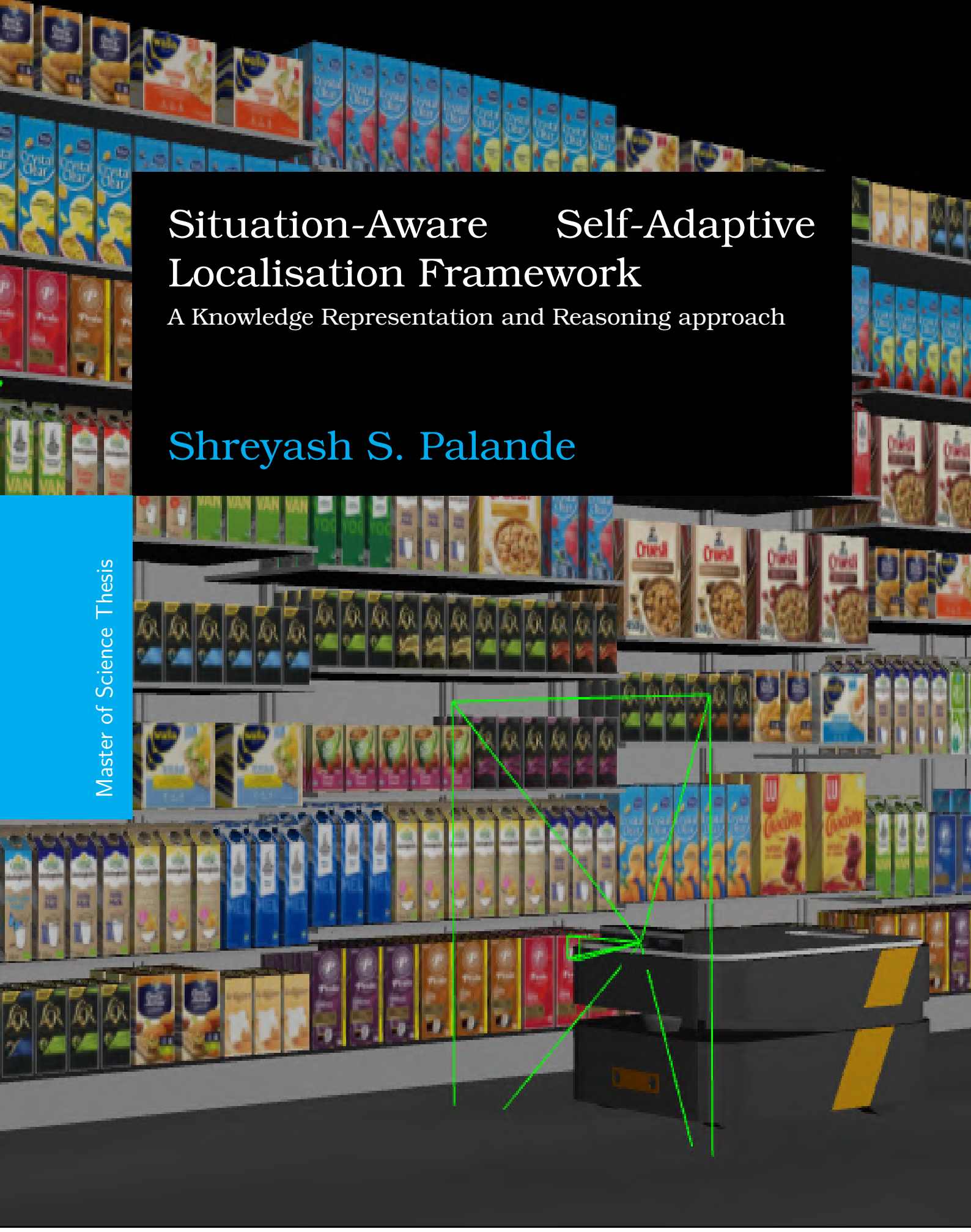

Situation-Aware Self-Adaptive Localisation Framework

A Knowledge Representation and Reasoning approach

Shreyash S. Palande

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Situation-Aware Self-Adaptive Localisation Framework

A Knowledge Representation and Reasoning approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Shreyash S. Palande

November 10, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by the department of Cognitive Robotics, Delft and AIRLab Delft. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Situation-Aware Self-Adaptive Localisation Framework

by

Shreyash S. Palande

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 10, 2020

Supervisor(s):

dr.ir. C. Hernández Corbato

dr.ir. J. Sijs

Prof.dr.ir. B. De Schutter

Reader(s):

dr.ir.P. Mohajerin Esfahani

Abstract

Substantial efforts are being made to make robots more reliable and safe to work around humans. Robots often perform flawless demos in a controlled environment under the supervision of an operator but tend to fail in the real world when deployed for a long period of time due to faults and environmental disturbances. A robotic system is composed of different physical and software components whose characteristics are likely to change over time. Assumptions made about the system during the design phase may change over time, especially when a system is deployed for long periods. Such changes that are often ignored, need to be considered. Environments in which a robot operates are dynamic with high uncertainty and unpredictability. In such scenarios, capabilities such as situational awareness and self-adaptation will be useful to create more robust, resilient and reliable solutions. The objective for this thesis work is to develop a framework which will embed capabilities such as situational-awareness, context-awareness and self-adaptation within a robot. This research provides a novel, reusable and generalised localisation framework called Situation-Aware Self-Adaptive (SASA) localisation framework for robotics application. This framework is developed using knowledge representation and reasoning which will provide a robot with the capability of adapting according to the situation. We have demonstrated the applicability of the SASA framework to a mobile robot localisation use case. In this research work we have demonstrated the performance of the framework during environmental disturbances due to poor illumination and featureless environment and internal fault due to component failure. We have also demonstrated the reusability, changeability and the consistency of SASA framework. This work showed that the situational-awareness and self-adaptation capability enhances the robot's localisation ability and provides reliable localisation even in the case of environmental uncertainties and internal faults where conventional localisation systems fail. This thesis represents a leap forward in the direction of creating more reliable and resilient solutions for robotic applications and it lays the foundations for further research in this direction.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to thank my thesis supervisor, Dr. Carlos Hernández Corbato for giving me an opportunity to work with your highly competitive and amazing research group. It was a great learning experience where I got to know a lot about Knowledge Representation and Reasoning (KR&R). During the past year, you pushed me to achieve the best out of my work, and always guided me in the right direction. I am grateful to you for making me discover the world of Meta-Controllers and giving me the freedom to develop my own ideas. I would also like to thank my colleagues from Meta-control group for giving me valuable feedback without which helped me a lot to improve my thesis work. I would also like to thank Dr. Joris Sijs for enlightening me about GRAKN.AI. and debriefing me about the environmental effects on the performance of autonomous robots which got things rolling in the right direction for this project.

Of course, I would like to thank my parents without whom I would have never made it here and also all the other people who always had faith in me and my choices. Mom, Dad, I will never be able to express all my gratitude to you. Even though it was hard sometimes, you always wanted the best for me, and you always let me follow my dreams. I would also like to thank my roommate and one of my best friends Vishwas Iyer. On days when I was overjoyed with my results or on days when I felt hopeless as to where this thesis was going, he always stood by me. Not just this thesis, but the entire journey of MSc Systems and Control would have been incomplete without you.

Finally, I want to thank all the amazing friends I made during these two years at TU Delft. I will always cherish these beautiful memories and I thank everyone for being a part of this extraordinary adventure.

Delft, University of Technology — Shreyash S. Palande
November 10, 2020

"You have to dream before your dreams can come true."
— *Dr. APJ Abdul Kalam*

Chapter 1

Introduction

Substantial efforts are being made to make robots more reliable and safe to work yet robots usually tend to fail in the real world when deployed for a long period of time. The work developed in this research project addresses this problem by exploiting self-awareness and self-adaptation capability of the robot in the direction of creating more reliable robotic solutions. This introductory chapter focuses on the motivations behind this research, posing the two fundamental questions that this thesis addresses and the main contributions of this work are highlighted. The chapter is then concluded with the outline of the document.

1-1 Research motivations

Robotics has gained a lot of attention in the past few years. Robots are being used for various applications from cleaning houses [7], to serving drinks [8] and autonomous cars [9]. Service robots are being developed for assisting humans in homes and offices [10]. With increasing autonomy, the complexity of the system is increasing and so is the need for reliability. This situation presents important challenges for the development of autonomous systems in real world applications.

Robots often perform flawless demos in controlled environments under the supervision of an operator, but usually tend to fail in the real world when deployed for a long period of time [11]. Murphay et al. describes in [12] that Unmanned Ground Vehicle (UGV) (in a disaster situation) fails 10 times as frequently as the same robot in a laboratory setting and UGVs in the field have have a Mean Time Between Failure (MTBF) of 6-20 hours. The problem lies in enhancing the robustness and fault tolerance capability of the systems for their autonomous behaviour. With increasing complexity, more number of components are added to the system thereby increasing the probability of it being faulty. Unexpected interactions of faulty components and uncertain environment changes are a threat to system reliability and functionality. Hence, it is essential to focus on the robustness of an autonomous system in order to improve its reliability.

Conventional control techniques such as feedback control only operate over a closed set of limited quantitative uncertainties and tend to fail if the system behaviour diverges from the model of the plant. The models are usually derived from a system identification technique which is a methodology for building mathematical models of dynamic systems using measurements of the input and output signals of the system. These models are derived using some assumptions during the design phase which may change over time especially when a system is deployed for long periods. The control strategy heavily relies on how closely these static models represent changes in an actual system over a longer deployment period. Intelligent control technique such as adaptive control can perform on-line identification of the process parameters, or modification of controller gains, thereby improving the robustness properties. Unfortunately, it is efficient only for linear systems whose parameters have slow variations. Sliding mode control alters the dynamics of a nonlinear system by switching from one continuous structure to another based on the region of operation. Though these types of controllers cover a large region of operation, the control strategy is static and can function properly provided the uncertain parameters or disturbances are found within some bounded set. In case of robotic systems, strong nonlinear dynamics and highly uncertain operational environment makes it difficult to model the system accurately. Even if the system and its environment are modelled well, the region of operation can't be predicted beforehand. Human intervention is required for decision-making in case of failures and uncertainties.

Having said that, in recent years, lot of advancements have been made in the direction self-adaptation for improving levels of autonomy in complex systems. IBM introduced the idea of automatic computing in 2001. The goal of autonomic computing was to realise computer and software systems and applications that can manage themselves with little or no guidance from humans. Kephart and Chess [3] proposed Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) reference model for automatic computing which enables a system to perform self management i.e. self-configuration, self-healing, self-optimisation and self-protection. The proposed model for self-adaptation divides the system into two parts i.e. a managed system and a feedback loop. The managed system consists of the plant and the control system, while the feedback loop acts as an external supervisor and deals with adaptation concerns of the managed system (e.g. heal the managed system when a fault is detected, and so forth). This model was originally proposed for Information technology (IT) systems for improving quality of service but it could be used in a robotic context to provide self-adaptation capability. Such capability would allow the system to make run-time decisions to modify it's behaviour in response to unexpected events such as changing availability of resources, faults and even environmental uncertainties.

Though MAPE-K reference model provides a solution for self-adaptation, there are two major challenges in implementing self-adaptation in robotic systems: 1) Situational-awareness: the ability of a system, to be aware of itself and its operational environment, 2) Knowledge Representation: Defining appropriate abstractions and models for capturing system's internal structure and environment in such a way that it must be inherent to the system. Though Knowledge Representation and Reasoning (KR&R) is an established research field in Artifical Intelligence (AI), little work has been done to realise knowledge processing mechanisms for robotic applications [13]. KR&R can be exploited to give machines the power of consciously making sense of things by applying logic and emulate the decision-making ability of a human expert based on beliefs and existing information.

In this thesis work, we have developed a Situation-Aware Self-Adaptive (SASA) localisation

framework for mobile robot application using KR&R which has the capability to adapt itself according to the situation. The robot localisation problem is selected due to two primary reasons: 1) The localisation problem in mobile robots still has an open loop nature, in the sense that no feedback mechanism is available to overcome the effects of uncertainties and faults. 2) It is the lowest level problem in robot navigation pipeline and even after receiving the greatest research attention in the past decades and significant advances [4], the localisation problem is not completely solved and more research is needed in the direction of reliability, fault tolerance and long term autonomy [14]. To conclude, the research motivations led to the formulation of two primary questions that this work addresses:

1. ***Can we enhance the localisation capability of a robotic system using situational-awareness and self-adaptation?***
2. ***Can explicit knowledge representation and automated reasoning about the robot's internal components be exploited to obtain an intrinsically fault tolerant and reliable systems?***

1-2 Main contributions

The goal of this research includes developing a novel Situation-Aware Self-Adaptive (SASA) localisation system which will enhance the localisation reliability and fault tolerance capability. This system is meant to be applied to mobile robots subject to uncertain environmental situations and component failures, providing a scalable, reusable and flexible solution. The main contributions of this work are detailed below:

- ***Self-adaptive situational-aware mobile robot localisation framework to perform run-time reconfiguration***

  In this work we have developed a self-adaptive localisation framework using MAPE-K reference model which can perform run-time reconfiguration in cases of environmental uncertainties and internal faults aiming for enabling robust long-term autonomy. We have also developed few environmental quality metric for quantifying environmental situations which will provide important information for taking run-time decisions.

- ***Fault detection, isolation and recovery for localisation system using deductive reasoning***

  In this work we have developed reusable semantic knowledge models in the form of ontology/knowledge graph to perform automated deductive reasoning about internal faults and constraint violations, and perform online recovery. The proposed approach simplifies the overall architecture for fault detection and recovery, exploiting the structure of the system and the inter-dependencies between the components and hierarchy.

1-3 Thesis outline

This document is organised as follows. Chapter 2 covers the localisation problem in mobile robotics, the limitations in current state-of-the-art localisation frameworks and the challenges

with localising in real world scenarios. Chapter 3 covers the main topics related to this thesis: the Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) reference model for self adaptation and Knowledge Representation and Reasoning (KR&R). In chapter 4 we describe the development process of Situation-Aware Self-Adaptive (SASA) localisation framework, the operational environment and the few examples of the scenarios that the developed system should handle. Chapter 5 discusses the development process of each element of the automatic manager in Situation-Aware Self-Adaptive (SASA) localisation framework along with the reasoning process required for performing self-adaptation. In chapter 6 we have validated the work done in this thesis using five experiments and observed the frameworks response during: 'Environmental Uncertainty' due to poor illumination, 'Internal fault' due to component failure, 'Environmental Uncertainty' due to featureless environment, 'System modifications' during robot life-cycle, and 'Consistency' of the designed framework. We have also discussed the corresponding results of the experiments and compared them with conventional localisation systems. Finally, chapter 7 concludes the work presented in this document, and it summarises the answers to the research questions previously mentioned. Besides, we have also included some recommendations for future research in this direction.

Chapter 2

Robot Localisation and it's Limitations

In this chapter we present the mobile robot localisation problem with the purpose of making this report as self-contained as possible. Section 2-1 discusses briefly about the robot localisation pipeline along with comparison of few state-of-the-art techniques used for localisation. Section 2-2 presents the limitations in the current state-of-the-art techniques and the need for situational-awareness and self-adaptation in the localisation system.

2-1 Localisation problem

Mobile robot localisation is the problem of determining the pose of a robot in the current environment. It is the most basic perceptual problem in robotics and often called position estimation or position tracking. Nearly all robotic tasks require knowledge of the location of the robots although not necessarily within a global map. Localisation can be seen as a problem of coordinate transformation. Localisation problem can be categorised into the following types:

- Local Localisation: In this problem, the initial robot pose is known and the subsequent poses are estimated as the robot moves in the environment. This is also called position tracking problem which is considered a local problem, since the uncertainty is local and confined to a region near the robot's true pose. [15].
- Global Localisation: This is a more complicated localisation problem. Here, the initial pose of the robot is unknown and the robot must determine its pose in the global map. The amount of uncertainty is much higher and assumptions cannot be made about boundedness of the pose error [15].
- Kidnapped robot problem: This problem is a variant of the global localisation problem, but it is even more difficult because the robot might believe it knows where it is while it actually does not. One might argue that robots are rarely kidnapped in practice, but

the practical importance of this problem arises from the observation that most state-of-the-art localisation algorithms cannot guaranteed 100% success and sensor failures might occur unexpectedly [15].

The mobile robot localisation pipeline at the highest level is shown in the Figure 2-1. The process of localisation can be divided into two stages: perception and state estimation. The robot uses the sensor to perceive the environment and then uses this perceived information to estimate the pose.

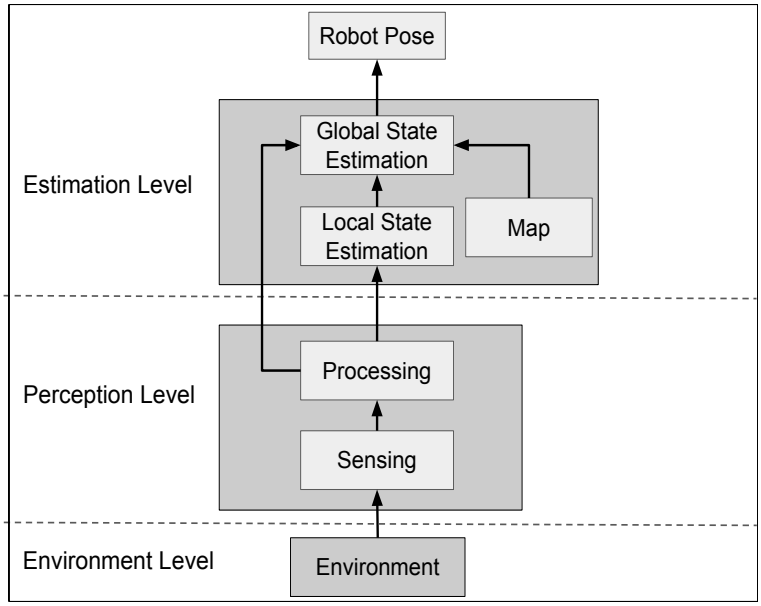

Figure 2-1: Robot Localisation Pipeline. The localisation system perceives the surrounding environment and the internal states using sensor which is then used for performing local state estimation. The local estimates are then used along with the environment map to preform global state estimation.

Various approaches are present for solving the localisation problem in mobile robots which are broadly classified into two categories: Relative localisation and Absolute localisation [16].

Relative Localisation: Relative localisation uses proprioceptive sensors[1] to extract the position and orientation information. The examples of sensors used are encoders, gyroscope, accelerometers, etc. Relative localisation can be accomplished either by odometry or by inertial navigation.

- Odometry: Odometry is the study of the position estimation of the robot using the rotations of it's wheels. Odometry is used to estimate the position relative to the starting location. Odometry uses encoders to count the number of rotations of the wheels to measure the distance traveled. The pose is determined relative to the world frame by using the robot's kinematics. To estimate an absolute position, the relative translation and orientation between two encoder readings is integrated. Odometry provides good short-term accuracy and is inexpensive but has various drawbacks such as errors due to irregular surface, wheel slippage, etc.[17].

[1]Proprioceptive sensors measure values internal to the system

- Inertial Navigation: It is performed by using the robot's motion state evaluation. It uses gyroscope and accelerometer to measure the rate of rotation and acceleration respectively. However, inertial sensor data encounters drift over time because of the integration of the acceleration data to get position. Hence, they are unsuitable to perform localisation over long time periods. They are usually used in supplement with other localisation techniques [17].

Absolute Localisation Absolute localisation uses exteroceptive sensors[2] such as beacons, landmarks, satellite-based signals to obtain the absolute position. Various methods to perform absolute localisation is given below:

- Active and passive beacons: This method computes the absolute position by measuring the distance to different beacons or by measuring the direction of incidence. Two popular methods of beacon systems are trilateration and triangulation. Trilateration uses time-of-flight information to compute the distance between the known stationary transmitter beacons and the on-board receiver. An example of trilateration is Global Positioning System (GPS). Triangulation uses multiple beacons mounted at known locations and a rotating sensor on the robot registers the angles at which it detects the transmitter beacons relative to the vehicle position or vice versa. The difference in the two is that trilateration measures distance and triangulation measures angles. Localisation using beacons is highly reliable and provides high sampling rate but it has high installation and maintenance cost. Also, trilateration based localisation faces many difficulties in indoor environment due to poor signal strength of GPS signal and triangulation based Localisation cannot be used for long range as it will require multiple beacons. [17].

- Landmark based: In this type of localisation the robot recognises specific environment features using its sensors. Landmarks can be classified into artificial landmarks (bar codes, QR codes) and natural landmarks. Robots recognise the known landmarks and calculate its position relative to the landmarks.

- Map based: The global map of the environment is stored in the robot's memory which can be a CAD model or a map built by using the sensor data. When robot moves around the environment it produces local maps which are compared with the global map stored in the memory and the actual position can be computed. This type of localisation is more used for indoor navigation because the environment is limited and hence a map can be easily developed. This method is advantageous because it uses the naturally occurring structure of typical indoor environments to derive position information without modifying the environment [16].

2-1-1 Sensors used for localisation

Sensors are used to perceive, analyse and interpret the state of mobile robot and its surroundings. Variety of sensors used for robot localisation which can be classified into proprioceptive[3]/

[2]Exteroceptive sensors acquire information from the robot's environment
[3]Proprioceptive sensors measures the internal state of the robot

exteroceptive[4] and passive[5]/active[6]. Table 2-1 shows the classification of sensors used for localisation. The reliability of the localisation system heavily relies on the sensor performance. Some sensors provide accurate results in a controlled environment but are prone to errors in the real world. Various environmental factors affect the performance of the sensors such as magnetic field affecting the magnetometer readings, low lighting affecting the image data, etc. Any physical damage can also cause sensor failures. In long term deployment conditions the reliability of the system can be improved if it is able to cope with uncertainties and failures.

Table 2-1: Classification of sensor use w.r.t. mobile robotic localisation modified from [4]

Sensor Type	Use	Sensing System	PC/EC	A/P
Wheel sensors	Wheel speed/position	Brush encoders	PC	P
		Optical Encoders	PC	A
		Magnetic encoders	PC	A
		Inductive encoders	PC	A
		Capacitive encoders	PC	A
Inertial Sensors	Acceleration, Orientation	Accelerometers	PC	P
		Compass	EC	P
		Gyroscopes	PC	P
		Inclinometers	PC	A/P
Ground-based beacons	Localisation in a fixed frame	GPS	EC	A
		Optical beacons	EC	A
		Ultrasonic beacons	EC	A
		Motion Capture	EC	P
Range Sensors	Time of Flight	Infrared sensor	EC	A
		Ultrasonic sensor	EC	A
		Laser rangefinder	EC	A
		Structured light	EC	A
Vision-based sensor	F2F Matching, F2M Matching, Optical flow	CMOS Camera	EC	P
		CCD Camera	EC	P

A, active; P, passive; P/A, passive/active; PC, proprioceptive; EC, exteroceptive

2-1-2 Techniques for state estimation

After perceiving the environment, the robot needs to maintain a meaningful position estimate in the long run. There are mainly two techniques for estimating the robot pose: Filtering-based and Optimisation-based [18, 19, 20, 21].

- Filtering-based: In this method the system state is estimated using probabilistic filtering algorithms such as Extended Kalman Filter (EKF), Particle Filter, Maximum likelihood estimation and expectation maximisation.
  - Kalman Filter: It is derived from recursive Bayesian rule [20] and is the optimal linear filter which can estimate the internal states of a dynamic system from noisy

[4]Exteroceptive sensors acquire information from the robot's environment
[5]Passive sensors measure the energy entering the sensor
[6]Active sensors emits the energy into environment and then measures the environmental reaction

measurements. However, Kalman Filter makes an assumption that the system is linear and the noise is Gaussian. Hence an extended version called EKF is used to solve probabilistic state estimation problem for non linear system. However, EKF is sensitive to erroneous data associations which can cause the filter to diverge. EKF is also difficult to apply for large scale map because the computation process increases quadratically with the landmarks [18].

– Particle Filter: Particle Filter is a nonparametric variant of recursive Bayes filter that uses random samples to describe arbitrary distributions. [22]. Particle Filters has good scalability for maps with multiple landmarks [18] which estimates the robot path and the landmarks position by several low dimensional EKFs [20]. Adaptive Monte Carlo Localization (AMCL) is one of the examples of particle filter used for robot localisation which treats the robot position as a set of randomised samples (particles), which are distributed evenly over the pose space. As the robot senses the environment, the filter is updated and it starts to converge to the real pose.

- Optimisation-based: Graph optimisation method is composed of three parts 1) motion estimation 2) loop-closure detection 3)graph optimisation. Motion estimation and loop closure acts as a front end part and graph optimisation is performed at the back end [18]. The graph based localisation approach abstract the sensor measurements and create a pose estimation problem. Each pose is represented as soft node and the landmarks which are obtained from map are treated as fixed nodes. The sensor measurements are treated as factors in the graph which are labelled with a probability distribution over relative location between two nodes. When the graph is formed, optimisation is performed to minimise the error induced by the factors which are sensor measurements [23].

2-1-3 Metric to evaluate localisation performance

To evaluate and compare performance of localisation methods, different metrics are available and among which most popularly used metrics are Absolute Trajectory Error (ATE) and %CPU usage. However, %CPU usage is highly dependent on the system configuration hence most of this metric is compared based on the performance of different methods on the same processor which will give a relative %CPU usage of different methods. To compare the trajectory we will also use ATE metric which is as follows:

$$\text{ATE} = \left(\frac{1}{N} \sum_{i=0}^{N-1} ||\Delta p_i||^2 \right)^{\frac{1}{2}} \tag{2-1}$$

where $\Delta p_i = p_i - \Delta R_i \hat{p_i}^{'}$, $p_i \in R^3$ is the position of the robot, R_i is the rotation matrix $p_i^{'} = Rp_i + t$, $t \in R^3$ = robot trajectory. N: Number of samples. For more information on ATE refer [24].

2-1-4 State of the art localisation frameworks

Localisation is quite a popular problem in robotics community and has gained lot of interest over past decades, which resulted in different types of localisation frameworks. As this thesis

Table 2-2: Best suitable LIDAR and Vision based localisation algorithm [5, 6]

Method	Framework	Accuracy [6] ($ATE_{max}[m]$)	Computation [5] (Mean % CPU)
Laser-based	Hector SLAM [25]	1.71	28.460
	GMapping [26]	4.59	25.601
Vision-based	RTAB map [27]	0.08	38.831
	ORB-SLAM2 [28]	0.31	51.042

does not focus on developing a new localisation framework but rather focuses on situational awareness and adaptation of localisation, we have decided to use the available frameworks directly for localisation. We have already compared various state-of-the-art frameworks in the literature survey which was done prior to the thesis work. The comparison was done based on two metrics: ATE and CPU usage in (%) for processing. The important results are summarised in Table 2-2 which shows the two best suitable LIDAR based and Vision based localisation frameworks according to the above two metrics. As Real-Time Appearance-Based Mapping (RTAB map) is best in terms of accuracy we have decided to use RTAB map as a default localisation method.

2-2 Limitations of the state of the art localisation frameworks

The localisation framework can be expressed in form of control loop as shown in Figure 2-2. The control loop is not fail safe as it is still open loop w.r.t. fault/failures of the different components and environmental uncertainties. If a fault occurs in any sensor, it will be propagated through the entire system and can cause erroneous pose estimation. Figure 2-3 shows the error in proprioceptive affecting the entire system causing errors in pose estimates. Erroneous estimation are not only caused due to sensor failure but also because of various other reasons which we have identified below.

2-2-1 Sources of localisation errors

The sensor characteristics can be measured in a laboratory environment. However, all factors which might affect sensor performance in the real world cannot be taken into account while testing the sensor. This is most relevant to sophisticated sensors such as laser ranging and vision-based sensors.

Wheel encoders which measure the angular velocity of the individual wheels are affected due to wheel slipping/skidding caused due to low friction surface, uneven ground, and accidental human push. Other sources of errors are misalignment of the wheels, uncertainty in the wheel diameter or reduction in the wheel diameter as the wheel wears with use and variation in the contact point of the wheel. Inertial sensors such as magnetometers might give false or noisy readings due to the disturbance of the magnetic field by other magnetic surfaces in the vicinity and their susceptibility to vibration. Sensors such as GPS can provide accurate pose information but have limitations in the indoor environment such as poor signal strength. Cameras are highly susceptible to motion blur and are dependent on lighting changes, lighting

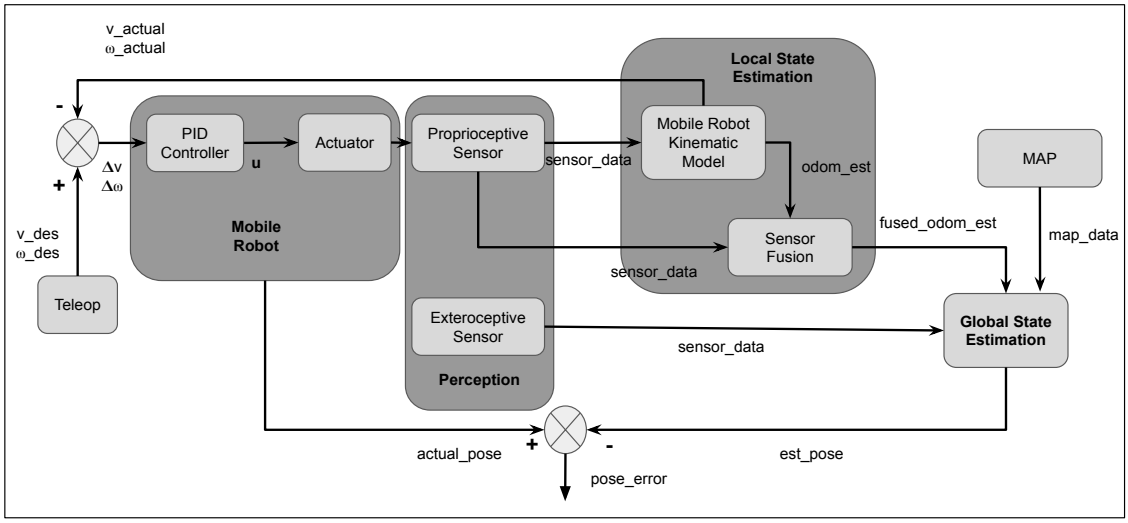

Figure 2-2: Mobile Robot localisation as a open loop with respect to fault and disturbances.

ω_des, v_des represents the desired angular and linear velocities, ω_actual, v_actual represents the angular and linear velocities, $\delta\omega, \delta v$ represents the error between the current velocities and the desired velocities.

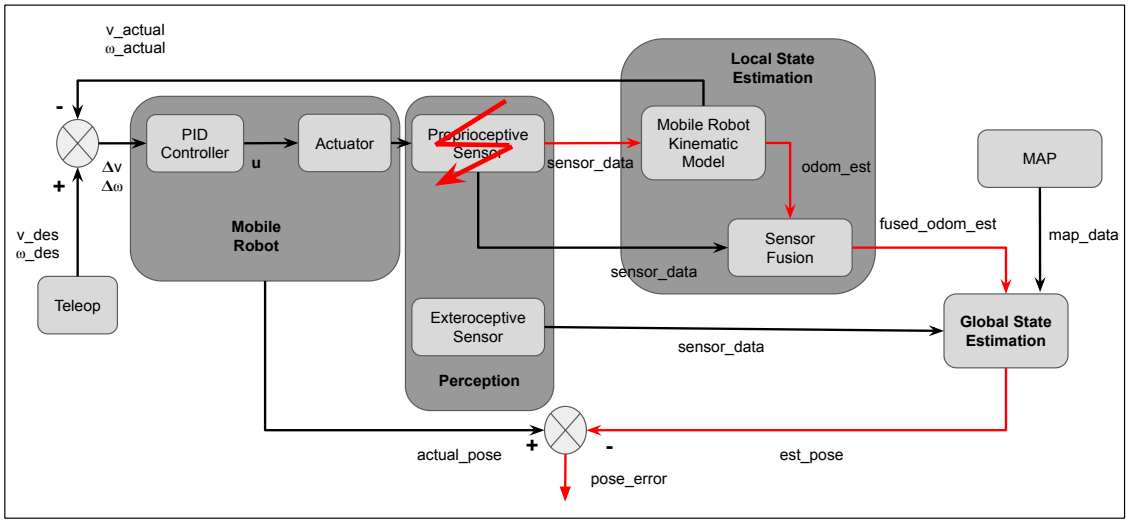

Figure 2-3: Sensor fault affecting the entire localisation system

specularity, reflections, shadows, etc [4, 29]. Narrow dynamic range in the camera might acquire saturated images in low lighting conditions. Long exposure time and large gain can cause motion blur and increased noise [30], smoke rendering on the cameras [31] and shadows caused due to dynamic obstacles can affect the quality of the camera data.

Laser scanner which comes under the category of active ranging sensor, are commonly used in robotic applications. Laser scanner uses time-of-flight technique to provide accurate distance information but is highly affected by various factors such as object surface characteristics in the surrounding, multiple returns, atmospheric transmittance [32] etc. One of the causes of error involves coherent reflection caused by highly polished surface such as mirror, polished

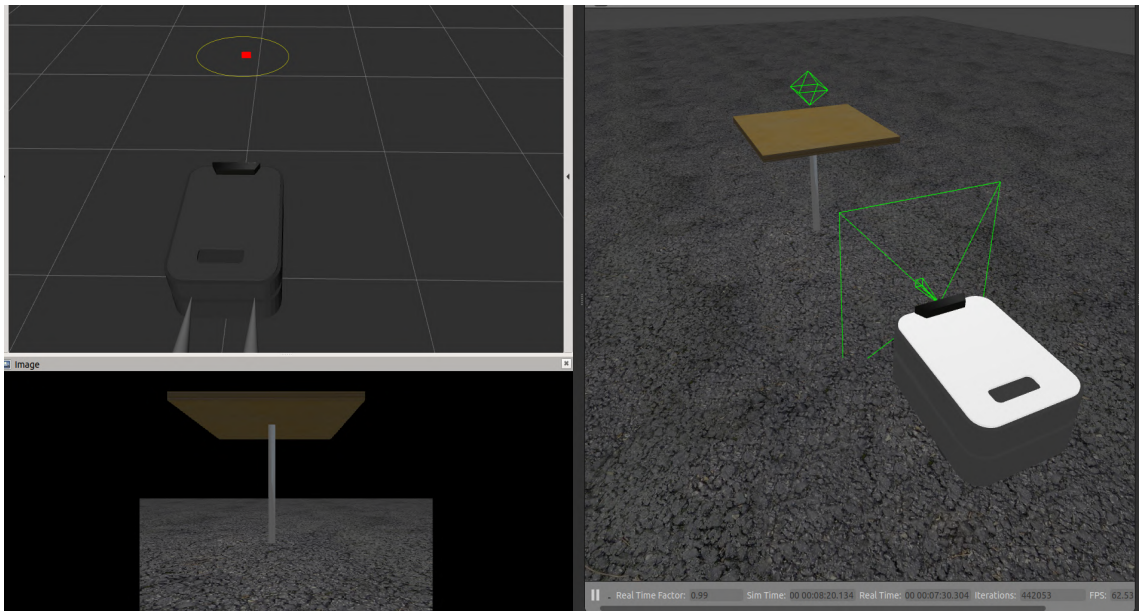

Figure 2-4: False perception due to incomplete information. The table is detected as a red dot by the 2D laser scanner.

metals and ghost readings caused due to transparent surfaces. Also laser rangefinders do not have the ability to detect transparent materials such as glass [4]. If multiple laser rangefinders are present in the close vicinity, it can cause cross-talk and interference [33, 1]. Figure 2-5 shows the affects of mutual interference on the laser reading shown in red due to the presence of other laser scanner in the vicinity. Ultrasonic range sensors will also be affected similar to laser scanners due to the reflection from smooth and angled surfaces. Also, it will face interference between multiple sonar emitters. 2D sensors such as laser scanner or ultrasonic sensors also might not produce enough information for the robot to perceive the environment correctly. For example: In case of tables, trolleys or humans the robot will only get the scan reading of the level on which the sensor is mounted which might result into faulty perception as shown in Figure 2-4 where the table is incorrectly perceived as a small red dot by the laser scanner but perceived correctly by the camera.

Situations may arise where data association between the sensor readings and available map may lead to multi modal hypothesis. For example in a repetitive feature environment the robot may believe that it is at different locations at the same time. This might also be caused due to symmetric environment and long parallel hallways. Sensor noise and environmental uncertainties reduce the ability of the robot to localise accurately. Solutions such as temporal fusion or multi-sensor fusion using Kalman Filter (KF), EKF, etc works well within a certain framework, and under certain conditions but may be unreliable and suffer divergence or convergence to an incorrect solution in presence of severe data fault. Also, the filter based sensor fusion does not allow any kind of flexibility to add and remove sensors as it models one specific system. Once the model represented in the filter matrices has been defined to accommodate certain sensors, all those sensors must give information at each iteration of the filter. If one of the sensors fails to provide new information for the current iteration, the output of the filter can be erroneous [34]. This could be solved by using Information filters

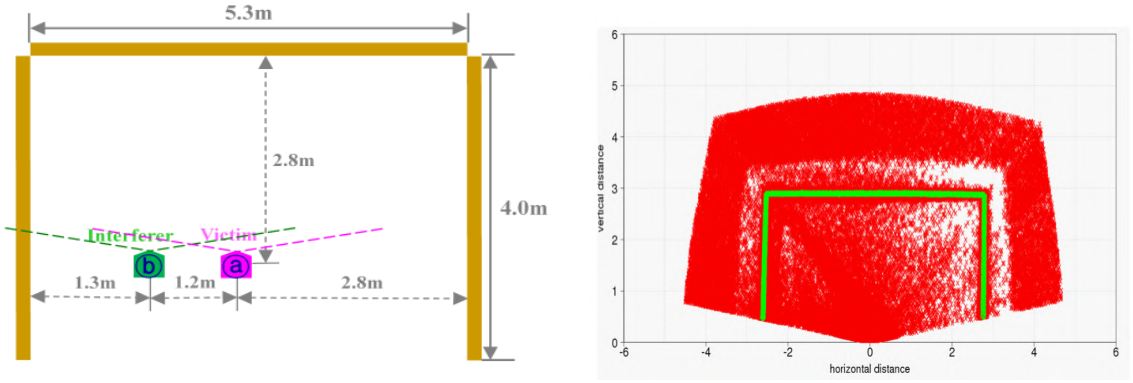

Figure 2-5: The figure in the left shows the placement of the two laser scanners placed closed to each other. The figure in the right shows the reflected rays detected by the scanner 'a' which has a lot of noise (shown in red) because of the mutual interference between the two lasers . The actual wall is shown in green.[1]

but they have other limitations such as updating the filter states is more complex than KF, the initial belief must be Gaussian [15], the inability to maintain multiple data association hypotheses which results in brittleness in the presence of ambiguous features, the static world assumption which makes the approach inapplicable to modeling moving objects [35].

There are solutions available in the literature which address the specific problem for failures and changing environment. Kim et. al [29] solves the problem of changing lighting conditions by building multiple maps of same environment for different lighting conditions, Shim et al. [36] solves the same problem by controlling camera exposure time, Koch et al. [37] solved the problem of laser reflection from transparent surface by Mirror Detector Approach which uses pre-filtering and post-filtering on laser scan data to detect reflecting surfaces. But in all these methods the problem was solved by creating a tailored solution. In an open environment predicting all such scenarios in advance is difficult. Hence, we feel a need for a solution which can manage an unexpected situations even if the system's response in that situation is not modelled in advance.

2-3 Summary

This chapter discussed the mobile robot localisation problem which can be divided into three layers i.e. perception, local state estimation and global state estimation. Different alternatives are available for developing each layer of the pipeline. Though different methods are available in literature for robot localisation there is still a gap with respect to the reliability of the localisation system which needs to be addressed. Some tailored solutions have already been developed to tackle the environmental disturbances and internal failures however as mobile robot operates in open world not all situations can be predicted in advance. To make the robot localisation more reliable, there is an urgent need to close the loop with respect to faults, failures and unexpected environmental disturbances. This motivated us to search for a technique which can make the localisation system more reliable and fault tolerant which we will discuss in the next chapter.

Chapter 3

Background

We have seen the limitations of the state-of-the-art localisation framework in chapter 2 and the need to close the loop of the localisation pipeline. In order to do this we started looking at different approaches available in literature for providing fault tolerance. Out of them the most relatable one was fault tolerant controllers. However such controllers needs an analytically model of the plant which at times if difficult to derive accurately due to its nonlinear nature. Also conventional fault tolerant controllers cannot account for qualitative uncertainties. Hence, we look further into approaches for fault tolerant control system where we came across data driven methods such as Artificial Neural Networks (ANN). But these types of approaches generally require a large amount of labelled faulty data for training diagnosis models which is quite costly and at times impossible to get. Also adaptability was another concern. In most cases they cannot extrapolate beyond the range of training data [38]. The operating conditions of the robot could be changed over its lifetime not only due to faults but also due to changing environmental conditions which will need re-training. Also they lack explanation facility. As our motivation was to develop a more reliable system, these artificial neural network based approaches could not provide more meaningful information and their reliability is still doubtful [38, 39]. Training a network could sometimes result in over-fitting which would have a negative impact on fault detection. Over-fitting tends to make a network conservative which in case of fault detection would increase false positive outcomes. Lastly, the training process is often time consuming and needs several iterations for learning.

We look further and came across self adaptive systems which could provide intrinsic fault tolerance capability. This motivated us to develop a Situation-Aware Self-Adaptive (SASA) localisation framework which could provide the robot with capabilities such as situational awareness and self-adaptation such that it could be used to obtain intrinsic fault tolerant and reliable systems. This chapter present a brief background on the topics used for developing SASA localisation framework. This chapter covers the fundamental theoretical concepts, the thinking behind the methodology used and tools chosen for Knowledge Representation and Reasoning (KR&R). Fault tolerance is of maximal relevance for our research which is covered in Section 3-1. Section 3-1-2 discuss briefly about the self-adaptive system especially using Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) reference model. Section

3-2 covers the methodology used for knowledge representation, process of ontology design and tools used for implementing the ontology in the robotic system.

3-1 Fault Tolerant Systems

Dependability have become a crucial challenge for autonomous control systems. Dependability can be defined as *"the ability to deliver service that can justifiably be trusted"* [40]. Dependability mainly combines together two system properties which are:

1. Reliability: It is the ability of the component/system to deliver the required function/services under the stated conditions for specific periods of time. Reliability basically evaluates how frequently the system becomes faulty. Reliability can closely be related to the Mean Time To Failure (MTTF) of the system.

2. Availability: It is the probability of a system to be operational when needed. Contrary to reliability, availability also depends on the system maintenance polices. Availability can closely be related to the Mean Time Between Failure (MTBF) of the system.

A dependable system is a fail-safe system with high availability and reliability. An internal failure within the system or environmental uncertainties can affect the system's functionality. In the general sense, the desired functionality of a system can be affected due to failures, uncertainties and disturbances. In case of failure, the system is unable to perform and might even cause a permanent breakdown. An error is a part of the system state which can propagate through the entire system and can cause the system to fail. It is a deviation of the system from the expected behaviour. Faults are the hypothesised cause of the error. A fault is a deviation of the system state or the system parameters from the nominal situation [41]. In the event of a fault, the system might be able to continue it's functionality with some acceptable degradation in performance but failure is an irrecoverable event and the system has to be shut down. The overall system works satisfactorily only if all components i.e. sensors and algorithms, provide the desired service. As there is so much inter-dependency between the internal components, a fault might change the performance of the whole system. In any case, the faults or disturbances are the primary cause of changes in the system functionality which eventually leads to a degraded system performance or loss of functionality [2]. Assume that the system behaviour can be described by the two variables y1 and y2. Then Figure 3-1 shows the different regions of the system behaviour. In the region *B1*, the system satisfies its function. This is the region where the system should ideally remain during its time of operation. On occurrence of faults or disturbances, the the behaviour of the system changes and the system enters the *B2* region which is acceptable because the system can still provide services with certain degradation in performance. Faults and disturbances bring the system from the region *B1* to *B2*. A fault-tolerant controller should be able to initiate recovery actions that prevent a further degradation of the performance towards the unacceptable or unsafe regions and it should move the system back into the region of desired performance. At the border between the two regions, the supervision system is invoked, which diagnoses the faults and adjusts the controlled system to the new situation [2].

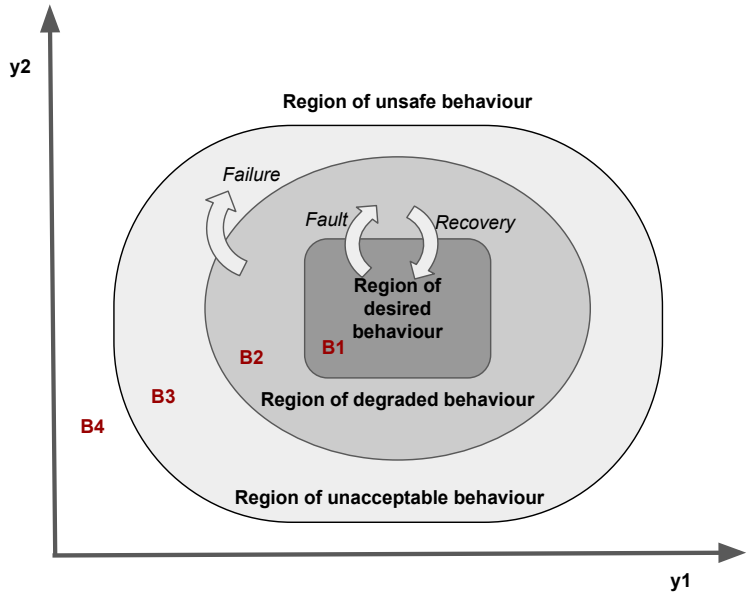

Figure 3-1: Regions of system behaviour. Faults and disturbances will take the system from B1 towards B4

3-1-1 Fault Tolerant Control Systems (FTCS)

There are certain control techniques which allow the accommodation of faults and disturbances to a certain extent [2]. They are as follows:

- Robust Control: It is an approach to design a fixed controller that tolerates changes of the plant dynamics. Fault tolerance is achieved without changing the control parameters and is hence a passive approach. However, robust controllers can perform over a restricted set of changes in plant behaviour and does not provide best performance in a nominal plant because its parameters are fixed so as to get a trade-off between performance and robustness [2].
- Adaptive Control: It is a control method in which the controller adapts its parameters with the changes in the controlled system. However, the theory of adaptive control shows that this principle is particularly efficient only for plants that are described by linear models with slowly varying parameters. These restrictions are usually not met by systems under the influence of faults, which typically have a nonlinear behaviour with sudden and large parameter changes [2].

A robust control policy is static and the controller is sub-optimal as it is designed to work assuming that certain variables will be unknown but bounded. As against robust controllers, a prior information is not required about the bounds on the uncertain or time-varying parameters in adaptive control but adaptation is limited to only slowly varying systems. On the contrary, Fault Tolerant Control (FTC) can deal with occurrence of fault on any magnitude by changing the control law to cancel the effects of the faults and unbounded uncertainties or to attenuate them to an acceptable level. In a faultless system, nominal performance can be achieved as the control law is only changed after occurrence of faults.

The general architecture of FTC is shown in Figure 3-2. Note that the faults f and disturbances d are different. Robust controllers usually take into account only the disturbances d. Moreover, it can be seen from the Figure 3-2 fault-tolerant control extends the usual feedback controller by a supervision system, which has to perform two tasks:

- Fault Diagnosis: The diagnosis block measures the control input and plant's output using sensor and test the consistency with the plant's model. It is responsible for detecting and diagnosing the fault in the system.
- Controller Redesign: The redesign block takes input from the 'diagnosis' and adjusts the controller setting to the faulty situation. Controller redesign considers the problem of changing the control structure and the control law after a fault has occurred in the plant. It aims to fulfil the performance requirements of the system even in case of faults.

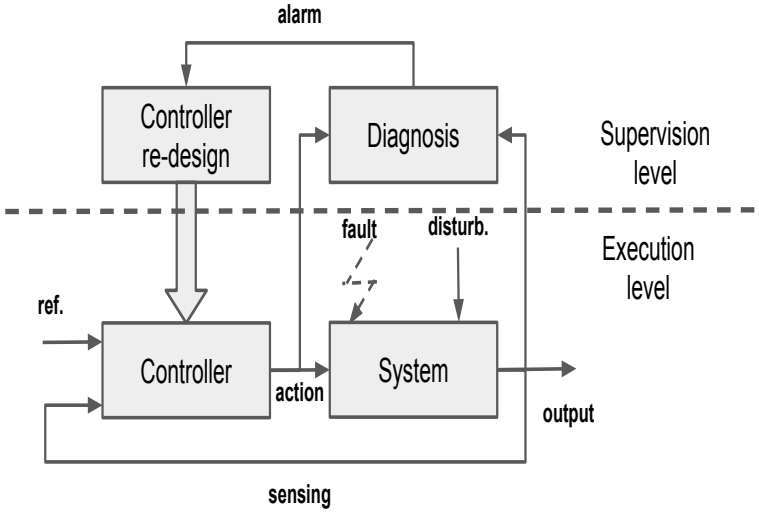

Figure 3-2: Architecture of FTC adapted from [2]

The main advantage of fault-tolerant control over other measures for fault tolerance is the fact that fault-tolerant control makes 'intelligent' use of the redundancies included in the system which makes it the most promising field for providing more reliable systems. However, in conventional fault tolerant control systems the process of fault diagnosis is divided into two parts: a residual generation module and a residual evaluation module. For residual generation, a mathematical model of the supervised process is required which is deduced from differential equations [41] and the output of the model is then compared with the actual output of the system for evaluation purpose. For the purpose of fault diagnosis, a fault model of the system is required which describes a specific fault effect. A robotic system is a complex, nonlinear and highly interdependent system which makes the process of deriving a process model and fault model very difficult and time consuming. Also a robot operates in an open environment and hence we cannot account for all the faults that might occur in the system at design time. And without information about the faults and the way in which the faults affect the system, no fault isolation and identification is possible [41]. Even if the model is derived after lot of efforts the model is specific to a single system which limits the

reusability of the model. Classical fault tolerant controllers cannot account for qualitative uncertainties. "Qualitative uncertainty refers to the occurrence of unexpected events that qualitatively change the behaviour of the plant. They cannot be accounted for in traditional control models as disturbances" [42]. This motivated us to look further into methods other than classical fault tolerant controllers which can provide fault tolerance without explicit fault models or mathematical process models and we came across Self Adaptive System (SAS). SAS system is a class of systems which are capable of modifying their runtime behaviour in order to achieve system objectives. Unpredictable circumstances such as changes in the system's environment, system faults, and changes in the priority of requirements are handled by such systems by performing runtime adaptation.

3-1-2 Self-adaptive Systems

With increasing autonomy, a robotic system is not just limited to a single controller but consists of multiple feedback loops at different levels of hierarchy with lots of inter-dependencies. Hence the controllers used for high autonomy applications are very software-intensive. In software systems a lot of research has been done on Autonomic computing, where computing systems can manage themselves given high-level objectives. This technique has been applied in areas of fault-tolerant systems where a software system can perform dynamic adaptation in relation to fault-tolerance. The dependability of a software-based system is improved by providing the system with the ability to adapt itself at run time to handle resource variability, changing user needs, and system faults. Such a system, which is able to automatically modify itself in response to changes in its operating environment is called Self Adaptive System (SAS) [43, 44]. The system modifies itself by manipulating the parameters or its behaviour in response to the changing environment or internal states. SAS can provide self-management properties such as self-configuration and self-healing in presence of failures.

Kephart and Chess [3] introduced MAPE-K reference model for SAS which is shown in Figure 3-3. Though the MAPE-K model have been there for quite some time it still remains as one of the influential reference model for autonomic and self-adaptive systems. It divides the SAS into Managed element and automatic manager. The Automatic Manager is responsible for adaptation and consists of:

- Monitor: It is responsible for monitoring the managed system and the environment. It collects and provides the information to the analyze phase.
- Analyze: It performs data analysis and reasoning on the data provided by the monitors. Analyze is responsible for determining whether adaptation actions are required using the monitored information of the managed system, the environment, and the adaptation concern of interest.
- Plan is responsible for planning the actions and adaptation necessary to achieve the goals or objectives.
- Execute: It changes the behaviour of the managed resources using the actions recommended from the plan phase.
- Knowledge: Knowledge are the models that provide abstraction of the relevant aspects of the system, its operating environment and the adaptation goals.

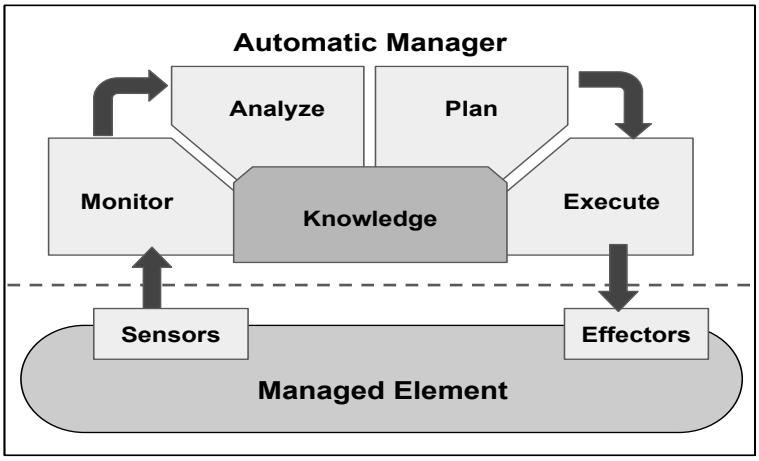

Figure 3-3: MAPE-K Feedback Loop for Self-adaptive system adapted from [3]

MAPE-K Feedback loop is an abstract representation of the reference control model which will provide fault-tolerance and self-adaptation capability. MAPE-K reference model can be mapped to the architecture of FTC which is shown in Figure 3-4. The Monitor and Analyze element of the MAPE-K will be used to perform fault detection and diagnosis while the Plan and Execute element will perform controller redesign i.e. reconfiguration. The knowledge is equivalent to the mathematical model the system used in FTC. Note that the sensors here are not just limited to the sensors used for feedback in control loop but can also include performance monitors, disturbance monitors, mission progress monitors, etc. In this research we will use MAPE-K to build SASA localisation framework which will provide robustness, fault tolerance and self-adaptation capability on top of the conventional localisation pipeline.

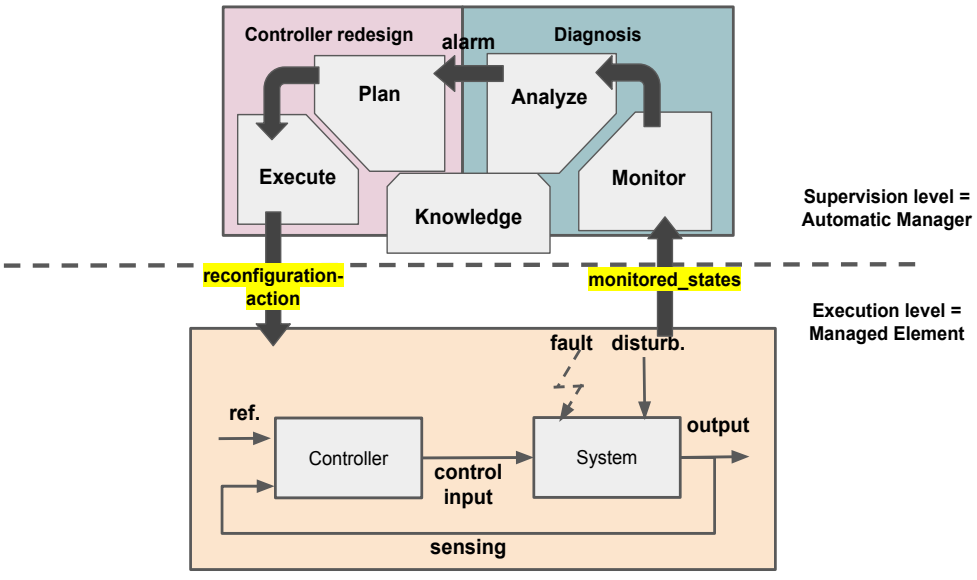

Figure 3-4: MAPE-K Feedback Loop mapped onto architecture of active FTC. The monitor and analyze element of MAPE-K reference model acts as a diagnosis task of active FTC and Plan and Execute element of the MAPE-K acts as controller redesign part of FTC

3-2 Knowledge Representation

For implementing MAPE-K loop, knowledge about the system and its components is required in order to perform run-time reconfiguration. Hence it is necessary to represent knowledge in the form that the system can understand. Knowledge Representation is the field of artificial intelligence that focuses on designing computer representations which can capture information related to the system and its surrounding world. Various methods are available to represent knowledge such as First-order Logic (FOL), Description Logic (DL), Ontology, Behavioural tree, etc. which were reviewed in the literature survey prior to this thesis. Among them, ontologies are the best suitable for our application because they provide shareable and reusable knowledge base using easy human-interpretable format for storing and amalgamating knowledge, readily available editing, consistency checking and reasoning tools [45]. Hence, we will use ontology as a form of knowledge representation for this thesis work.

An ontology is an explicit specification of a conceptualisation [46]. Ontologies represent knowledge as a taxonomy and hierarchy of concepts with their attributes, values and relations. This type of representation will be beneficial for Fault Tolerant System (FTS) as it can model the different components of the system and their dependency effectively. They provide a platform which facilitates the sharing and reuse of knowledge between groups in a computational form. Ontology has been strongly developed as a form of semantic knowledge base[1]. Using ontology for symbolic knowledge representation can provide re-use and sharing of expert knowledge. The ontology can be used for finding fault causes in the system by reasoning over data. It can directly or indirectly provide repair or maintenance strategy [47]. Ontology can also represent the available data in a structured manner on which rules can be applied to perform reasoning, targeting at finding out the root fault causes and inferring some repair or maintenance proposals. Hence, a decision was made to use ontology for knowledge representation. Use of ontology in robotics is increasing. Projects such as Open Robot Ontology (ORO) by Lemaignan et al. [48], Knowledge Processing Framework (KnowRob) by Tenorth et al. [49], Smart and Networking Underwater Robots in Cooperation Meshes (SWARMs) [50], Perception and Manipulation Knowledge (PMK) [51] use ontology for semantic representation. The survey work done in literature survey prior to this thesis work and the study mentioned in [52] and [53] covers the recent use of ontology in robotics and shows in-depth comparison. However, none of the methods mentioned covers the fault and failure of the components and the reconfiguration strategy in robotics domain.

3-2-1 Ontology Design and Development process

Before developing an ontology, it is important to specify the purpose. In our case, the purpose of developing an ontology is to represent a robotic system especially a localisation system in the form of different components and functionalities these components could provide. This kind of knowledge representation in the form of components could be used to find alternatives during faults and failures, using the redundancy present in the system. This ontology should not be specific to a single use case or a single system and should cover the general components

[1] A semantic knowledge, or frames is a knowledge base that represents semantic relations between concepts in a system or a domain. It is a graph consisting of vertices, which represent concepts, and edges, which represent semantic relations between concepts

of the localisation system. The developed ontology will be used to perform reasoning and to search for alternatives and reconfigure the system by using the available redundancy such that the system will be able to provide with the functionality even in case of faults and disturbances.

In principle, there are two different approaches to construct ontologies given in [54]: 1) Bottom-up and 2) Top-Down

- Top-Down: In top-down approach the process of ontology building starts by analysing and studying the relevant information about the domain and then modelling the top level concepts in, which are then subsequently refined in the next step. This approach is usually carried out by domain experts which results in development of a high quality upper ontology [54].
- Bottom-Up: In bottom-up approach the process starts from the most specific concepts and conceptual structure of the domain and then they are generalised for building the ontology in incremental fashion.

Both the approaches have their own pros and cons. Ontologies developed from Top-down approach can be reused across multiple applications and scenario. This approach helps creating a reusable and shareable ontology with a better control on the level of detail [55]. However developing an ontology using top-down approach requires a domain expert and is costly and time consuming [54]. Starting with high level concepts can result in choosing arbitrary high-level categories causing a risk of less stability in the model [55]. Also due to highly dynamic and constantly evolving nature of the knowledge related to a certain domain, the ontology needs to be updated and refined. The advantage of using bottom-up approach is the possibility of discovering knowledge at larger scale and faster pace and detecting and revising human biases is also possible [54]. The ontologies developed by this approach are domain and/or application specific and are not reusable. But it supports the refining and expanding of existing ontologies by adding new knowledge emerging from texts [54]. Due to the high level of detail this approach makes it difficult to spot commonality between related concepts and increases the risk of inconsistencies [54, 55].

The *"middle-out"* as proposed by Uschold and Gruninger [56] was chosen for ontology development. In this approach of ontology development the process is started by specifying the most common concepts, branching out to the most general and then to the most specific ones which allows to focus on most relevant knowledge [57]. The main advantage of using middle-out approach is that the identification of primary concept of the ontology is possible. After defining the relevant concepts, it is possible to specialise or generalise them if necessary. This causes more stable ontology and requires less rework [58]. The reason for choosing this approach is because of availability of the relevant concepts related to the localisation system. Hence we just need to structure those concepts to develop a ontology schema.

3-2-2 Ontology Implementation

To be able to use ontologies in an actual autonomous system, a method for implementing them in the system is required. There are several ontology languages and tools that can be used to implement ontologies. The in-depth review of the tools and languages used for ontology was done in the literature survey which will be summarised here.

Choice of the language and tools used for ontologies

Languages available for building ontologies can be classified into two types: Traditional ontology languages and Ontology markup languages. The first type consists of languages based on first order logic (e.g., KIF), on frames combined with first order logic (e.g., CycL, Ontolingua, OCML and FLogic), and on description logics (e.g. LOOM). The second category is Web standards, which are used to facilitate interchange on the Internet, and ontology languages, which are web standards compatible, are named Web-based ontology languages. Examples of languages of this group are: SHOE, RDF, RDFS, OIL, DAML and OWL. Among these the most popular ontology languages are KIF, OWL, RDF + RDF(S) and DAML+OIL [59]. However OWL, RDF + RDF(S) and DAML+OIL has advantage over KIF as it receives strong support from other communities besides the ontology community, which means that more tools are available for editing, handling, and documenting the ontologies [60]. Other languages are also available such Graql which was considered most promising in the literature survey. Graql is a querying language for GRAKN.AI. GRAKN.AI which will be refereed as Grakn throughout the thesis is an open source distributed, hyper-relational database for managing complex data in the form of a knowledge graph that implements a concept-level schema for cognitive and Artifical Intelligence (AI) systems [61]. Knowledge graph and ontology though are different have same meaning in our use case and hence will be used interchangeably throughout the thesis.

Table 3-1: Comparison between OWL and Graql based on the evaluation from the literature study done prior to this thesis

Criteria	OWL	Graql
Modeling Flexibility	Two individuals can only have binary relations. In order to create N-ary relationship each OWL property needs to be separately modeled and then combine together into N-ary relation patterns using auxiliary class names and property restrictions.	Graql's ontology language contains higher level constructs that can define N-ary relationships and objects
Querying Capability	In OWL querying is performed in a set of triples	In Graql multiple information can be extracted in a single query which makes the querying process easy.
Third party dependability	OWL require components from many different systems, such as ontology editors (e.g. Protege), storage (RDF triple stores), query engines (e.g. SPARQL), reasoners (e.g. Pellet, other OWL reasoners).	Graql has a fully integrated knowledge-base environment, with storage, querying, validation, reasoning and visualization.
Ease of use	OWL need lot of different components which need to be learn for building, querying and reasoning with ontology	Graql can be used to perform both ontology building an reasoning which has a simple and intuitive syntax

The languages mentioned above need tools for ontology editing and reasoning. Protégé is an

open source tool used for ontology editing and has an OWL API which is a Java API for developing and editing OWL ontologies. Web Ontology Language (OWL) constructs should be used for developing ontology in Protégé which results in very restrictive schema thus limiting the flexibility of the tool making it hard to add new concept. Protégé has interfaces to reasoners such as Fact++ and Pellet. The tool also has plugins for visualisation, such as OWLViz and Ontograf. Querying the ontology is done using SPARQL Protocol And RDF Query Language (SPARQL) and reasoning Semantic Web Rule Language (SWRL) using rules. Large number of options, multiple plugins and different API's makes it confusing to use the tool.

Graql however uses Grakn for ontology representation and reasoning. This knowledge representation system has its inbuilt automated reasoning engine that performs automated deductive reasoning during query time. The ontology can be modelled by programming using Graql language or graphically using Grakn Workbase[2]. Grakn provides Python, JAVA and Node.js APIs for querying and updating the data in the ontology. Grakn provides a complete package for representing concept-level model, a type system, a query language, a reasoning engine and schema verification. To do the same with OWL requires multiple standards and their respective implementations, each with their own inherent complexities. In particular, OWL is extremely feature rich, which has resulted in a high degree of complexity making it unsuitable for most software applications [62]. However Grakn also has few limitations. Grakn does make use of a schema, but does not use semantic standards, which makes it incompatible with the other ontologies which are developed using OWL. Also, Grakn is quite young tool and hence has a smaller user community and thus less support from fellow users. The advantages of Grakn were more significant for us than its limitations. Grakn also provides a suitable balance between complexity and expressivity for knowledge representation and an automated reasoning and integrated package which was a decisive factor in choosing Grakn over OWL.

3-3 Summary

In this chapter we discussed about the background topics which helped us to develop SASA localisation framework. Instead of looking in the direction of conventional fault tolerant control system, we decided to use a slightly different approach in the direction of SAS using MAPE-K reference model. We made these decisions because it is difficult to model different environmental properties mathematically and fault tolerant controllers don't account for qualitative uncertainties in the system. MAPE-K reference model can provide self adaptation capability which can be used to develop intrinsic fault tolerant system. As against the classical fault tolerant controllers, a knowledge based framework can also be reused for other similar types of systems, provided the knowledge is shareable and reusable. Hence we decided to make use of ontology for knowledge modelling. Ontologies have various advantages over other forms of knowledge representations among which reusability and provision for reasoning were more important for us. Using heuristic relationships and performing deduction using the knowledge and facts expressed as rules and parameters, the system can mimic the intelligence of a human expert. Also, reasoning can provide explanation to justify the decisions made which is an important property for developing reliable systems. In the later part of the chapter, we

[2]Grakn Workbase is a tool which provides an interface through which we can read from a Grakn knowledge graph or model ontology graphically

saw different tools for ontology modelling among which we chose Grakn for implementation due to its modelling flexibility, querying capability and ease of use. In the next chapter we will discuss about the development process of the SASA localisation framework.

Chapter 4

Development process of Situation-Aware Self-Adaptive Localisation System

In this chapter we discuss the development process of the Situation-Aware Self-Adaptive (SASA) localisation framework. To develop the framework we have followed Integrated Systems Engineering and Pipelines of Processes in Object-Oriented Architectures (ISE&PPOOA) process from [63]. This chapter discusses about the operational context, different operational scenarios, the desired behaviour of SASA framework and the framework requirements from a Systems Engineering perspective. This chapter is important to get a clear idea about the scope and the functionality of SASA localisation framework. This chapter will be concluded with the skeleton of the functional architecture of SASA framework which will be completed in Chapter 5

4-1 Operational Context, Operational Scenarios and desired behaviour

The first step of the system design process according to ISE&PPOOA process is the identification of the operational scenarios. The goal of this process is to identify the possible operational context of the system and describe its operational scenarios for the different modes of operation. Note that this step is just to get a clear idea about how the system should operate in some limited scenarios. However the adaptation should not only be limited to these scenarios but also to other types of situations which could occur in real life which are not discussed here or not known to the framework in advance.

4-1-1 Identify Operational Context

The first step of the system development is to identify the operational context. In this step we will identify the context and the environment in which our system will be operating.

Our goal is to develop a situation aware localisation system which at any moment provides a reasonable estimation of the robot position available for display in a map of the retail store, so a human supervisor or the navigation planner could make decisions concerning robot operation. The robot will operate in a retail store environment. We are using a retail environment because this work is part of a bigger project under AIR Lab Delft which is a TU Delft-Ahold Delhaize collaboration focused on developing state-of-the-art innovations in the retail industry. The environment is created using Gazebo simulator [64] [1] and shown in Figure 4-1. The environment has multiple shelves with different types of products. The environment has nonuniform illumination and featureless regions to replicate the actual store which is created using Gazebo plugins and can be changed using input commands. The robot will use the SASA localisation framework and provide its accurate location while moving in the store which can be visualised in Rviz which is a 3D visualisation tool for Robot Operating System (ROS)[2]. The robot will be operated manually by user commands for its motion in the environment. The teleoperation can be replaced with navigation planner in future to perform the autonomous mission. The localisation system might fail due to environmental disturbance such as uncertain lighting conditions, low visual features, smoke rendering on sensor which will affect stereo vision, and long, narrow aisle, cross-talk between active sensors, transparent and reflective surfaces which will affect laser based localisation, and various other reasons which are mentioned in section 2-2-1. However, the localisation system might also fail due internal failure such as algorithmic failure (i.e. pose estimator divergence from actual value), transient failure caused due to driver malfunction, permanent failure caused due to physical damage to the sensor, and system freezing due to resources exhaustion. Figure 4-2 establish the context of our SASA localisation framework by using System Modelling Language (SysML) block definition diagram in which the relations between the main elements are defined. SASA framework is not just limited to this context and should also perform in other types of stores. This context is taken just to show a proof of working of SASA localisation framework.

4-1-2 Operational Scenarios and desired behaviour

After identifying the operational context and the environment, the next step is to identify scenarios which the localisation system may have to face. The scenarios identified are with respect to environmental uncertainties, internal faults and unavailability of resources. The scenarios are given below and Table 4-1 describes the desired behaviour of the localisation system in detail in each scenarios.

- S1: Normal Conditions. The environmental conditions are proper for estimating robot's state, all components are working properly without any faults in any part of the system. Everything is working as desired.

[1] Gazebo is an open-source 3D robotics simulator

[2] Robot Operating System is a collection of software frameworks for robot software development.

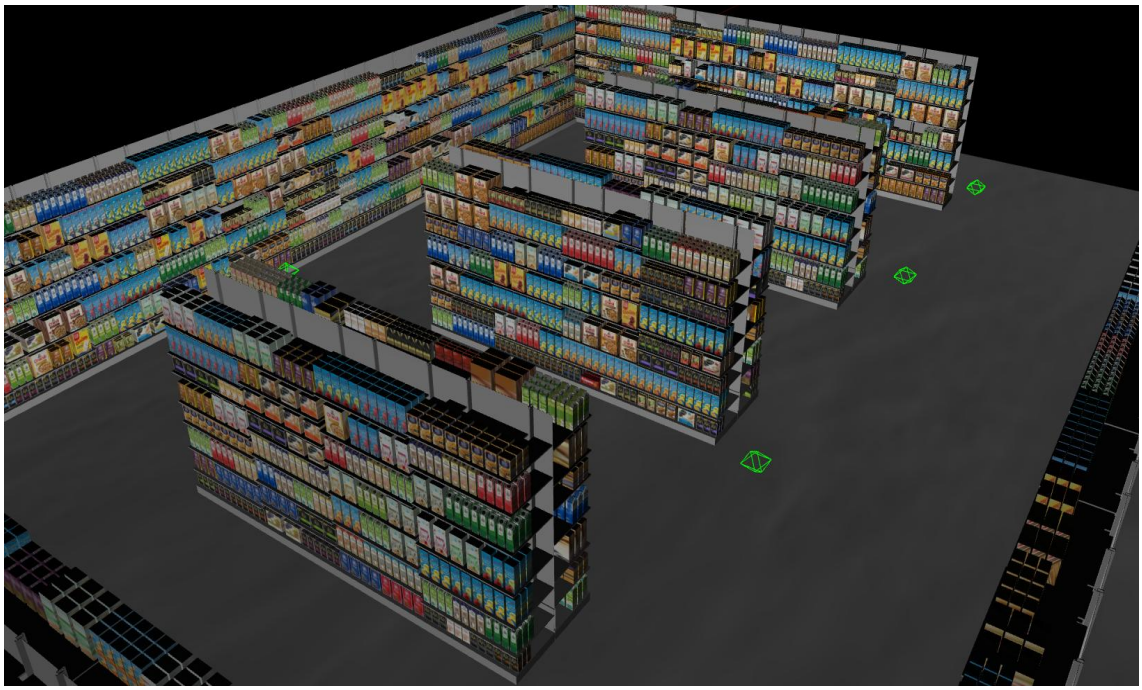

Figure 4-1: Retail store simulation environment

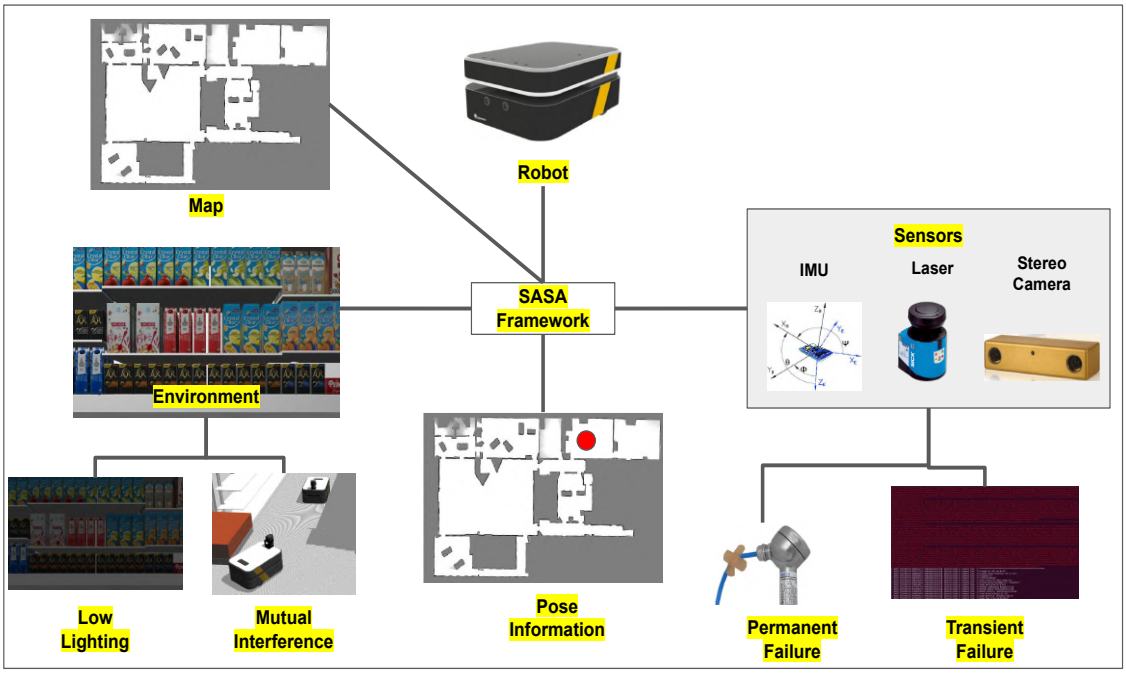

Figure 4-2: Context diagram for SASA localisation framework which can face unexpected situations such as environment disturbances and internal failures. Even in the presence of such type of disturbances and failures, the SASA localisation framework should provide the pose information. Here we are assuming that we have a prior map of the environment.

- S2: Unacceptable environmental disturbance: The environment has poor illumination conditions, or low visual features or obstructed Field of View (FOV) which highly affects vision based methods and causes erroneous pose estimates. The errors are not caused due to internal components of the robot but due to the environmental disturbance. The environmental uncertainties described in Section 2-2-1 fall under this category. Such types of situation should be handled by SASA framework.
- S3: Transient failure. The environment conditions are proper, however during operation there are transient failures such as algorithmic failure, data synchronisation or processing problem, hardware driver failure which can give rise to errors in conventional localisation system. These type of failures are not permanent and can be solved just by re-initialising the the faulty component which should be done by SASA framework.
- S4: Permanent hardware failure. These types of failures are permanent in the system and caused due to some damage to the hardware or software component. The SASA framework should recover from such type of failures by using redundancy available within the system.
- S5: Temporary unavailability of the resources: There can be situations when the resources which are otherwise available are not available for certain period of time. Such as computational power or battery. During such scenarios the localisation system might freeze due to resources exhaustion.

4-2 Operational Needs and System Requirements

We have identified different operational scenarios in the previous section and how the system should behave in each scenario. In this section, we will identify the operational needs which will cover all the operational scenarios and convert those needed into system requirements, capabilities and functions.

4-2-1 Identify Operational Needs

The process of identifying operational need is important as it will help us define the objective of our system and which will be converted into the system requirements and system capabilities. At this stage, we don't need a rigorous list of requirements as they will be converted into precise system requirements later. The identified operational needs are as follows:

- **ON1:** The developed system shall be able to perform the localisation as that of a conventional localisation system which we have discussed in chapter 2.
- **ON2:** The system shall be able to monitor the internal states and the external environment in order to detect and identify run-time errors and failures caused in the internal components and also the disturbances caused due to environmental changes.
- **ON3:** The system shall be able to identify the redundancies available in the system so that those redundancies can be used to overcome the situation of internal failure or environmental disturbances.

Table 4-1: Description of desired behaviour in different Operational Scenarios

Scenario	**S1: Normal Conditions**
Preconditions	The localisation system is initialised and running
Periodic event	The localisation system works as desired with reasonable performance[a] without any internal error or environmental disturbances.
Description	The localisation system reads the sensor data and extract necessary information to preform state estimation to provide robot's pose.
Scenario	**S2: Unacceptable Environmental disturbances**
Preconditions	The localisation system is initialised and running
Triggering event	Unexpected situation affects the ability of pose estimator.
Description	During operation the an unexpected environmental disturbance occur which affects the quality[b] of the sensor data or the performance[a] of localisation. For example in a featureless or a dark environment, the sensor information is insufficient to perform vision based pose estimation. The framework should be able to detect low performance and the cause of low performance and should try to overcome this situation using the redundancy available in the system.
Basic Flow	1. Detect the inability/low performance of localisation and the cause for low performance *(which is environmental disturbances in this case)* 2. Search for alternatives using available redundancy 3. Deploy the alternatives by performing run-time adaptation which can provide reasonable performance
Post-conditions	The system resumes its localisation even with disturbances using available redundancy
Scenario	**S3:Transient failure**
Preconditions	The localisation system is up and running with the default configuration.
Triggering event	Any internal component fails either due to algorithmic failure or due to abrupt stop of any process.
Description	During operation the estimator fails and diverges from actual values causing high variance in the estimates or an component stops providing data to the other components. The framework should detect this kind of abnormality and actuate the recovery from it by re-initialising the failed component.
Basic Flow	1. Detect errors in the components and the reason behind them 2. Update the state of components 3. Re-start the faulty component
Post-conditions	The localisation system resumes its operation as previously with same the performance.

a, b: The metric for performance and quality will be discussed later in Chapter 5

- **ON4:** The system shall be able to change the configuration in run-time without affecting the other processes of the system. For example: in case of poor illumination the system shall be able to stop the vision based localisation and start the laser based localisation without affecting the other process in the system.

Table 4-2: Description of the Operational Scenarios (Continued...)

Scenario	**S3:Permanent failure**
Preconditions	The localisation system is not running due to the unavailability of data or a component.
Triggering event	A component ceases to be available
Description	During operation a sensor used for localisation is damaged physically or any component such as an estimation algorithm stops providing the data to the the localisation system. Restarting as in the previous scenario doesn't solve the problem. The SASA framework should detects this abnormality and actuate to recover from it by searching for an alternative from the available redundancy. If any redundancy is available, the system should perform run-time adaptation and use the alternate method to achieve same functionality
Basic Flow	1. Detect fault in the components of the localisation system 2. Updates its status of the component 3. Re-start the faulty component: This won't work as the component is not available anymore 4. Search for alternative exploiting the available redundancy 5. Use that redundancy to perform estimation
Postconditions	The system resumes its operation with same or acceptable performance degradation of the localisation system with an alternate configuration
Scenario	**S5: Temporary Unavailability of resources**
Preconditions	The localisation system is initialised and running
Triggering event	High computational load due to other computation demanding tasks reduces the localisation performance
Description	During operation other task which need high computational load starts in parallel. As a result the performance of the localisation system degrades causing it to lag or update at lower rate. This should be detected by the SASA framework and the system should shift to a configuration which has lower computational requirements.
Basic Flow	1. Detect high computational load *(Note that here we are making an assumption that a high computational load will also results in performance degradation of a localisation system)* 2. Search for configuration which has lower computation requirements 3. Makes changes to the current configuration according to alternate configuration
Post-conditions	The system continues its operation with some modifications in the current configuration which might results in trade-off such as low accuracy due to lower computation. For example: for lower computation the number of particles in particle filter based estimator can be reduced.

4-2-2 Specify system capabilities and NFR

After describing the operational scenarios of our localisation system, the next step is to convert the operational needs into a set of capabilities, functions, and quality attributes that

the SASA framework needs to posses. The goal of this step is to transform scenarios and needs into a set of system capabilities and high-level system requirements. The identified high level capabilities of our localisation system is given in Table 4-3. The first capability is required for performing localisation while the other two are required for the system to be fault tolerant and reliable.

Table 4-3: Identified high level capabilities of our system

C1: Self-localisation	The self-localisation capability will provide the robot with the ability to estimate its pose within the operational environment. This is the primary capability of our system.
C2: Situational awareness	This capability is required by the robot to understand what is going around in the environment and within the internal system. This capability will provide the robot with the ability to detect faults and failures within the system and the detect disturbances which will affect the performance of the C1.
C3: Self adaptation	The capability C2 will just allow the robot to analyse and understand the faults in the system and the environmental conditions but it won't provide the ability to act on it. Hence self adaptation capability is required which will provide robot with the ability to react to the situations such that the robot will be able to preform operation even in the presence of disturbances and faults.

Non-Functional requirements (NFR) of a system specify the criteria that can be used to judge the operation of a system, rather than specific behaviours. In simple terms, the functional requirements define what the system is supposed to do or how it is supposed to behave and the NFR specifies how the system is suppose to be. We have seen in the literature survey and even in chapter 2 that there are various methods available for mobile robot localisation and each has their own pros and cons. In this research work we are not really focusing on developing the localisation system by our self but we want to develop a framework which can be used on top of any localisation system and provide self adaptation. Hence for us, the *reusability* and *modularity* of the developed framework is of great importance. The framework should be such that it can be applied to any of the localisation methods currently available or which might be developed in future. Also, it should possible to add, remove and modify components from the system with minimum efforts and should hence be modular.

Apart from *reusability* and *modularity*, we are dealing with adaptation and hence the system should have *adaptability*, however we prefer to consider adaptation as part of behaviour because adaptation as a functionality can be allocated. But the system should have other quality attributes related to adaptation performance such as low latency and adaptation response time [3]. Response time is the time taken by the system to overcome a situation after the occurrence of that situation. Latency is the time taken by the SASA framework to detect any unexpected event.

[3]The adaptation latency and adaptation response time are discussed in chapter 6 while testing the framework

4-3 Functional Architecture

Once the localisation system has been characterised from a operational viewpoint through the capabilities and quality attributes, we have to design the functional architecture that addresses them. This step is the most important one in the ISE&PPOOA process. A Functional architecture is an architectural model that identifies system function and the interactions between different components in the system [65]. It defines how the functions will operate together to perform a particular objective or an operation. Generally, more than one architecture can satisfy the system requirements. However, each architecture will have different operational cost, performance, risk, etc. The functional architecture represents the problem space at the highest level of hierarchy independent of the technical solution used. Thus a functional architecture is a permanent architecture in contrast to a physical architecture which is more dependent on the technical insights of a solution. The functional architecture of a system is reusable for similar systems of the same product family or systems with similar missions or in the same application domain, saving development time and money [63].

The path we have followed to develop the functional architecture of our system is slightly different from ISE&PPOOA process. The reason for this is we have used a ROS based architecture which has developed localisation packages. These packages are already defined in a certain functional architecture which we have adapted to our requirements. This means that instead of beginning by defining the functional architecture of the system in design, and then implementing it, we had performed functional analysis of the ROS localisation packages, and used those packages in our SASA localisation framework. Note that the functional analysis done in this chapter is only with respect to *capability C1* i.e. self-localisation and the other capabilities will be presented in the chapter 5

4-3-1 Functional Analysis of Localisation System

The localisation system available in ROS can be can be grouped into two different subsystems based on the functionality i.e. perception and state estimation. Figure 4-3 depicts the functional breakdown of the localisation layer of SASA framework into subsystems and the subsystem further into components. It consists of:

- Perception subsystem: It consists of sensors mounted on the robot which are a part of the robotic system used for sensing robot's internal states and environment states relevant for localisation. It is also responsible for processing the sensor data before it is used by other sub-systems.
- State-estimation subsystem: It is responsible for estimating the state of the robot i.e. the current pose of the robot in the environment. It is composed of local state estimation and global state estimation. For local state estimation, the robot estimates the relative state from the start of the motion and for global state estimation it uses a map of the environment stored in the database to estimate the global pose.
- MOTION subsystem: It contains the mobile base and receives the motion commands in our case given manually as input from user using keyboard/joystick. This teleoperation component can however be replaced by an autonomous navigation planner

which will give the motion commands to the mobile robot. *Though it is not a part of localisation system we have mentioned it to show the complete flow of the motion-perception-estimation cycle.*

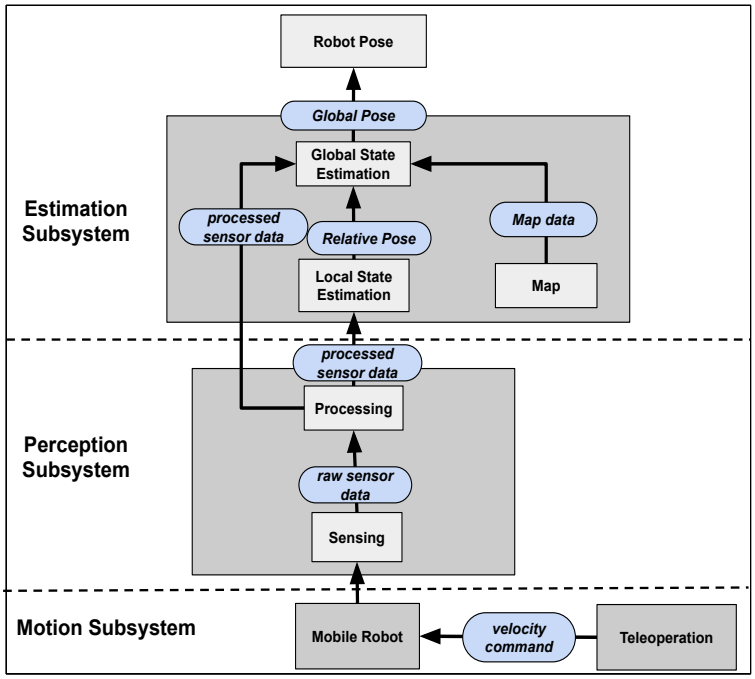

Figure 4-3: The functional decomposition of the self-localisation layer of our framework.
In this diagram the motion subsystem is not a part of the localisation system but it shown here because it controls the robot motion which is detected by the localisation system due to change in position.

Note that the functional decomposition shown in Figure 4-3 only shows the conventional localisation with capability C1. The other capabilities C2: Situational awareness and C3: Self-adaptation will be added later to the functional architecture. The decomposition shows the components involved in the localisation and their functionality. For example, the sensing component will preform sensing function, local state estimation will perform local estimation function and so on.

4-3-2 Design Alternatives

The functional architecture of the localisation system is unique. There are multiple ways to design a physical system which can provide same functionality. In this step we have identified different design alternatives which can provide the same functionality. Knowledge about design alternatives is important to identify the available redundancies. We will identify the alternatives mainly based on two approaches used for localisation i.e. vision based localisation and laser based localisation. Note that there can be alternatives apart from one shown in Figure 4-4 and Figure 4-5 where the method can combine both the sensors i.e. laser and vision. But such methods do have a primary sensor and they tend to fail if any disturbance

[3]The arrows show the direction of the data flow and blue boxes shows what kind of information is exchanged

or fault affects the data quality of the primary sensor. For example, some versions of Real-Time Appearance-Based Mapping (RTAB map) though uses laser scan as an input to increase the accuracy of localisation fails if vision data is unavailable [6].

Figure 4-4 shows the vision based localisation system. The perception layer consists of a stereo camera component which will capture the stereo images and provide it to other components, and an IMU sensor which provides inertial measurements of the mobile robot. The Stereo odometry estimation component will take the stereo images produced by the sensor to perform local pose estimation. The sensor fusion component will take the input from the local state estimation component and IMU sensor and produce more reliable pose estimation. In the end, the vision-based localisation component will take the local pose estimates from the sensor fusion component, the sensor data from stereo camera and map data from the map server component to provide the global pose estimation of the robot.

Figure 4-5 shows alternative configuration using laser scanner. On the perception level it consists of a laser scanner and a IMU sensor. The laser scanner will produce a 2D range scan of the surrounding which can be used by other components. The data from the laser scanner is used by the laser odometry component to estimate local pose. The estimated pose is then used by the sensor fusion component along with the IMU sensor data to reduce uncertainty in pose estimates. Finally, the laser based localisation component will take input from the sensor fusion component, map server and the laser scanner sensor to provide global pose estimates.

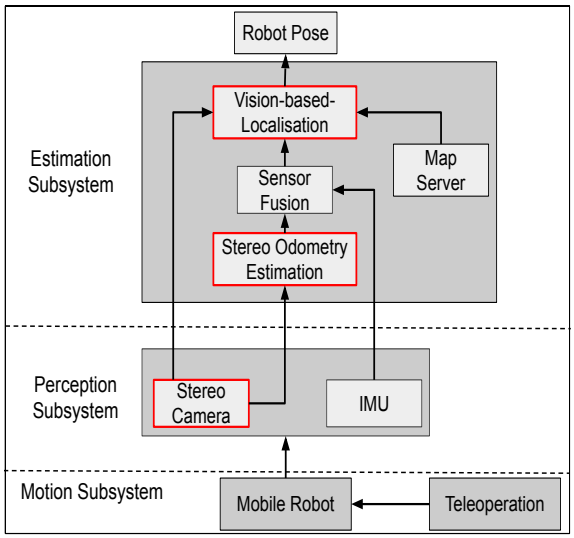

Figure 4-4: Vision-based Configuration

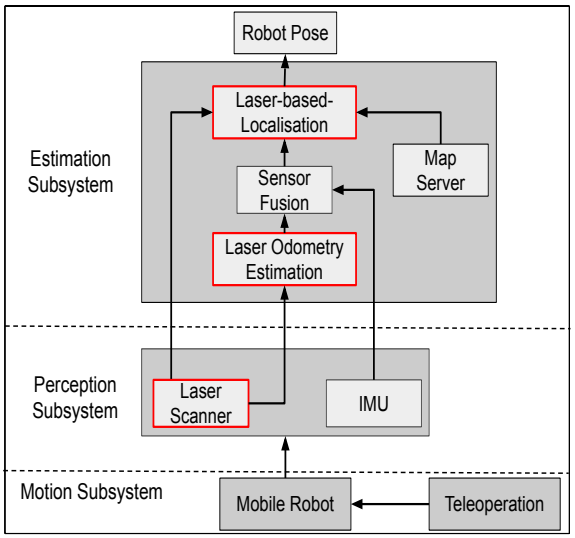

Figure 4-5: Laser-based Configuration

4-4 Addition of automatic manager

So far, we have a functional design of a localisation system which can perform reasonable pose estimation of the robot. But as shown in the Figure 2-3 any fault or uncertainty can affect the localisation system and there is no method by which the system will be able to overcome the faulty situation. In order to create a situation aware self-adaptive localisation system we will use Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) as a

reference feedback loop. The system developed till now will act as a managed element of the MAPE-K loop and now we have to develop automatic supervisor which consists of elements such as monitor, analyze, plan, execute and knowledge which will work in combination to analyze the situation and provide adaptation alternatives according to the situation. After completion, the MAPE-K loop for SASA system will be similar to the one shown in Figure 4-6. The monitor component consists of the ROS nodes which will continuously monitor the environmental and internal status of the robot. The analyze component will analyze the information extracted by the monitor component to detect if any fault is present in the system. The plan component will only be used when any fault or uncertainties is detected in the system. It will use reasoning to overcome the fault and provide the same functionality. The execute component will acts as an effector for switching the ROS nodes in the run-time. All of the components are explained in the detail in the next Chapter 5. Note that the missing elements will be added to Figure 4-6 in the next chapter where the complete functional architecture of the SASA will be shown.

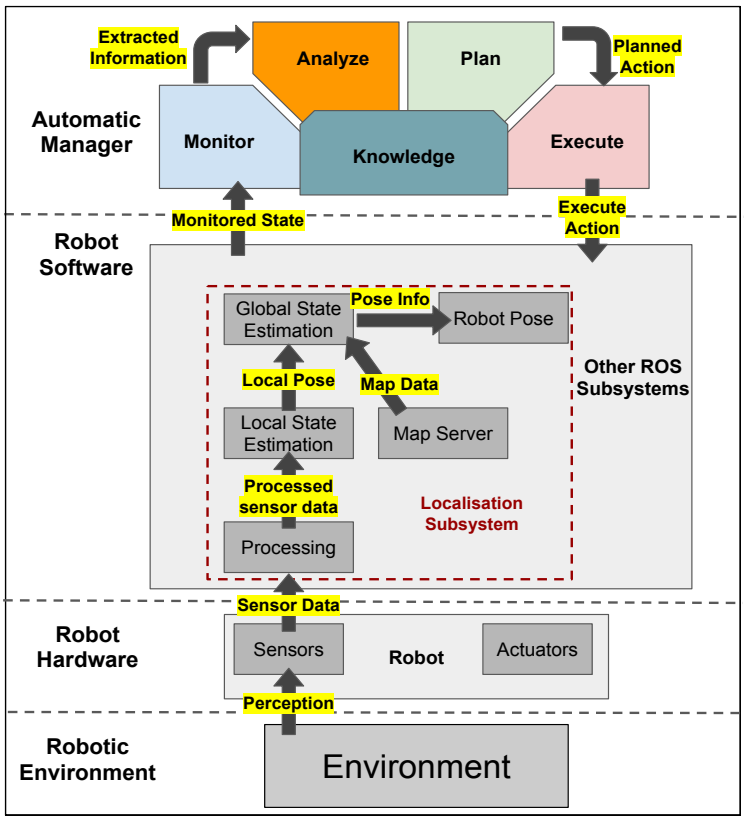

Figure 4-6: Functional Architecture of the SASA localisation framework with with information about the managed system layer

4-5 Summary

In this chapter we discussed the development process of the SASA localisation framework. We started with an example of operation context, and the operational scenarios. While analysing

the operational scenarios we understood that the conventional localisation framework can only perform in normal situation but in order to perform in presence of environmental uncertainties and internal failures, the system needed additional capabilities such as situational awareness and self adaptation. The operational needs helped us identify the functional and Non-Functional requirements (NFR) of the SASA localisation framework such as reusability, modularity, adaptation performance. We started designing the functional architecture of the SASA framework where we identified different design alternatives for self-localisation capability. In the later part of the chapter we presented the functional architecture of the localisation layer of the SASA localisation framework. However this functional architecture is still incomplete and still needs elements which will provide the capability to perform during unexpected situations. In the next chapter, we will discuss the automatic manager which will provide the system with capabilities such as self adaptation and situational awareness using which the system will be able to handle unexpected situations.

Chapter 5

Development of MAPE-K elements

The functional architecture defined in Figure 4-6 does not have any information about the automatic manager layer of Situation-Aware Self-Adaptive (SASA) framework. In this chapter we will present the adaptation layer and discuss in detail different elements of the adaptation layer i.e. Monitor, Analyze, Plan, Execute and Knowledge. This layer is responsible to provide the system with situational awareness and self adaptation capability. This chapter will be concluded with the complete functional architecture of the SASA localisation framework.

5-1 Knowledge

The automatic supervisor needs knowledge of the system to perform adaptation. These knowledge can be represented in different form such as First-order Logic (FOL), Description logics (DL), Ontologies, Behaviour trees, etc [45]. Among them ontology have various advantages over others. 1) It is easily extensible: It is possible to extend ontology by adding new class and relationships to the already available ontology. So no need to build the complete knowledge model from scratch. 2) It is reusable: The same knowledge model can be applied to different applications. 3) It provides high granularity: The amount of detail captured using ontology can be varied. It can capture knowledge about the general concepts in a system to the most specific ones. 4) Allows reasoning: Ontology models support reasoning. It is possible to infer logical consequences from a set of asserted facts or data.

The overview of the primary concepts of our knowledge schema is shown in Figure 5-1. A schema is a schematic representation or a blueprint of concepts and relationships within domain. Note that these concepts are not specific to the localisation but are quite generalised because we have used 'middle-out approach' for ontology development where we start from these generalised concepts. We will start with these concepts and move towards more detailed ones in the upcoming section.

5-1-1 General Schema

The core concepts of our schema are shown in Figure 5-1. The *purple-box* represents the concepts which are the main entities of our domain, *green-diamond* represents the relationships between the concepts and the *yellow-highlighted* text shows the roles the concept plays in that relationship. A robot is composed of different components. We start with `robot component` where a robot component can be anything such as software, hardware or a communication interface. Each component can provide some `functionality`, where a functionality is the purpose which the component can serve when used. Each component can serve multiple functions such as a camera can be used for localisation as well as obstacle identification. So the functionality which a component can provide, depends on the component setting i.e. how it is configured. This is modelled using `functionality-relation` relationship in the schema. The `processing-relation` relates: a `Robot Component` concept where it plays a role of `component`, the `Process input` and `Process output` concepts which plays the role of `has-input` and `has-output` respectively. A `processing-relation` can be considered as a black box which will process some inputs and provide some outputs. `Process input` and `Process output` concepts are used to model inputs and outputs of the process.

A `processing-relation` also has `Property` which acts as a constraint. The idea behind modelling this relation is that every process has some constraints and the process would only be successful if the constraints are satisfied. To represent it in control terminology, a linear controller would only be effective near the region of operation. In our case, the `Property` could be anything from an environmental property, the system's internal property such as battery, computational load to the performance of estimation algorithms. The `property` will be used to model a state of an environment, internal system or an estimator which can affect the way in which a component can function. For example: Low illumination in the environment or a large error in pose estimation will affect the localisation functionality. Finally the `status` will be used to representation the status of the process. If all the constraints are satisfied then the process will have a health status and if any fault occurs in the system then the status will be unhealthy.

The schema explained above consists of general components which can be used to represent different types of processes in a robotic system. But we want to focus on the concepts which are specific to localisation which we have mentioned in the last chapter. These include concepts like local state estimation, perception, etc. To do this we will create sub types of the primary concepts. The sub-types will act as a child concept and will inherit all the properties of the parent concept.

The process of creating sub-types will drive the ontology towards specialisation. A robot component can be anything from a hardware sensor to a software component or a communication component. Hence we have created them as sub-types of the robot component. As we are using Robot Operating System (ROS) [66] based architecture, the `ROS-Message` concept will be a sub-type of `Communication` concept and `ROS-nodes` will be sub-type of `Software` concept. A robot component could also be a `Hardware` which can then be more specific to `Power Supply`, `Sensor`, `Computational Resources`. Figure 5-2, Figure 5-3 and Figure 5-4 shows the sub-types of concepts in a hierarchical manner. Note that the mentioned sub-types are just an example and does not cover the exhaustive list of all the sub-types in our system.

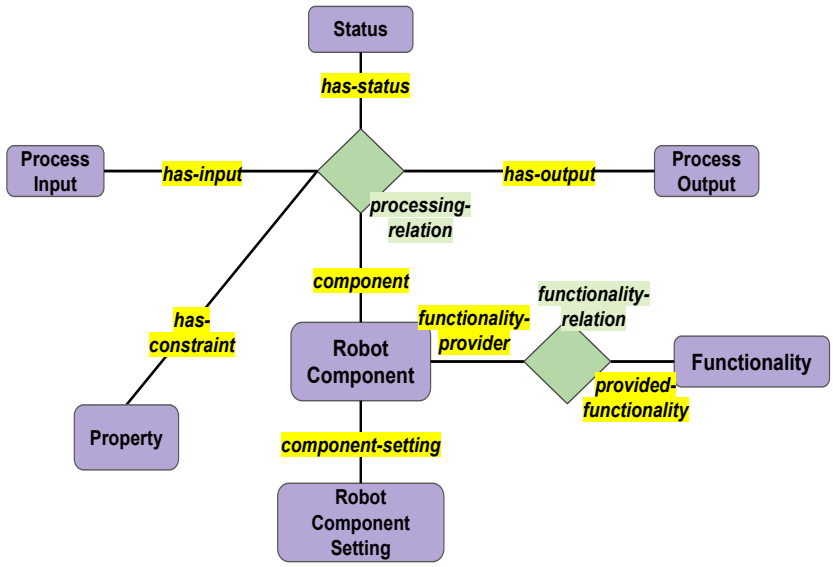

Figure 5-1: Core concepts of the schema

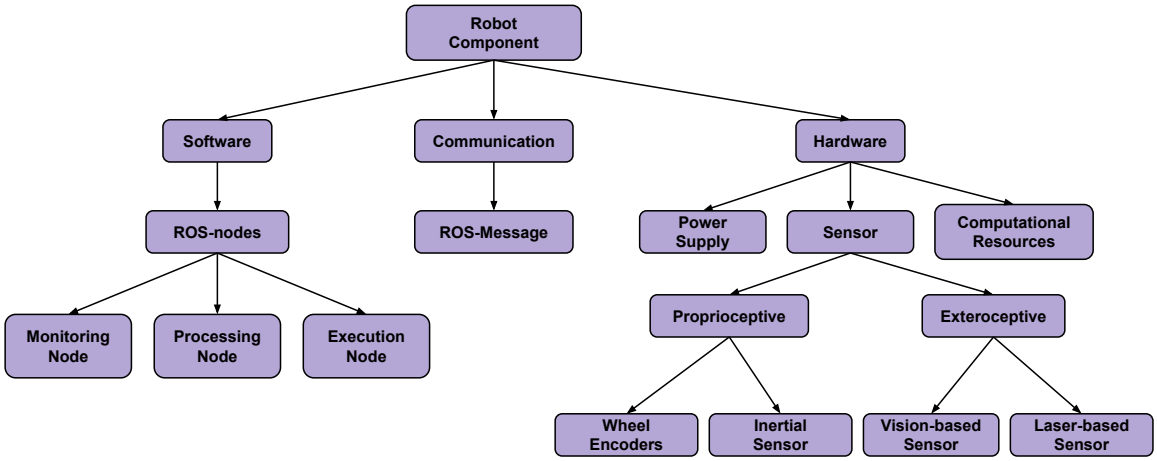

Figure 5-2: Sub-types of Component Concept

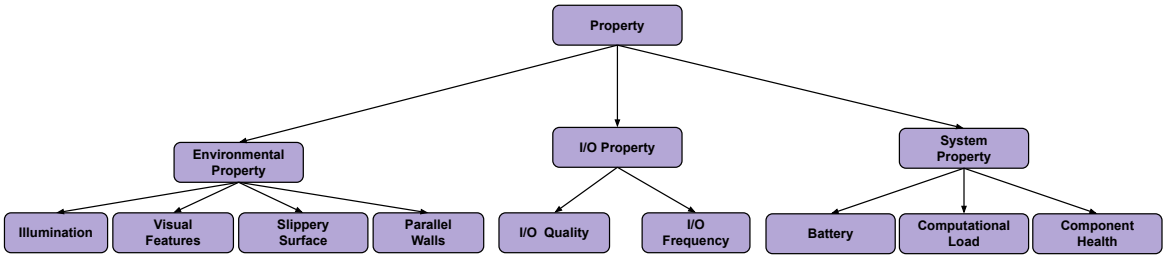

Figure 5-3: Sub-types of Property Concept

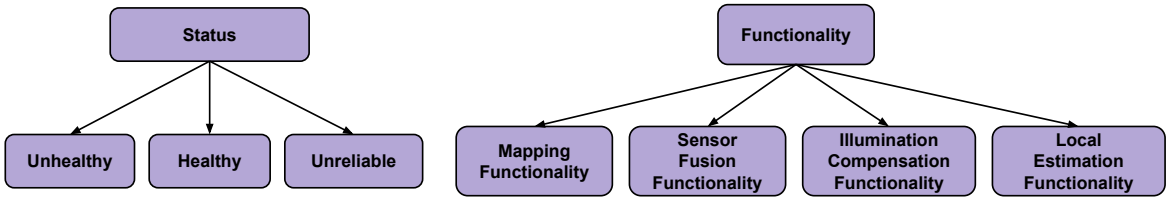

Figure 5-4: Sub-types of Functionality and Status Concept

5-1-2 Schema for Localisation use case

In the previous section we have discussed the general concepts in the schema and how we can create more specific sub-types according to our requirement. In this section we will see the final schema for SASA localisation framework.

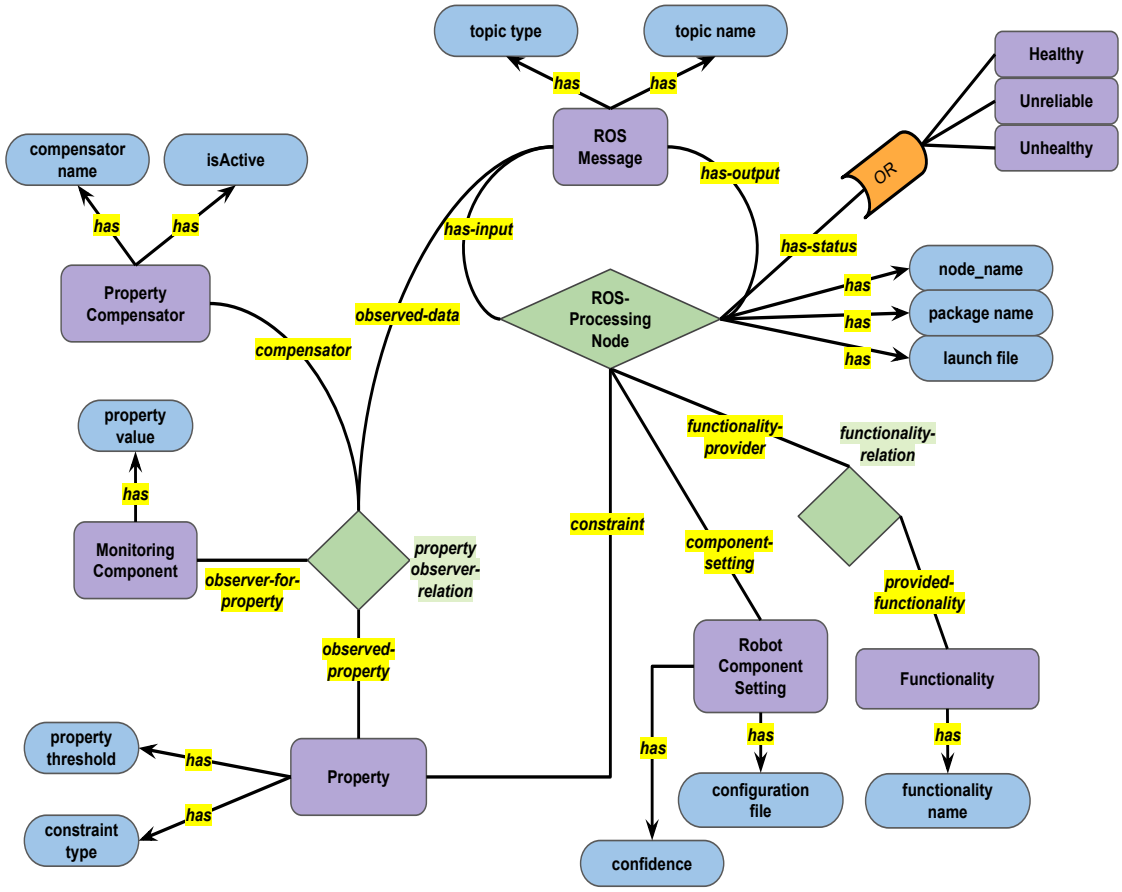

Figure 5-5: Developed Schema of the SASA localisation framework

Figure 5-5 shows the complete schema of our SASA localisation framework. This acts as a primary skeleton for conceptualisation of elements related to the localisation. Note that, as we are using ROS based architecture, the inputs and outputs of the process will be in the form of ROS messages and the processing will happen in a ROS node. Let us start from

`ROS-Processing Node`. The node will have some input in the form of a `ROS-Message` and those messages will have some name and type. This is modelled using attributes which are shown in *blue boxes*. Each message is differentiated using a name and the node will only accept an input of certain datatype, hence we have modelled them in our schema. Now same as the general schema, the `Property` will act as a constraint. But in this schema we have give some attributes to the property such as `property threshold` and `constraint type`. The threshold will be used in future to identify the violation of constraints. The `constraint-type` attribute will hold the value of type of property. For example: the component can still provide the functionality if some constraints are violated and hence such properties will act as a soft constraint, and some constraints if violated will stop the component's ability to provide the functionality. Hence they will act as hard constraints. Note that all those sub-types which we had mentioned in the previous section i.e. environment property, I/O property, etc will all have these attributes. Now we need a monitor which will observe those properties. These monitors are required to update the current states of the properties in the knowledge and these updated values will be compared with the threshold values to check for constraint violation. There is a concept in the schema called `Property-compensator`. These compensator are such that they can alter the effects of the property. To given an example: consider illumination property. If the robot is mounted with flashlights it can use that flashlights when the illumination is low. The attribute `isActive` will be used later in the reasoning section. It is to update the status of the compensator when it is deployed.

Now the `ROS-processing-node` has attributes such as `node-name`, `launch-file` and `package-name`. These attributes will be useful to start or stop a process during run-time in ROS. The `ROS-processing-node` has a relationship with `Functionality` and `Component-setting`. So as mentioned before, each component can provide functionalities with different performance. For example: The performance of a particle filter used for state estimation depends on the amount of particle used for estimation. Higher the amount of particle higher is the accuracy and vice versa. Hence the concept of component setting will be used to model the parameters for that particular component. Note that `component setting` has an attribute `confidence`. This user defined attribute will be used to encode the knowledge of how well the component can accomplish the functionality. The `ROS-processing-node` also has some status. If all the constraints are satisfied then the node will have the status of healthy. While if only the soft constraints are violated then the status of the node will be unreliable and if the hard constraint is violated then the status will be `Unhealthy`.

Apart from the schema in Figure 5-5 there are few more concepts which are shown in Figure 5-6. These concepts are used to establish a hierarchical network within the knowledge by inference. Note that the roles which are shown by red lines are not defined by user and will be inferred using reasoning which will be explained in the coming sections of this chapter.

5-1-3 Data instantiation of schema concepts

The schema described in Figure 5-5 will just provide a structured model of the concepts related to localisation but does not have any significance unless it is grounded. Grounding is the process of making physical quantities correspond to their conceptual counterparts, where we actually map the physical data in our system to the knowledge. To perform grounding, we need to instate the concepts according to components available in our system. To explain

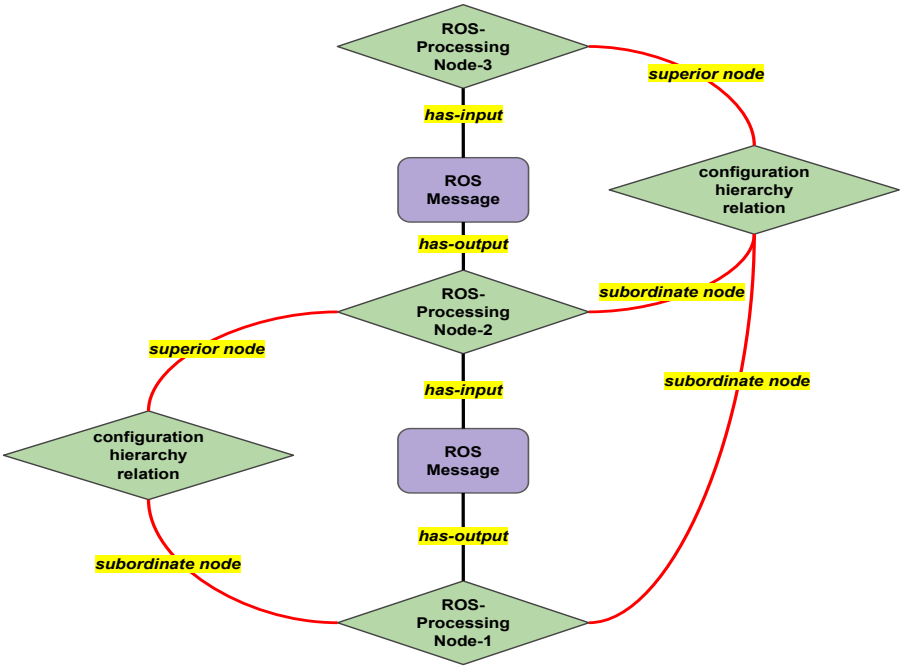

Figure 5-6: Configuration Hierarchy relation between processing nodes

in a form of object-oriented methodology the schema are equivalent to the classes and the instances are equivalent to the objects. Instances are needed to perform reasoning which is needed for fault analysis and system reconfiguration. We will try to explain the process of instantiation with an example.

Consider we have a ROS node in our system which will take the stereo images from the camera and produces the local pose estimates of the robot. The schema grounding of for this particular node is shown in Figure 5-7. The node will produce the output in form of ROS message. The functionality of this node is to provide *local pose estimation*. The functionality of the node will be affected due to the poor illumination and low visual features. Note that we have only shown the illumination property in the figure. The functionality of the node will also be affected due to poor quality of estimates. There may be a situation when the robot would still be able to estimate it's pose in low illuminated environment. So here we cannot really put a number as a threshold to the environmental property hence we have assigned it as a soft constraint. In this case, poor illumination will affect the reliability of the node but not it's functionality. The primary function of the node is to estimate the pose and hence the pose quality should be good. If the pose quality is poor, it will affect all other components which are using estimated pose as an input. Hence pose quality is assigned as a hard constraint because this should never be violated and if violated, should raise an error. The two monitors i.e. *Illumination monitor* and *Pose monitor* will observe the respective property i.e. environmental illumination in former and pose and quality of pose in later, and will update the current value to the knowledge base. In this way the same schema can be used for grounding all the ROS nodes and hardware components present in the system. Grounding of all the components according to our system will be shown in the next chapter.

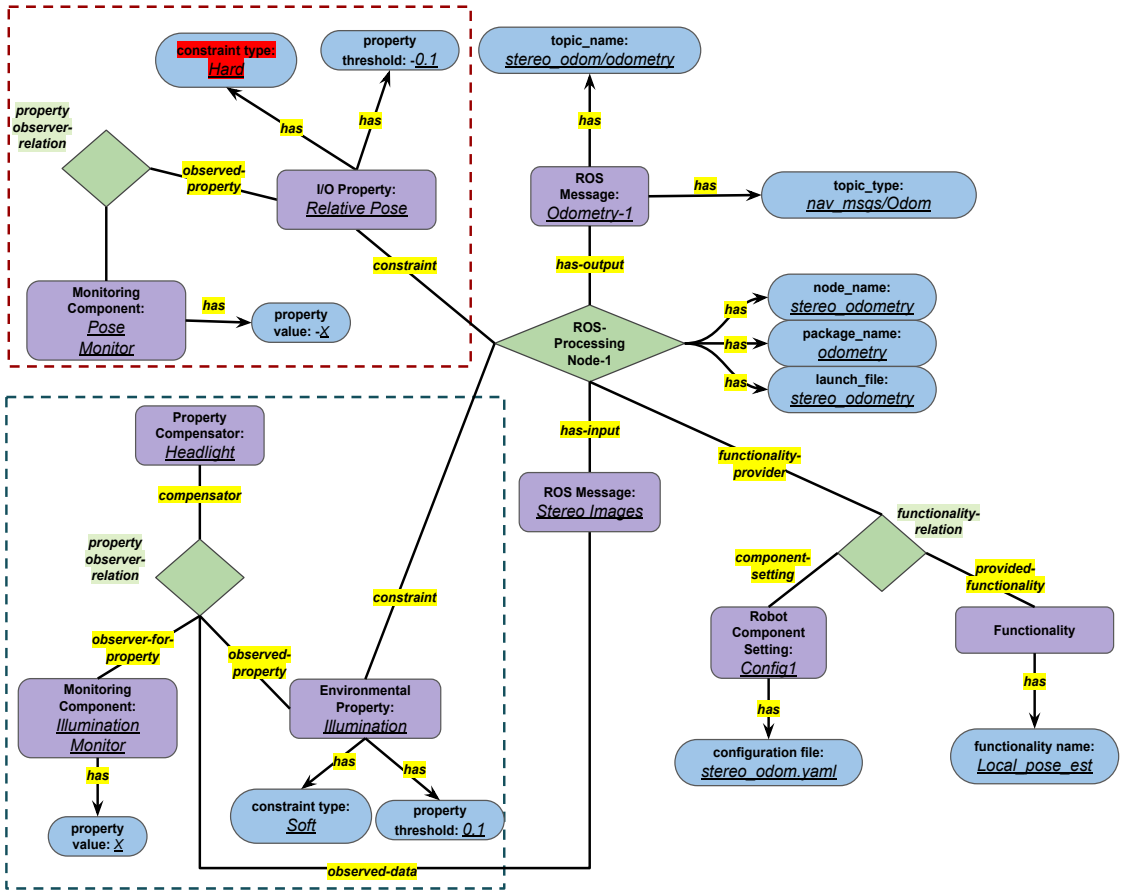

Figure 5-7: Data Instance of the concepts related to stereo camera ROS node

5-2 Monitor

The monitor element of Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) loop is responsible for gathering particular data from the underlying managed system and the environment through probes or sensors of the managed system, and saving the data in the Knowledge. This element along with analyse which is discussed in the next section is responsible for achieving situational awareness. We have seen examples of few monitors in the last section. Here we will discuss all the monitors which we have created for our localisation system. We have used some guidelines for developing the monitors which are as follows:

- **Monitor should be as general as possible:** We want the the monitors to be such that they can be used for maximum possible cases. However, a single monitor can not cover all the properties. But we can create monitors such that they will capture the environmental properties independent of the method used for estimation. For example: Consider a feature extraction algorithm being used for vision based localisation. The performance of that algorithm would be dependent on the tuning of the algorithm parameters and the environmental conditions which affects the vision. We will focus on the monitors which will capture environmental conditions because those same conditions

would also affect all vision based methods.

- **Use minimum number of parameters for monitor:** We want our monitors to use less number of parameters
- **Monitors should not be computationally expensive:** As these monitors would only observe the current conditions and have no contribution to the primary task of the robot. Hence they should not use much computation.

Using the above guidelines we have developed four types of monitors to observe the quality of the environmental conditions and estimation algorithm for vision based localisation. We have not developed any monitors for laser based localisation due to the time limitation of this thesis. But it can be done using some of the methods available in literature such as accuracy estimation for laser point cloud [67] and analysing the quality of matched 3D point clouds [68].

5-2-1 Pose quality monitor

The main task of the localisation system to estimate the pose of the robot. Several methods have been proposed in literature for quantifying and dealing with the uncertainty in the robot pose. The most commonly used metric is the co-variance matrix as the uncertainty measure for the location estimator [69]. We will also use the same metric as a pose quality metric to monitor the quality of the pose estimation. Other metric such as Absolute Trajectory Error (ATE) can also be used as a performance metric but for ATE the ground truth measurement is required which is not available in real life unless some external tracking system is used. Hence during runtime operation we will use pose co-variance as a performance metric. The co-variance measures the squared expected deviation from the mean values and is used to expresses position uncertainty after error correction [70]. Almost all the methods for robot localisation give co-variance matrix as a side product along with the robot pose. Hence we have decided to use co-variance matrix to quantify the quality of the localisation method. The higher the co-variance matrix the poorer is the quality of the localisation method and vice versa. The robot pose in 2D has 3 states i.e. x position, y position and the yaw angle and each state is uncorrelated. Hence the co-variance matrix is

$$\sum_{pose} = \begin{bmatrix} \sigma_x^2 & \sigma_x * \sigma_y & \sigma_x * \sigma_\theta \\ \sigma_y * \sigma_x & \sigma_y^2 & \sigma_y * \sigma_\theta \\ \sigma_\theta * \sigma_x & \sigma_\theta * \sigma_y & \sigma_\theta^2 \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}$$

where σ is variance in each state.

But instead of representing each individual variance, we used root sum squared variance as the pose quality metric L_pose.

$$L\_pose = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_\theta^2} \tag{5-1}$$

Analysing only the pose co-variance metric is not sufficient as it will only tell about the quality of the robot localisation but not the reason behind the poor quality. Hence we also need to analyse the environmental conditions to find the cause behind the poor quality of localisation.

5-2-2 Environment illumination monitor

In order to identify the reason behind the poor quality of localisation we need to monitor the components involved in the localisation and the environmental situations. Environmental properties have lot of effects on the localisation capability. For vision based localisation, we need to monitor the factors that might affect the robot vision such as illumination, noise, visual features, etc. Hence we have developed monitors to observe those features.

Intensity histogram provides the information about various characteristics of the image such as brightness and contrast. If the histogram is concentrated in a narrow region, it means the image has low contrast and broad histogram reflects an image with significant contrast. The mean and standard deviation of the histogram gives an idea about the overall histogram distribution which can be used to estimate the environmental illumination. Hence we will use intensity histogram as illumination monitor.

The RGB image obtained using camera can be converted to Gray scale image(Y) which denotes the intensity of the each pixel in the image. We have done conversion using YCbCr color encoding system [71]

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{5-2}$$

Gray image consists of intensity pixels Y and histogram of the Gray image gives the overall intensity distribution of an image. Using the gray image histogram we can define the illumination metric as follows:

$$L_{illumination} = \sum_{i=1}^{N} i * P(i) \tag{5-3}$$

where *P(i)* is the probability of gray value and N is the number of possible gray values in the histogram (256 for 8 bit image)

The intensity histogram varies with the change in the illumination of the surrounding which is captured by illumination metric. Instead of using image to detect the environmental brightness, we can directly use a light meter or lux sensor which is a device used for measuring the illuminance, but we don't have the hardware available so we will use indirect methods to determine the environment illumination.

5-2-3 Visual information monitor

The image gradient is the attribute related to the fine details, texture and edges of the image and can be used to extract information from images. Various image features such as Histogram of oriented gradients (HOG) [72], Scale-invariant feature transform (SIFT) [73], edge detection, object detection are computed using image gradient information. As these features use gradient information, we can use a gradient based metric to monitor the amount of visual information in the surrounding. We could have used the feature detection method itself such as HOG or SIFT for monitoring visual information. But there are different feature based methods and any of the methods can be used for the localisation. So those types of monitors will not provide generality and would be dependent on the method used. Also, there

are lot of parameters for each of these methods which are tuned differently for each system. Hence we have used a monitor which will use image gradient information directly without depending on any specific method.

The purpose of the visual information monitor is to quantify the visual information available in the image. We have adopted the nonlinear gradient mapping function from Shim et al. [36], and made additional improvements similar to [30]. First the grayscale image obtained from Equation 5-2 is use to calculate the gradient of the image. The gradient is calculated using Sobel operator [74]. The operator uses *3x3* kernel which is convoluted over the image to calculate the approximate derivative. The horizontal and vertical changes are calculated separately and are merged as shown in Equation 5-5.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I \tag{5-4}$$

where * denotes 2-dimensional convolution operation, G_x, G_y represent the image gradient in horizontal and vertical direction respectively.

The overall gradient magnitude of the intensity image Y is given by:

$$G = \sqrt{G_x^2 + G_y^2} \tag{5-5}$$

The mapping function relates the gradient magnitude to the gradient information and is calculated as follows.

$$\tilde{g_i} = \begin{cases} \frac{1}{N_g}(\lambda(g_i - \delta) + 1), & g_i \geq \delta \\ 0, & g_i < \delta \end{cases} \tag{5-6}$$

$$\text{s.t. } N_g = log(\lambda(1 - \delta) + 1),$$

where $g_i \in [0, 1]$[1] which is the gradient magnitude at pixel i of gradient image obtained from Equation 5-5, δ is the activation threshold which is used to filter out noise, λ is the control parameter to adjust the mapping behaviour, N_g is the normalisation factor and $\tilde{g_i}$ is the of gradient information at pixel i. However this mapping function is strongly biased to the gradients caused by the high exposure [30]. In order to remove the bias the image is divided into grids and the gradient information in each grid is aggregated. This modification in the mapping function reduces the bias caused due to non-uniform distribution. The gradient information in each grid is given by

$$G_j = \sum_{i \in C_j} \frac{\tilde{g_i}}{Wj * Hj}, \qquad j = 1, 2, \ldots, N_c \tag{5-7}$$

where G_j is the j^{th} grid cell, N_c is the total number of grid cells, Wj, Hj are the width and height of the j^{th} grid and $\tilde{g_i}$ is the amount of gradient information.

The final gradient metric is given by mean gradient information in each tile i.e.

$$L_{gradient} = E(G) \tag{5-8}$$

where $E(G)$ is the mean value of the distribution of the gradient information in each grid cells G_j. Large $L_{gradient}$ indicates that the gradient information is strong and uniformly distributed and low $L_{gradient}$ indicates weak and biased gradient. This gradient information metric is independent of the method used and will capture the information about the amount of visual feature in the environment.

5-2-4 Image Noise monitor

The visual localisation methods can also suffer when the image noise reaches a certain intensity [75]. As none of the above monitors explicitly capture the noise in the image, we will use a noise based metric to quantify the noise available in the image. There are various methods to estimate the noise in the image such as Peak signal-to-noise ratio (PSNR), Feature similarity index (FSIM) which compares the distorted images with original image. However in our case there is no reference image to compare with and so we will use the noise metric without a reference image. Also, the methods proposed in [76] based on eigen value analysis and [77] based on principal component analysis give more accurate results but are computational expensive. Hence we use the method proposed in [78] which used a filter based approach which is computational fast.

The metric estimates the noise using laplacian operator proposed by Immerkaer in [79] with some modifications of adaptive edge detection proposed in [78] to prevent over-estimation of the noise.

Immerkaer [79] proposed a noise estimation operator which mask the image using 3x3 kernel given by

$$M = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{5-9}$$

However the above laplacian kernel is sensitive to image objects which will result in inaccurate noise estimates. To improve the accuracy the laplacian filter should be applied to remove the saturated regions and only to keep unsaturated homogeneous regions. Hence we uses two masks one for unsaturated regions which is a simple threshold mask given by:

$$U(i) = \begin{cases} 1, & \tau_l \leq Y(i) \leq \tau_h \\ 0, & otherwise \end{cases}$$

where τ_l, τ_h are the lower limit and upper limit for unsaturated pixel, $Y(i)$ is the intensity image.

The homogeneous region mask H which is an adaptive threshold mask is given by

$$H(i) = \begin{cases} 1, & g_i \leq \delta_p \\ 0, & g_i > \delta_p \end{cases}$$

where the adaptive threshold δ_p is the p^{th} percentile of gradients in the image, and $g(i)$ is the gradient magnitude of the pixel i.

The noise based metric is given by

$$Ł_{image-noise} = \sqrt{\frac{\pi}{2 * Ns}} \sum_{i} H(i).U(i).|I * M|(i) \tag{5-10}$$

where N_S denotes the number of valid pixels in the mask $H \cdot U$, * denotes the convolution operator and $|\cdot|$ denotes the absolute operator.

Changing lighting conditions [4, 29, 30, 80], areas of inconstant visual features [4] and camera noise [75, 30] are the major issues for visual perception which are observed by the monitors which we have developed. There are other problems that have a close influence on the performance of visual perception such as photometric calibration, motion bias, and rolling shutter effect [81]. However motion bias and rolling shutter were not of concern here as the mobile robot does not preform any high speed manoeuvre. Hence we choose to ignore them. These monitors will extract meaningful information from the environment and update the current states to the knowledge base which will be used to check the violation of constraints and take the adaptation decision. The results of these visual perception monitors on TID2008 dataset [82] is shown in Appendix B. Note that in this thesis work we have not covered issues related to the laser perception due to the time constraint.

5-3 Analyse using Deductive reasoning

Different approaches for analysing and planning the adaptation are present which needs a criteria for identifying the need for adaptation and for choosing a suitable adaptation plans, respectively. These metrics are based on: models, rules, goals, or policies. In order to analyse the system for faults and uncertainties and plan the reconfiguration, we will use deductive reasoning. Deductive reasoning or deductive logic is the process of reasoning from one or more statements (i.e. premises) to reach a logical conclusion. If all premises are true, the terms are clear, and the rules of deductive logic are followed, then the conclusion reached is necessarily true [83]. An example of deductive reasoning is

- All men are mortal. (First premise)
- Socrates is a man. (Second premise)
- Therefore, Socrates is mortal. (Conclusion)

The first premise states that all objects classified as "men" have the attribute "mortal." The second premise states that "Socrates" is classified as a "man" – a member of the set "men." The conclusion then states that "Socrates" must be "mortal" because he inherits this attribute from his classification as a "man." [83]

This type of deductive reasoning will be performed on the data which will be used to analyse the faults and its effects on the system and to come up with reconfiguration strategy. The advantage of using deductive reasoning is that a conclusion is reached deductively by applying general rules which hold over the entire closed domain. These general rules can be reused or shared for other system which fulfils our Non-Functional requirements (NFR) of reusability

and generality. We will now discuss how reasoning is performed in our system to analyse the situation.

After having created the data instances of the concepts specific to our system, reasoning can be performed over the data using pre-defined rules, which can be included in the Graql definition of the ontology schema or by using `match insert` queries. The `match-insert` query together in combination can be used to perform reasoning such as when a particular pattern of the instances are found in the match clause new data can be inserted into knowledge graph using insert clause. The rules though somewhat similar to the match insert queries have slightly different functionality. The rule-based reasoning in Grakn allows us to capture the evolution of patterns within the knowledge graph. These rules are composed of a set of conditions (the "when" part of the rule) and a conclusion (the "then" part of the rule). When all the conditions of the premises are satisfied, Graql rules will search for a given pattern in the data and if found, it will create a queryable relation only for the time being of the given transaction, so a queryable relationships is inferred without actually adding new data [84].

The advantage of using rules is that the set of rules included in the Grakn schema can be used for any data that is inserted in the ontology and the inferred conclusions do not have to be deleted separately when the corresponding pattern is deleted. On the other hand, inserting the data using insert queries will only take into account the data at that specific moment and the inserted relationships have to be deleted separately if the corresponding pattern is deleted. Hence we will use rules to perform deductive reasoning for two main reasons: 1) We don't want tailored queries to perform reasoning as this will limit the reusability of the rules 2) We don't want to complicate the process of reasoning by adding data instances and deleting separately if a corresponding pattern is deleted.

Table 5-1 presents the rules in logical formalism which we have developed for analysing the situation. Note that these rule are expressed using logic notation for easy understanding. However in our system, these rules are modelled using Graql language which makes them much more expressive. We have used 7 rules all of which are general and can be used for any ROS based system. In the rules "?x" indicate a variable x, "A(?x,?y)" indicate two variables x and y are in a relationship "A", and "B(?x,var)" indicate the variable x has a attribute B which holds the value var. The meaning of all the rules is mentioned in Table 5-3.

Note that all of these rules are general and do not have any special encoding for localisation. Hence these rules can be reused for any ROS based system and not just limited to localisation. For reusing the rules one needs to follow the schema shown in Figure 5-5. For example, consider a robot manipulator which is developed using ROS architecture. The robot has to move an object from position A to position B. There are three ROS nodes one for producing image data, one which will take the image data to detect the object, and third the planning node which will take the information of the detected object to perform planning. Using rule R1 and R2 hierarchy at all the levels will be inferred. Now if the obstacle is blocking the view of camera and there is one monitor detecting the quality of camera feed similar to the visual quality monitor which we have developed, it will update the knowledge that even though the camera node is producing data it doesn't have much information. Hence using rule R3 and R4 the camera node will become unhealthy. This will make the object detection node and the planning node unhealthy because of rule R6. Hence the rules we have developed are general and can be reused for any ROS based system, provided the schema if followed.

Table 5-1: Rules for analysing the situation represented in boolean logic

Purpose	Rule
Node Hierarchy	**R1:** Node(?N1) $\wedge hasop(?N1, ?IO2) \wedge$ Node(?N2) $\wedge hasip(?N2, ?IO2)$ $\rightarrow ConfigHierarchy(subordinate :?N1, superior :?N2)$
	R2: Node(?N1) $\wedge Node(?N2) \wedge Node(?N3)$ $\wedge ConfigHierarchy(subordinate :?N1, superior :?N2) \wedge$ ConfigHierarchy(subordinate:?N2,superior:?N3) $\rightarrow ConfigHierarchy(subordinate :?N1, superior :?N3)$
Health Detection	**R3:** Property(?P) $\wedge hasthresh(?P, ?Th) \wedge Monitor(?M) \wedge$ propobvrelation(?P,?M) $\wedge hasvalue(?M, ?V) \wedge (?V <?Th)$ $\rightarrow issatisfy(?P, false)$
	R4: Node(?N) $\wedge Property(?P) \wedge hasconstraint(?N, ?P, hard)$ $\wedge issatisfy(?P, false) \rightarrow Nodestatus(?N, unhealthy)$
	R5: Node(?N) $\wedge Property(?P1) \wedge Property(?P2)$ $\wedge hasconstraint(?N, ?P1, soft) \wedge hasconstraint(?N, ?P2, hard)$ $\wedge issatisfy(?P1, false) \wedge \neg issatisfy(?P2, false)$ $\rightarrow Nodestatus(?N, unreliable)$
Health Propagation	**R6:** Node(?N2) $\wedge Node(?N1) \wedge Nodestatus(?N1, unhealthy)$ $\wedge ConfigHierarchy(subordinate :?N1, superior :?N2)$ $\rightarrow Nodestatus(?N2, unhealthy)$
	R7: Node(?N2) $\wedge Node(?N1) \wedge Nodestatus(?N1, unreliable) \wedge$ $\wedge ConfigHierarchy(subordinate :?N1, superior :?N2)$ $\wedge \neg Nodestatus(?N2, unhealthy)$ $\rightarrow Nodestatus(?N1, unreliable)$

5-4 Plan and Execute

The monitor and analyze together will provide situational awareness and will answer questions such as: is the environment good for performing localisation, is the estimator working properly, etc. However, if any event is detected which affects the localisation functionality, the system should try to overcome it using other available resources. Hence we need plan and execute element which are the part of the MAPE-K feedback loop. In MAPE-K the plan and execute agent is responsible for finding an alternative for reconfiguration and executing those alternatives. To perform adaptations, plan and execute elements use the knowledge of the components present in the system to search for a configuration which will provide the same functionality to overcome the faulty behaviour.

5-4-1 Plan element using rules and queries

As we know that Graql queries can be used to retrieve or modify information stored in a knowledge graph, we will use queries to plan the reconfiguration and update the reconfigured components into the knowledge. The planning could also be done by using a decision making or planning approach such as STRIPS solver, Planning Domain Definition Language (PDDL), behaviour tree based method. However planing using languages like PDDL are often con-

structed for a specific application and can be difficult to reuse [85]. Instead we have used Graql queries for planning.

Table 5-2: Rules an queries for planning the reconfiguration in boolean logic

Purpose	Rule
Action recommendation	**R8:** Node(?N) $\wedge Nodestatus(?N, Unreliable)$ $\rightarrow action(?N, optionalreconfigure)$
	R9: Node(?N) $\wedge Nodestatus(?N, Unhealthy)$ $\rightarrow action(?N, mandatoryreconfigure)$
Query	**Q1:** action(?a,?type) $\Rightarrow get(?a, ?type)$
Action Query for adaptation	**Q2:** action(?N, mandatoryreconfigure) $\wedge Property(?P)$ $\wedge hasconstraint(?N, ?P) \wedge issatisfy(?P, false)$ $\wedge propobvrelation(?P, compensator :?C)\wedge$ isActive(?C,false) $\Rightarrow get(?C)$
	Q3: action(?N1, mandatoryreconfigure) $\wedge Node(?N2)$ $\wedge Functionality(?F, localisation) \wedge Nodestatus(?N2, healthy)$ $\wedge functionalityprovider(?N1, ?N2, ?F)$; ConfigHierarchy(?N3,?N2) $\Rightarrow get(?N3, ?N2, ?N1)$
	Q3: action(?N1, mandatoryreconfigure) $\wedge Node(?N2)$ $\wedge Functionality(?F) \wedge Nodestatus(?N2, healthy)$ $\wedge functionalityprovider(?N1, ?N2, ?F)$; ConfigHierarchy(?N3,?N2) $\Rightarrow get(?N3, ?N2, ?N1)$
Query for initialization	**Q4:** functionalityprovider(?N,localisation) $\wedge Node(?N) \wedge Posedata(?p) \wedge hasop(?N, ?p)$ $\wedge hasposex(?P, ?x) \wedge hasposey(?P, ?y)\wedge$ hasposeyaw(?P,?yaw) $\Rightarrow get(?x, ?y, ?yaw)$

*Note that **Q3** and **Q3** are similar queries. The only difference is that **Q3** is generic and in Q3 we are searching for alternative configuration which can provide the localisation functionality. We will only use Q3 hereafter for our use case*

Table 5-2 show the primary queries and rules for planning. The meanings of these rules and queries are given in Table 5-3.

Note that the reconfiguration action is optional if the constraint is soft and node status is unreliable, and the it is mandatory if the constraint is hard and node status is unhealthy. These rules are static and will be added while defining the schema. The queries on the other hand can be used to add, update and delete data from the knowledge base. These queries will be used to perform adaptation. A ROS nodes will continuously query the knowledge graph at certain frequency with query Q1 to check if any action is needed. Ideally this query should be **NULL** if all the components are working properly and all constraints are satisfied. But if any constraints are violated, the query will return with the action which should be taken. If the action is reconfiguration the planning node will query the knowledge with Q2 to check if any compensator is available for violated constraint. If it is available the compensator will be activated and the state will be updated in the knowledge. However if there is no compensator for the violated constraint the node will query the knowledge with Q3 to search for other configuration which can provide the localisation functionality.

Note that till now all the rules and queries are general and reusable for other ROS based

Table 5-3: Meaning of rules and queries which are presented in Table 5-1 and Table 5-2

Rule/ Query	Generality	Meaning
R1	general	Node N1 produces output IO2 and node N2 has input IO2 then Node N1 act as a subordinate node to the node N2 in the configuration hierarchy relation
R2	general	Node N1 acts as a subordinate node to node N2 and Node N2 acts as a subordinate node to node N3 then node N1 will also act as a subordinate node to node N3
R3	general	There is a property P and it has some threshold Th, there is a monitor M observing that property and has current value V then if vale V <Th the property will be violated
R4	general	There is a node N which has property P as a constraint and the P is a hard constraint and if P is violated then the node will have status unhealthy
R5	general	There is a node N which has property P1 and P2 as constraints and P1 is a soft constraint and P2 is a hard constraint. Now if P1 is violated and there is no P2 which is violated then the node will have an unreliable status.
R6	general	There is a node N1 and N2 and they are in a configuration hierarchy, where N1 is a subordinate node and N2, is a superior node. Now if N1 is unhealthy then N2 will also be unhealthy
R7	general	There is a node N1 and N2 and they are in a configuration hierarchy, where N1 is a subordinate node and N2, is a superior node. Now if N1 is unreliable and N2 is not unhealthy then N2 will also be unreliable
R8	general	There is a node N1 which has node status unreliable. Then the recommended action is to optionally reconfigure node N1
R9	general	There is a node N1 which has node status unhealthy. Then the recommended action is to compulsory reconfigure node N1
Q1	general	If there is an action recommendation then get the type of action to be performed and the node on which the action needs to be performed
Q2	general	If there is an action recommendation on node N1 which is a compulsory reconfiguration and the node N1 has constraints P and the constraints are violated. If the property has a compensator and if the compensator is not active then get the compensator name.
Q3	Localisation Specific	If there is an action recommendation on node N1 which is a compulsory reconfiguration and the node N1 provides localisation functionality then is there any other node N2 which can provide localisation functionality and has a status healthy
Q4	Localisation Specific	There is a node N which provides the functionality of localisation. The node has an output P which is of type pose data and pose data has x,y, yaw information. Then get x,y, yaw

application. However, while changing the configuration we want the other configuration to be initialised from some initial state i.e. the initial pose of the robot. Once we get the launch

files of the node for the alternative configuration we would initialise them with the last saved robot pose. This is done by again querying the knowledge graph to give the the latest saved pose provided by the node which was offering localisation functionality. This is done using query Q4 which will get the initial states of the robot. These states will be passed as an argument to the execute element along with the launch name of the nodes.

5-4-2 Execute element using ROS node manipulator

The Execute element is responsible for executing the adaptation actions of the generated plans. Execute requires an effector to adapt the managed element (in our case ROS nodes) according to the results of the planning element. We have used ROS action server as an execute agent. The planning ROS node acts as a ROS action client and will send a request to ROS execute node for executing adaptation actions. In our case the action would be starting new ROS nodes, stopping working ROS nodes or changing some parameter values. The ROS execute node will execute the actions and send the results of the actions back to the planning ROS node which will update the knowledge base. For switching the ROS nodes we are using the python subprocess module which allows to spawn new processes, connect to their input, output, error pipes, and obtain their return codes in runtime.

5-5 Functional Architecture of SASA localisation framework

Now as we have explained each element of our framework we will discuss the functional architecture of the SASA localisation framework. Figure 5-8 shows the functional architecture of the complete system. In the developed SASA framework there would be different number of monitors which will observe and update the current status of the environment and the internal states of the robot to the knowledge. These client monitoring nodes will send a request to the monitor manager which will provide the service of updating the current status to the knowledge. There will be one monitor manager and multiple monitoring nodes as per the requirement. The planning node will query the knowledge graph at certain frequency to check for action recommendation. This planning node will be a ROS action client node and will send request to the execution manager node which is a ROS action server node and will act as a execution manager which will execute the the necessary actions. The complete flow of the activity in the system is shown in the Figure 5-9.

5-6 Summary

In this chapter we completed the development process of the SASA localisation framework. We added the automatic manager layer to our framework using MAPE-K reference model. This automatic manager provided our framework with the capability of self-adaptation and situational awareness. The knowledge schema which we have developed is generic and reusable to any ROS based system. However to make it more specific to the localisation use case we created the sub-types of the schema concepts and instantiated those sub-types according to the components available within our system. We developed the quality monitors to observe the performance of the localisation and the surrounding environment. These monitors would

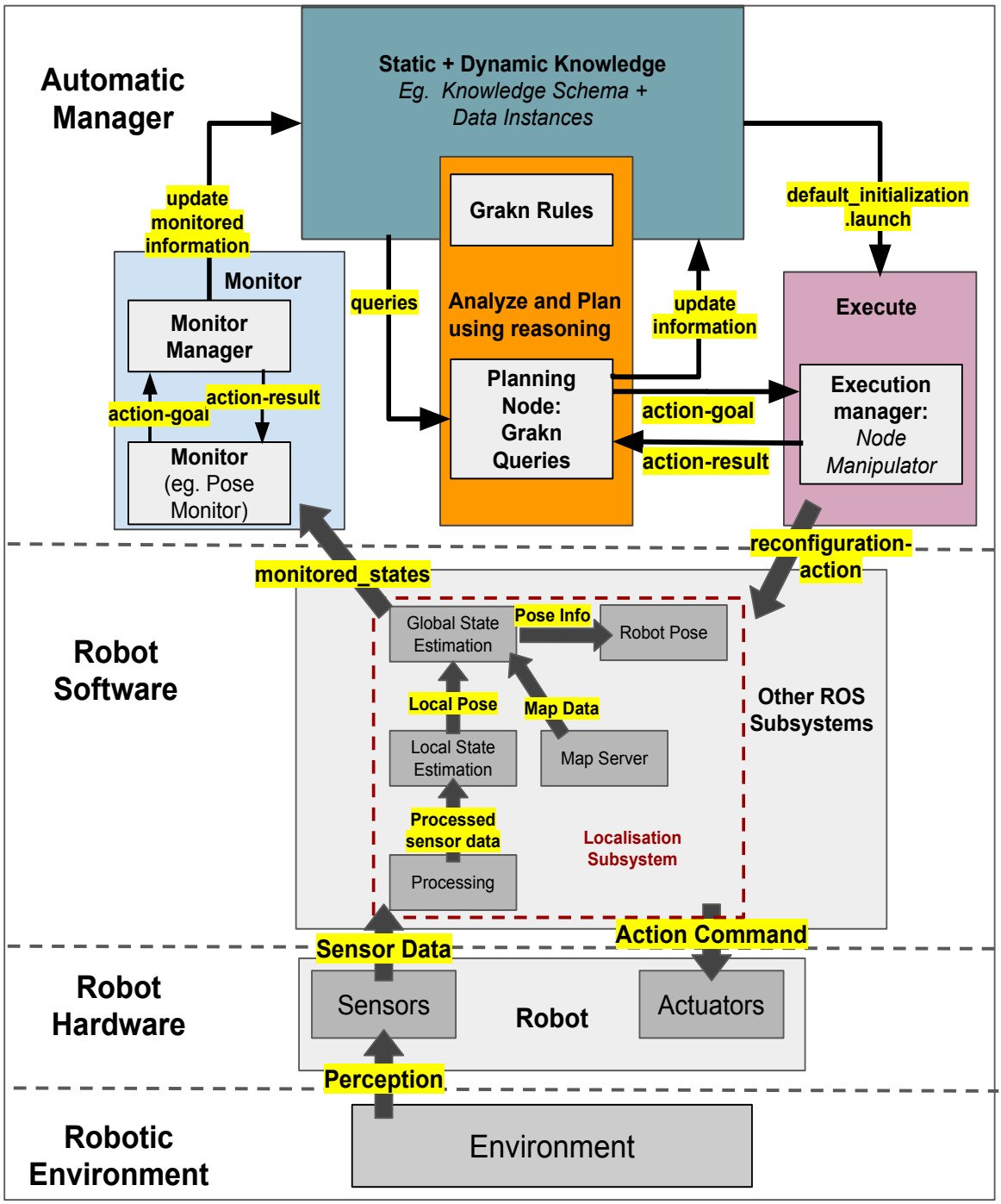

Figure 5-8: Functional architecture of SASA localisation framework

help the SASA framework to understand the current situation of the robot and its environment. We also developed the rules which will be used to analyse the situation and plan the reconfiguration action in case of unexpected situations. In the end, we presented the complete

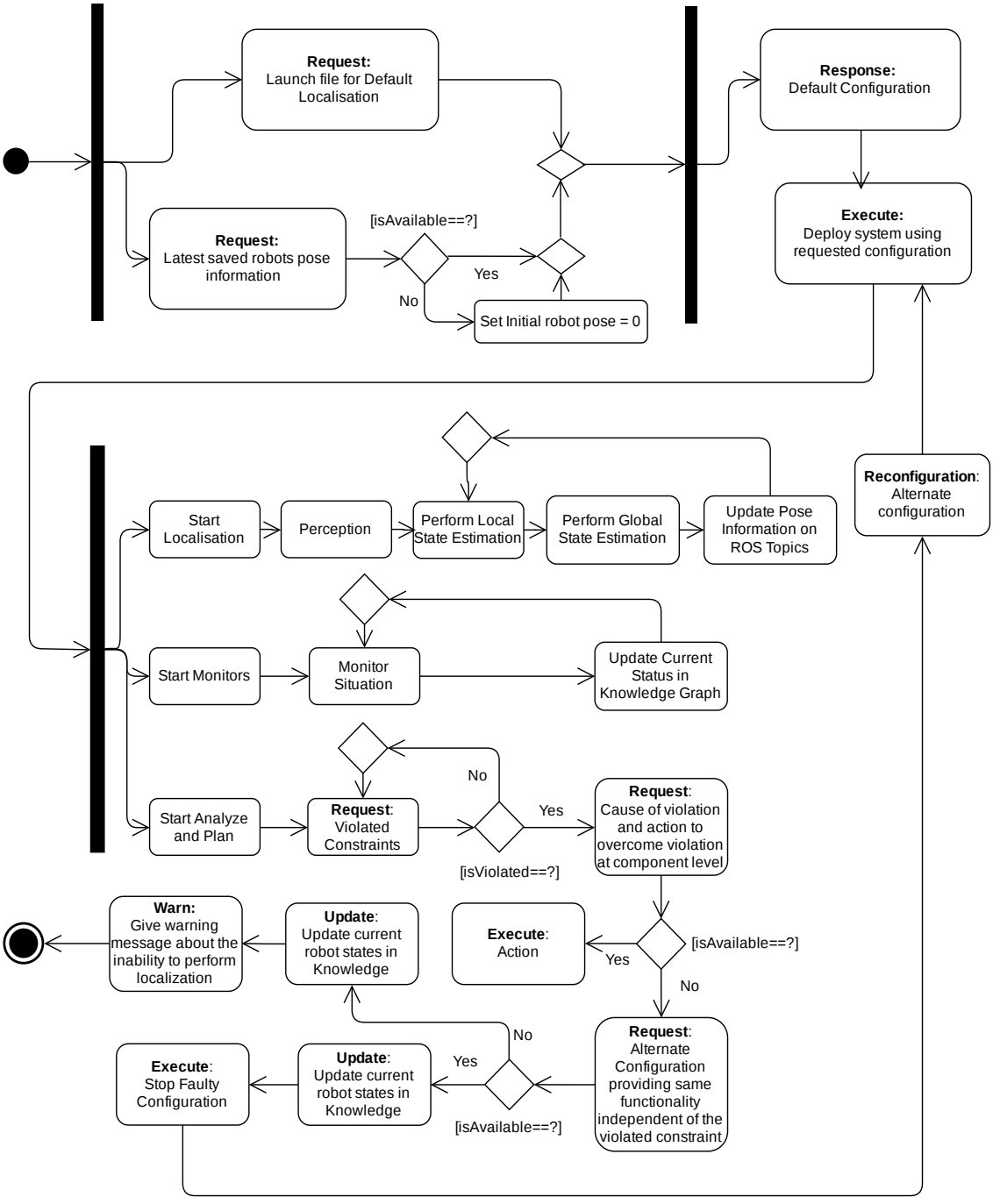

Figure 5-9: Activity diagram of situational aware localisation system

functional architecture of our SASA localisation framework and the activity diagram of our framework.

Chapter 6

Experiments and Results

The research questions mentioned in Chapter 1 were answered using experiments. This chapter covers the experiments carried out to validate the developed Situation-Aware Self-Adaptive (SASA) localisation framework. Section 6-1 will cover the experimental design and the scenarios tested along with the technical details of the system. Then we will discuss the results of each test scenarios. Section 6-7 will discuss the general analysis about the SASA framework.

6-1 Experimental Setup

In order to validate the developed framework we carried out 5 experiments. These experiments were preformed to test the fault tolerance capability of the framework and the capability to perform during environment disturbances. Experiments were also carried to test the reusability, changeability and consistency of the framework. The behaviour of the system will be analyzed by comparing the pose estimation before and after the occurrence of an event. The trajectories of different methods will be compared based on Absolute Trajectory Error (ATE).

6-1-1 T1: Environmental disturbance

To test the system's behaviour during environmental disturbances we performed tests by changing the lighting conditions in the simulation environment. The lighting in the retail store environment is not uniform and might change over the period of time or get damaged or could be blocked due to some shelves which will cause nonuniform illumination. This will affect the robot's ability to perform vision based localisation. To validate our developed SASA localisation system for this situation, we preformed tests by changing the lighting in the environment. For changing the lighting conditions we have created our custom plugin in Gazebo simulator which can modify the lighting intensity of the environment.

6-1-2 T2: Fault tolerance

To test the reconfiguration due to internal failure we performed tests by injecting a fault in the localisation system. As the experiments were carried out in the simulation environment, it was not possible to replicate the failure of the sensor in run-time. However we intentionally stopped a running process (i.e. ROS node) which was required for other Robot Operating System (ROS) nodes at upper hierarchical level.

6-1-3 T3: Reusability

To test the reusability of the developed rules we performed tests on a different property which acted as a constraint to the stereo pose estimation node. For this test we created a new monitoring client node to detect the amount of visual information in the surrounding environment. The monitor was created using the metric we discussed in Section 5-2-3. No modification was made to any of the rules. To replicate it in the simulation environment, empty shelves were placed in some part of the retail store environment and some regions had plain wall. This made those regions featureless which will affect the robot's ability to perform vision based localisation.

6-1-4 T4: Changeability

To test the changeability of our framework we made some modification to the robot's hardware. We have added a flashlight on the robot which can be actuated by the robot using gazebo plugins. This experiment was performed to replicate the situation where some modifications are made to the robot during it's life-cycle. To do this, one instance of `compensator` class was created which is subtyped from `Robot component`, and it was added to the `property-observer-relation` with `Illumination` property. The new `flashlight component` will play the role of compensator in `property-observer-relation`. No modifications were made to the rules.

6-1-5 T5: Consistency

To test the consistency of the SASA framework, experiments were performed where we simulated the same situation 15 times at different locations in the retail store to check whether the behaviour of the framework consistent. This test was done to replicate the real life scenario where the robot needs to behave consistently in real world.

6-1-6 Technical details of the system

The carry out the above test scenarios we used a laptop with Intel Core i7 8th generation processor, 16 GB RAM, NVIDIA GeForce GTX 1050Ti GPU, Ubuntu 18.04 Operating System. The simulator and ROS packages used for experiments are as follows:

- Gazebo Simulator: Multi-robot simulator, version 9.15.0. All experiments are performed in this simulator with added plugins for controlling the illumination in the environment

with the help of $light/modify$ gazebo topic. The properties of the lights in the simulation can be modified using the plugins in run-time.

- Mobile Robot: The robot models are created in Gazebo using Universal Robotic Description Format (URDF) which is an Extensible Markup Language (XML) format for representing a robot model. Using URDF we can specify the kinematic and dynamic properties of the robots. The robots is equipped with sensors which are mentioned below.

- Sensors: The robots are equipped with 9-DOF IMU sensor, Bumblebee Stereo RGB Camera and a SICK LMS111 2D range scanner. They are attached to the robot model and are interfaced using gazebo sensor plugins which allows complete access to the physical properties of sensors and their underlying elements.

- RViz: The rviz node is a graphical interface provided by the ROS visualization packages that displays geometric information of the running system: positions, maps, sensor readings, etc. This node is used for the visualizing the robot during operation.

- Teleoperation: The teleop node is a very simple node that allows to teleoperate the mobile robot using the computer's keyboard. It publishes messages in the *cmd vel* ROS topic, which send linear and angular velocity commands to the robot.

- Execution manager: The *execution_manager* node is responsible for initializing the system and starting the different nodes as per the operation requirement and controlling the nodes during runtime. It acts as an execute element of the Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) loop. The *execution_manager* node is a ROS action server which takes the request from the planning node to start and stop different processes in run-time.

- Monitor manager: The *monitor_manager* node is responsible for updating the monitored states to the knowledge graph. It acts as a server node which takes the requests from different monitoring node and update the information in the knowledge graph.

- Monitors: It consists of different monitoring node used for monitoring internal states of the localisation system and the environmental status such as environmental illumination monitor, node status monitor, pose monitor, etc. These monitoring nodes are started and stooped as per the requirement by execution manager.

- Planning node: The *planning_node* is responsible for querying the knowledge graph at certain frequency interval to check the need for adaptation. If any adaptation is needed i.e. configuration needs to be changed, it will send the request to the *execution_manager* node.

- Odometry Estimation: To estimate odometry we have two alternative nodes i.e. stereo odometry estimation node provided by `rtabmap_ros` package [86] and laser odometry estimation node provided by `rf2o_laser_odometry` package [87]. Both the nodes use different primary sensors to estimate odometry because of which they can be used in different situations. The node estimates the odometry and publishes the odometry information on a ROS topic as well as publish the transform between robot link and the odometry frame. The *stereo_odometry node* performs Feature to Feature (F2F) visual

feature matching and hence is a feature-based, which extracts image feature points and tracks them in the image sequence. The *laser_odometry node* performs estimate the planar motion of a laser from consecutive range scans by formulating the range flow constraint equation in terms of the sensor velocity, and minimize a robust function of the resulting geometric constraints to obtain the motion estimate [87]. Both of the packages are used in default configuration without modifying the parameters.

- Localisation: To perform localisation we have two nodes. One approach uses laser based localisation using the Adaptive Monte Carlo Localization (AMCL) approach as described by Dieter Fox in [88]. The `acml node` uses laser scan and odometry readings from *laser_odometry_node* in the form of a tf message, along with a given occupancy map of the environment, and produces an estimation of the robot pose which it published as a ROS message. Another alternative approach uses `rtab-map` localisation [6] which consumes the stereo images provided by Bumblebee Stereo RGB Camera along with the odometry readings provided by odometry estimation node, IMU sensor readings and a point cloud map and produces an estimation of the robot pose which it published as a ROS message.

- Map server: The `map_server` ROS node loads the saved map data of the retail store from a file and publishes it as an occupancy grid map. The map is essential for the operation of the `amcl` localisation. The `map_server` node is only required for `amcl` localisation as `rtab-map` localisation uses the map internally which is saved in the form of database file.

Here we have two different alternatives for localisation i.e. vision-based alternative and laser-based alternative, from the available components and ROS package. Both of these alternatives were tested in simulation environment where a robot was operated manually and the ATE between the ground truth and the estimation method was compared after the test run. In total 8 experiments were conducted 4 using stereo estimation and 4 using laser estimation for the path length of around 20m in each experiment. The ATE of the estimation method is shown in Table 6-1. Note that all experiments were conducted with the default parameter values without any tuning. The performance of the two methods for all the experiments are shown in the Figure 6-1. The two design alternatives have reasonable yet slightly different performances. For vision based estimation, we have used `stereo_odometry` node from `rtab-map` ROS package [86] and for laser based configuration we used `laser_odometry` node from `rf2o_laser_odometry` package [87]. The vision-based configuration has a mean ATE of 0.031m while the laser-based configuration has a median ATE of 0.05m. High ATE for laser based localisation configuration could be due to the parallel stocking of shelves in the stores which acts as a long parallel wall which is one of the limitations of laser based estimation. From the results of this test-run and the discussion from section 2-1-4, we have decided to use vision-based localisation as our default configuration as it provides greater accuracy than laser based localisation. The two different approaches for pose estimation will be tested in different situations and the performance of these two methods will be compared with the performance of the SASA localisation framework.

Table 6-1: ATE(m) measure for stereo estimation and laser estimation for the path traveled of length 20 m

Experiment	Exp1	Exp2	Exp3	Exp4
Stereo Est	0.03	0.035	0.032	0.028
Laser Est	0.0447	0.052	0.049	0.051

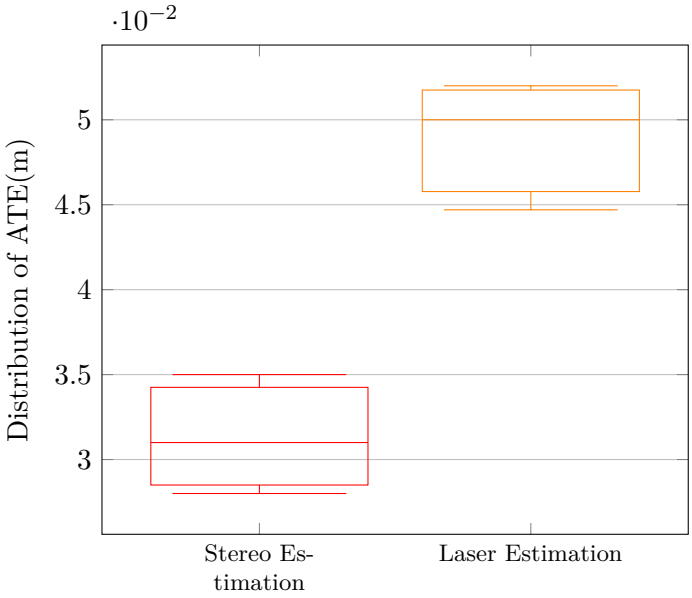

Figure 6-1: Comparison between distribution of ATE for laser-based pose estimation and vision-based pose estimation for simulation in performed in Gazebo simulator

6-1-7 Schema Instantiation

Now, to perform reasoning, we need to instantiate the concepts from the schema equivalent to their physical counterparts. The packages mentioned above will act as ROS nodes of the system. The instantiated knowledge graph with the entities, relation and attributes is shown in Figure 6-2. In total the knowledge graph contains **19 entities** and **10 relations**. They are horizontally divided into three levels i.e. sensor-data , local state estimation and global state estimation. As `rtabmap` localisation node launches its map in form a database it was not instantiated in the knowledge graph. There are two alternate configurations to perform localisation i.e. laser-based localisation and vision-based localisation. Here for the experiments we are only focusing on vision based localisation and its adaptation to laser based localisation. Hence we have created monitors for visual pose estimation node only.

The threshold of the pose covariance property was set after analysing the response of *L_pose* which is shown in Figure 6-3. During normal situations the *L_pose* metric is observed to be less than 0.1 and that value increases abruptly like a step response on occurrence of any unexpected situation. This causes the pose estimates to diverge from the original values leading to high covariance. Hence the value was set to 0.1. For node health property the threshold was set to 1 because the node monitor keep the health 1 until the node is active and sets to 0 when the node is unavailable. The threshold of the illumination property was

set empirically to 0.2. Note that the $L\_illumination$ value below 0.2 does not mean that the robot will suddenly stop providing pose estimation. Instead it just means that the robot has entered a region with poor illumination.

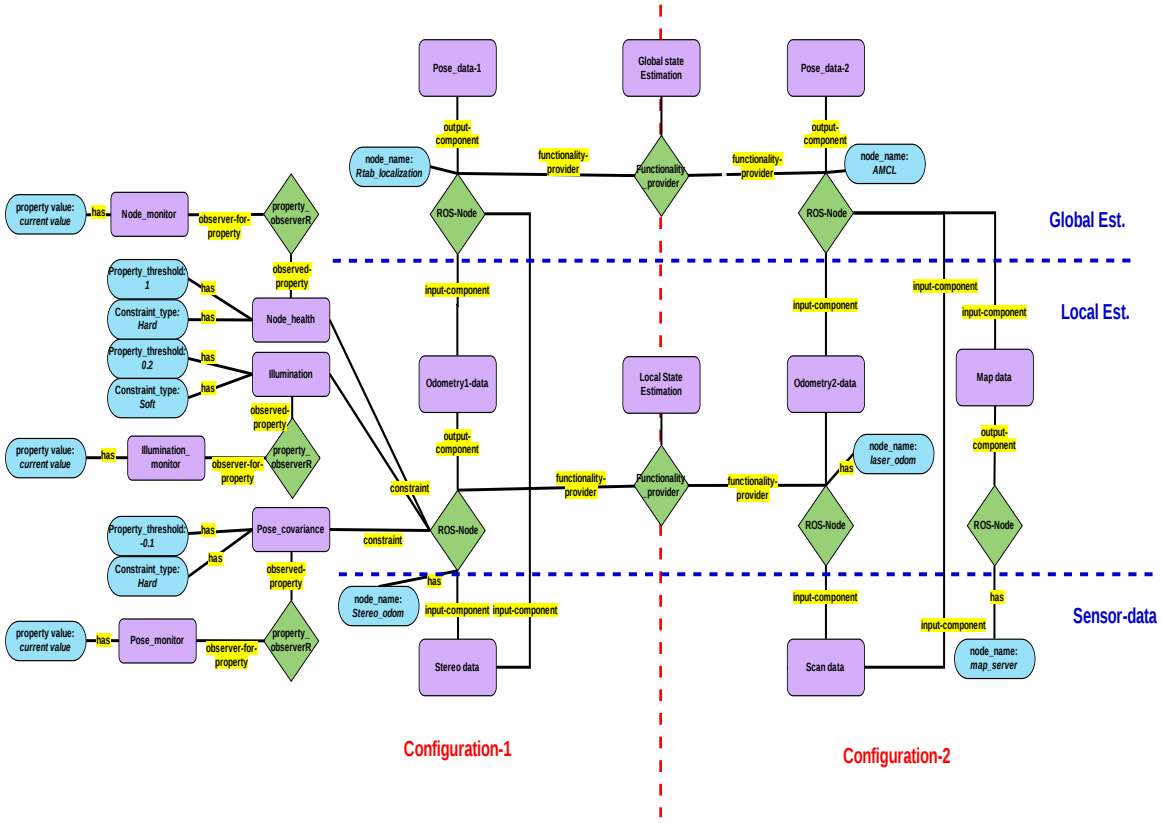

Figure 6-2: Knowledge grounding for our localisation system

We have created three monitors for stereo odometry node which act as a subordinate node for vision based localisation. If the stereo odometry node is unhealthy then the robot cannot perform vision based localisation. The functionality of the stereo odometry node depends on three constraints. Pose variance and node health are hard constraints because if any of the constraint is violated then it will affect local estimation. Note here that we have created a negative threshold for pose covariance. This is because, according to **rule R3** the constraint would only be violated if the current value is less then the threshold. But in case of covariance the less accurate estimation gives higher covariance. Hence to match the the property with the rule we are updating the negated value of the current pose covariance. And when the value is more due to poor estimation the **rule R3** will be satisfied. In future also we will put a negative threshold and update the negated current value of the property which will follow a similar trend as that of covariance.

The functional architecture of the instantiated system would also follow the same structure of the one shown in Figure 5-8. Here we have three monitors which will act as ROS action clients and update the current states in the knowledge by sending a request to the monitor manager. The structure of the monitors and monitor manager is show in Figure 6-4. Having

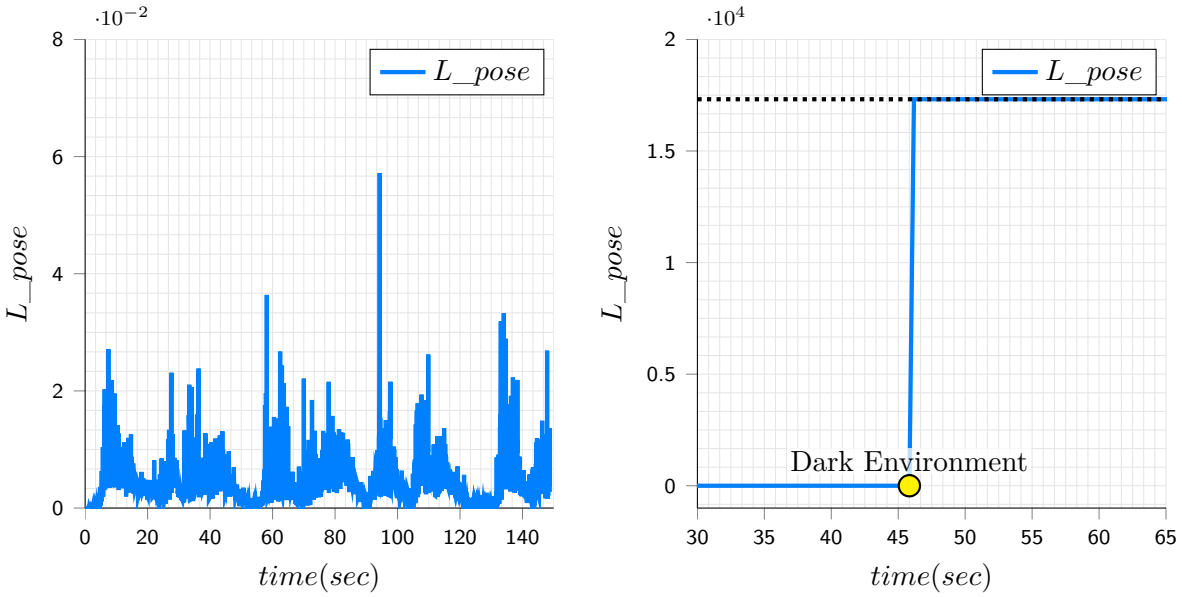

Figure 6-3: The left figure shows the values of L_pose metric in normal situations. the figure in the right shows the values of L_pose metric during an unexpected situation such as dark environment

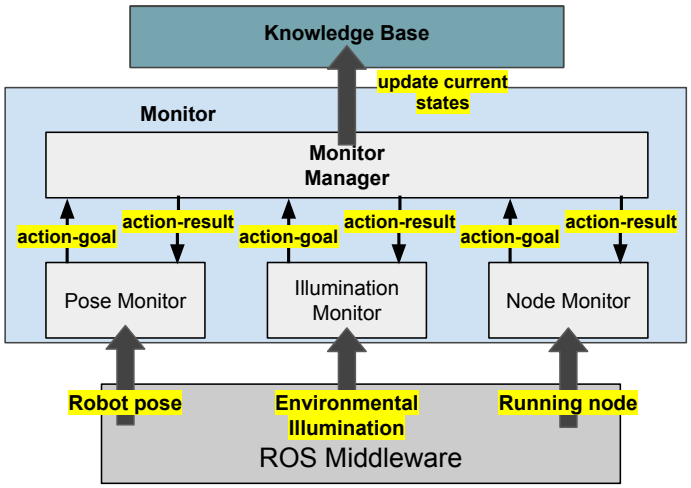

Figure 6-4: Monitors for our localisation system

discussed the system in detail, we will now analyse the results of our experiments.

6-2 Results of T1:Environmental disturbance

This test was carried to observe our system response for environmental disturbances. The sequence of activity in the system proceeds as follows. While the robot is in motion and performing localisation, the illumination was changed manually. This illumination change

causes low brightness and high co-variance in the position estimates. High pose co-variance and the low environmental illumination is detected by the monitoring nodes which updates the current states to the knowledge graph. This will cause the premises of **rule R3** to be satisfied due to which **rule R4 and R6** will also be satisfied. This will cause the stereo odometry node and vision based localisation node to change its status to unhealthy. Now the planning node which is querying the knowledge graph should ideally get *NULL* values will get some action recommendation due to satisfaction of **rule R9**. As an action is recommended, the planning node will again query the knowledge graph for additional information such as configuration hierarchy, property compensator availability using **Query Q3** which will be returned with the set of unhealthy nodes, alternate nodes and the subordinate node which will be sent to execution manager for execution. The sequence of the inference is as follows:

$$R3 \rightarrow R4 \rightarrow R6 \rightarrow R9 \Rightarrow isCompensatorQ2?activateC : (Q3 \rightarrow Q4) \tag{6-1}$$

Figure 6-5 shows the response of the SASA localisation system to the poor illumination. Figure 6-5.b displays the comparison between the ground truth and the pose estimation. Note that low illumination will cause a change in configuration in runtime. The response time of the system is 3.8 sec. *Ts: Response time: It can be defined as the total time taken by the system to overcome a situation after the occurrence of that situation. Response time is sum of the latency, reconfiguration time and action time.* In this scenario the SASA framework was able to adapt and reconfigure successfully to a new configuration within reasonable time and accuracy.

6-3 Results of T2: Fault tolerance

This test was carried to observe our system response during internal fault. The sequence of activity in the system proceeds as follows. While the robot is in motion and performing localisation, an executable file of a process (stereo odometry estimation node) was deleted and the process was stopped intentionally. As the node was stopped, the node monitor which was monitoring whether the node is active will detect it and update the current status of the node being inactive i.e. current value 0. This will cause the premises of **rule R3** to be satisfied due to which **rule R4 and R6** will also be satisfied. This will cause the stereo odometry node and vision based localisation node to change its status to unhealthy. The planning node which is querying the knowledge graph which should ideally get *NULL* values, will get some action recommendation due to satisfaction of **rule R9**. As an action is recommended, the planning node will again query the knowledge graph for additional information such as configuration hierarchy, property compensator availability using **Query Q3** which will be returned with the set of unhealthy nodes, alternate nodes and the subordinate node and be sent to the execution manager for execution. The sequence of the inference is as follows:

$$R3 \rightarrow R4 \rightarrow R6 \rightarrow R9 \Rightarrow isCompensatorQ2?activateC : (Q3 \rightarrow Q4) \tag{6-2}$$

Figure 6-6 shows the response of the SASA localisation system for internal failure. Figure 6-6.b displays the comparison between the ground truth and the pose estimation. Note that the internal failure has caused the configuration to be changed during runtime. The SASA framework was able to respond in is 2.8 sec. In this scenario, the SASA framework was able to adapt and reconfigure successfully to a new configuration within reasonable time and accuracy. The ATE before configuration was 0.0538 m and after adaptation was 0.11 m.

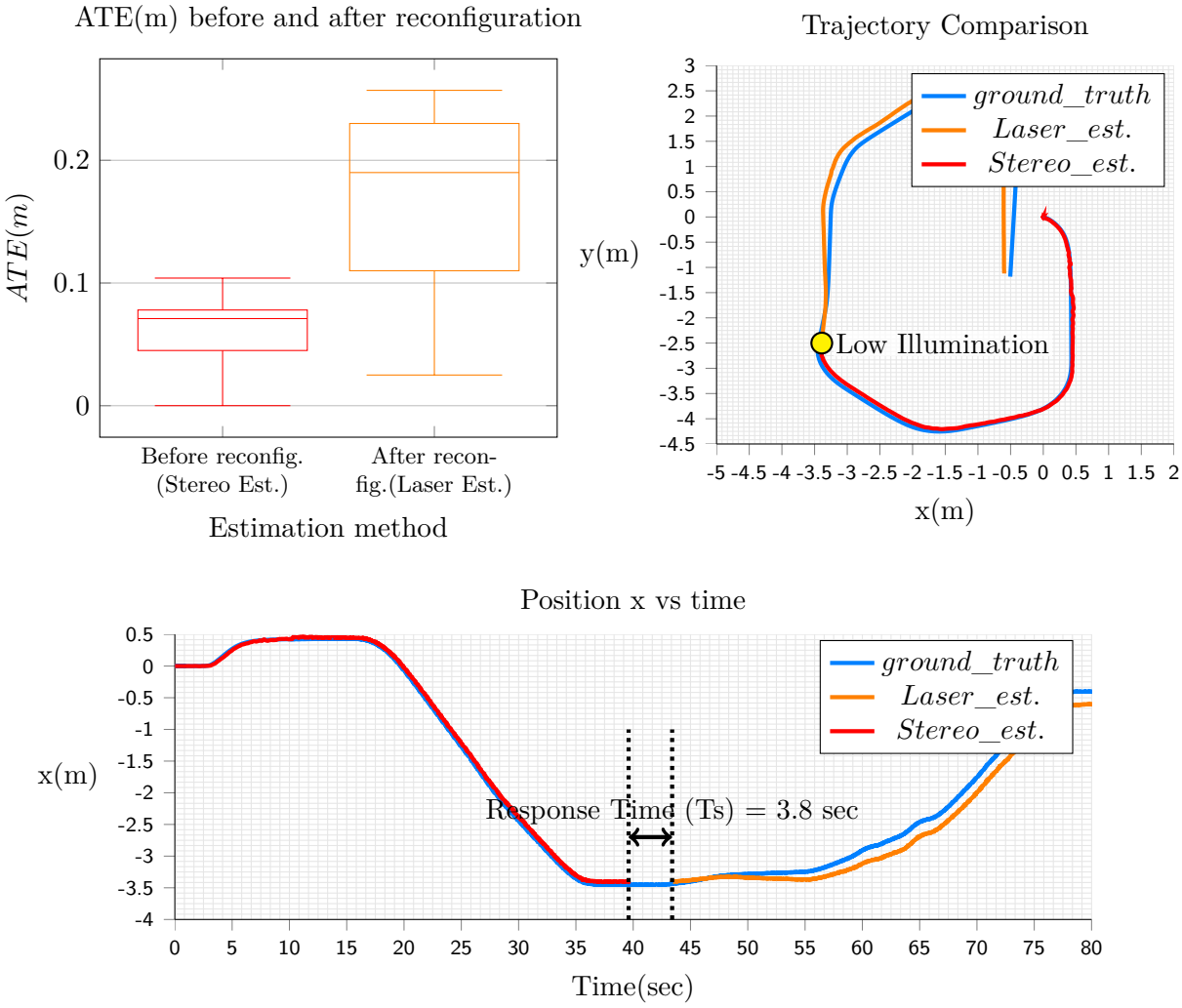

Figure 6-5: Low illumination affects the visual localisation ability of the robot which is solved by performing runtime adaptation by the robot. The yellow circle is the moment at which the illumination was changed manually

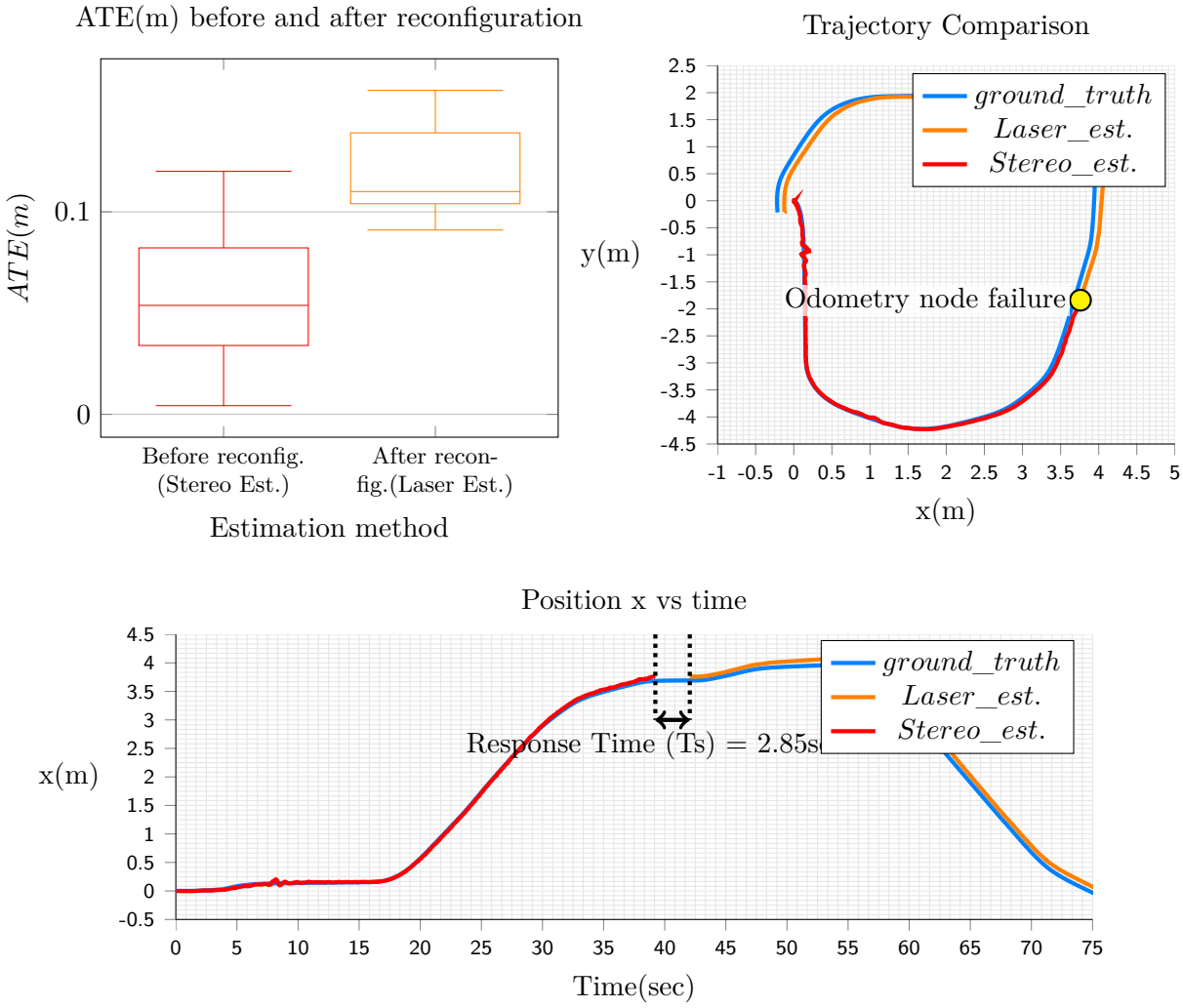

Figure 6-6: The failure of stereo odometry node cause the vision based localisation to fail which is detected by the robot and reconfigures its delft to laser based localisation. The yellow circle is the moment at which the node was switched off

6-4 Results of T3: Reusability

This test was carried to check the reusability of the rules for which we decided to add a new environmental constraint about the visual information to the stereo odometry estimation node. To do this, we created a monitoring node which acts as a client node and sent the visual information status to the monitoring manager node. On the data instantiation side, we created 2 concepts for `visual information` property and `visual information monitor` and one property observer relationship between them. The property was set to soft constraint and a new role player was added to the stereo odometry estimation node. So in total 2 entities were added, one relation was added and 1 relation was modified. No modification was made to any of the rules. The sequence of activity in the system proceeds as follows. While the robot is in motion and performing localisation, the robot was made to enter a region which had a featureless wall. This would affect the vision based localisation performance because the vision based localisation is not able to preform in a featureless environment. The monitor which is created by the metric mentioned in section 5-2-3 detects the low visual information and the pose monitor detects the high covariance in the pose estimates. This will cause the premises of **rule R3** to be satisfied due to which **rule R4 and R6** will also be satisfied. And the same process will repeat again as the previous scenarios.

Figure 6-7 shows the response of the SASA localisation system in featureless environment. Figure 6-7.b displays the comparison between the ground truth and the pose estimation. Note that the featureless environment has caused the configuration to be changed during runtime from vision based to laser based. The time SASA framework was able to respond in is 3.72 sec. In this scenario the SASA framework was able to adapt and reconfigure successfully to a new configuration within reasonable time and accuracy. The ATE before configuration was 0.039 m and after adaptation was 0.072 m.

6-5 Results of T4: Changeability

This test was carried to check the changeability of the SASA framework. We decided to add a new component to the robot which is a flashlight using which the robot could perform vision based localisation even in a low illumination environment. After adding this new component, the system will not need to change its configuration to laser even during low illumination. On the data instantiation side we created 1 new `flashlight` entity and property observer relation for illumination property was modified with a new role player. The modified relation can be seen in Figure 6-9. No modification was made to the rules. So in total 1 entity was added and 1 relation was modified.

The sequence of activity in the system proceeds as follows. While the robot is in motion and performing localisation, the illumination was changed manually. This illumination change causes low brightness and high co-variance in the position estimates. High pose co-variance and the low environmental illumination is detected by the monitoring nodes which updates the current states to the knowledge graph. This will cause the premises of **rule R3** to be satisfied due to which **rule R4 and R6** will also be satisfied. This will cause the stereo odometry node and vision based localisation node to change its status to unhealthy. Now the planning node which is querying the knowledge graph and should ideally get *NULL* values, will get some action recommendation due to satisfaction of **rule R9**. Now as a new compensator is

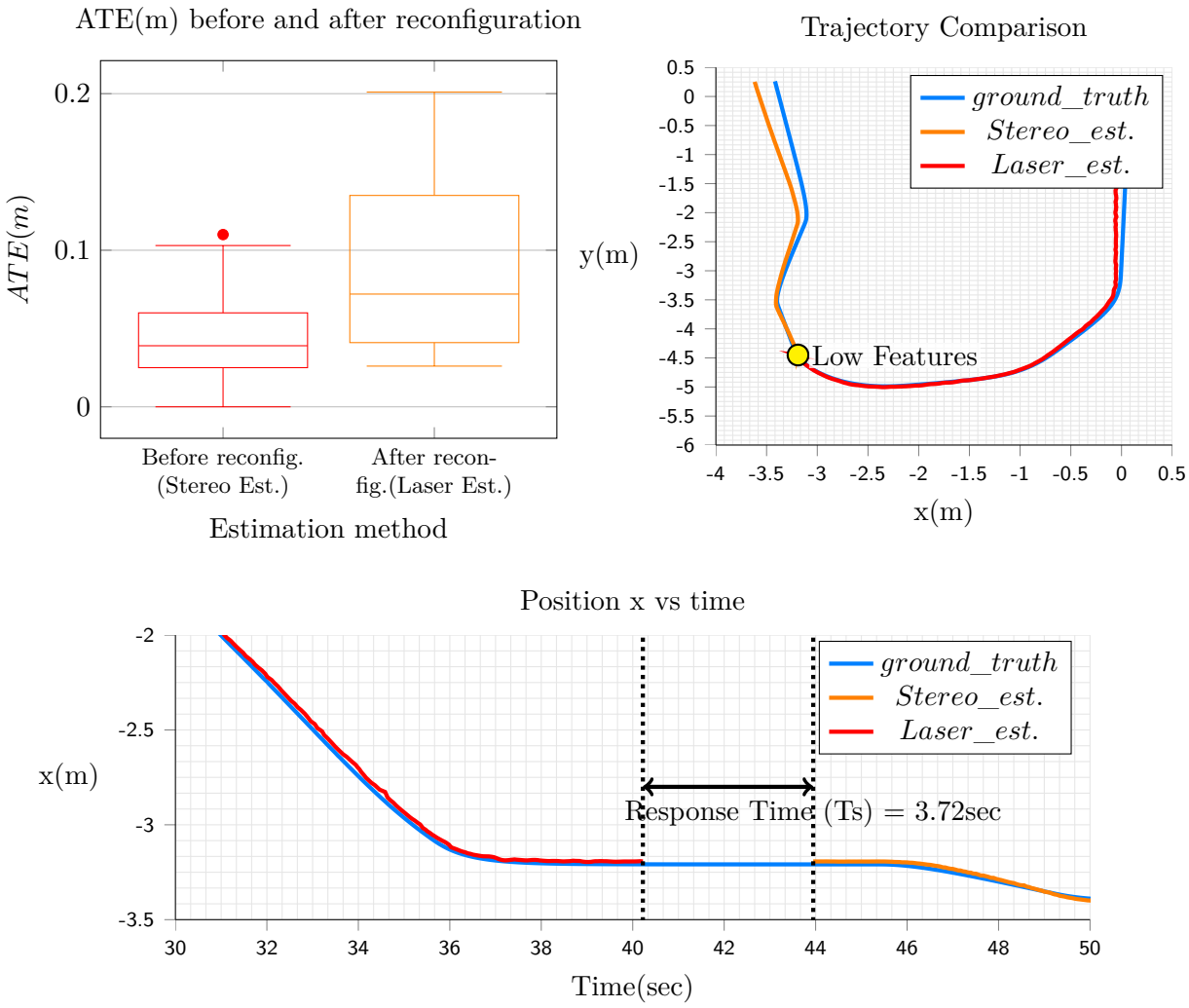

Figure 6-7: The featureless environment affects the functionality of vision based localisation. This is detected by the robot which adapts to the situation. The yellow circle is the moment at which the robot enters the region of low features.

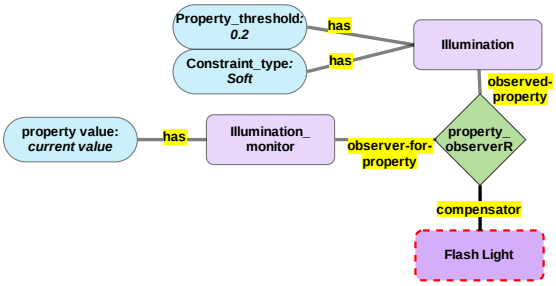

Figure 6-8: Compensator added for illumination

added to the illumination property the query Q2 will be answered with the recommendation to activate the compensator. Hence the conditional statement $isCompensatorQ2?activateC : (Q3 \rightarrow Q4)$ is satisfied and the system won't query Q3. This will cause the activation of the component and the robot will switch on its flashlight.

Figure 6-9 shows the response of the SASA localisation system in low illuminated environment. Figure 6-9.b displays the comparison between the ground truth and the pose estimation. Note that the low illuminated environment has made the robot use its component to perform localisation functionality. The SASA framework was able to respond in 2.02 sec. In this scenario the SASA framework was able to adapt and reconfigure successfully to a new configuration within reasonable time and accuracy. The ATE before configuration was 0.055 m and after adaptation was 0.068 m.

6-6 Results of T5: Consistency

This test was carried to check the consistency of the SASA framework. In this experiment we simulated the same situation 15 times at different locations in the retail store to check whether the behaviour of the framework is consistent. So the situation for the previous test i.e. T4 was repeated 15 times. Figure 6-10 shows the response of the SASA localisation system for the same situation. Here the time means as follows

Tl: Latency: It is defined as the total time taken by the monitor element to detect an event after occurrence of an event.

Tr: Reconfiguration time: It is defined as the total time taken by the automatic manager to provide some action recommendation after detecting an event.

Ta: Action time: It is defined as the total time taken by the system to perform an action after the action is recommended by the automatic manager.

Ts: Response time: It can be defined as the total time taken by the system to overcome a situation after the occurrence of that situation. Response time is sum of the latency, reconfiguration time and action time.

Note here *Ta* depends on the action which is executed and hence it will be different for different actions. Figure 6-11 displays the distribution of time for test performed. For all the tests, the system took on average 1.052 sec to detect the environmental state once it occurred

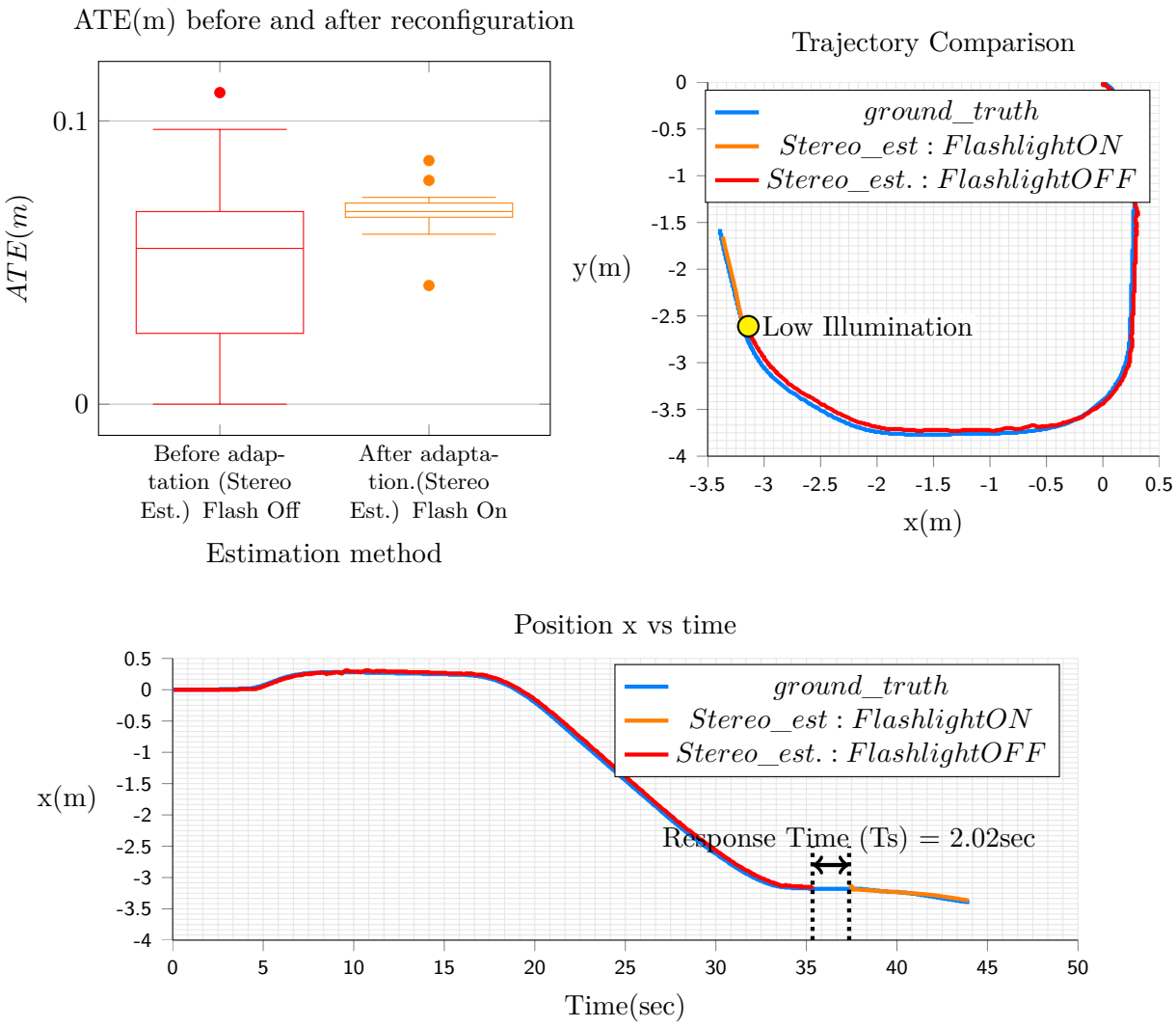

Figure 6-9: The low illuminated environment affects the ability of the robot to perform vision based pose estimation. This is detect by the robot and it preforms reconfiguration. This robot has a component which can alter the illumination. Hence it will use the component instead of searching for alternative configuration

and 0.664 sec to analyse the situation and provide action recommendation. For all the test the system was able to overcome the environmental disturbance situation on an average of 2.3465sec. The system was consistent with the behaviour and showed same behaviour every time with a consistency rate of 100%.

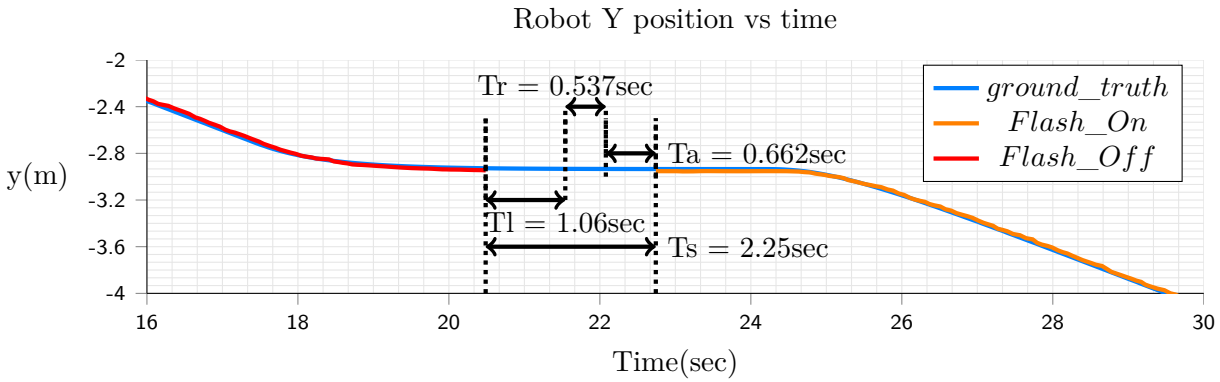

Figure 6-10: Ts:Response Time, Tr:Reconfiguration time, Ta: Action Execution Time, Tl: Latency for 1 of the 15 test performed using test T4 scenario.

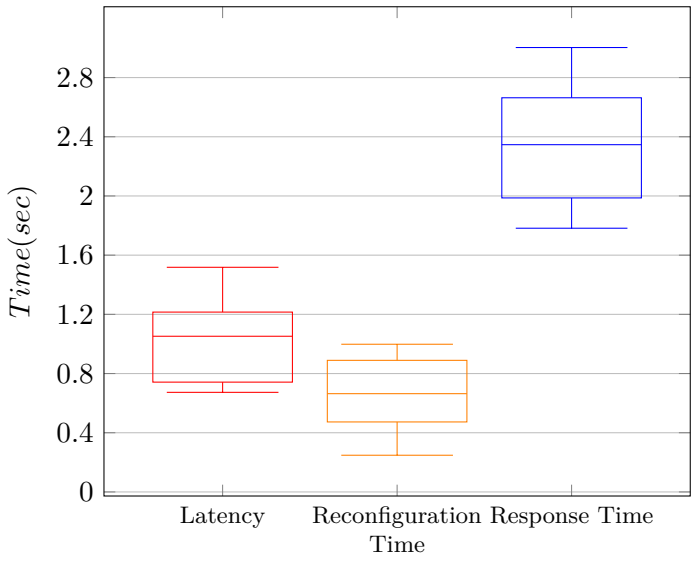

Figure 6-11: Latency, Reconfiguration Time and response time for adaptation for 15 test performed using test T4 scenario. The system performed adaption ever time which make it 100% consistent

6-7 Analysis

Experiments were performed to validate the behaviour of SASA localisation framework in different situations. The localisation framework was able to cope with all the situations given

to it and provided reliable and reasonable response in all the scenarios. Being a rule based system it was possible to predict the behaviour of the system in all the scenarios and the system behaved as per the predictions. The framework made it easy to change or modify different components in the system with ease. There was no need to add new rules for any of the scenario. No modification were made even to schema concepts. Just new instantiation was made according to the added physical components. Table 6-2 summarises the modification made to the system for different scenarios. The modifications made to the system is difficult to express in terms of a quantity. However, we will use a software metric which is a standard of measure of a degree to which a software system or process possesses some property. One commonly used metric is *Logical Lines of code (LLOC)*[1]. To give an example, grounding one new concept from the knowledge needs adding 1 LLOC to the Graql file. So adding new knowledge to the SASA is not costly. On the ROS system side adding one monitoring node will cost only one script which will consist of property monitoring algorithm and 2 Graql queries one for updating the property states. So adding new components to the SASA framework on adaptation layer is not costly and would hardly take any time.

Table 6-2: Summary of the changes made to the knowledge w.r.t. different experiments

	Initial System	New Property Constraint T3	New Compensator T4
Number of changes made to rules	N/A	0	0
Number of changes schema	N/A	0	0
Modification made to classes and relation data	N/A	+2 entity +1 relation 1 relation modifed	+1 entity 1 relation modified
Cost for adding new knowledge	N/A	+2 LLOC for 2 entity +1 LLOC for 1 new relation	+1 LLOC for 1 entity
Total classes and relations	19 entities 10 relation	19 + 2 entities 10+1 relation	21 + 1 entities 11 relation
Changes made to ROS system	N/A	+1 monitoring client	+1 ROS message for compensation action
Cost for adding new ROS component	N/A	+1 ROS Action client node. The action client will contain the algorithm for monitoring and two graql queries. One for deleting the old states and one for adding current states.	+1 ROS topic publisher which will publish the action to activate the component

Table 6-3 summarises the response of the conventional localisation pipeline with the novel SASA localisation system. In normal situations, the SASA framework provided comparable performance with the other conventional framework. The SASA localisation system was able to handle different situation which the laser-based pose estimation framework or the

[1]LLOC measures the number of executable statements

vision-based pose estimation could not handle. One would argue that the laser based pose estimation showed better results than the SASA localisation framework in almost all cases except internal failure. This was because, the situations tested were the limitations of vision based pose estimation method and the situations where the laser based pose estimation could fail such as active interference between different sensors, transparent and reflective surfaces were not tested due to the time constraints. In such scenarios the laser based pose estimation could have failed. For the experiments performed and their results we can make a statement that the developed SASA localisation framework could provide benefits over conventional localisation pipeline. One could also argue that the adaptation logic could be hard-coded in the localisation pipeline. But hard-coded logic will only take into account the scenarios considered while designing the logic. Adding new scenarios or new components to the localisation system would make the hard-coded logic useless and new rules and logic would be required which is time consuming and cumbersome process. The SASA localisation framework provided addition and modification to the localisation pipeline without any change to adaptation logic. Just modification made to the system should be added to the knowledge following the schema structure and the SASA framework would take that into consideration during adaptation. Hence the SASA localisation framework provided the robot with the ability of self-awareness and self-adaptation which can handle different situations on its own.

Table 6-3: Comparison of SASA localisation framework with Vision-based pose estimation and laser-based pose estimation localisation in different scenarios for the path length on 20 m

	Vision-localisation ATE(m)	Laser-localisation ATE(m)	SASA localisation framework ATE(m)
Normal Scenario	0.03	0.05	0.055
Featureless Environment	Fail	0.0447	0.072
Dark Environment	Fail	0.051	0.068
Internal node Failure	Fail	Fail	0.11

The localisation system developed and the experimental scenarios described throughout this chapter, has demonstrated the validity of the approach for the designing self-awareness and adaptive properties in localisation system. The designed MAPE-K loop developed for the localisation system has provided the capability to detect environmental uncertainties which affect the localisation performance, malfunctioning of internal components and overcome it by adapting through reconfiguration. Regarding the online operation the SASA localisation system demonstrates successful adaptation in reasonable time for various scenarios regarding environmental uncertainty and internal component failure which provides the system with fault-tolerance capabilities exploiting system redundancy. The system also demonstrates architectural redesign using automated deductive reasoning during run-time.

Chapter 7

Conclusion

This conclusive chapter summarises the content presented in this work, and provides answers to the initially posed research questions. Besides, some guidelines for future work are also mentioned.

7-1 Summary

In this work we proposed a Situation-Aware Self-Adaptive (SASA) localisation framework based on Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) reference model. The starting point was the analysis of the mobile robot localisation problem and the limitations of the current state-of-the-art localisation framework in the direction of fault tolerance and environmental uncertainty. This motivated us to look in the direction of fault tolerant control systems. However, classical fault tolerant controllers were difficult to apply for our use case due to their requirement of a mathematical model of the process and their inability to handle qualitative uncertainties. The resuability of the classical fault tolerance controllers was also a concern which motivated us to look for approaches which can embed the fault tolerance capability within the system intrinsically. While exploring relevant literature, we came across MAPE-K feedback loop for Self Adaptive System (SAS) which lead to the origin of SASA localisation framework. The SASA framework could provide the conventional localisation system with the added advantage of situational awareness and self adaptation. This framework was indented to use the system's knowledge to provide run-time reconfiguration when any unexpected situation occurs without the added cost of developing fault models of the system. Chapter 4 and chapter 5 discusses the development process of the SASA localisation framework. In chapter 6 we validated the developed SASA framework on different scenarios such as environment disturbances and internal failures. We also demonstrated the reusability and low modification cost of the framework. We also compared the SASA localisation framework with the state-of-the-art conventional localisation framework available in literature. The SASA framework showed comparable performance with the other conventional framework and outperformed them in case of failures.

7-2 The answers to the research questions

The research goal of this thesis was to develop a self-adaptive situational aware framework for mobile robot localisation application. At the time when this research was conducted there was no self-adaptive framework which was focusing explicitly on mobile robot localisation use case to the best of our knowledge. Also there was no method available which took into account the current situation and the effects of environmental disturbance and internal failure, on the localisation ability up to our knowledge. The outcome of this work led to the following answers to the initially posed research questions.

Can we enhance the localisation capability of a robotic system using situational-awareness and self-adaptation?

The functional architecture of the system developed in Figure 5-8 could be used to embed the self-adaptation capability in the robot. Also the monitors developed in Section 5-2 along with general rules developed in Table 5-1 and Table 5-2 provide the system with situational awareness capability. These two capabilities made the robot localisation more reliable without affecting the normal operation of the localisation system. It even provided functionality in a situation which conventional localisation system could not handle. Hence we can state that the localisation capability of a robotic system was enhanced using situational-awareness and self-adaptation capability.

Can explicit knowledge representation and automated reasoning about the robot's internal components be exploited to obtain an intrinsically fault tolerant and reliable systems?

Many advantages regarding Knowledge Representation and Reasoning (KR&R) emerged during this study. The knowledge in the form of schema explained in Section 5-1 and the rules developed in Table 5-1 to perform automated reasoning showed that using KR&R, the robot was able to cope with the internal faults and even environmental disturbances. The recovery was performed by the framework on its own without causing the system to enter the state of failure. Overall, the knowledge model and the functional architecture in Figure 5-8 can be seen as an intrinsically fault tolerant scheme, which facilitates the detection, isolation and recovery from internal faults and environmental disturbances. Hence we can say that explicit knowledge representation about the robot's internal components and automated reasoning made the system intrinsically fault tolerant and more reliable.

To conclude, the SASA localisation framework provided a general and reusable schema and rules which allows addition of more constraints about the environmental property and allows modification of the internal components with ease and without worrying about the adaptation logic. Due to its symbolic nature, it also provides explanation about the violated constraints and how those affect the rest of the system. Lastly, the developed schema and rules are general and not limited to localisation use case thereby making the framework scalable and applicable to other Robot Operating System (ROS) based systems.

7-3 Future work

The framework developed in this thesis was able to handle all the situations for which it was tested i.e. 'Environmental Uncertainty' due to poor illumination, 'Internal fault' due

to component failure, 'Environmental Uncertainty' due to featureless environment, 'System modifications' during robot life-cycle, and 'Consistency' of the designed framework. However, there are many ideas for future developments and improvements.

The proposed framework was tested in a simulation environment. However it would be interesting to see how the framework performs in real world environment. According to us there won't be any problem to transfer the framework on a real robot and the performance would be dependent on how closely the developed monitors are able to detect the situations.

Another interesting direction for further development would be to add rules using fuzzy logic. In our system the rules were developed using boolean logic i.e. something is either true or false. However rules developed using fuzzy logic would provide multi-valued logic in the region close to the thresholds.

One more addition to the SASA framework would be to add a metric of resource utilisation to the schema where the knowledge would be updated during run-time operation about the computational load and the battery utilised by a particular configuration. This knowledge could be used in a situation when the battery is low or computation load is high.

Appendix A

Definitions

Functional requirements: A functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs [89].

Non-Functional requirements (NFR): NFR are the requirements that specify a criteria that can be used to judge the development or operation of a system rather than specific behaviors.

Reusability: “The degree to which an asset can be used in more than one software system, or in building other assets” [90] p. 307

Modifiability: A system’s modifiability is defined as the system’s receptiveness to change

Reliability: “The ability of a system or component to perform its required functions under stated conditions for a specified period of time” [90] p.297.

Resilience: "The ability to adapt to changing conditions and prepare for, withstand, and rapidly recover from disruption" [91]

Adaptability: "Degree to which a product or system can be effectively and efficiently adapted for different or evolving hardware, software or other operational or usage environments” [92]

Survivability: "The degree to which a product or system continues to fulfil its mission by providing essential services in a timely manner in spite of the presence of attacks. cf. recoverability".

Resource utilisation: It is the ability of the system to use appropriate amounts and types of resources when the system performs its mission functions under stated conditions.

Latency: It is defined as the total time taken by the monitor element to detect an event after occurrence of an event.

Reconfiguration time: It is defined as the total time taken by the automatic manager to provide some action recommendation after detecting an event.

Action time: It is defined as the total time taken by the system to perform an action after the action is recommended by the automatic manager.

Response time: It can be defined as the total time taken by the system to overcome a situation after the occurrence of that situation. Response time is sum of the latency, reconfiguration time and action time.

A-1 Definitions of schema concepts

This section gives definitions for the concepts used in the Situation-Aware Self-Adaptive (SASA) localisation framework.

Robot Component: A physical, software or communication element that can be part of a robotic system and that fulfils certain functions that is relevant for one or more configurations.

Robot Component Setting: Concept to represent the parameter setting of the component

Functionality: The purpose a component can serve when used.

Process Input: Concept that is used to model the inputs of the process.

Process Output: Concept that is used to model the outputs of the process.

Property: Anything which can affect or limit the functionality of the component.

Status: It describes the health of a particular component.

Communication component: It is a concept which can be used to represent the elements which are responsible for the communication between different components.

Hardware component: It is a concept which can be used to represent the hardware elements present in the system.

Software component: It is a concept which can be used to represent the algorithms or processes which are responsible for the controlling or processing in a system.

Monitoring component: It is a concept which can be used to represent the components which are responsible to observe the state of a system or the environment.

Property Compensator: It is a concept which can be used to represent the components which are alter or affect the constraints on a component.

ROS Message: Concept to represent the communication between different components within a ROS system.

Healthy: Entity that represents the state of the component when ever constraint is satisfied.

Unreliable: Entity that represents the state of the component when only soft constraints are violated.

Unhealthy: Entity that represents the state of the component when any of the hard constraints are violated.

ROS processing node: Relationship which processes some inputs and produces outputs in a Robot Operating System (ROS) based system .

Property observer relation: Relationship which connects together the property and the monitor which is observing that property.

Functionality relation: Relationship which connects together the component concept and the functionality which that component provides.

Configuration hierarchy relation: Relationship which connects set of components connected through input/output that together provide some functionality.

Appendix B

Results of developed visual perception monitors on TID2008 dataset

Figure B-1: TDI2008 Dataset Change in intensity

Figure B-2: TDI2008 Dataset Change in blur

Figure B-3: TDI2008 Dataset Change in gaussian noise

Figure B-4: TDI2008 Dataset Change in intensity

Figure B-5: TDI2008 Dataset Change in blur

Figure B-6: TDI2008 Dataset Change in gaussian noise

Figure B-7: TDI2008 Dataset Change in intensity

Figure B-8: TDI2008 Dataset Change in blur

Figure B-9: TDI2008 Dataset Change in gaussian noise

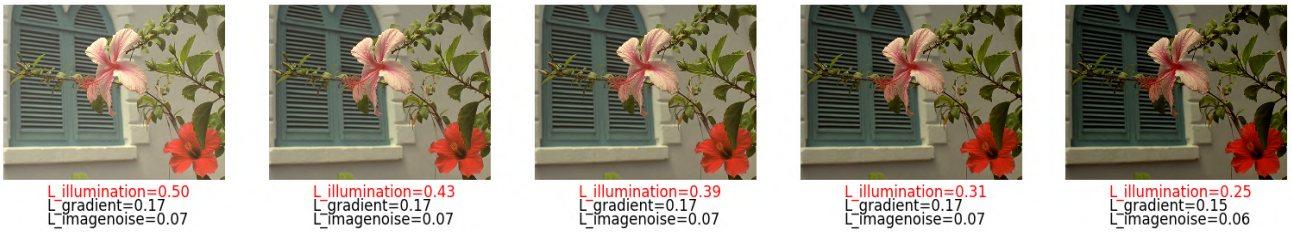

Figure B-10: TDI2008 Dataset Change in intensity

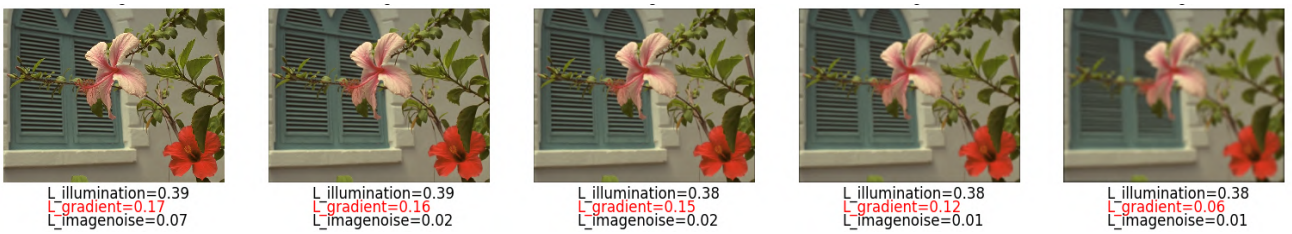

Figure B-11: TDI2008 Dataset Change in blur

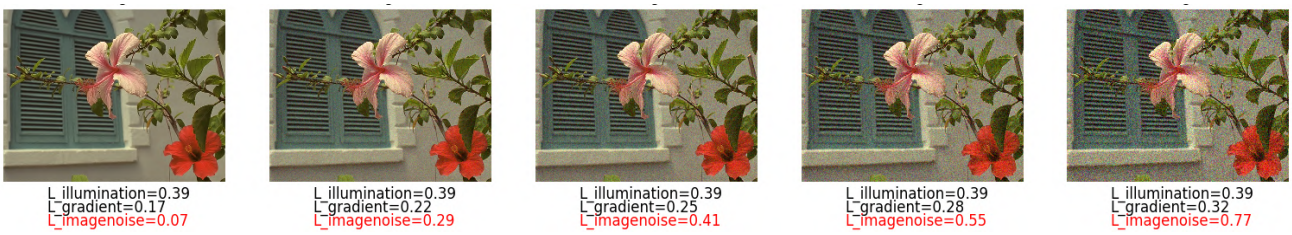

Figure B-12: TDI2008 Dataset Change in gaussian noise

Bibliography

[1] J. Eom, G. Kim, and Y. Park, "Mutual interference potential and impact of scanning lidar according to the relevant vehicle applications," in *Laser Radar Technology and Applications XXIV*, vol. 11005, p. 110050I, International Society for Optics and Photonics, 2019.

[2] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder, *Diagnosis and fault-tolerant control*, vol. 2. Springer, 2006.

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[4] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[5] B. M. da Silva, R. S. Xavier, T. P. do Nascimento, and L. M. Gonsalves, "Experimental evaluation of ros compatible slam algorithms for rgb-d sensors," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pp. 1–6, IEEE, 2017.

[6] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[7] "Rooboma-the cleaning robot," Last accessed on 20 January 2020. https://www.irobot.nl/roomba.

[8] M. Diab, A. Akbari, M. Ud Din, and J. Rosell, "Pmk—a knowledge processing framework for autonomous robotics perception and manipulation," *Sensors*, vol. 19, no. 5, p. 1166, 2019.

[9] P. LeBeau, "Waymo starts commercial ride-share service," 2018.

[10] M. Tenorth and M. Beetz, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[11] S. Wang and H. I. Christensen, "Tritonbot: First lessons learned from deployment of a long-term autonomy tour guide robot," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 158–165, IEEE, 2018.

[12] R. R. Murphy, S. Tadokoro, and A. Kleiner, "Disaster robotics," *Springer Handbook of Robotics, chapter 60, pp. 1577–1604,B. Siciliano and O. Khatib, eds.*, 2016.

[13] M. Tenorth and M. Beetz, "Knowrob—knowledge processing for autonomous personal robots," in *2009 IEEE/RSJ international conference on intelligent robots and systems*, pp. 4261–4266, IEEE, 2009.

[14] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[15] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[16] G. A. Demetriou, "A Survey of Sensors for Localization of Unmanned Ground Vehicles (UGVs)," *Frederick Institute of Technology*, 2006.

[17] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," . *SpringerPlus (2016) 5: 1897*, 2016.

[18] Z. Kong, "A brief review of simultaneous localization and mapping," *IEEE Conference Proceedings: IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Oct. 2017.

[19] P. Skrzypczyński, "Mobile robot localization: Where we are and what are the challenges?," in *AUTOMATION*, 2017.

[20] T. S. Ho, C. F. Yeong, and E. L. M. Su, "Simultaneous localization and mapping survey based on filtering techniques," *2015 10th Asian Control Conference (ASCC)*, pp. 1–6, 2015.

[21] A. Mohamed, J. Ren, M. El-Gindy, H. Lang, and A. Ouda, "Literature survey for autonomous vehicles: sensor fusion, computer vision, system identification and fault tolerance," *International Journal of Automation and Control (IJAAC), Vol. 12, No. 4*, 2018.

[22] V. Imani, K. Haataja, and P. Toivanen, "Three main paradigms of simultaneous localization and mapping (slam) problem," *Tenth International Conference on Machine Vision (ICMV 2017)*, 2018.

[23] C. Wu, T. A. Huang, M. Muffert, T. Schwarz, and J. Grater, "Precise pose graph localization with sparse point and lane features," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4077–4082, 2017.

[24] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7244–7251, IEEE, 2018.

[25] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*, pp. 155–160, IEEE, 2011.

[26] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[27] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2661–2666, IEEE, 2014.

[28] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[29] P. Kim, B. Coltin, O. Alexandrov, and H. J. Kim, "Robust visual localization in changing lighting conditions," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5447–5452, IEEE, 2017.

[30] U. Shin, J. Park, G. Shim, F. Rameau, and I. S. Kweon, "Camera exposure control for robust robot vision with noise-aware image quality assessment," *arXiv preprint arXiv:1907.12646*, 2019.

[31] C. Hernández, J. Bermejo-Alonso, and R. Sanz, "A self-adaptation framework based on functional knowledge for augmented autonomy in robots," *Integrated Computer-Aided Engineering*, vol. 25, no. 2, pp. 157–172, 2018.

[32] A. G. Kashani, M. J. Olsen, C. E. Parrish, and N. Wilson, "A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, vol. 15, no. 11, pp. 28099–28128, 2015.

[33] A. L. Diehm, M. Hammer, M. Hebel, and M. Arens, "Mitigation of crosstalk effects in multi-lidar configurations," in *Electro-Optical Remote Sensing XII*, vol. 10796, p. 1079604, International Society for Optics and Photonics, 2018.

[34] L. Drolet, F. Michaud, and J. Côté, "Adaptable sensor fusion using multiple kalman filters," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 2, pp. 1434–1439, IEEE, 2000.

[35] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The international journal of robotics research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[36] I. Shim, T.-H. Oh, J.-Y. Lee, J. Choi, D.-G. Choi, and I. S. Kweon, "Gradient-based camera exposure control for outdoor mobile platforms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1569–1583, 2018.

[37] R. Koch, S. May, P. Murmann, and A. Nüchter, "Identification of transparent and specular reflective material in laser scans to discriminate affected measurements for faultless robotic slam," *Robotics and Autonomous Systems*, vol. 87, pp. 296–312, 2017.

[38] Y. Zhao, T. Li, X. Zhang, and C. Zhang, "Artificial intelligence-based fault detection and diagnosis methods for building energy systems: Advantages, challenges and the future," *Renewable and Sustainable Energy Reviews*, vol. 109, pp. 85–101, 2019.

[39] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part iii: Process history based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[40] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[41] R. Isermann, *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2006.

[42] C. H. Corbato, *Model-based self-awareness patterns for autonomy.* PhD thesis, Universidad Politécnica de Madrid, 2013.

[43] C. Krupitzer, M. Breitbach, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems (extended version)," 2018.

[44] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 3, pp. 54–62, 1999.

[45] G. Burroughes, *Reconfigurable autonomy for future planetary rovers.* PhD thesis, University of Surrey, 2017.

[46] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[47] B. Liu, J. Wu, L. Yao, and Z. Ding, "Ontology-based fault diagnosis: a decade in review," in *Proceedings of the 11th International Conference on Computer Modeling and Simulation*, pp. 112–116, 2019.

[48] S. Lemaignan, R. Ros, M. Mosenlechner, R. Alami, and M. Beetz, "ORO, a knowledge management platform for cognitive architectures in robotics," *Conference: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[49] M. Tenorth and M. Beetz, " KnowRob — knowledge processing for autonomous personal robots," *Proc. of the 2009 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, p. 4261–4266, 2009.

[50] X. Li, S. Bilbao, T. Martin-Wanton, J. Bastos, and J. Rodriguez, "Swarms ontology: A common information model for the cooperation of underwater robots," *Sensors*, 2017.

[51] M. Diab, A. Akbari, M. U. Din, and J. Rosell, "PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation," *Sensors (Basel)*, vol. Volume 19, Issue 5, 2019.

[52] M. A. Cornejo-Lupa, R. P. Ticona-Herrera, Y. Cardinale, and D. Barrios-Aranibar, "A survey of ontologies for simultaneous localization and mapping in mobile robots," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–26, 2020.

[53] A. Olivares-Alarcos, D. Beßler, A. Khamis, P. Goncalves, M. K. Habib, J. Bermejo-Alonso, M. Barreto, M. Diab, J. Rosell, J. Quintas, *et al.*, "A review and comparison of ontology-based approaches to robot autonomy," *The Knowledge Engineering Review*, vol. 34, 2019.

[54] E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia, "Integrating a bottom–up and top–down methodology for building semantic resources for the multilingual legal domain," in *Semantic Processing of Legal Texts*, pp. 95–121, Springer, 2010.

[55] M. El Ghosh, H. Naja, H. Abdulrab, and M. Khalil, "Towards a middle-out approach for building legal domain reference ontology," 2016.

[56] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *The knowledge engineering review*, vol. 11, no. 2, pp. 93–136, 1996.

[57] J. I. Olszewska, M. Barreto, J. Bermejo-Alonso, J. Carbonera, A. Chibani, S. Fiorini, P. Goncalves, M. Habib, A. Khamis, A. Olivares, E. P. de Freitas, E. Prestes, S. V. Ragavan, S. Redfield, R. Sanz, B. Spencer, and H. Li, "Ontology for autonomous robotics," *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, p. 189–194, 2017.

[58] M. Fernandez, A. Gomez-Prez, and N. Juristo, "METHONTOLOGY: From ontological art towards ontological engineering," *AAAI Spring Symposium on Ontological Engineering*, pp. 33–40, 1997.

[59] D. Kalibatiene and O. Vasilecas, "Survey on Ontology Languages," *10th International Conference in Business Information Processing*, 2011.

[60] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, "Book on Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web," *Springer Verlag*, 2004.

[61] "Knowledge base of relational and nosql database management systems, solid it," last accessed on 9 October 2020. https://db-engines.com/en/.

[62] "Comparing grakn to semantic web standards," last accessed on 9 October 2020. https://dev.grakn.ai/docs/comparisons/semantic-web-and-grakn.

[63] J. L. Fernandez and C. Hernandez, *Practical model-based systems engineering*. Artech House, 2019.

[64] O. S. R. F. (OSRF), ""gazebo". gazebo simulator," last accessed on 22 October 2020. http://gazebosim.org/.

[65] D. Davis *et al.*, "Smc systems engineering primer & handbook," *United States Air Force Space & Missile Systems Center*, pp. 13–17, 2005.

[66] W. Garage, "Robot operating system," 2011. https://www.ros.org/.

[67] P. Schaer, J. Skaloud, S. Landtwing, and K. Legat, "Accuracy estimation for laser point cloud including scanning geometry," in *Mobile Mapping Symposium 2007, Padova*, no. CONF, 2007.

[68] I. Bogoslavskyi and C. Stachniss, "Analyzing the quality of matched 3d point clouds of objects," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6685–6690, IEEE, 2017.

[69] C. M. Wang, "Location estimation and uncertainty analysis for mobile robots," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 1231–1235, IEEE, 1988.

[70] N. Doh, H. Choset, and W. K. Chung, "Accurate relative localization using odometry," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2, pp. 1606–1612, IEEE, 2003.

[71] "Color conversions opencv," https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html.

[72] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[73] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[74] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," *a talk at the Stanford Artificial Project in*, pp. 271–272, 1968.

[75] L. Cao, J. Ling, and X. Xiao, "Study on the influence of image noise on monocular feature-based visual slam based on ffdnet," *Sensors*, vol. 20, no. 17, p. 4922, 2020.

[76] G. Chen, F. Zhu, and P. Ann Heng, "An efficient statistical method for image noise level estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 477–485, 2015.

[77] S. Pyatykh, J. Hesser, and L. Zheng, "Image noise level estimation by principal component analysis," *IEEE transactions on image processing*, vol. 22, no. 2, pp. 687–699, 2012.

[78] S.-C. Tai and S.-M. Yang, "A fast method for image noise estimation using laplacian operator and adaptive edge detection," in *2008 3rd International Symposium on Communications, Control and Signal Processing*, pp. 1077–1081, IEEE, 2008.

[79] J. Immerkaer, "Fast noise variance estimation, computer vision and image understanding," 1996.

[80] I. Shim, J.-Y. Lee, and I. S. Kweon, "Auto-adjusting camera exposure for outdoor robotics using gradient information," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1011–1017, IEEE, 2014.

[81] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[82] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti, "Tid2008-a database for evaluation of full-reference visual quality assessment metrics," *Advances of Modern Radioelectronics*, vol. 10, no. 4, pp. 30–45, 2009.

[83] "Deductive reasoning„" last accessed on 27 October 2020. https://en.wikipedia.org/wiki/Deductive_reasoning.

[84] S. Vredeveldt, "Semantic knowledge to assess the capabilities of auvs for planning," 2019.

[85] E. Bouillet, M. Feblowitz, Z. Liu, A. Ranganathan, and A. Riabov, "A knowledge engineering and planning framework based on owl ontologies," *Proceedings of the Second International Competition on Knowledge Engineering*, vol. 191, 2007.

[86] "Rtab map ros package," http://wiki.ros.org/rtabmap_ros.

[87] M. Jaimez, J. G. Monroy, and J. González-Jiménez, "Planar odometry from a radial laser scanner. a range flow-based approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4479–4485, 2016.

[88] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *The international Journal of robotics research*, vol. 22, no. 12, pp. 985–1003, 2003.

[89] R. Fulton and R. Vandermolen, *Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA/DO-254*. CRC Press, 2017.

[90] . I. IEE, "Ieee and iso/iec standard 24765: Systems and software engineering vocabulary.," *New York and Geneva: Institute of Electrical and Electronics Engineers and the International Organization for Standardization and the International Electrotechnical Commission*, 2010.

[91] W. House, "National security strategy of the united states. washington, dc: The white house," 2010.

[92] K. M. Adams, "Introduction to non-functional requirements," in *Nonfunctional Requirements in Systems Analysis and Design*, pp. 45–72, Springer, 2015.

Glossary

List of Acronyms

HOG	Histogram of oriented gradients
SIFT	Scale-invariant feature transform
SAS	Self Adaptive System
PMK	Perception and Manipulation Knowledge
NFR	Non-Functional requirements
ROS	Robot Operating System
MTBF	Mean Time Between Failures
UGV	Unmanned Ground Vehicle
ISE&PPOOA	Integrated Systems Engineering and Pipelines of Processes in Object-Oriented Architectures
MAPE-K	Monitor-Analyze-Plan-Execute over a shared Knowledge
FOV	Field of View
SysML	System Modelling Language
AMCL	Adaptive Monte Carlo Localization
PSNR	Peak signal-to-noise ratio
FSIM	Feature similarity index
CCD	Charge coupled device
CMOS	Complementary metal oxide semiconductor
F2F	Feature to Feature
F2M	Feature to Map
EKF	Extended Kalman Filter
KF	Kalman Filter
FTCS	Fault Tolerant Control Systems
FTC	Fault Tolerant Control

FTS	Fault Tolerant System
MTTF	Mean Time To Failure
MTBF	Mean Time Between Failure
CAD	Computer-aided design
ANN	Artificial Neural Networks
AI	Artifical Intelligence
UGV	Unmanned Ground Vehicle
UGV	Unmanned Ground Vehicle
OWL	Web Ontology Language
ORO	Open Robot Ontology
KnowRob	Knowledge Processing Framework
SWARMs	Smart and Networking Underwater Robots in Cooperation Meshes
SWRL	Semantic Web Rule Language
PMK	Perception and Manipulation Knowledge
SPARQL	SPARQL Protocol And RDF Query Language
GPS	Global Positioning System
RTAB map	Real-Time Appearance-Based Mapping
PDDL	Planning Domain Definition Language
STRIPS	Stanford Research Institute Problem Solver
URDF	Universal Robotic Description Format
XML	Extensible Markup Language
SASA	Situation-Aware Self-Adaptive
IT	Information technology
KR&R	Knowledge Representation and Reasoning
ATE	Absolute Trajectory Error
LLOC	Logical Lines of code