines (frame 9). In each frame, noise is artificially added to the contact point locations, which are shown in red. The median of these contact points is shown in green, and is the point that the surface placement controller is attempting to place within the desired boundary defined by the cone outline. The median perpendicular distance to the points is not shown in the figure and is instead described in the main text.

The process of ensuring a consistent placement strategy includes:

1. placing the median of the contact points within a specific field of view of the whisker array

2. positioning the whisker array such that the median distance to all contact points are within a specific perpendicular range

3. Fixing the orientation of the whisker-array with respect to the direction of gravity

Figure 6.1 illustrates this process, where frames 1, 3 and 6 show the process of orienting the whisker-array towards the median of the contacts so that it falls within the field of view, which is represented by the cone outline. Frame 9 shows the process of translating towards the surface such that the median perpendicular distance is within a specific perpendicular range, which is represented by the pair of parallel lines.

The consistent surface placement controller (SPC) is implemented in the form of a function that returns the desired goal coordinate frame belonging to the end-effector/whisker-array. The function first attempts to correct the orientation of the whisker-array, followed by its translational position. At all times, the function returns a goal coordinate frame that maintains a fixed yaw value relative to the direction of gravity. The direction of gravity is assumed to be known, otherwise this could be obtained from an accelerometer. The function is shown in Algorithm 7 and its internal function *maintain_yaw*, which is responsible for correcting the roll of the target pose, is shown in Algorithm 8.

The *maintain_yaw* function defines the target pose's coordinate frame by first defining its y-component. The y-component would determine the direction that the whisker-array would point towards, as such, the normal to the surface region is used to define its value.

The next step includes defining the z-component, which would be equivalent to setting the whisker-array's yaw value. The z-component would need to be defined such that it lies on the same plane defined by the gravity vector and the y-component, and since the y and z components are perpendicular, the z-component can be defined as the vector 90° away from the y-component, in the direction of the gravity vector. The x-component can then easily be calculated as the cross product of the y and z components.

Thus, given the definition of the *correct_pose* function, the pose of the whisker-array is constantly checked to ensure that all conditions are met before a new surface region is targeted.

Algorithm 7: Consistent surface placement function

function correct_pose $\{\boldsymbol{tf}^i_{ee}, \boldsymbol{c}_i, \mu, \lambda_\theta, \lambda_l, \lambda_u, \delta_\theta, \delta_x\}$;

Input : End effector coordinate frame for current iteration $\boldsymbol{tf}^i_{ee} \in \mathbb{R}^{4\times4}$
Estimated contact points $\boldsymbol{c}_i = [[x_1, y_1, z_1] \dots [x_n, y_n, z_n]]^T$
Minimum number of contact points μ
Half angle of cone volume where median must lie in λ_θ
Minimum number of contact points μ
Lower and upper bound of perpendicular distance that median contact distance must lie in $\lambda_l \& \lambda_u$
Rotation and translation division steps $\delta_\theta \& \delta_x$

Output: Goal coordinate frame $\boldsymbol{tf}_{goal} \in \mathbb{R}^{4\times4}$

$\boldsymbol{ee}_y = \boldsymbol{tf}^i_{ee}[1:3, 2]$;
$\boldsymbol{ee}_{position} = \boldsymbol{tf}^i_{ee}[1:3, 4]$;
if $size(\boldsymbol{c}_i) \geq \mu$ **then**
    /* correct rotation */
    $\widetilde{\boldsymbol{c}}_i = median(\boldsymbol{c}_i)$;
    $\boldsymbol{n} = normalize(\widetilde{\boldsymbol{c}}_i - \boldsymbol{ee}_{position})$;
    $\theta = acos(\boldsymbol{ee}_y \cdot \boldsymbol{v})$;
    if $\theta > \lambda_\theta$ **then**
        $region\_normal = -\boldsymbol{v}$;
        $\boldsymbol{tf} = maintain\_roll(\boldsymbol{tf}^i_{ee}, region\_normal)$;
        $\boldsymbol{tf}_{goal} = divide\_steps(\boldsymbol{tf}^i_{goal}, \delta_\theta)$;
        return $\boldsymbol{tf}_{goal}$
    /* correct distance */
    $\boldsymbol{v}_{ee\to c} = \boldsymbol{c}_i - \boldsymbol{ee}_{position}$;
    $\boldsymbol{v}^y_{ee\to c} = \boldsymbol{v}_{ee\to c} \cdot \boldsymbol{ee}_y$;
    $\widetilde{d}_y = median(\boldsymbol{v}^y_{ee\to c})$;
    if $\widetilde{d}_y < \lambda_l$ *or* $\widetilde{d}_y > \lambda_u$ **then**
        $mid_d = mean([\lambda_l, \lambda_u])$;
        $\boldsymbol{v} = \widetilde{d}_y - mid_d$;
        $\boldsymbol{v}_{trans} = \boldsymbol{v}/\delta_x$;
        $\boldsymbol{tf} = translate(\boldsymbol{tf}^i_{ee}, \boldsymbol{v}_{trans})$;
        $\boldsymbol{v}_y = \boldsymbol{tf}[1:3, 2]$;
        $\boldsymbol{tf}_{goal} = maintain\_roll(\boldsymbol{tf}, -\boldsymbol{v}_y)$;
        return $\boldsymbol{tf}_{goal}$

Algorithm 8: Maintain yaw according to the direction of gravity

function maintain_yaw $\{\boldsymbol{tf}, \boldsymbol{n}\}$;
Input : Target coordinate frame before roll correction $\boldsymbol{tf} \in \mathbb{R}^{4\times4}$
Target region's normal vector $\boldsymbol{n} = \in \mathbb{R}^{3\times1}$
Output: Goal coordinate frame $\boldsymbol{tf}_{goal} \in \mathbb{R}^{4\times4}$

$\boldsymbol{ee}_{position} = \boldsymbol{tf}\,[1:3,4]$; // target frame's position
$\boldsymbol{y}_{target} = -\boldsymbol{n}$; // target frame's $y$component vector
$\boldsymbol{v}_{gravity} = [0,0,1]$; // vector directed away from gravity i.e. -z
$\theta_{vy} = acos(\boldsymbol{y}_{target} \cdot \boldsymbol{v}_{gravity})$;
$\theta_{vz} = \theta_{vy} - \frac{\pi}{2}$; // angle to orient $\boldsymbol{v}_{gravity}$ to target frame's $z$component
$\boldsymbol{r}_{vz} = \boldsymbol{v}_{gravity} \times \boldsymbol{y}_{target}$; // define rotation axis from which $\boldsymbol{v}_{gravity} \rightarrow \boldsymbol{z}_{target}$
$axis\_ang_{vz} = [\boldsymbol{r}_{vz}, \theta_{vz}]$;
$rot_{vz} = RotationMatrix(axis\_ang_{vz})$;
$\boldsymbol{z}_{target} = rot_{vz}\boldsymbol{v}_{gravity}$;
$\boldsymbol{x}_{target} = \boldsymbol{y}_{target} \times \boldsymbol{z}_{target}$; // target frame's $x$component is perpendicular to y&$z$components
$\boldsymbol{tf}_{goal} = \begin{bmatrix} \boldsymbol{x}_{target} & \boldsymbol{y}_{target} & \boldsymbol{z}_{target} & \boldsymbol{ee}_{position} \\ 0 & 0 & 0 & 1 \end{bmatrix}$;

6.1.2 Surface Region Search

Reminding the reader that each unique feature is associated with a particular pose cell, and that, when a feature is recognized, that particular pose cell's activity is increased leading to the formation of a new activity packet. As the whisker-array moves, pose cell activity will be shifted around the grid according to the motion described by the odometry. In the event that pose cells, which are in the path of the newly formed activity packet, are activated via feature recognition, the activity of the packet will increase and given sufficient excitation, will lead to a re-localization if the current pose estimate is not in agreement. These series of consecutive feature recognitions must therefore correspond to neighboring regions that are along the path taken by the whisker-array.

Linked experience nodes within the object map represent such neighboring regions. It is therefore conceivable to induce re-localizations by moving the whisker-array from one experience node to another linked experience node. However, a choice must be made in selecting a node to move to, since experience nodes could potentially have multiple links. Two selection criteria are proposed: 1) Following the experience history generated during the object map's generation, 2) Moving towards a region that is most dissimilar to all plausible regions.

6.1.2.1 Surface following

Each experience includes a description of its pose relative to a linked experience, and a transformation matrix that moves the whisker-array to the desired region, described by the target experience, can be formulated. The problem, however, is that these measurements are corrupted by some form of noise.

The relative pose between experiences may be determined either based on their pose in object map space, or otherwise, the recorded odometry input that led to a change from one experience to the other. The poses in object map space are constantly being adjusted at each iteration based on the occurred re-localizations. These corrections are part of the inherent SLAM mechanism of updating the algorithm's latest belief in map layout. Thus the relative pose of each experience in object map space should contain less error than those calculated from the odometry. For brevity the relative pose of experiences in object map space is referred to as the 'map pose' and those from the odometry the 'odometry pose'.

Irrespective of the choice, the successive execution of transformations, based on either the map or odometry poses, across multiple experiences would inevitably drift from the intended target. The proposed method for resolving this issue is to: segment the motion for an experience step (one experience to another) and limit the requested motion to 2-dimensions. By implementing these conditions following all changes in pose, the whisker-array should reduce the variation in observed features and facilitate feature recognition for previously observed regions. In case the whisker-array drifts beyond the intended target region, the proposed search methods both monitor re-localization events and update their trajectories. Logically, the continuous updating of trajectory should ensure the arrival of the whisker-array to its intended target, provided correct re-localizations.

An example of this proposed method can be seen in Figure 6.2. To generate this image, the simulated whisker-array was set to move across several faces along a line of latitude. The transformation matrix describing the motion at each iteration was then corrupted with noise and two additional whisker-arrays are included. The two whisker-arrays' pose highlight the effect that a surface placement controller has on pose of the whisker-array when trying to move from one region to another based on noisy odometry input. The figure shows that, by using the proposed surface placement controller (SPC), the whisker-array's whiskers maintain contact with the object's surface and its yaw angle is fixed relative to the direction of gravity (-z).

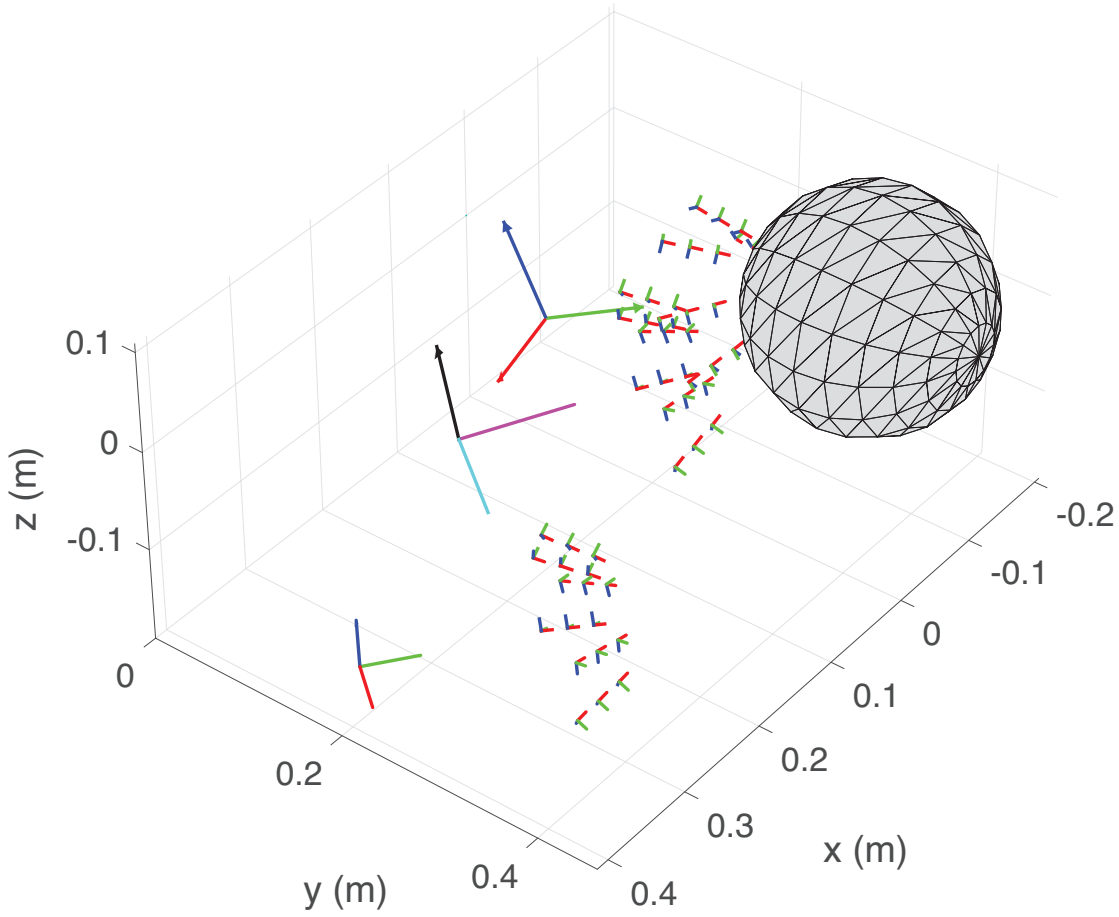


Figure 6.2: The consequence of the surface placement controller is that the whiskers are kept in constant contact with the surface of the object. In this example a whisker-array is simulated to move around a sphere and the motion is re-created using noisy odometry ($\mathcal{N}(0, 1cm)$ for translation and $\mathcal{N}(0, 5^{\circ})$ for orientation). Two RGB coordinate frames shown represent the final pose of the whisker-array when its pose is determined via path integration while the CMK frame shows the final pose of the ground truth. The larger of the two RGB frames illustrates the case where the placement controller is activated, while the smaller of the two illustrates the contrary case.

6.1.2.2 Follow experience history

The agent is confident of an object's identity when the rate of re-localization is high enough to cause its confidence level to exceed a preset threshold. The rate of re-localization is the highest when the agent follows a similar path as that during the generation of the object map as shown in (Salman & Pearson, 2018). However, this path cannot be replicated exactly and will instead be approximated according to the stored

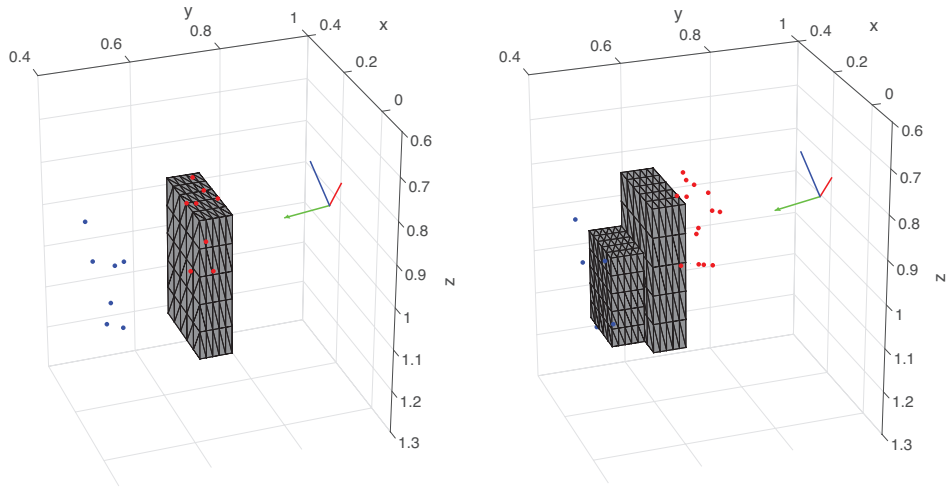
experiences.

The history of active experiences is stored to allow for the segmentation and merging of object maps, which is part of the recognition process described in Section 5.2. The path way can therefore be approximated by following each successive experience as indicated by the history. It is an approximation since the relative positioning of one experience to another is determined by odometry data and will therefore include some form of noise. The vector describing the placement of the next experience relative to the supposed current one will be reduced to a 2D component. To ensure the agent does not drive into the obstacle and to give the surface placement control mechanism ample time to react and correct its pose, the translation is split into smaller step sizes (3 cm in this work).

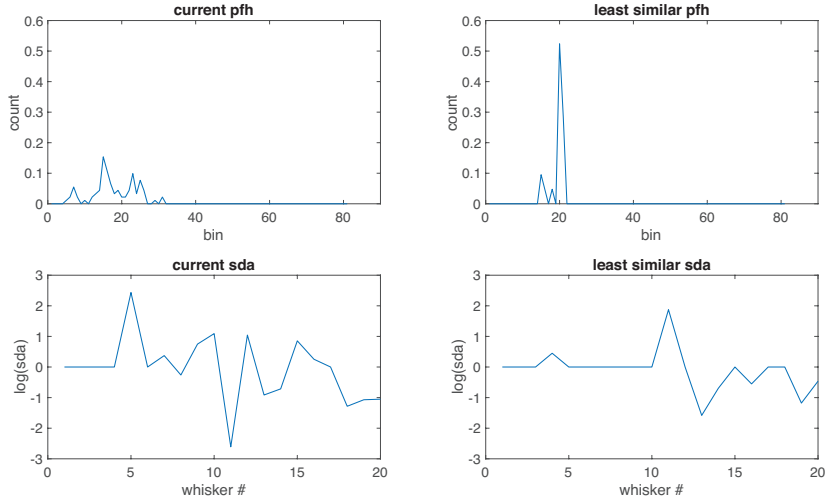
By following the experience history, the motion of whisker-array attempts to play back the series of movements made when generating the original object map. The experience history refers to the log of the active experience at each WhiskerRatSLAM iteration. For example, when exploring the surface of an object, at each iteration there is an experience that represents the current state of the whisker-array, which is referred to as the *current* experience. As the exploration proceeds, an occurrence of a re-localization, movement or new observation, results in a change of of state and thus, *current* experience. The experience history is therefore a log of which experience was active at each specific iteration.

In Section 5.2 Figure 5.16 illustrates the concept of using the rate of re-localization, to an object's set of experiences, as a measure of confidence in that object's identity. The top plot includes the experience history of the exploration run (black line) in which the identity of the object is initially unknown. It can be seen that as the whisker-array is set to explore the object, WhiskerRatSLAM recognizes, at several iterations, that the experience that best describes the whisker-arrays current state, is from a set belonging to the Plane object.

The trajectory executed in the case of Figure 5.16 is the same as that used for obtaining the Plane object map, hence the near constant occurrence of correct re-localizations. The proposed *follow history* search strategy attempts to recreate the original trajectory by moving from one experience to the next. Thus, given a re-localization to a particular object's experience, the next experience as indicated in the experience history, would be the target.



(a)



(b)

Figure 6.3: Using the least similar region approach, the region whose features are least similar to the current region are targeted. The red markers shown in Figure 6.3a highlight the region that the whisker-array is currently at, and the blue markers highlight the contacts associated with the least similar region. The aim is to visit the least similar region so as to reduce ambiguity in object identity. In the following example, the true identity of the object being probed is the Box object. By targeting the region on the opposite side, which belongs to the Box-stack object, the system's confidence in Box-stack object would be lower than the Box object as the array would fail to observe the least similar region. The least similar region is calculated based on the similarity of regions' features, which include the point feature histogram and slope distribution array. The region features are shown in Figure 6.3b and the current region's features are in the left column while the least similar region's features are in the right column. The method used to calculate the least similar features is detailed in the main text.

6.1.2.3 Least similar feature

When exploring an unknown object, there is a possibility of having a non-zero confidence in multiple object identities. To cope with ambiguity, the proposed approach is to search for the region that least resembles the current region, from the entire set of possible objects.

It is computational expensive to search for the global solution regarding the least similar region from among the sets of likely objects. This is because each feature would have to be compared against the remaining features, resulting in a complexity of $\mathcal{O}(n^2)$. The proposed solution is to instead define the current observation as the reference from which other regions are measured against.

$$\mathbf{S}(o_n, O_p) = \begin{bmatrix} [o_n \ , \ o_{i_1}] \\ \vdots \\ [o_n \ , \ o_{i_{end}}] \end{bmatrix} \tag{6.1}$$

$$= \begin{bmatrix} [\chi^2(\mathbf{f}_n, \mathbf{f}_{i_1}) \ , \ NMSE(\mathbf{s}_n, \mathbf{s}_{i_1})] \\ \vdots \\ [\chi^2(\mathbf{f}_n, \mathbf{f}_{i_{end}}) \ , \ NMSE(\mathbf{s}_n, \mathbf{s}_{end})] \end{bmatrix} \tag{6.2}$$

Equation 6.1 shows the similarity matrix comparing each of the likely objects' features $o_i \in O_p$, $i \neq n$, against the current observation o_n. Each row in the matrix is the similarity vector defined in Equation 5.5, and the first column is a measure of point feature histogram similarity, while the second is a measure of slope distribution array similarity.

The similarity matrix is sorted according to the first column, followed by the second, in descending order. Thus, the least similar feature, s_{min}, will correspond to the first row.

s_{min} could potentially be associated with multiple experiences $E_{s_{min}}$. The solution is to determine which target experience $e \in E_{s_{min}}$ has the lowest associated cost of getting to. The cost from one experience node to another is defined in Equation 6.3 where w_t is the translational distance, w_r is the angular displacement and n_l is the number of connections to and from other nodes.

$$w = w_t + w_r + \frac{1}{n_l} \tag{6.3}$$

The path from the *current* experience to a potential experience node is calculated using Matlab's *graphshortestpath* function (The MathWorks, 2018), which is set to use

Dijkstra's algorithm for calculating the shorted path based on the cost defined in Equation 6.3.

Once the paths from the *current* experience to all the potential experiences $E_{s_{min}}$ are calculated, the experience with he shortest path is targeted. The whisker-array is moved towards the target experience, until a desired confidence level is reached. If the confidence of another object identity exceeds the current highest object identity, the procedure is repeated and a new least similar region is targeted. Figure 6.3 illustrates an example of a region and its associated least similar region.

6.1.2.4 Object identification condition

In the work described in Section 5.2, the algorithm concludes that a particular object has been identified given a confidence level that is higher than a pre-defined threshold (RC method). Through further analysis it has been observed that a better measure would be to integrate the confidence and observe when it exceeds a pre-defined threshold (IC method).

Through integration, the history of the confidence level for a particular object is taken into account. If an identity has a non-zero confidence for a larger period of time, it would suggest that more evidence has been gathered in its favour. The previous method of choosing the identity whose confidence first exceeds a threshold is premature as the sudden increase in confidence may be attributed to false re-localizations. The integral approach is an attempt to minimize the effect of false re-localizations and improve the robustness of the recognition system.

Figure 6.4 illustrates the raw and integrated confidence plots for the Box-Stack and Box objects. Using the RC method, and a threshold value of 3, the correct identity will be selected. However, this given a slight change to the threshold value, for example 4, the incorrect identity would be selected. Looking at the confidence integral plot, the IC method allows for a larger margin of error with respect to a specific threshold value.

6.1.3 Simulation Setup

6.1.3.1 Object map generation

Before object recognition can be carried out, each object needs to be mapped. A map is generated by exploring each object and visiting multiple regions across its surface, saving the resulting object-map generated by WhiskerRatSLAM. The order and placement of each visited region, in the case of the current work, is defined by an ellipsoid mesh that is fit around each object. The whisker-array is set to visit each face of the ellipsoid mesh

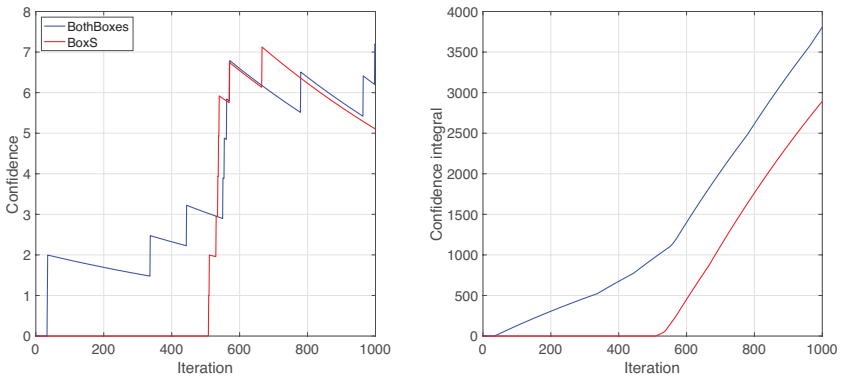


Figure 6.4: Confidence and integrated confidence plots. The integrated confidence is found to be a more robust measure of object identity as it takes into account past confidence levels for a particular object. The previous method (RC) that only considers the raw confidence value is vulnerable to noise as a threshold can be crossed given a sudden spike in confidence levels. Using the IC method, a continuous high confidence level is required before a change in dominant object identity can occur.

(a) Box Stack (b) Box

Figure 6.5: The simulation runs include the whisker-array exploring 2 unique objects. The Box and Box Stack objects were specifically selected to observe how well the object recognition system performs when presented with ambiguous objects.

and, following contact, adjust according to the surface-placement strategy. Once the conditions of the surface-placement strategy was met, the whisker-array moved on to the closest unvisited face. A single loop is defined as the visitation of all the mesh faces once. A total of two loops were executed to ensure that a good variety of features were observed for each surface region.

Noise was added to the odometry of the whisker-array, as well as to the simulated whisker-contact features.

6.1.3.1.1 Odometry noise To calculate an iteration's odometry input, the displacement from the previous iteration's pose to the next is derived using Equation 6.4, where $\boldsymbol{T}_{i-1\to i}$ is the transformation matrix describing the change in pose and T_{i-1}^{-1} is the inverse of the previous iteration's pose, and T_i the current iteration's pose.

$$\boldsymbol{T}_{i-1\to i} = T_{i-1}^{-1} T_i \tag{6.4}$$

To obtain a plausible velocity value that was within the expected range of a physical run, the angular velocity is drawn from a normal distribution of the form $\omega \sim \mathcal{N}(0.03, 0.0087)$, where the parameters are based on the observed distribution of angular velocities in the physical data-set used in Section 5.2.

$$\boldsymbol{R}_{i-1\to i} = \boldsymbol{T}_{i-1\to i}[1:3, 1:3] \tag{6.5}$$

Angular displacement is calculated by first converting the rotation matrix $\boldsymbol{R}_{i-1\to i}$, which is defined in Equation 6.5, to an axis-angle representation of the form $[\mathbf{e}^T, \theta]$. The rotation axis is a column vector $\mathbf{e}$, and the magnitude of rotation is θ.

Further noise is added to the rotational component of the odometry by rotating the rotation matrix $\boldsymbol{R}_{i-1\to i}$. The rotation noise is defined according to a vector of euler angles, where each element is drawn from the normal distribution $\mathcal{N}(0, 2.5^\circ)$. Converting the Euler vector to rotation matrix $\boldsymbol{R}_\eta$, the resulting noisy rotation matrix can be derived using Equation 6.6

$$\boldsymbol{R}_{i-1\to i_\eta} = \boldsymbol{R}_{i-1\to i} \boldsymbol{R}_\eta \tag{6.6}$$

Converting $\boldsymbol{R}_{i-1\to i_\eta}$ back to an axis-angle representation $[\mathbf{e}_\eta^T, \theta_\eta]$, the rotation velocity vector included in the odometry input can be calculated according to Equation 6.7.

$$\boldsymbol{\omega}_\eta = \omega \mathbf{e}_\eta \tag{6.7}$$

To calculate linear velocity, the change in time is first set according to the values of the noisy angular displacement and velocity values:

$$\Delta t = \theta_\eta / \omega \tag{6.8}$$

Using the displacement vector $\boldsymbol{\delta} = \boldsymbol{T}_{i-1\rightarrow i}[1:3,4]$, noise is added to its magnitude $\|\boldsymbol{\delta}_\eta\| = \|\boldsymbol{\delta}\| + \eta$, where $\eta \sim \mathcal{N}(0, 1\text{ cm})$.

$$\boldsymbol{\delta}_\eta = \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|} \cdot \|\boldsymbol{\delta}_\eta\| \tag{6.9}$$

The noisy displacement vector is calculated according to equation 6.9 and the noise velocity vector according to equation 6.10.

$$\boldsymbol{v}_\eta = \boldsymbol{\delta}_\eta / \Delta t \tag{6.10}$$

The noisy odometry input is thus of the form shown in Equation 6.11.

$$\boldsymbol{o}_\eta = [\boldsymbol{v}_\eta, \boldsymbol{\omega}_\eta, \Delta t] \tag{6.11}$$

Noise was also added to the simulated whisker-contact features that include the positions of the contact points and the contacted surface slope.

6.1.3.1.2 Whisker-contact feature noise Contact points were obtained using the same simulated ground truth process described in Chapter 4. Noise is added to the estimated contact location according to $\mathcal{N}(0, r_\eta)$, where r_η is equal to 25% of the whisker's length. In the simulated environment, the whisker-lengths were 60 mm, 100 mm and 160 mm, for each row going from the rostral to caudal direction.

To calculate the slope measure at each whisker, the normal vector of the contacted mesh face is first determined according to Equation 6.12. Given the pose of the whisker's base at the time of contact, which is defined by a transformation matrix $T^{t_c}_{BW_i}$, the face's normal vector is projected onto the coordinate frames$x$and y components i.e. the vectors corresponding to the direction of the Hall-effect sensors. The slope is then equal to the ratio of the$y$and$x$projected normal.

$$n_x = -n \cdot T^{t_c}_{BW_i}[1:3,1] \tag{6.12}$$

$$n_y = -n \cdot T^{t_c}_{BW_i}[1:3,2] \tag{6.13}$$

$$slope = {}^{n_y}/_{n_x} \tag{6.14}$$

Slope noise is defined by the normal distribution $\mathcal{N}(0, s_\eta)$, where $s_\eta = 0.33$, which is based on the inspection of the physical data-sets used in Section 5.2.

Furthermore, a contact was registered at a probability of 98%, which is an arbitrary value chosen to test the robustness of the algorithm.

Having obtained the object maps, an unknown object is presented to the whisker-array, which is tasked with exploring the surface until a desired confidence threshold is reached, or otherwise, a time limit is exceeded.

6.1.3.2 Object Exploration

The current chapter explores the effectiveness of the surface placement controller, as well as the two proposed region search strategies for improving the recognition capabilities of the whiskered system, using the objects illustrated in Figure 6.5.

The simulation based experiments are carried out such that the state of the whisker-array during initialization can be controlled by a seed parameter. For a given unique seed value, the simulation is run three times so as to observe the effect of using: 1) the default trajectory, 2) the least similar region search strategy and 3) the follow experience history strategy. A total of 10 unique seed values are used to result in a total of 30 runs.

The default trajectory generated for the whisker array follows a spherical mesh, which is unlike the ellipsoid mesh used during the creation of the object map. The seed number affects the starting face that the whisker-array visits and thus the initial pose of the whisker-array. Once a confidence in a particular object identity increases, one of the proposed region search strategies takes over as the method of generating trajectories. The default method is activated given a zero confidence level for all objects, or otherwise if the whisker-array loses contact with the surface.

6.1.4 Physical Setup

The reality-based setup for gathering the object maps and running the object-recognition experiments is similar to the one described in Section 5.2. Like the simulation-based set up, the Box and Box-Stack objects are used, which are shown in Figure 6.5.

To generate an object map, the default trajectory was generated according to a virtual ellipsoid mesh that is fit to the object, and each mesh face is targeted. Following contact, the whisker-array is free to adjust itself according to the surface placement strategy. The number of surface placement adjustments were limited to 10 iterations to reduce the data-set collection time. A total of two loops, where one loop included visiting all the ellipsoid faces, are executed when generating an object map.

An RCP whisker control scheme was implemented at all times to reduce the likelihood of whisker breakages. RCP, which is described in Chapter 4, is set so that the whisker retracts only after the deflection sensor reaches its maximum value, thereby avoiding any

early saturation of deflection-amplitude. The support vector regression model ($svr10$) generated in Chapter 4 is used as the radial-estimation model.

Once the object maps are collected, the object exploration can be started. The object exploration trial involves presenting the whisker-array with an unknown object and observing its confidence in object identity. Each object is explored in manner where the trajectory, by default, is generated according to a spherical mesh. Given a rise in object identity confidence, the most promising search strategy suggested by the results of the simulation-based experiments will be used for confirming object identity.

6.2 Results

Figure 6.6 shows the effect that the surface placement strategy has on the confidence level for object identity. The results were obtained by setting the whisker-array to explore the Box-Stack object using a spherical mesh based trajectory. The correct object identity in this case would be the Box-Stack object while the incorrect identity would refer to the Box object.

With the surface placement controller (SPC) enabled, the average confidence value for the correct object identity is significantly higher than when the SPC is disabled. Enabling SPC also is seen to increase the average confidence in incorrect identity, however, the increase is not as great as that for the correct identity.

Considering the simulation-based runs, the performance is measured using two metrics. The first metric is the size of the margin between the time for the correct IC value to reach the chosen threshold, and that for the incorrect IC value to reach the same threshold. This metric is referred to as the Identity Margin (IM) and a higher positive value measures how well the algorithm performed at discriminating between the correct and incorrect object identities. The second metric measures how quickly the correct object identity is recognized and is referred to as the successful recognition time (SRT).

A suitable integrated confidence threshold was selected by searching for a value that resulted in a high IM median and low spread. Figure 6.7 includes four box plots that vary the IC threshold between 100, 300, 600 and 900. It can be seen that by increasing the threshold, a higher IM value can be achieved at the expense of higher variance. The optimum value is observed to be approximately 300.

With an IC threshold of 300, the LSR strategy is seen to improve the performance of the recognition system as seen by its higher IM median relative to the default search strategy and FEH. However, with 95% confidence, the medians are not considered different and the improvement in performance is not statistically significant. The same is

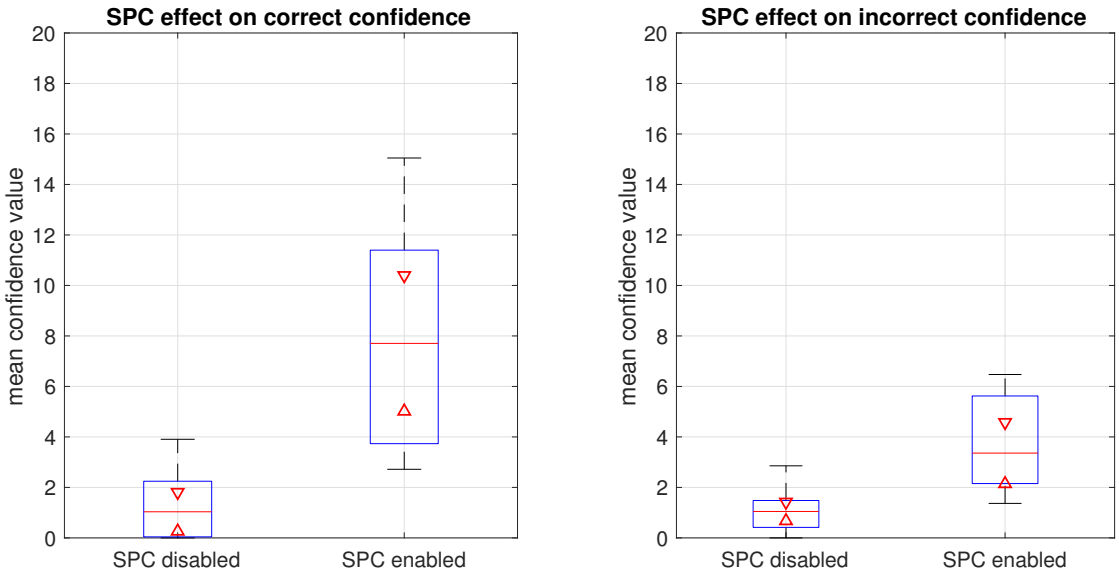


Figure 6.6: The mean confidence levels for the correct (left) and incorrect (right) object identity across multiple runs where the SPC was either enabled or disabled. For each of the two cases, 10 runs were made, using the default trajectory as the search strategy. The red line indicates the median of the data set, and the triangle markers indicate the boundary that the neighbouring group's median must lie beyond, in order for their medians to be considered significantly different at a 5% significance level. In the case of the results shown in this figure, the enabling of SPC results in a significant increase of confidence in the correct object identity. Similarly, the enabling of SPC results in a significant increase of confidence in the incorrect object identity, however, the increase is not as large as that for the correct case.

true for the FEH case, which relative to both LSR and the default strategy, has the highest IM variance.

Figure 6.9 plots the distributions of the successful recognition times for each search strategy. An improvement of performance will be illustrated by a reduction in SRT value as this would imply a quick recognition of correct object identity. It can be seen that the LSR median is the lowest out of the three strategies, however, neither the LSR or FEH strategies result in a statistically significant improvement of performance.

The results from the reality-based experiments using the LSR strategy are shown in Figure 6.10, and include both the raw and integrated confidence values for each particular object identity. Two runs were carried out, including one where the Box-Stack object was the correct object identity (top row) and a second where the Box object was the correct identity (bottom row).

Looking at the bottom plot in Figure 6.10b, it can be seen that the IC value of

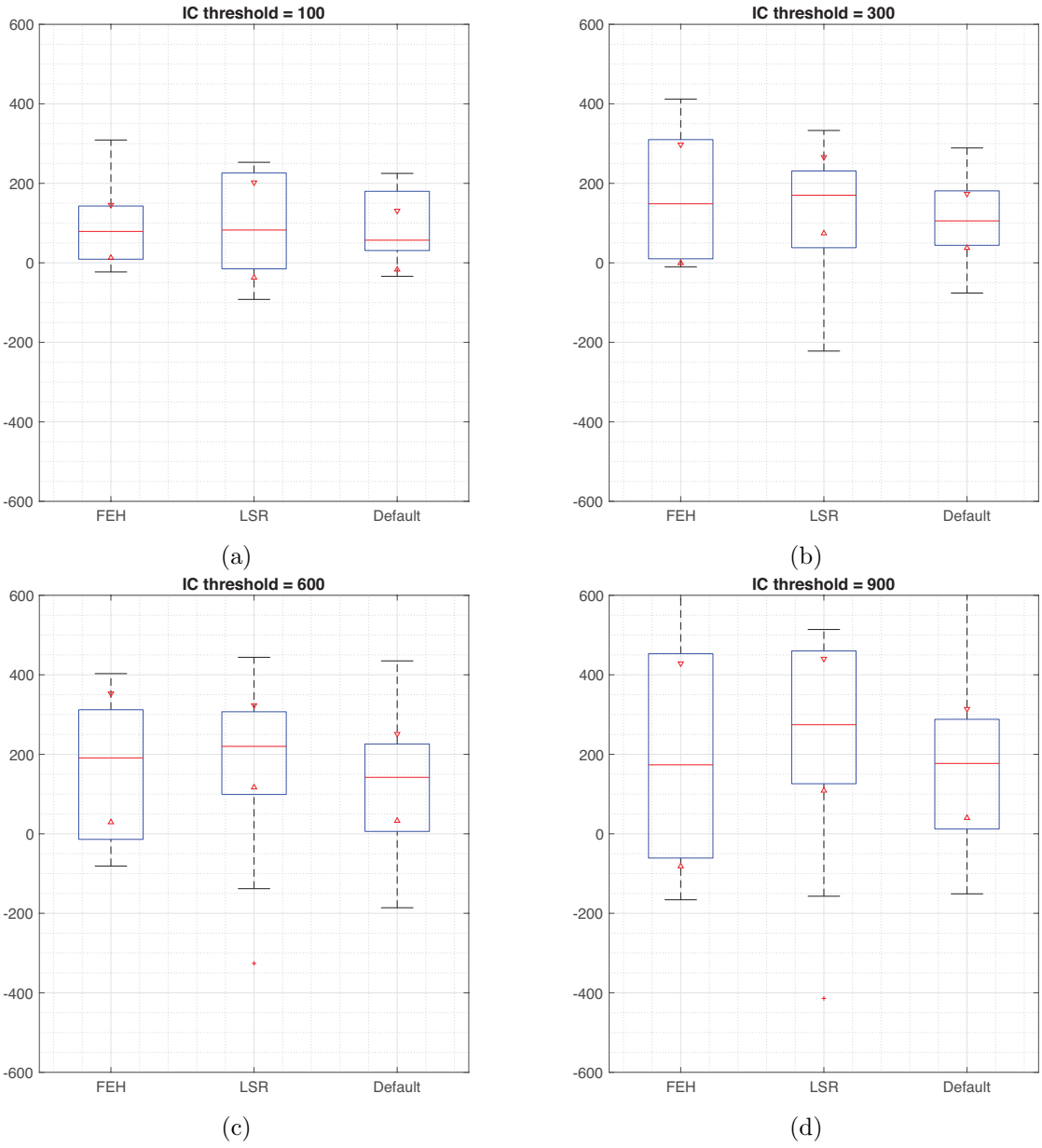


Figure 6.7: The effect of integrated confidence (IC) threshold on the distribution of the time difference between incorrect and correct identity classification. It can be seen that in the direction of increasing IC threshold, a desirable increase in Identity Margin (IM) is observed, however, this is at the expense of increasing variance. Optimizing for a high IM median and low variance for all three search strategies, an IC value of 300 is obtained.

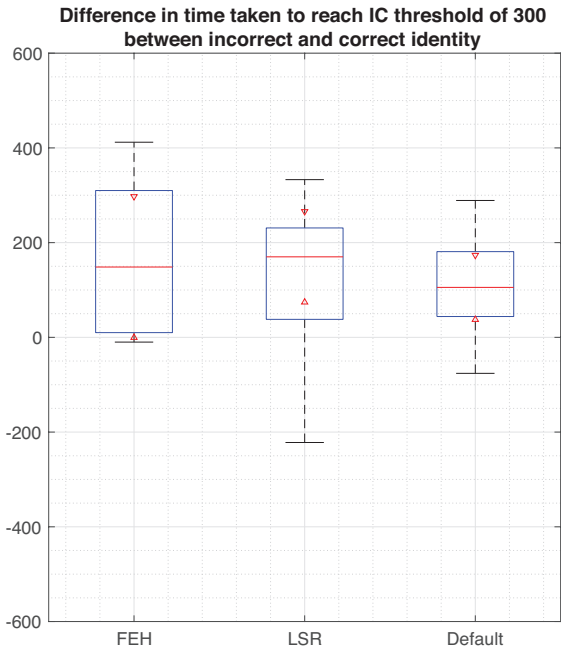


Figure 6.8: By calculating the difference between the time taken for the incorrect object identity to reach an integrated confidence value of 300, and that for a correct object identity, the performance of the recognition system can be measured. This performance measure is referred to as the Identity Margin (IM) and its distribution for each of the three search strategies is illustrated in this figure. With the highest IM median, LSR is seen to improve the performance of the recognition system. However, this improvement is not considered significant at a 5% significance level.

the incorrect (Box-Stack) object rises first, which is eventually beaten by the correct (Box) identities IC value. In the case of the upper plot where the correct object is the Box-Stack, the correct object identity always maintains a higher IC value. In both cases ambiguity in object identity occurs since both identities have a non zero IC value before the threshold is reached. However, in both cases, the correct object identity's IC value crosses the threshold first, which is considered as a successful recognition.

6.3 Discussion

The next step from Section 5.2 was to address the significantly lower confidence values experienced when using an exploration trajectory that is not the same as that used during object-map generation. The work described in this chapter addresses this problem by implemented the surface placement controller (SPC), which is shown in the previous

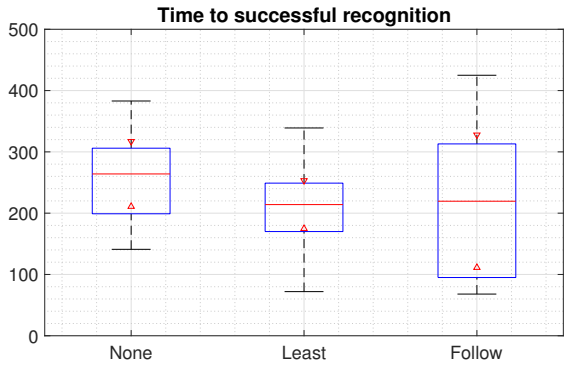
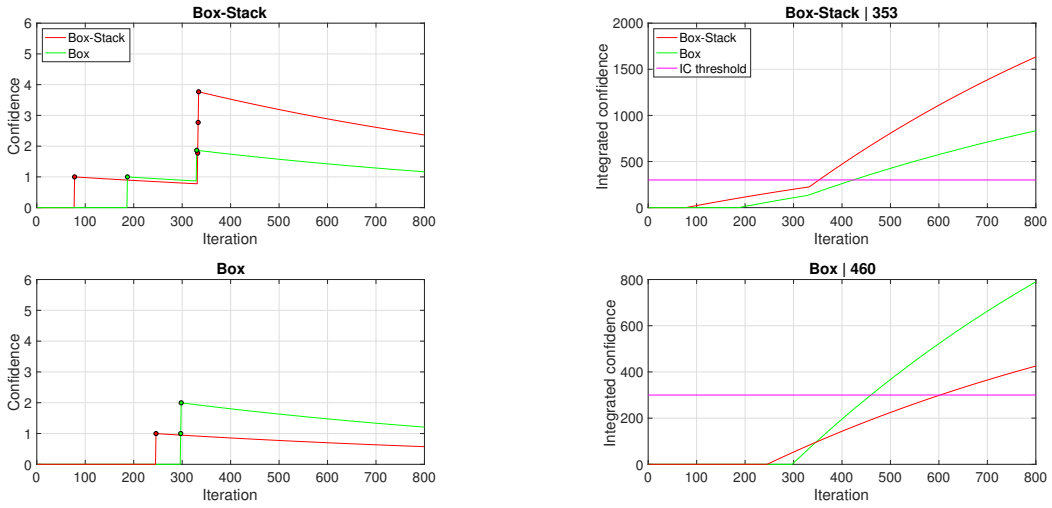


Figure 6.9: The figure includes a box plot that illustrates the distribution of successful recognition time (SRT) for each search strategy. SRT measures the time taken for the correct object identity's integrated confidence to reach the set threshold of 300, and a lower value would imply a quicker recognition process. A statistically significant improvement is not observed for either FEH or LSR relative to the default strategy.

results section that it does significantly raise the raw confidence value for the correct object identity.

The second goal from this chapter included developing a search strategy that minimizes the time taken to correctly identify an object, as well as deal with ambiguity. Figure 6.9 shows that the LSR search strategy does reduce the time taken for a successful recognition, while Figure 6.8 shows that when exploring partially similar object shapes, the LSR strategy does improve the algorithms ability to discriminate between correct and incorrect object identities. For both results, using a 5% significance, the improvements are not significant, however, the improvements do suggest that the concept of moving towards least similar regions does cause an increase of performance, which would warrant further investigation.

Ambiguity also is not always resolved as observed in other experimental runs. This could potentially be due to a low time constant for pose cell activity. Reminding the reader that the two object used are ambiguous in the upper region where the Box and Box-stack are most similar. Going lower down the body, the two object differ since the Box-stack has a protruded smaller Box object attached. If the time constant describing the decay of the pose cell activity were increased, effectively increasing the memory length of WhiskerRatSLAM, the system can better discriminate ambiguous objects as older evidence can be utilized. In the case of the two objects, is the whisker-array moves from the upper region towards the lower, less ambiguous region, the pose-cell activity packet associated with the correct object identity would more likely survive the journey and be able to receive the activity associated with observing the less ambiguous region.



(a) Raw confidence value.

(b) Integrated confidence value.

Figure 6.10: The reality-based object recognition experiments present the whisker-array with two different objects and use the LSR strategy for confirming object identity. Two runs are highlighted in the results where the top row belongs to the run where the Box-Stack object is the correct object identity and the bottom, the Box object is the correct object identity. Ambiguity occurs in both runs as there is a non-zero IC value for multiple identities before the IC threshold is crossed. For both cases the correct object identity is recognized as the IC value for the correct identity is the first to reach the chosen threshold of 300. It must be noted that the y-axis limit of the plots in Figure 6.10b are not the same — this is to show more clearly the state of the IC levels at the time of threshold crossing. In summary, both runs highlight successful object recognition.

Observing the effect that pose-cell decay has on WhiskerRatSLAM's ability to deal with ambiguous objects would be an interesting topic for future work.

In conclusion an effective strategy for ensuring consistent region observations is developed for the whisker-array system. The WhiskerRatSLAM system is shown to be more confident using SPC of the correct object identity, and a trajectory other than the one used to generate the object-map may be used. This work however was not able to conclude which method was best for confirming object identity as neither strategy resulted in significant performance gains.

Chapter 7

Conclusions

The scope of this work was limited to exploring an office like environment using whisker-tactile and odometric sensors that are mounted to a mobile wheel based platform with a 6-dof arm.

The environment is assumed to be made of a uniformly textured and flat ground with spurious number of simply shaped objects such as chairs, tables, trash cans and multiple rooms. It's been discussed that such a space would be well suited to being mapped using a topological mapping approach, since the environment was populated with landmarks across a uniform surface.

Chapter 3 details the tasks undertaken to set up the localization and mapping system, which includes the integration of whisker-tactile sensing data with the RatSLAM algorithm. The work included the mapping of whisker-array sensing data to an image representation of contact features, the evaluation of different contact-angle estimating methods, and the evaluation of the bio-inspired whisker control strategy, rapid cessation of protraction. The chapter's findings illustrate how the use of RCP, and a deflection based method for estimating contact angle, result in contact features that are less ambiguous for different surface regions that in turn improve the localization performance of the algorithm. RCP was also observed to reduce the likelihood of whiskers breaking. Furthermore, the compatibility of a RatSLAM-Whisker combination was confirmed via a successful loop closure over a 2D surface with detectable random deformities.

Such an ideal terrain would not typically be encountered in an office-like environment, and would instead consist of a uniformly textured floor with sparsely distributed objects. To address this case, the proposed robotic-whisker system would need to extract higher quality whisker-contact features so as to recognize higher level environmental features such as shapes. The set of features known to be available to whisker sensors are

geometric and textural features. The current work focuses on geometrical local features for object/shape recognition. In an effort to improve the distinguishing capability of the future 3D object recognition system, a contact localization method that is of a higher fidelity than the crude tip-assumption method used in Chapter 3 is needed.

Given the system's hardware set up, a machine learning approach is taken for estimating radial distance based on whisker-deflection data. Chapter 4 thus includes an investigation into which regression model is better suited for improving the radial-distance accuracy. The results show that given the sacrifice of a slightly lower precision, a support vector regression model can be used to map sensory features to a radial-distance value, without the need to measure whisker length. With an improved reconstruction fidelity the system is better able to discriminate different object shapes, which desirable given the planned design of an object recognition system.

A particularly important issue that needed to be addressed is the inherent requirement for tactile-sensors' to orient and translate along a surface in order to sense and identify an object, which would call for a SLAM solution that is capable of handling the movement through higher dimensions. Chapter 5 presents a description of a novel, RatSLAM based algorithm dubbed WhiskerRatSLAM, which operates in 6-dimensional physical space.

The capabilities of the new algorithm is explored in Section 5.2 and is shown to be able to identify an object shape by having the whiskers whisk along its surface while performing 6D SLAM. The results include the precision of end-effector localization along an objects surface and the identification of an object from among a series of other previously mapped objects. The results indicated that the recognition performance was poor when the exploration trajectory of the whisker-array differed from the trajectory used to initially map the object. Chapter 6 addresses this problem by implementing a surface placement controller that reduces the variation in observed features at a given surface region.

Chapter 6 also explored different region search strategies that attempt to reduce the time taken to confirm an object's identity. The results from this chapter show a statistically significant improvement in correct object recognition when using the surface placement controller and no significant changes to recognition speed when using the proposed Least Similar Region or Follow Experience History search strategies.

It has thus far been illustrated that two sub-systems that are each required for object and terrain exploration, have been shown to operate successfully in Chapters 3 and 6 respectively.

7.1 Future work

Future work should include the implementation of an active surface exploration approach, which is described in the work of Mitchinson (Mitchinson & Prescott, 2013b), the integration of *object maps* into the terrain exploration model as described in Figure 5.1 such that the platform can navigate efficiently through a sparsely populated landscape defined by richly represented tactile landmarks.

With regards to Chapter 6, the strategies for increasing object recognition times failed to show any significant improvement and further investigation is needed to address this problem of dealing with object ambiguity. Adding more regional features such as textures could potentially result in improvement for both the Least Similar Region and Follow Experience History approaches. Another potential avenue for investigation is the way similarity is measured, currently a normalized mean square error is used to measure similarity of Slope Distribution Arrays. Alternative approaches could include the consideration which whisker experiences a particular value of slope, and therefore take into account changes in whisker-array orientation.

To finalize the integration of the terrain and object exploration modes, the challenge of identifying the cues needed to switch between modes needs to be addressed. This includes investigating how the system can differentiate between obstacles and objects, such as encountering a wall or when encountering a large box shaped object.

There is also the problem of whisker being caught in gaps, or at an angle that prevents it from whisking. A real-time system that can adapt the pose of the whisker-array, instantaneously and organically once a whisker is found to be in such a scenario needs to be implemented if the whisker-array is to be used in real-life applications.

The overall speed of the system is also very slow. The whisking speed is limited to avoid nose in the deflection data. This could be addressed by using: advanced filtering techniques found in research (Anderson et al., 2010), changing the whisker's material to one with a higher damping coefficient, and utilizing more complex whisking trajectories that exhibit spread reduction behavior.

The whisker-array also would require additional sensors as a last resort obstacle avoidance mechanism for deployment in real-life operations. The current system is heavily controlled and the environment is exactly mapped prior to trajectory generation, thereby allowing the system to avoid collision. If the system is to be implemented in real-life, a more robust avoidance mechanism is needed, such as the nose sensor on the Shrewbot (Pearson et al., 2011) that include an array of passive microvibrissae.

References

Ahissar, E., & Knutsen, P. M. (2008, Jun 01). Object localization with whiskers. *Biological Cybernetics*, *98*(6), 449–458. Retrieved from `https://doi.org/10.1007/s00422-008-0214-4` doi: 10.1007/s00422-008-0214-4

Ahn, S., & Kim, D. (2017). Radial distance estimation with tapered whisker sensors. *Sensors*, *17*(7), 1659.

Anderson, S. R., Pearson, M. J., Pipe, A., Prescott, T., Dean, P., & Porrill, J. (2010). Adaptive cancelation of self-generated sensory signals in a whisking robot. *IEEE Transactions on Robotics*, *26*(6), 1065–1076.

Arabzadeh, E., Panzeri, S., & Diamond, M. E. (2006). Deciphering the spike train of a sensory neuron: counts and temporal patterns in the rat whisker pathway. *Journal of Neuroscience*, *26*(36), 9216–9226.

Arleo, A., & Gerstner, W. (2000). Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. *Biological Cybernetics*, *83*(3), 287–299.

Aulinas, J., Petillot, Y. R., Salvi, J., & Lladó, X. (2008). The SLAM problem: a survey. *CCIA*, *184*(1), 363–371.

Ball, D. (2018a). *OpenRatSLAM C++.* Retrieved from `https://github.com/davidmball/ratslam`

Ball, D. (2018b). *OpenRatSLAM Matlab.* Retrieved from `https://github.com/davidmball/ratslam_matlab`

Ball, D., Heath, S., Wiles, J., Wyeth, G., Corke, P., & Milford, M. (2013, Apr 01). OpenRatSLAM: an open source brain-based SLAM system. *Autonomous Robots*, *34*(3), 149–176. Retrieved from `https://doi.org/10.1007/s10514-012-9317-9` doi: 10.1007/s10514-012-9317-9

Ball, D., Heath, S., Wyeth, G., & Wiles, J. (2010). iRat: Intelligent rat animat technology. *Proceedings of the 2010 Australasian Conference on Robotics and Automation*, 1–3.

Ball, D., & Milford, M. (2015, January). *OpenRatSLAM datasets.* Retrieved from

https://wiki.qut.edu.au/display/cyphy/openRatSLAM+datasets

Bazeille, S., & Filliat, D. (2011). Incremental topo-metric slam using vision and robot odometry. In *IEEE International Conference on Robotics and Automation 2011 (ICRA)* (p. 4067-4073).

Berg, R. W., & Kleinfeld, D. (2003, Jan). Rhythmic whisking by rat: Retraction as well as protraction of the vibrissae is under active muscular control. *J Neurophysiol*, *89*(1), 104–117. doi: 10.1152/jn.00600.2002

Bird, C. M., & Burgess, N. (2008). The hippocampus and memory: insights from spatial processing. *Nature Reviews Neuroscience*, *9*(3), 182.

Birdwell, J. A., Solomon, J. H., Thajchayapong, M., Taylor, M. A., Cheely, M., Towal, R. B., ... Hartmann, M. J. Z. (2007). Biomechanical models for radial distance determination by the rat vibrissal system. *Journal of Neurophysiology*, *98*(4), 2439–2455. Retrieved from http://jn.physiology.org/content/98/4/2439 doi: 10.1152/jn.00707.2006

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Brecht, M., Naumann, R., Anjum, F., Wolfe, J., Munz, M., Mende, C., & Roth-Alpermann, C. (2011, 10). The neurobiology of Etruscan shrew active touch. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *366*(1581), 3026–3036.

Burak, Y., & Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS computational biology*, *5*(2), e1000291.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., ... Leonard, J. J. (2016). Past, present, and future of Simultaneous Localization And Mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, *32*(6), 1309–1332.

Carvell, G. E., & Simons, D. J. (1990). Biometric analyses of vibrissal tactile discrimination in the rat. *Journal of Neuroscience*, *10*(8), 2638–2648.

Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical science*, 2–30.

Chitta, S., Sucan, I., & Cousins, S. (2012). Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, *19*(1), 18–19.

Craig, J. J. (2009). *Introduction to robotics: mechanics and control, 3/e*. Pearson Education India.

Dahiya, R. S., Metta, G., Valle, M., & Sandini, G. (2010, Feb). Tactile sensing from humans to humanoids. *IEEE Transactions on Robotics*, *26*(1), 1-20. doi: 10.1109/

TRO.2009.2033627

Diamond, M. E., & Arabzadeh, E. (2013). Whisker sensory system – from receptor to decision. *Progress in Neurobiology*, *103*, 28 - 40. Retrieved from http://www.sciencedirect.com/science/article/pii/S0301008212000895 (Conversion of Sensory Signals into Perceptions, Memories and Decisions) doi: http://dx.doi.org/10.1016/j.pneurobio.2012.05.013

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, *58*(15-16), 1–35.

dos Santos, J. P. M. (2013). *SmokeNav-Simultaneous Localization And Mapping in reduced visibility scenarios*. University of Coimbra.

Durrant-Whyte, H., & Bailey, T. (2006, June). Simultaneous Localization And Mapping: part I. *Robotics Automation Magazine, IEEE*, *13*(2), 99-110.

Etienne, A. S., Maurer, R., & Séguinot, V. (1996). Path integration in mammals and its interaction with visual landmarks. *Journal of Experimental Biology*, *199*(1), 201–209.

Evans, M., Fox, C., Lepora, N., Pearson, M., Sullivan, J. C., & Prescott, T. (2013). The effect of whisker movement on radial distance estimation: a case study in comparative robotics. *Frontiers in Neurorobotics*, *6*, 12. Retrieved from http://journal.frontiersin.org/article/10.3389/fnbot.2012.00012 doi: 10.3389/fnbot.2012.00012

Fend, M., Bovet, S., Yokoi, H., & Pfeifer, R. (2003, Oct). An active artificial whisker array for texture discrimination. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* (Vol. 2, p. 1044-1049 vol.2). doi: 10.1109/IROS.2003.1248782

findpeaks. (2017, Nov). Retrieved 15/11/2017, from https://www.mathworks.com/help/signal/ref/findpeaks.html

fitrsvm. (2017, Nov). Retrieved 14/11/2017, from https://uk.mathworks.com/help/stats/fitrsvm.html

Fox, C., Evans, M., Pearson, M., & Prescott, T. (2012, May). Tactile SLAM with a biomimetic whiskered robot. In *IEEE Int. Conf. Robotics and Automation (ICRA)* (p. 4925-4930). doi: 10.1109/ICRA.2012.6224813

Fox, C. W., Mitchinson, B., Pearson, M. J., Pipe, A. G., & Prescott, T. J. (2009). Contact type dependency of texture classification in a whiskered mobile robot. *Autonomous Robots*, *26*(4), 223–239. Retrieved from http://dx.doi.org/10.1007/s10514-009-9109-z doi: 10.1007/s10514-009-9109-z

Georgopoulos, A. P., Schwartz, A. B., & Kettner, R. E. (1986). Neuronal population

coding of movement direction. *Science*, *233*(4771), 1416–1419.

Gere, J. M. (2004). *Mechanics of materials* (6th ed.). Thomson Learning.

Gnana Sheela, K., & Deepa, S. N. (2013, 06). Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, *2013*.

Gordon, G., Fonio, E., & Ahissar, E. (2014, October). Learning and control of exploration primitives. *J. Comput. Neurosci.*, *37*(2), 259–280. doi: 10.1007/s10827-014-0500-1

Grant, R. A., Mitchinson, B., Fox, C. W., & Prescott, T. J. (2009). Active touch sensing in the rat: Anticipatory and regulatory control of whisker movements during surface exploration. *Journal of Neurophysiology*, *101*(2), 862–874. Retrieved from `http://jn.physiology.org/content/101/2/862` doi: 10.1152/jn.90783.2008

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., & Kwok, N. M. (2016). A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, *116*(1), 66–89.

Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., & Nordlund, P.-J. (2002). Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, *50*(2), 425–437.

Hayman, R. M. A., Casali, G., Wilson, J. J., & Jeffery, K. J. (2015). Grid cells on steeply sloping terrain: evidence for planar rather than volumetric encoding. *Frontiers in Psychology*, *6*, 925. Retrieved from `http://journal.frontiersin.org/article/10.3389/fpsyg.2015.00925` doi: 10.3389/fpsyg.2015.00925

Hetzel, G., Leibe, B., Levi, P., & Schiele, B. (2001). 3d object recognition from range images using local feature histograms. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (Vol. 2, p. II-394-II-399 vol.2). doi: 10.1109/CVPR.2001.990988

Hodaň, T., Matas, J., & Obdržálek, Š. (2016). On evaluation of 6d object pose estimation. In *European Conference on Computer Vision* (pp. 606–619).

Huet, L. A., Rudnicki, J. W., & Hartmann, M. J. (2017). Tactile sensing with whiskers of various shapes: determining the three-dimensional location of object contact based on mechanical signals at the whisker base. *Soft robotics*, *4*(2), 88–102.

Jadhav, S. P., & Feldman, D. E. (2010). Texture coding in the whisker system. *Current Opinion in Neurobiology*, *20*(3), 313 - 318. Retrieved from `http://www.sciencedirect.com/science/article/pii/S095943881000036X` (Sensory systems) doi: http://dx.doi.org/10.1016/j.conb.2010.02.014

Jazar, R. N. (2010). *Theory of applied robotics: kinematics, dynamics, and control*. Springer Science & Business Media.

Jeffery, K. J., Jovalekic, A., Verriotis, M., & Hayman, R. (2013, 10 008). Navigating in a three-dimensional world. *Behavioral and Brain Sciences*, *36*(5), 523-543. doi: 10.1017/S0140525X12002476

Kajiura, S. M., & Holland, K. N. (2002). Electroreception in juvenile scalloped hammerhead and sandbar sharks. *Journal of Experimental Biology*, *205*(23), 3609–3621.

Kalman, R. E. (1960, 03). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35-45. Retrieved from `https://doi.org/10.1115/1.3662552` doi: 10.1115/1.3662552

Karamizadeh, S., Abdullah, S. M., Halimi, M., Shayan, J., & Javad Rajabi, M. (2014). Advantage and drawback of support vector machine functionality. In *Computer, Communications, and Control Technology (I4CT), 2014 International Conference on* (pp. 63–65).

Kim, D., & Möller, R. (2007). Biomimetic whiskers for shape recognition. *Robotics and Autonomous Systems*, *55*(3), 229 - 243. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0921889006001400` doi: http://dx.doi.org/10.1016/j.robot.2006.08.001

Kim, J.-H., Starr, J. W., & Lattimer, B. Y. (2015). Firefighting robot stereo infrared vision and radar sensor fusion for imaging through smoke. *Fire Technology*, *51*(4), 823–845. Retrieved from `http://dx.doi.org/10.1007/s10694-014-0413-6` doi: 10.1007/s10694-014-0413-6

Krupa, D. J., Matell, M. S., Brisben, A. J., Oliveira, L. M., & Nicolelis, M. A. L. (2001). Behavioral properties of the trigeminal somatosensory system in rats performing whisker-dependent tactile discriminations. *Journal of Neuroscience*, *21*(15), 5752–5763. Retrieved from `http://www.jneurosci.org/content/21/15/5752`

Kummerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., & Kleiner, A. (2009). On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, *27*(4), 387–407. Retrieved from `http://dx.doi.org/10.1007/s10514-009-9155-6` doi: 10.1007/s10514-009-9155-6

Liang, C.-H., Chuang, C.-L., Jiang, J.-A., & Yang, E.-C. (2016). Magnetic sensing through the abdomen of the honey bee. *Scientific reports*, *6*, 23657.

Liu, H., Wu, Y., Sun, F., & Guo, D. (2017). Recent progress on tactile object recognition. *International Journal of Advanced Robotic Systems*, *14*(4), 1729881417717056. Retrieved from `https://doi.org/10.1177/1729881417717056` doi: 10.1177/1729881417717056

Lottem, E., & Azouz, R. (2009). Mechanisms of tactile information transmission through whisker vibrations. *The Journal of Neuroscience*, *29*(37), 11686-11697. Re-

trieved from http://www.jneurosci.org/content/29/37/11686.abstract doi: 10.1523/JNEUROSCI.0705-09.2009

Lozano-Perez, T. (2012). *Autonomous robot vehicles.* Springer Science & Business Media.

Lucianna, F. A., Albarracin, A. L., Vrech, S. M., Farfan, F. D., & Felice, C. J. (2016). The mathematical whisker: A review of numerical models of the rat's vibrissa biomechanics. *Journal of Biomechanics*, *49*(10), 2007 - 2014. Retrieved from http://www.sciencedirect.com/science/article/pii/S0021929016305905 doi: https://doi.org/10.1016/j.jbiomech.2016.05.019

Melexis. (n.d.). *MLX90316 rotary position sensor IC.* Retrieved 2019, from https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwip4d284cPmAhU0BWMBHYwzBfsQFjAAegQIAxAC&url=https%3A%2F%2Fwww.melexis.com%2F-%2Fmedia%2Ffiles%2Fdocuments%2Fdatasheets%2Fmlx90316-datasheet-melexis.pdf&usg=AOvVaw3Zvu1SXMZ3RpxisnrhySMo

Milford, M., & Wyeth, G. (2008, Oct). Mapping a suburb with a single camera using a biologically inspired SLAM system. *Robotics, IEEE Transactions on*, *24*(5), 1038-1053.

Mitchinson, B., Pearson, M., Pipe, A., & Prescott, T. (2012a). The emergence of action sequences from spatial attention: Insight from rodent-like robots. In T. Prescott, N. Lepora, A. Mura, & P. Verschure (Eds.), *Biomimetic and Biohybrid Systems* (Vol. 7375, p. 168-179). Springer Berlin Heidelberg.

Mitchinson, B., Pearson, M. J., Pipe, A. G., & Prescott, T. J. (2012b). Predictive prey pursuit in a whiskered robot. In *Conference Towards Autonomous Robotic Systems* (pp. 343–353).

Mitchinson, B., & Prescott, T. J. (2013a, 09). Whisker movements reveal spatial attention: A unified computational model of active sensing control in the rat. *PLOS Computational Biology*, *9*(9), 1-16. Retrieved from https://doi.org/10.1371/journal.pcbi.1003236 doi: 10.1371/journal.pcbi.1003236

Mitchinson, B., & Prescott, T. J. (2013b). Whisker movements reveal spatial attention: Unified computational model of active sensing control in the rat. *PLoS Computational Biology*, *9*(9).

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). *FastSLAM: A factored solution to the simultaneous localization and mapping problem.*

Morais, J. P., Georgiev, S., & Sprößig, W. (2014). *Real quaternionic calculus handbook.* Springer.

Mortimer, B., Rees, W. L., Koelemeijer, P., & Nissen-Meyer, T. (2018). Classifying

elephant behaviour through seismic vibrations. *Current Biology*, *28*(9), R547–R548.

Moser, E. I., Kropff, E., & Moser, M.-B. (2008). Place cells, grid cells, and the brain's spatial representation system. *Annual review of neuroscience*, *31*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Negnevitsky, M., & Intelligence, A. (2005). *A guide to intelligent systems*.

O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain research*.

Pammer, L., O'Connor, D. H., Hires, S. A., Clack, N. G., Huber, D., Myers, E. W., & Svoboda, K. (2013, 04). The mechanical variables underlying object localization along the axis of the whisker. *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, *33*(16), 6726–6741. Retrieved from `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3733083/` doi: 10.1523/JNEUROSCI.4316-12.2013

Pearson, M., Mitchinson, B., Pipe, A., & Prescott, T. (2011). Biomimetic vibrissal sensing for robots. *Phil. Trans. of the Royal Society, B*, *366*, 3085-3096.

Pearson, M. J., Fox, C., Sullivan, J. C., Prescott, T. J., Pipe, T., & Mitchinson, B. (2013). Simultaneous localisation and mapping on a multi-degree of freedom biomimetic whiskered robot. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (pp. 586–592).

Petersen, K. B., Pedersen, M. S., et al. (2008). The matrix cookbook. *Technical University of Denmark*, *7*(15), 510.

Pezzementi, Z., Plaku, E., Reyda, C., & Hager, G. D. (2011, June). Tactile-object recognition from appearance information. *IEEE Transactions on Robotics*, *27*(3), 473-487. doi: 10.1109/TRO.2011.2125350

Pillai, S., & Leonard, J. (2015). Monocular SLAM supported object recognition. *arXiv preprint arXiv:1506.01732*.

polyfitn. (2017, Nov). Retrieved 14/11/2017, from `https://uk.mathworks.com/matlabcentral/fileexchange/34765-polyfitn`

Prescott, T. J., Mitchinson, B., & Grant, R. A. (2011). Vibrissal behavior and function. *Scholarpedia*, *6*(10), 6642. (revision #153103) doi: 10.4249/scholarpedia.6642

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, p. 5).

Ramsden, E. (2011). *Hall-effect sensors: theory and application*. Elsevier.

Russell, R. A., & Wijaya, J. A. (2005, Aug). Recognising and manipulating

objects using data from a whisker sensor array. *Robotica*, *23*(5), 653–664. Retrieved from https://www.cambridge.org/core/article/recognising-and-manipulating-objects-using-data-from-a-whisker-sensor-array/121F1D0AB2031AEE4E845642F96982D7 doi: 10.1017/S0263574704000748

Rusu, R. B., Blodow, N., & Beetz, M. (2009, May). Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation* (p. 3212-3217). doi: 10.1109/ROBOT.2009.5152473

Rusu, R. B., Marton, Z. C., Blodow, N., & Beetz, M. (2008). Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on* (pp. 643–650).

Salman, M., & Pearson, M. (2016). Advancing whisker based navigation through the implementation of bio-inspired whisking strategies. In *IEEE International Conference on Robotics and Biomimetics (ROBIO 2016).*

Salman, M., & Pearson, M. J. (2018). Whisker-RatSLAM applied to 6d object identification and spatial localisation. In V. Vouloutsi et al. (Eds.), *Biomimetic and Biohybrid Systems* (pp. 403–414). Cham: Springer International Publishing.

Solomon, J. H., & Hartmann, M. J. (2006, 10 05). Biomechanics: Robotic whiskers used to sense features. *Nature*, *443*(7111), 525–525. Retrieved from http://dx.doi.org/10.1038/443525a

Stevens, B. L., Lewis, F. L., & Johnson, E. N. (2015). *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons.

Sullivan, J. C., Mitchinson, B., Pearson, M. J., Evans, M., Lepora, N. F., Fox, C. W., ... Prescott, T. J. (2012, Feb). Tactile discrimination using active whisker sensors. *IEEE Sensors Journal*, *12*(2), 350-362. doi: 10.1109/JSEN.2011.2148114

Sünderhauf, N., & Protzel, P. (2010). Beyond RatSLAM: Improvements to a biologically inspired SLAM system. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on* (pp. 1–8).

Szwed, M., Bagdasarian, K., Blumenfeld, B., Barak, O., Derdikman, D., & Ahissar, E. (2006). Responses of trigeminal ganglion neurons to the radial distance of contact during active vibrissal touch. *Journal of Neurophysiology*, *95*(2), 791–802. Retrieved from http://jn.physiology.org/content/95/2/791 doi: 10.1152/jn.00571.2005

The MathWorks, I. (2018, May). *graphshortestpath.* Retrieved from https://uk.mathworks.com/help/bioinfo/ref/graphshortestpath.html

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation.

Artificial Intelligence, *99*(1), 21–71.

Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous robots*, *15*(2), 111–127.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT press.

Tomatis, N., Nourbakhsh, I., Arras, K., & Siegwart, R. (2001). A hybrid approach for robust and precise mobile robot navigation with compact environment modeling. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (Vol. 2, p. 1111-1116 vol.2).

Towal, R. B., Quist, B. W., Gopal, V., Solomon, J. H., & Hartmann, M. J. (2011). The morphology of the rat vibrissal array: a model for quantifying spatiotemporal patterns of whisker-object contact. *PLoS computational biology*, *7*(4), e1001120.

Universal-Robots. (2018a). *Ur10 technical document.* Retrieved from `https://www.universal-robots.com/media/1801323/eng_199901_ur10_tech_spec_web_a4.pdf`

Universal-Robots. (2018b). *UR5 technical document.* Retrieved from `https://www.universal-robots.com/media/1801303/eng_199901_ur5_tech_spec_web_a4.pdf`

Universal Robots ROS. (2018). Retrieved from `https://github.com/ros-industrial/universal_robot`

Velleman, P. F., & Hoaglin, D. C. (1981). *Applications, basics, and computing of exploratory data analysis*. Duxbury Press.

Wahl, E., Hillenbrand, U., & Hirzinger, G. (2003). Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification. In *3-d digital imaging and modeling, 2003. 3dim 2003. proceedings. fourth international conference on* (pp. 474–481).

Wang, R., Thakur, C. S., Cohen, G., Hamilton, T. J., Tapson, J., & van Schaik, A. (2017). Neuromorphic hardware architecture using the neural engineering framework for pattern recognition. *IEEE transactions on biomedical circuits and systems*, *11*(3), 574–584.

Williams, C. M., & Kramer, E. M. (2010). The advantages of a tapered whisker. *PLoS One*, *5*(1), e8806.

Xu, T., Wang, W., Bian, X., Wang, X., Wang, X., Luo, J. K., & Dong, S. (2015, 08 13). High resolution skin-like sensor capable of sensing and visualizing various sensations and three dimensional shape. *Scientific Reports*, *5*, 12997. Retrieved from `http://dx.doi.org/10.1038/srep12997`

Yartsev, M. M., & Ulanovsky, N. (2013). Representation of three-dimensional space

in the hippocampus of flying bats. *Science*, *340*(6130), 367–372. Retrieved from http://science.sciencemag.org/content/340/6130/367 doi: 10.1126/science.1235338

Zaffari, G. B., dos Santos, M. M., Duarte, A. C., d. A. Fernandes, D., & d. C. Botelho, S. S. (2016, April). Exploring the dolphinslam's parameters. In *Oceans 2016 - shanghai* (p. 1-5). doi: 10.1109/OCEANSAP.2016.7485531

Zangwill, A. (2013). *Modern electrodynamics*. Cambridge University Press.

Zhang, H., Xu, F., & Zhou, L. (2010, July). Artificial neural network for load forecasting in smart grid. In *2010 international conference on machine learning and cybernetics* (Vol. 6, p. 3200-3205). doi: 10.1109/ICMLC.2010.5580713

Zuo, Y., Perkon, I., & Diamond, M. E. (2011). Whisking and whisker kinematics during a texture classification task. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *366*(1581), 3058–3069. Retrieved from http://rstb.royalsocietypublishing.org/content/366/1581/3058 doi: 10.1098/rstb.2011.0161