REAL-TIME PATH-PLANNING USING DEPTH/INS SENSOR FUSION FOR LOCALIZATION

by

Fares Amin E Alkhawja

A Thesis Presented to the Faculty of the

American University of Sharjah

College of Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in

Mechatronics Engineering

Sharjah, United Arab Emirates

December 2019

Approval Signatures

We, the undersigned, approve the Master's Thesis of Fares Amin E Alkhawja

Thesis Title: Real-Time Path-Planning using Depth/INS Sensor Fusion for Localization.

Signature Date of Signature

(DD / MM / YY)

_____ _____

Dr. Lotfi Romdhane

Professor, Department of Mechanical Engineering

Thesis Advisor

_____ _____

Dr. Mohammad Jaradat

Professor, Department of Mechanical Engineering

Thesis Co-Advisor

_____ _____

Dr. Mamoun Abdel-Hafez

Professor, Department of Mechanical Engineering

Thesis Committee Member

_____ _____

Dr. Salam Dhou

Assistant Professor, Department of Computer Science and Engineering

Thesis Committee Member

_____ _____

Dr. Mohammad Jaradat

Director, Mechatronics Graduate Program

_____ _____

Dr. Lotfi Romdhane

Associate Dean for Graduate Affairs and Research, College of Engineering

_____ _____

Dr. Naif Darwish

Acting Dean, College of Engineering

_____ _____

Dr. Mohamed El-Tarhuni

Vice Provost for Graduate Studies

Acknowledgement

I would like to thank my advisors Dr. Lotfi Romdhane and Dr. Mohammad Jaradat for providing knowledge, guidance, support, and motivation throughout my research stages. I'm deeply beholden for their great assistance, worthy discussion and suggestions.

I would also like to thank the professors of the Mechatronics Engineering graduate program who taught me the master level courses with mighty teaching methods and skills. I really appreciate their dignified advices and motivation.

And finally, I would like to thank the American University of Sharjah for the opportunity to be a Graduate Teaching Assistant, as it greatly benefited me in this research both financially and intellectually.

Dedication

Firstly, I would like to thank God for this achievement,

I dedicate this achievement to my beloved mother and my dear father. This achievement is too small to repay their moral and material support for me. They stood beside and believed in me and made a tremendous effort until I finished this difficult journey that lasted for two years and had many obstacles. Additionally, I would like to dedicate this achievement to my brothers, my friends, and for everyone who stood beside me, either by encouragement or physical help, thank you from the bottom of my heart.

Father and mother: Here I am again, finishing another big step in my life, which would not exist without you two. I hope to always keep you proud of me.

My sincere brothers: Proud to be a source of inspiration for you, hope your achievements will be even greater than mine, and to always be able to help you in your lives.

Friends: I hope the best for you in your lives, and I hope you can make your dreams come true.

Koshish: you were a true friend and partner in the past two years. I have enjoyed being with you while facing all the difficulties that we had together

Abstract

This thesis discusses a hardware implementation of a navigating robot that plans its path and localizes itself in indoor and outdoor environments. It aims to enhance the process of navigation by enhancing the localization module of it. After feeding the cost-map along with the obstacles, path-planning

algorithm (A*) chooses the lowest cost until it reaches the destination. The path planning controller uses the dynamic state of the robot using different localization techniques for outdoor and indoor environments. The objective is to make the localization process more stable to improve the navigation. The enhanced localization technique uses the depth camera localization data in indoor environments and outdoor environments, to enhance the results obtained by the IMU raw data through a depth-inertial fusion. The fusion algorithm is based on feed-forward cascade correlation network, which is part of the neural networks and it is assessed using different sensors and different neural networks methods. The robot uses the localization data in addition to the planned path to control its movement towards the destination. Results section shows the enhancement that was made to the localization process compared to the IMU localization or the depth camera localization through using the mentioned fusion algorithm.

Keywords: Navigation, Path planning, Depth/inertial localization

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Autonomous Robots interact with the environment through highly sophisticated sensors and actuators, and they are becoming increasingly capable of executing complex tasks, thanks to their high computing power. Package delivery is one of the tasks that utilizes the Unmanned Ground Vehicles (UGVs) or the mobile robots by navigating autonomously. Several companies are interested in this type of delivery. Autonomous robots must navigate “intelligently” in their environment, which could be indoor or outdoor. Navigation is built on three main pillars: Mapping, Localization and Path-planning. Whether the environment is an indoor environment or an outdoor one, mapping is needed to update the global map that the robot is using in the navigation process. Localization is another important factor in the navigation process, since it is essential for updating the state of the robot to create a cognition between the robot and its environment. Path planning is the third pillar of navigation, which uses the localization to monitor the position and update the path (or waypoints) accordingly.

Thesis Objective

The main objective of the thesis is to enhance the process of navigation, by enhancing the localization part. This is performed by fusing the depth camera localization data with the raw IMU (Inertial Measurement Unit) localization data, using feed-forward cascade correlation network (CCN). CCN is a methodology to build an optimized topology of the neural network layers, which best fits the application. Since the depth camera could lose frames due to various reasons, – as will be discussed throughout this document – the fusion with the IMU would firstly increase the precision of the localization, since the IMU is much faster than the depth camera in terms of data recording. Secondly, it would compensate for the loss in signal availability that might happen to the depth camera.

The fusion is assessed by different references such as wheel encoders and a GPS device, which indicates the feasibility of this project in indoor and outdoor environments. Since navigation requires mapping, a predefined map is given to the system, which allows the path-planning algorithm to plan the path and extract the waypoints. Then, the PI controller program takes over by reading the waypoints, gets the location of the robot and actuates the signal to reach to the destination.

Chapter 2. Literature Review

2.1. Localization

2.1.1. Indoor localization. Multiple techniques exist for localization and navigation in indoor environments. Some of these techniques require pre-setup for the environment where devices are added to the environment in order to get the state of the object or the robot. Examples for these techniques can be ultra-wideband, Wi-Fi, RFID, and Bluetooth beacons. Each one of these techniques has its own advantages and limitations. This will dictate the choice of a certain method for a given application.

On the other hand, visual, inertial, assisted GNSS, sound, and magnetic systems are examples of ready techniques that can be used without the need of a prior knowledge of the environment. Therefore, the sensors are mounted only on the robot itself. Since the local infrastructure is not required for these types of technologies, they are called infrastructure independent technologies.

The difficulty of indoor positioning arises due to different factors like environment dissimilarities, signal attenuation and signal distortion due to variety of noises, which also leads to light scattering. In addition, the accuracy requirement, which depends on the application, could reach to micrometers. At this level of accuracy, noises will be the main enemy of the object localization.

Since most of the applications are real-time, availability of the sensor measurement plays a crucial role in determining the best technology to use. If the motion of the object is unpredictable or untraceable, localization of an object would be impossible when the signal is lost, since it relies on the previous state. A good example for this is an inertial measurement unit, which uses the initial state in determining the next ones. Cost is another considerable limitation when a designed localizing product comes to mass production stages. Range and accuracy are usually two opposite attributes. In other words, if a certain technology gives wide range, most probably, its accuracy is relatively low [1].

Several mathematical and trigonometric approaches are used for the estimation methodology in the localization techniques. Time of Arrival (TOA) is based on measuring the time between a mobile target and three minimum surrounding beacons,

15

which results in three circles; accordingly, the point at which these three circles intersect is the object coordinates. TOA requires a timestamped-synchronized signal. Synchronization is partially dropped in Return Time of Flight (RTOF) method and it is totally eliminated in the Time Difference of Arrival (TDOA) method to be replaced by time reference.

Received Signal Strength Indicator (RSSI) indicates the difference between the transmitted and received signal strengths that makes it better than the previous mentioned techniques, since it does not depend on time. Angle of Arrival (AOA) or Direction of Arrival (DOA) are other approaches that use the intersection of different angled lines from different sources (antennas) to estimate position of an object in a limited radius of the antenna itself. All the previous methodologies are derived from a main methodology, which is called Triangulation method [2].

The pre-setup-based techniques are considered as wide range techniques, since their coverage goes up to tens of meters. However, as a compensation, their accuracy is poor and barely falls to tens of centimeters except the ultra-wideband, which is accurate since it relies on TOA of the radio waves.

Local infrastructure dependent techniques

Ultra-wideband

Ultra-wideband (UWB) uses a radio wave generator (emitter) and antennas (receiver) to detect static and moving objects. The estimated state depends on the scattered waves (coming from the emitter) of the object that are received by the receiver. By using this information, the position of the object can be determined using the TOA.

Since the 3D geometry of the room is predefined in this technique, reflections, which are caused by walls or objects, are taken into consideration. This makes this technology robust to potential noises. However, UWB is still susceptible to metallic walls interference when the environment is not predefined, and it is considered to be more expensive than other existing technologies [3].

Wireless Beacons (Wi-Fi, Bluetooth, and RFID)

Wi-Fi (IEEE 802.11 WLAN) is another good wireless approach of state estimation, which relies on the strength of the signal received. Additionally, Bluetooth

16

is considered as another secure, low power (BLE) and low-cost wireless technology [4]. However, it cannot be used for real-time localization due to its comparative high delays. On the contrary, RFID, which consists of an antenna that receives the surrounding ID tags data to estimate the position, can penetrate obstacles and walls in the environment but lacks security [5]. However, Wi-Fi is the best option for many researchers, since the technology itself is available in most indoor environments.

Wi-Fi indoor localization's high accuracy arises from the scattered or multipath-transmitted waves. A new Wi-Fi indoor localization method has been introduced in the past few years, known as "channel state information". It estimates the quality between the transmitter and the receiver by using the physical layer to depict the undesired propagated waves and lessen them as much as possible [6]. Each technology uses a unique operating frequency. The frequency and the strength of the signal dictate the accuracies and ranges of mentioned protocols. Figure 1 shows the ranges and accuracies of each technology.

Figure 1. Different ranges and accuracies of each technology [7].

Local infrastructure independent techniques

Robot-mounted techniques are used more often when portability is required. The previous methods are based on the electromagnetic waves to detect objects and localize them. Unlike the previous mentioned techniques, ultrasound uses sound waves to localize objects. It is based on transmitting and receiving high frequency (non-

audible) sound waves, therefore a medium at which the waves propagate is a requirement (vacuum will not allow the propagation of sound waves).

Ultrasound

The ultrasound emitter and receiver can be integrated in one device, which enables them to be mounted on the same platform. Then, by using TOA, the receiver at the other end estimates the distance travelled by the transmitted wave, and accordingly, it calculates the position as shown in Figure 2. Ultrasound is widely used in the medical field, since it is helpful in many surgeries and medical diagnosis. Moreover, it is crucial for constructing an image based on the attenuated reflected signals by using B-scan technique (2D) [8], as is the case for fetal ultrasound. Since ultrasound is known to be a mechanical wave, it depends on the type of the medium and its temperature. Therefore, pre-attained information about the environment is required for the sake of accuracy. For example, the speed of sound in crystals [9] varies for the same substrate at different temperatures.

Figure 2. Ultrasonic sensor methodology [10].

Assisted-GNSS (AGNSS)

Global Navigation Satellite Systems (GNSS) are increasingly gaining traction in the industry due to their superior accuracy compared to GPS only systems. GNSS is a general term for GPS, GLONASS and BeiDou. It uses satellites to determine the position with few meters of accuracy. However, satellite signals are attenuated as they propagate to the earth's surface, making it difficult to lock on a target when obstacles appear in the way, which is the case for the indoor environment. Assisted GNSS

(AGNSS) uses the internet access to help the receiver identify the satellites relative velocities and positions (ephemeris data or assistance data) [11]. Therefore, AGNSS can use the attenuated signals in addition to the mentioned prior information to localize an object (AGNSS receiver) in indoor

environments. AGNSS and other assistive GNSS techniques suffer from the lack of accuracy that is within a few meters [12].

Magnetic Localization

Magnetic localization has multiple types and multiple subsections. In Near-Field Electromagnetic Ranging (NFER), as the electromagnetic waves are propagating from a local transmitter on earth, an antenna is used to estimate the difference between orthogonal electric phase and magnetic field phase. The difference reaches up to 90° at infinite distances, therefore, the smaller the difference, the shorter the distance is. Another universal type of magnetic localization is known as “magnetic fingerprinting”, which uses the magnetic flux density. A magnetometer is used to detect the variations in the earth’s magnetic field to determine the current position. This method showed a vast range of adoption due to its ease of use and relative accuracy [13].

Compasses and inclinometers can be fused to estimate the three DOF orientation of an object [14]. However, if the environment magnetic field is not static, the fingerprinting algorithm cannot be applied.

Inertial Navigation Systems (INSs)

Inertial Navigation Systems (INSs) are devices that estimate the acceleration, the velocity and the position using the IMU. By knowing the initial state, the previous data can be extracted from the IMU. Usually, IMUs consist of three main sensors, which are the accelerometer, the gyroscope and the magnetometer. The accelerometer is used for acceleration calculation and linear motion sensing, where gyroscope is used for angular motion sensing. Many smartphone companies use Micro Electro-Mechanical Systems (MEMS) to estimate steps count and distance [15]. The technology is based on inertial measurements of the foot or the body movement [16].

Some IMUs come with an integrated barometer that helps in estimating the altitude. The advantage that the IMU has over the other technologies is its universal use due to its compact size and its availability, which makes it efficient for portable device application. Figure 3 shows an example of the IMU (the black chip) compactness.

Figure 3. Compact size of the IMU sensor [17].

IMUs are effective and helpful in many robotic projects. However, the accumulative increasing error in addition to the need of initial position knowledge make the IMU an unreliable if used as a standalone localization sensor, especially in indoor environments, where no GPS can be added. Therefore, IMU is most likely to be used with other sensors (depth sensors and odometers) to show more reliable results. Moreover, INS can be fused with other technology such as visual sensors [18] or wireless beacons (Wi-Fi, Bluetooth, RFID... etc.) to estimate the initial position and give more accurate results in a dynamic operating environment [19]. Ground-based robots can also use wheel encoders to observe angular position and velocity, then convert them to linear measurements to generate odometry [20].

Visual Localization

The use of cameras is another approach for indoor localization. Since cameras give an image of a certain place, this image can be trained by neural networks to give an estimate of the current position [21]. In

fact, visual localization is scalable depending on the requirements of the application. Cameras can be used in sensitive applications such as endoscopic examination with an accuracy near to 1.8 mm [22]. Some types of camera systems are based on AOA that was illustrated before, which is known as “synthetic stereo vision” [23]. These cameras give accurate images but require intensive computation to build the image. A good example for this type of cameras is the ZED camera.

Other techniques use pattern recognition approaches by performing image processing. The RGB camera reads all the different markers in the environment and the system estimates the current position based on their sizes and patterns. Each marker has

a unique pattern as shown in Figure 4. Motion capture using passive markers is another utilization of the visual localization technique. This method requires certain types of retroreflective materials to help the cameras estimate the position by sensing the brightness of the reflected light. These cameras usually use infrared light spectrum to spot the object. Vicon [24] and OptiTrack [25] are products that use this technique to estimate the position. Although this setup is highly accurate and can be used for online measurement, the cost involved is relatively high. Moreover, the field of view is dependent on the geometry of the setup and the cameras performance. Therefore, this type of visual localization is an infrastructure-dependent technique, since it requires a camera setup as shown in Figure 5.

Figure 4. Different patterns for Aruco markers [26].

Figure 5. Example of an OptiTrack setup along with the calibration stick [25].

The simple infrared localizer requires an emitter and a receiver as shown in Figure 6. The different infrared emitters project an infrared light grid and each light point is reflected to the receiver. Then, the sensor estimates the location of each index using the triangulation method, resulting in knowing the position within the constructed surrounding map with depth information, as shown in Figure 7. LIDARs and IR cameras are getting more popular in the robotic field, since their accuracy has currently improved to millimeter level.

The reason behind the popularity of this technique is its portability and documentation availability. Kinect V1 for example is getting a huge amount of support, even after the discontinuation of using the device for entertainment purposes. Kinect V1 uses the structured light to estimate the position to get the depth map [27]. 3D LIDAR is fast and accurate; therefore, it is increasingly adopted by carmakers due its use in developing autonomous cars [28].

It is worth mentioning that many researchers prefer to fuse some of the sensors that have accuracy limitations (e.g. IMU) with another localization method to have better results [31]. Table 1 shows the expected accuracies and ranges for the different techniques.

Figure 6. Infrared Transmitter/Receiver [29].

Figure 7. Methodology of Infrared Technology [30].

Table 1: Expected Accuracies and Coverage ranges for selected Technology [32].

Technology

Accuracy

Coverage (m)

Ultrasound

cm

2-10

AGNSS

6-61.2 m

global

Magnetic Localization

mm-cm

1-20

INS

1 %

10-100

Visual Localization

0.1mm-dm

1-10

Infrared

cm-m

1-5

22

Visual/Depth Sensors

The main reason for using Visual/Depth sensors in many research fields is the availability of extensive amount of resources, which makes their implementation easy and fast. The technology adopted by these devices vary from one device to another. In this section, multiple devices will be reviewed and will be compared to each other. The following devices are going to be divided based on the used technology. As mentioned earlier, the devices are going to be compared based on their cost, weight and the availability of resources and support. All the upcoming devices are compatible with Windows, Linux and the framework Robot Operating System (ROS).

Structured Light Technology

Structured Light Technology is being adopted by most of the visual/depth sensors due to its low cost and availability of drivers for Windows, Mac and Linux. In this technology, light grid is projected on the surroundings. Each light will have an aura that belongs to the light point in the middle, as shown in Figure 8.

Figure 8. Projected infrared lights by Kinect V1 into three different regions [33], [34].

In Figure 9, the receiver estimates the distance to each light point using the triangulation methodology [35]. In Structured Light Technology, light sensitivity is an important factor. It could cause problems when moving from indoor to outdoor environments. Therefore, some sensors come equipped with the ability to change the sensitivity based on the environment such as Kinect V1. Microsoft Kinect V1 [36], which is one of the most widely adopted sensors, uses this technique with a resolution of 320x240 for the depth camera and 640x480 for the RGB camera, and it takes pictures at a rate of 30 FPS. It is also equipped with a motor that enables it to tilt when needed. The Kinect V1 was originally a project for Xbox 360 and Xbox one consoles that did not gain enough interest in the gaming society, therefore it was discontinued. Even

though it is not being sold by Microsoft, it is still there in online stores in abundance. This is due to its wide range of resources and ready codes that can be used by researchers for learning. However, the portability of the Kinect V1 is poor due to its size and weight; it weighs 600g, which makes it a considerable load, especially when it is mounted on a robot. In addition to that, it needs an external DC power source (battery), which will increase the weight even further.

Figure 9. Triangulation principle of estimating d (distance).

PrimeSense and Xtion 2 sensors [37] manufactured by Asus are good options due to their lightweight and their high resolution. PrimeSense Carmine 1.09 has both depth (640x480) and RGB (1280x1024) cameras. It is also equipped with a microphone that can be used for triggering it. Asus Xtion 2 is even better since it is lighter (145g) and its RGB camera has more resolution with an FPS that goes up to 60. Asus sensors resources are not as abundant as the ones for Kinect V1. However, it still has a good amount of resources available to support users.

Occipital Structure [38] is another portable solution for depth localization, since it only weighs 95g. It has a depth and RGB cameras with a resolution similar to PrimeSense Carmine 1.09. Although, it is an iPad-based hardware that has its own

battery (3-4 hours), it possesses the compatibility to work with computers and other tablets as well. Since it is still a relatively new project, resources and support are still limited.

Similar to PrimeSense Carmine 1.09, Orbbec Astra [39] has an integrated microphone. Additionally, it has almost the same resolution for both depth and RGB camera, but with a lower FPS (FPS=10). Orbbec Astra and Occiptal Structure use the second version of Openni driver (Openni 2), which is used in ROS to recognize the device itself.

The last sensor to be considered for this section is Intel SR300 [40]. It supports USB 3.0 to be able to handle the 1080p 30 FPS resolution of the RGB camera along with a depth resolution equal to 640x480. Although SR300 is motorized, its weight does not exceed 110g, which is much lower than the motorized Kinect V1.

Pulsed Light Technology

Similar to the structured light technology, Pulsed Light Technology cameras have an emitter and a receiver using a different mechanism. The emitter sends light pulses to the object. The receiver, which is an array of photodiodes, will capture the reflection of these pulses. Once they are received, the Time-of-Flight (TOF) technique uses the speed of light to measure the distance of the reflected beam at each point of the photodiodes. Unlike Structured Light Technology, Pulsed Light Technology is compact and fast, since it uses pulses of light speed to estimate the distance, which also reduces its power consumption. However, this comes at a price, which is the vulnerability of the sensors (CMOS detectors) to the ambient light [41].

Microsoft Kinect V2 [34] – shown in Figure 10 – uses TOF to estimate the depth distance (DD) by using the reflected beams captured by the wide-range camera. This gives depth information to the user with a resolution of 515x424. Since Kinect V2 uses TOF, the FPS can reach up to 160 in addition to the RGB camera that has 1920x1080 pixels and 30 FPS. This huge amount of data requires a fast way of communication (USB 3.0) as 2 gigabytes are being sent per second. However, Microsoft managed to reduce the power consumption by abandoning the tilting motors that existed in the older model. Kinect V2 is equipped with a microphone that allows waking up from sleep, which also increases the energy efficiency and reduces the power consumption. Since

Kinect V2 is relatively newer than the previous version, it has fewer resources than the older model Kinect V1. However, many researchers are now adopting it in their projects due to its efficiency and practicality. Kinect V2 weighs 900g, which is significantly higher in comparison with the previous sensors, which makes it a considerable drawback if portability is needed in a particular application.

Figure 10. The internals of Kinect V2: showing the principle of TOF.

Sony DepthSense [42] is another famous extremely low powered and low cost TOF-based camera that is being adopted by BMW luxurious cars. DepthSense 325 is equipped with two microphones and two cameras: depth (320x240 60 FPS) and RGB (720p 30 FPS) cameras. This sensor targets the short-range applications, and it is designed for indoor uses.

2D LIDARs and 3D LIDARs use Pulsed Light Technology to estimate the surroundings as well. The difference between 3D LIDARs and the previous Pulsed Light Technology cameras is the pulsating field of view; 3D LIDARs pulsate the lights at one portion of the field of view at a time, then the portion keeps on rotating until it covers all the portions of the grid [43]. 3D LIDAR is a recent technology and it is getting more popular, especially in the automated automotive field, due to their accuracy and

their lightweight. An example of the field of view (FOV) in 3D LIDAR is shown in Figure 11.

Figure 11. FOV of 3D LIDAR developed by LeddarTech [44].

Stereo Camera

The idea behind Stereo Cameras is that it imitates the human eyes, with the lens and the retina, allowing them to sense the depth of the image along with the colors. Stereo camera platforms can have more than two lenses as well; each lens has its own – known – focal length and it is equipped with an image sensor representing the retina in human eyes [45]. Using the Similar Triangles theorem, the distance to a certain point can be estimated as shown in Figure 12.

Although stereo cameras are relatively an old technology, Stereolabs has recently adopted this technology for CPU based projects in their ZED Cameras. More recently, Intel RealSense d435 was developed with outdoors capability since it uses laser projector, which also allows it to perform in dark environments. Stereo cameras are usually light and reliable, but they require high processing speeds due to the high amounts of data being processed and sent. In other words, they require Graphical Processing Units (GPUs) to be able to handle the huge amount of data being processed.

ZED Camera [40] is a powerful camera that can give a resolution up to 4416x1242 with a speed of recording that goes to 100 FPS. Unlike the previous sensors, the ZED camera has identical RGB and depth resolutions, which increases the accuracy, and the quality of the image being fed. The ZED Camera weighs 159g, while the new mini ZED camera [46] weighs only 62.9g. Both cameras require a fast USB 3.0, since it needs to send the data rapidly to the PC to be processed by the GPU. The compatible GPUs for ZED cameras are exclusively manufactured by NVidia, since they support the use of parallel processing to get the resultant image.

27

Figure 12. The structure of the stereo camera [47].

ZED Camera must be equipped with an NVidia chip such as NVidia Jetson TX Development KIT (1.6 Kg) when used on a portable platform; it. Since the ZED camera is considered as a passive sensor, low-light environments reduce the accuracy of the depth camera resulting in losing track of the position.

RealSense D4xx [48] is a recent stereo-based technology project that has two versions of devices similar to each other, which are d435 and d415 that weigh 72g each. They both need an Intel GPU kit to power them up, such as “Aaeon UP Board” that is the size of a raspberry pi. However, it is more powerful with a Quad Core 64bit Atom x5-Z835, Intel HD 400 GPU and a 4 GB DDR3L-1600 RAM Memory. The kit supports Ubuntu OS 14.04 and 16.04. However, the use of a certain kit is not a necessity, since any Intel GPU power kit works perfectly with either camera. Both cameras are equipped with two 1080p 30 FPS cameras in addition to standard IR projector for d415 and a wide IR Project for d435. Therefore, unlike the ZED camera, they are considered as Active sensors. In addition, they both share the same 1280x720 90FPS depth resolution using the Intel® RealSense™ Vision Processor D4. Intel claims that both cameras are adaptive for the background light and can adjust easily with outdoor environments as well as indoor environments. The main difference between the two cameras is the shutter technology being used. Unlike d415, which uses a Rolling-Shutter, d435 uses a Global-Shutter, which is important for capturing dynamic objects, since it projects all the reflected lines of the projection at once, making the object looks like they are frozen at that moment. Furthermore, the lenses are different for each device resulting in different FOV angles as shown in Table 2. Intel is exerting many efforts to support these

devices after the discontinuity of the old R200 device. Intel uses RealSense driver and many demonstrative examples for using the device can be found online.

28

Table 2: Specification of the motioned cameras.

Device name

Max DD (m)

Min DD (m)

Horizontal FOV

Vertical FOV

Price

Weight

Kinect V1 [30]

4

0.8

62°

48.6°

109$

600 g

Asus PrimeSense Carmine 1.09 [31]

1.4

0.35

57.5°

45°

295$

395 g

Asus Xtion 2 [31]

3.5

0.8

74°

52°

270$

145 g

Occipital Structure [32]

3.5

0.4

58°

45°

379$

95 g

Orbbec Astra [33]

8

0.4

60°

49.5°

149$

300 g

Intel SR300 [34]

1.5

0.2

71.5°

55°

270$

109 g

Kinect V2 [30]

4.5

0.5

70°

60°

105$

900 g

Sony DepthSense 325 [36]

1

0.014

74°

58°

38$

227 g

ZED mini [40]

12

0.15

90°

60°

449$

62.9 g

ZED Camera [40]

20 - 15

0.5 -1

90°

60°

449$

159 g

Intel RealSense d415 [41]

10

0.3

69.4°

42.5°

149$

72 g

Intel RealSense d435 [41]

10

0.3

91.2°

65.5°

179$

72 g

29

2.1.2. Outdoor localization. When it comes to outdoor localization GPS and differential GPS combination is the best global and accurate solution. Since its range is unlimited globally, and the error of the GPS can be corrected by a differential GPS (dGPS). GPS uses three satellites or four (increased accuracy) to estimate the position in 3D by obtaining an intersection of three or four projected estimated circles on the ground. GPS signals are attenuated due earth's upper layers, which also causes delays for signal travel time leading to deviated readings in the receiver end. Although the delay does not exceed 16 ns for a certain satellite, the error goes up to 5 m [49]. In some applications, the accuracy of the location is needed to deliver an effective performance.

GPS navigation equation can be given as:

$(Xi-UX)2+(Yi-UY)2+(Zi-UZ)2=(Pi-c\Delta TB)2$

(1)

Where UX,UY and UZ are components of receiver position, ΔTB is the time difference between receiver and satellite, X,Y, and Z are three components of satellite position, Pi is the exact distance of ith satellite from receiver in ideal conditions, and c is the light velocity. The distance calculation should be done for three satellites at least. An introduced solution (dGPS) was to have a fixed known position called static based station, to average the noises and delay changes, caused by the earth layers out of the measurement scope, leading to accurately localizing the point with 20 cm error. Another type of dGPS uses geostationary satellites as shown in Figure 13. Since they follow the earth's direction rotation above the equator, they appear to be stationary and can be used as stationary base stations, which reduces the error to < 1 m. Combining the above two solutions results in a very accurate positioning system with a 2 cm deviation error [49].

Figure 13. Illustrates the importance of the reference station in dGPS [49].

30

The importance of the dGPS is illustrated in Figure 14 and Figure 15. As shown, due to the different factors, the delay that the signal has towards the GPS receiver affects the estimation of the distance.

Therefore, the station tries to correct the error by correcting the distance between the receiver and the satellite, since the ephemeris data is already known to the station.

Figure 14. Difference between GPS and dGPS [50].

Figure 15. The enhancement that dGPS does for the accuracy of the state [50].

2.2. Path Planning

Path planning is another wide domain that involves many algorithms that dictates the path for the different types of robots. Path-planning can be defined as finding a continuous path from the start node to the end node in a certain environment that contains time-independent small defined segments or nodes connected to each other. Another related part of robotics navigation is the trajectory planning, which is related to the path planning itself. Trajectory planning refers to the problem of determining the modality of the path curve in a certain robot [51]. It is strongly related to time, since it can be described as recursive or continuous states that can be formulated mathematically.

A certain classification model has been chosen to categorize path-planning algorithms based on their unique properties [51]. The first category is the Sampling Based Algorithms, which samples the environment and then reaches the goal using the taken samples. Secondly, comes the Node Based Optimal Algorithms, which use a set of nodes that are predefined already to construct an optimal path. Thirdly, is the Mathematics Model Based Algorithms, which consider the dynamic and kinematic constraints of a certain robot and optimize the trajectory of a path to make it feasible by the robot. The Fourth category, namely Bioinspired Algorithms are inspired by the behavior of the biological systems and colonies to imitate them to reach to a certain achievement. The last category is called Fused Algorithms, which include some combinations of different algorithms resulting in a better performance.

2.2.1. Sampling based algorithms. Sampling based algorithms are divided into active and passive. The active division classifies the algorithms that are capable of reaching the goal independently without the need to combine with another algorithm. Active algorithms can be used for a real time path planning application, since they generate the path (waypoints) as they are running. However, a predefined map needs to be registered in the database for these algorithms to run. Simultaneous Localization and Mapping (SLAM) [52] can be used to allow real time mapping and path planning at the same time. Therefore, convergence time is a very critical factor in real-time applications.

Active algorithms

Rapidly exploring random trees (RRT)

RRT was proposed firstly in [53]. One of its advantages is the ability of handling multiple degrees of freedom. Furthermore, it tries to solve different robot constraints, such as non-holonomic, kinetic and dynamic constraints. Firstly, random trees are growing rapidly and randomly in the configuration space trying to connect the defined starting goal with the ending one. As the tree is growing, new nodes are being generated randomly in the map where the path is free (no obstacles in the path between the generated node and the starting one). Then, one of the registered nodes (nearest node) is selected based on its distance to the random generated node, which dictates the direction of the tree branch growth.

The step size of the branch is a predefined value that can be set based on the size of the environment and the potential obstacles. The connection is made if the path within the step size area is obstacle free. The process of the RRT is shown in Figure 16.

Figure 16. The process of RRT [54].

Dynamic domain RRT (DDRRT)

DDRRT was introduced in [55]. It was meant to solve the biasing disadvantage that RRT has, due to the use of Monte Carlo random sampling. In DDRRT, an r-radius sphere is constructed at each center of a near node representing the maximum length of an upcoming generated branch for that particular node. The existence of this condition will make sure that any generated node is unbiased towards the other generated registered nodes. This biasing problem in RRT is also known by Voronoi Bias. Although, it is a property of the RRT and can be used as an advantage, it is undesirable in certain applications where time is an important factor to reduce. Therefore, DDRRT could be a better alternative for RRT depending on the application.

RRT-Star (RRT*)

This method and RRG method (mentioned in the Passive Algorithms section) were introduced in [56]. RRT* is considered to be a great improvement for RRT. It is an enhanced version of RRG that can rewire the constructed paths to lower the cost of the entire path for a certain node. As soon as the new node is generated, the algorithm rechecks for the surrounding nodes connections with a certain radius from the center of new node, after the new connection has been made. If a certain path for any surrounding node can be cost optimized, the node is rewired to the lowest possible cost node in the

surrounding region. The cost values are assigned at each node, generating iterations based on different factors such as node distance, path smoothness, global goal distance and potential obstacle existence. Figure 17 shows the enhancement that the RRT* has compared to the conventional RRT.

Figure 17. Enhancement that the RRT* has over RRT [54].

Artificial potential algorithms

Artificial Potential Algorithms – shown in Figure 18 – use a different method than the previous mentioned algorithms [57]. It utilizes a potential function that describes the relationship between the free path and the obstacle in the given environment. By using the sum of forces of the attraction to the goal and the repulsion of the existing obstacles, the path will tend to guide the robot towards the goal, while simultaneously being repulsed by the obstacles.

Figure 18. The potential created in accordance to the existing obstacle [57].

Passive algorithms

Rapidly exploring random graph (RRG)

RRG is based on RRT [56]. However, the new node tries to extend the connections with a certain radius to all surrounding registered nodes. This results in a graph with more possible efficient connections with a complex looking map. Although this might be a disadvantage, it enables swarm or multiple robots to take different paths on the constructed network or map.

Probabilistic road map (PRM)

PRM – shown in Figure 19 – [58] is the first used multiple query method that uses sampling approach, which builds a map called a roadmap. In every iteration, the system selects a configuration in the configuration space randomly and tests whether it is in the free space using a collision check function. When a random node is generated, it checks if it is in the free space or not. If so, it then considers the possibility of forging routes between this new node and the closest existing node in the configuration space. Every created path is recorded as a new edge in the graph that the system is building. If obstacles exist in the connection between the closest node and the new one, the connection fails, and it will be disregarded. The PRM methodology is capable of introducing kinematic constraints for a certain robot. Once PRM is generated, a path can be generated by trying to connect the designated start and end node to the nearby nodes on the roadmap using the Node Based Optimal Algorithms that will be discussed in the next section.

Figure 19. Probabilistic Road Map (PRM) while creating new nodes [59].

2.2.2. Node based optimal algorithms. These types of algorithms use the generated nodes to find an optimal path by reducing the cost as much as possible.

These algorithms can be also referred to as network algorithms [60], since they use the generated network that was created by another algorithm. They can also be used in a grid map where each node has its own cost.

Dijkstra's algorithm

This algorithm was introduced by Dijkstra [61]. It determines the shortest path in a certain configuration space where the cost of each configuration is known. The cost defines how the path should look like. This is done by generating weights for every node towards the goal, then, the algorithm tries to connect the goal to the starting node through the other existing nodes with the lowest cost possible.

A-Star algorithm (A*)

A* which was introduced in [62] is an extension of the previous demonstrated algorithm (Dijkstra's Algorithm). It uses an increased number of states by involving another factor in the cost equation called the heuristic estimation.

A* tries to find the lowest cost path using the registered information about the cost of the path from the starting node to the nth node, further, it uses the cost of the node itself (heuristic value). These two pieces of information can be added resulting in a total representation of the cost from a start point to the end. As it proceeds, the cost will be added and compared with other possible options (nodes) until it reaches to the destination. Then, the path will be generated based on the nodes that the algorithm has selected. The process of A-Star Algorithm is shown in Figure 20, where the orange node is the starting

node, and the blue one is the end node. As shown, each node has its own cost, in addition to the heuristic cost which exists between two connected nodes.

Figure 20. Process of A-Star Algorithm [63].

36

Lifelong planning A*(LPA*)

LPA* [64] is an extension for A* that allocates for edge cost change of the registered nodes. Then, it determines the shortest route repeatedly from the starting node until the last one. Unlike A*, it incrementally adds or deletes vertices or nodes which results in a reduced overall time.

D-Star algorithm (D*)

D* stands for Dynamic A* [65], which allows the nodes to have a variable heuristic value. This property gives the authority to the algorithm to change cost of the nodes based on updated circumstances that is read by sensor. Unlike the A*, D* can access each node's cost, read and modify the cost if needed. This makes D* always ready for any abrupt changes in the environment, resulting in rerouting the nodes for the updated lowest cost path. With the introduction of D* Lite algorithm significant improvements were realized over the D* Algorithm with lower number needed nodes to be accessed and the nodes to be expanded.

D-Star lite (D* Lite)

D* Lite [66] is a recent and important algorithm currently, due to the less computation complexity that it possesses and less execution time. D* is based on LPA* that selects the first node and estimates the cost of the nodes reaching to the goal. However, D* Lite is a goal-oriented algorithm that uses the goal node to construct the node cost network. Therefore, it shortens the path at any starting point (non-predefined) towards the goal point. Table 3 compares the different grid/node based optimal algorithms:

Table 3: Grid/node-based algorithms.

A*

LPA*

D*

D* Lite

Execution time

Fast

Fast

Slow

Fastest

Computational power

Low

Relatively low

High

Relatively Low

Remarks

No nodes info

Start-node oriented

Registers multiple paths

Goal oriented

Cannot be used in real time

Can be used in real time

Static Cost

Dynamic costs

37

2.2.3. Mathematical model-based algorithms. These kinds of algorithms tend to optimize the path by considering the kinetic and dynamic constraints induced by a non-holonomic robot. Therefore, optimizing the path is required to abide the limitations of the robot while reaching the goal using the segments of the different paths provided by the nodes.

Linear algorithms

The used algorithms in this type of optimization is linear that sums up different terms to yield the total cost function. This cost function considers the kinetic and dynamic constraints, the reachable region at each step and the control factor. Furthermore, Linear Algorithms can work under control disturbances and it can handle node uncertainty. Representing the system and environment can be closely represented using Mixed-Integer Linear Programming (MILP). MILP is widely used, and it has a MATLAB toolbox called OPTRAGEN developed by [67].

Optimal control

It is a form that can describe a path planning problem in a set of a differential equations that accordingly control the path by knowing what the state is at a certain time. Optimal control uses state estimation techniques such as Kalman filter to estimate the state and accordingly control the robot towards a certain predefined non-optimal path. By controlling the robot, the resultant path is going to be globally optimal as the robot is approaching the goal.

2.2.4. Bioinspired evolutionary algorithms (B-EA). This method has multiple algorithms [68], and each one imitates a biological system. As the complexity increases in bigger environments, the number of variables increases as well, resulting in failure in some of the traditional linear and dynamic

programming sets. The idea of BEA is to have a stochastic search technique that can inherit the social behavior and the biological evolutionary systems in nature.

Genetic algorithms method (GA)

GA method was first introduced in [69]. In this algorithm, a cost function is initialized in order to evaluate the potential solution available. The idea is to use the chromosomes as potential solutions for a certain problem. Each population cluster is evaluated by a fitness function [70] that will dictate a following mechanism for these

individual clusters to survive. This process will repeatedly have an evolution of the generations until the population cluster converges to a good quality solution. This method became the most popular population-based optimization method.

Ant colony optimization (ACO)

Many animal species have interesting path planning systems such as hawks, flying snakes and ants. ACO [58] uses the fact that ants try establishing the shortest path possible from their colony to a certain location. This all happens by a group cooperation that enhances the intensity of trail and the visibility, which are two main concepts in this kind of systems that result in forming the shortest path. ACO can be incorporated in RRT algorithms resulting in a better path and a shorter one [70].

2.2.5. Bioinspired neural networks algorithms (B-NN). Neural Networks was first introduced in [71]. It is a subcategory of Bioinspired Algorithms, since it generates a path in neural-like form. It uses the attraction and repulsion forces that are used in Potential Field Algorithms. However, another equation is introduced in neural network to represent the dynamics of a certain robot. It tries to maximize the neural activity and select indexed neuron that results in this event. Since the convergence time is not predictable and is not guaranteed, Bioinspired Algorithms share the weakness of unreliability.

2.2.6. Fused algorithms. In [60] GA method was used in addition to A* to generate a Fast algorithm for path planning. Each chromosome was used to represent a feasible path network. Efficient chromosome encoding strategy provided guarantees, and each chromosome represents a feasible path, avoiding the searching circulation. Variation happens at certain positions, making the algorithm assemble quickly. Experimental results in [72] showed that the GA combined with A* speeds the path planning process. The combination of RRT* with ACO in [73] is another Fused Algorithm. The ACO algorithm works as an optimization technique based on the concept of swarm intelligence (decentralized, self-organized systems).

Although RRT* has been proven to be efficient for planning in environments populated with obstacles, it does not necessarily lead to the most efficient solution. Therefore, it would make it significantly faster if it is possible to learn from experience using heuristics to optimize the path as much as possible. ACO allows RRT* to learn

from experience to choose a different sample rather than the previous one and estimates what is the contribution of each sample towards the path improvement.

A new method that was introduced in [74], which uses obstacle vertices to rebuild a path in a real-time manner. It has a list of nodes that keep on expanding for every edge of an obstacle registered [74]. This method is an improvement of the Visibility Graph algorithm (VG) [75] that generates a path based on a roadmap technique. It is based on the use of the vertices of the obstacles to create a roadmap that will generate a path using any node based optimal algorithm.

Sensor-based Random Tree [76], which was improved in [77], is another sampling-based algorithm that inherits many RRT characteristics. However, the step size in SRT is dynamic and changes based on the surrounding obstacles. Unlike RRT, SRT is more oriented towards a certain angle.

In [77], the authors tried to improve the SRT by reassigning the random points to be non-random ("candidate points") based on the sensors' viewpoint. It showed better results in terms of the path-generated length.

For Indoor Path Planning, localization can be a challenging problem. The authors in [78] used Kinect V2 sensor to localize their robot. The path planning algorithms that were used are RRT and improved RRT that tries to bias the raw RRT towards the goal. This resulted in a smoother route for the robot, which abides to its kinetic and dynamic constraints. Although RRT* would also give a smoother route, its execution time would be considerably higher.

Another novel idea in [79] was to use fuzzy logic in analyzing the horizontal plane and vertical plane separately to reconstruct the total path. In each plane, there are two main algorithms: obstacle avoidance and goal seeking, then, the output of each plane for each algorithm is evaluated using the fusion at the weighted fuzzy logic controller. This results in giving the final motion command to generate the full path.

A conceptual modification of D* is proposed in [80]. The authors proposed an incremental path finding in 3D space, which allows the algorithm to change the cost of each node for the sensors ranged nodes only. Unlike the D* the number of nodes to be handled at once is lower, which leads to lower computation needs and shorter

calculation time. The authors assumed all the paths are affordable, but the planned path is the least costly path.

Another modification of D* was proposed in [81]. By intelligently dividing the search space with navigation meshes, the calculation time of the optimal path was reduced. Similar to the incremental path finding, this algorithm allows the starting of the movement without the full knowledge of the total space. It also introduced a terrain factor that determines the moving cost for different terrains. This is performed by declaring two lists; OPEN list and CLOSED one. OPEN list will have the nodes propagated information about path cost. CLOSED list will contain nodes that need to be expanded.

In [82], the authors tried to solve the shortcoming of RRT*, which is the difficulty to solve dynamic path planning problem. They exploited D* lite for fast convergence and applied it to a generated roadmap. The generated roadmap is constructed based on the existing nodes and the collision-free segments along with their cost in the workspace. This algorithm requires re-planning whenever a new threat is detected, since it might end up with a collision with the constructed roadmap.

The authors in [83] used an improved bioinspired neural networks algorithm to reach a static target underwater, autonomously. The neural activity values are characterized by an improved shunting equation that tries finding a short and a smooth path at the same time. This method improved the efficiency and the real-time performance. Although no actual experiments were presented, the simulations showed the ability of the algorithm to converge.

Another Fused Algorithm that uses Genetic Algorithms and Bezier curves is presented in [84]. Bezier curve is a parametric curve that is used to model smooth curves especially in computer graphics. It is a very compelling combination since the resulted curves are obligated to smoothness using Bezier curves. The authors presented a grid-cost map, which included obstacles in the map. Bezier curves were used to smoothen the path using eight control points in the map.

41

Chapter 3. Methodology

As shown in the previous section, many approaches exist for localization (whether it is indoor or outdoor) and for path planning (whether it is active or passive). The thesis will focus on using an affordable sensor in order to be mounted on the robot that is able to perform complex computation. This will allow the robot to use the information coming from the sensor and to localize itself in different environments. Moreover, this information is to be used in planning the optimal path of the robot. In addition, a trigger to start the process is to be used wirelessly. The following sub-sections will include an expansion to every block in the below flowchart (Figure 21).

Figure 21. Process of navigation.

3.1. Mapping and Path-Planning Algorithms

After the previous review, it is shown that RRT* gives an optimal path and offers a good solution, when the map is predefined. However, the time needed by the RRT* is relatively longer than the basic RRT. Since RRT and RRT* are sampling-based algorithms that create nodes in the environment, having these nodes generated in advance can save time.

Time is an important factor, since the purpose of this project is to have a real-time algorithm. Therefore, with the prior knowledge of the indoor environment, a grid-cost map can be introduced based on the starting node and the goal node. Then, a node-based algorithm such as A* can be used to give the path in real-time, to generate the path and associated waypoints.

Part of the path-planning algorithm is obstacle avoidance, which involves using a pre-existing map. Therefore, obstacles should have an infinite cost to force the algorithm to avoid the cost-map nodes of the obstacle. A 2D grid map and 2D grid-cost map are shown in Figure 22 and Figure 23, respectively.

Mapping

Mapping

path

path-planningplanning

Generating

Generating waypointswaypoints

Localization (CCN)

Localization (CCN) -Proposed SolutionProposed Solution

Control

Control

42

Figure 22. 2D GRID map where X is the goal and O is starting point.

Figure 23. 2D GRID-cost showing the cost (intensity of the color) with the presence of obstacle.

The black box in Figure 23 represents the change to the values of the cost map to infinity (where the obstacle exists). Then, the different obstacles of the map can be

43

added similarly. The reason for using A* is due to its lesser computational cost and faster path planning. For outdoor environment, the grid is based on the location of the starting node and the ending one in addition to the distance between them, which will define the step size for the grid-cost map.

The shown grid cost-map in Figure 23 creates the waypoints by knowing which direction it is heading. In other words, since the robot is non-holonomic, the rotation of the robot will be considered as an extra waypoint. Moreover, to reduce the number of waypoints (generate a more efficient navigational path), particular waypoints will be selected only when the orientation is changing. An illustrative example is shown in Figure 24.

Figure 24. Path resulted from the selected points according to the 2D cost-map.

The resolution of the provided grid – which represents the map – depends mainly on the object sizes, shapes in the environment and size of the environment. The shown Figure 23 and Figure 24 happen to be in two-dimensional frame. However, if a drone is being used or another three-dimensional navigational robot, this method can be extended to be a three-dimensional grid. Table 4 shows the generated waypoints, which will be fed to the robot navigation program. The epoch number is just a representation of the time at each waypoint passage.

44

Table 4. The generated waypoints of the previous path.

Epoch number Desired X Desired Y Operating Orientation

0 10 10 45°

7 20 10 0°

18 50 40 45°

48 50 50 90

Figure 25 shows the change of orientation – on the go – as epochs are advancing.

Figure 25. Orientation of the robot with respect to epochs.

Figure 26 and Figure 27 show a three-dimensional grid (map), which includes the waypoints in it. It is also worth mentioning that the objects (obstacles) in the map are slightly larger than their actual sizes, to ensure that the robot will have enough space even when there are small uncertainties in its localization.

Figure 26. 3D cost-map generated in MATLAB with presence of obstacle (black x's).

Figure 27. Path resulted from the selected points according to the 3D cost map.

With the consideration of holonomic dynamics of the drone, Table 5 of waypoints can be generated.

Table 5. 3D Generated waypoints for the previous 3D map.

Epoch number

Desired X

X(0)=4

Desired Y

Y(0)=5

Desired Z

Z(0)=1

0

5

6

2

7

7

10

3

18

7

10

5

3.2. Localization

The current sub-section will start discussing the details of localization techniques adopted in this thesis, by discussing the different chosen methods of localization for indoor and outdoor environments.

3.2.1. Indoor localization techniques. As discussed before, many alternatives exist for the indoor localization techniques. However, for this type of application, a portable independent technique is required. The level of accuracy needed is around 1 dm error, which is needed for the applications presented in this work.

In this chapter three different localization techniques will be discussed, in addition to the fourth localization technique that comes in the “outdoor localization techniques” section. First, the depth camera localization is discussed. Then, the wheel

46

odometry and heading (Dead Reckoning) is introduced and explained. Finally, the localization using IMU (Accelerometer and Gyroscope) is analyzed.

Depth camera localization

As shown before, many options are there for depth cameras, however, the chosen depth camera was the Intel d435. The mentioned camera was preferred over Microsoft Kinect V1 for two feasibility reasons and one technical reason. First, the weight of the Intel d435 (71 g) is lower than the Microsoft Kinect V1 (605 g). Secondly, Intel d435 has a USB-C port that is responsible for power and data transfer. USB-C port is capable of providing high power with no compromise on the speed of the data transfer. Additionally, Microsoft Kinect V1, requires a separate 12V DC power supply, which cannot be provided by a portable computer. Rather, it needs a battery that will increase the weight of the whole system. On the top of that, the Intel d435 provides two ways of depth localization, which makes it less vulnerable or sensitive to noises and vibrations.

The selected method of depth localization in Intel d435 performs localization by taking features from the environment by its stereo cameras and IR sensors. These features will be continuously compared to previously saved features to detect closed loop and enhance the localization process. This method is known as RTAB-MAP (Real-Time Appearance-Based Mapping) [85]. RTAB-MAP uses SLAM to keep track of the location. The camera frames will help construct the map; in turn the depth images will be used to localize the camera with respect to its environment.

The map will contain distances from each point feature to the camera reference. Once the camera moves, a new frame will be recorded and added to the global map, creating another set of distances. Therefore, comparing the previous distances to the new ones will re-localize the camera with respect to its starting point. RTAB-MAP uses the hardware of the camera to eliminate noises, and it is able to detect previous frames to adjust the map and position accordingly, this is known by Loop Detection. Loop Detection is done using bag of words method [86].

Loop Detection could result in a shortage of available memory if the map size becomes too big. Therefore, RTAB-MAP offers a different type of memory management, by introducing two types of memories. The first type is called Working

47

Memory, this type is continuously storing frequent and recent frames for a certain number of locations only. The second memory type is called Long Term Memory, in which loop closure detections are saved to remember and update the associated locations. This method makes the computation time feasible and reduces the complexity of the computation.

Figure 28 shows RTAB-MAP memory management model, where the Sensory Memory extracts the features from the image that is fed by Perception and sends the location to the Short-Term Memory (STM) by creating a location and signature for the image sent. STM is responsible for increasing the weight of similarities detections along the frames helping the working memory to decide whether it is a loop closure or not.

Figure 28. RTAB-Map memory management model [87].

Points projected and received by the sensor vary depending on the environment and the GPU (Graphical processing unit) used to operate the Intel d435 camera. The more distinctive the environment is the more the features will be in the frame. Moreover, the fastest frequency at which the camera can operate is 1 Hz, this frequency is set as a default setting. However, this frequency can be reduced if a low-end GPU is used. Lower frequencies cause less features in the frame, which reduces the quality of the localization accordingly.

48

Dead reckoning localization

To be able to understand the process of this type of localization, the robot model needs to be introduced. The robot used in hand is called “Kobuki” manufactured by “yujinrobot” [88]. Figure 29 shows the Kobuki robot. As any mobile robot, – non-holonomic robot – Kobuki has its own two-dimensional kinematics equations and matrices. These equations are based on rigid frame and rigid wheels [89].

Figure 29. The mobile robot platform (Kobuki) [88].

The global velocity vector G shown below, includes three variables, x, y (to represent the state in 2D frame) and θ (heading).

G=($\dot{x}\dot{y}\theta$)

(2)

where the local frame velocity matrix,

P=(r(∅

R+∅

$L)20r(\emptyset$

$R-\emptyset$

$L)2l)$

(3)

where, r is the radius of the Kobuki robot wheel which equals to 0.06 m, l is the distance between the two wheels which equals to 0.15 m, and $\emptyset$ represent the robot wheel angle for right and left wheels.

49

To be able to relate the robot's local frame to the global frame, a rotation matrix needs to be introduced (R).

R=(cos (θ)sin (θ)0−sin (θ)cos (θ)0001)

(4)

The rotation matrix calculates the global frame state matrix in relation to the local frame state matrix. The relationship between local frame and global frame is shown in Figure 30.

$G=R-1P$

(5)

Figure 30. Relationship between global frame and Kobuki's local frame [89].

where XR and YR represent the Kobuki's local frame. Moreover, XI and YI represent the global frame.

As shown previously, three variables were included in the local state matrix. Those variables are derived by the wheel odometry (wheel encoders) and the heading provided by the gyroscope.

Heading (θ) in Kobuki is given through quaternion units. Quaternion is a universal mathematical representation for orientations and rotations of a certain object

50

in three dimensions. Quaternion unit offers a simpler, more efficient and more stable representation compared to Euler angles, which makes it more convenient. However, the need to convert to Euler angles is necessitated by their convenience and ability to be grasped by users. In order to convert to Euler angles a rotation matrix needs to be used.

Eq. 6 shows a matrix for the conversion from quaternion to Euler angles, followed by an explanation of the parameters in Table 6.

($\emptyset\theta\psi$) =(tan−1(2($q0q1+q2q3$)1−2($q12+q22$))sin−1(2($q0q2-q3q1$))tan−1(2($q0q3+q1q2$)1−2($q22+q32$)))

(6)

where,

Table 6. Parameters explanation

∅ Rotation about the X-axis

$\boldsymbol{\theta}$ Rotation about the Y-axis

$\boldsymbol{\psi}$ Rotation about the Z-axis

$\boldsymbol{q0}$

qw

cos($\alpha 2$)

$\boldsymbol{q1}$

qx

cos($\alpha 2$)cos(βx)

$\boldsymbol{q2}$

qy

cos($\alpha 2$)cos(βy)

$\boldsymbol{q3}$

qz

cos($\alpha 2$)cos(βz) where, α is the value in radians of the angle of rotation. Moreover, $cos(\beta x)$, $cos(\beta y)$ and $cos(\beta z)$ are the direction cosines locating the axis of rotation.

IMU (Accelerometer + Gyroscope) localization

Accelerometer and Gyroscope are basic components in the IMU. They work together, to measure the acceleration in three dimensions, and the orientation in three dimensions respectively.

51

Accelerometers in principle, mimic the idea of a damped mass on a spring [90]. The spring constant is known, so that its displacement will lead to the acceleration in the spring orientation, as shown in Eq. 7, where the explanation of the parameters is shown in Table 7,

$dg=Mgksp$

(7)

Table 7. Explanation of the Eq. 7

$\boldsymbol{dg}$

Displacement per g

$\boldsymbol{M}$

Mass of the device

$\boldsymbol{ksp}$

Spring constant of the device

$\boldsymbol{g}$

Gravity = 9.8 m/s2

Nowadays, multiple ways of retrieving the data of accelerometers exist such as capacitive approaches, which are very reliable and accurate as discussed in [91].

Since the main aim is the position of the robot, integration of the acceleration needs to be done on the accelerometer measurement twice. However, since the integration amplifies the error over time, this will end up in false results in the future. On the other hand, gyroscopes are devices that consist of a wheel (flywheel) mounted by three gimbals [92]. Each gimbal is attached to one axis allowing it to rotate only on that particular axis. Figure 31 shows a three-dimensional Gyroscope.

Figure 31. Three-dimensional Gyroscope [92].

Gyroscopes accuracy decrease after some time, causing it to measure false readings of the state in three/two dimensions. Therefore, the assistance of the

52

accelerometer is needed to correct the values over time [93]. Hence, the accelerometer can help in correcting the gyroscope readings using Eq. 8 [94].

θ=0.98$*$(θ+$gyrData*dt$)+0.02($accData$)

(8)

As mentioned earlier, MEMS is the new trend for accelerometers and gyroscopes due to their compactness and portability. Figure 32 shows a zoomed view of the MEMS accelerometer construction.

By using fixed capacitor plates, the total capacitance will depend on the location of the mass of flexible mass as shown in Figure 32 [94]. Eq. 9 shows the capacitance value resulted from a certain movement in the 3-axis MEMS accelerometer shown in Figure 33 followed by Table 8 that explains its parameters.

Figure 32. Zoomed MEMS accelerometer model [94].

Figure 33. Detailed view of MEMS accelerometer [95].

C=ϵ0$*\epsilon r*L*Wd$ (pF)

(9)

53

Table 8. Meaning of parameters.

$\boldsymbol{\epsilon 0}$

dielectric constant of air = 8.85x10-12 Farad/meter

ϵr

dielectric constant of a certain substrate relative to air

L

length of adjoining fixed capacitive plate and mass

W

thickness of fixed capacitive plate and mass

d

separation between fixed capacitive plates and mass

To enhance the performance of the accelerometer, jerk has been taken into consideration to remove the abrupt changes of velocity. The use of jerk will require four consecutive accelerations readings in order to filter out the noises or unwanted readings. Eq. 10 – 12 shows the derivation needed to accomplish the discussed methodology.

$$Jerk1=acc2-acc1dt$$

(10)

$$Jerk2=acc4-acc3dt$$

(11)

$$Accelerationt=Jerk1-Jerk22dt$$

(12)

These equations will be applied to the three dimensions, so that it gives the filtered acceleration in the local axes. Therefore, to convert to the global frame a conversion needs to be done as shown earlier. However, different rotation matrix is used, to account for the different degrees of freedom (DOF) that the IMU has [96]. Eq. 13 shows the rotation matrix that needs to be applied to convert from local frame to global frame using accelerometer and gyroscope orientation data.

$$RIMU=(cos\theta cos\psi sin\emptyset sin\theta cos\psi-cos\emptyset sin\psi cos\emptyset sin\theta cos\psi+sin\emptyset sin\psi cos\theta sin\psi sin\emptyset sin\theta sin\psi+cos\emptyset cos\psi \; cos\emptyset sin\theta sin\psi-sin\emptyset cos\psi-sin\theta sin\emptyset cos\theta cos\emptyset cos\theta)$$

(13)

Consequently, the global acceleration values will be integrated twice using Eq. 14 and Eq. 15, resulting in the three-dimensional position.

$$vk=Accelerationtdt+vk-1$$

(14)

$$Pik=vkdt+Pik-1$$

(15)

where, vk represents the current velocity in a certain direction and Pik represents the current position in ith dimension with respect to the first position $Pi1$.

54

3.2.2. Outdoor localization technique (GPS). For outdoor environments, GPS technology can be additionally used to localize the robot. GPS is accurate enough for the outdoor terrain, since the size of the objects that the robot needs to deal with is relatively larger than the ones in the indoor environment. GPS provides a global ranged solution that makes the robot ready to operate without any prior knowledge about the environment as long as it is outdoor, and the GPS signal can reach the receiver without jamming or attenuation.

In places where high buildings exist, IMU can also be used for short time, since the GPS signal maybe out of reach. In other words, having the GPS and IMU fused by the extended Kalman filter EKF (since the model is nonlinear) will improve the accuracy and the reliability of the system. The used IMU, provide GPS support and include the corrected solution in its algorithms. The information about connected satellites is provided by the GPS, indicating the accuracy and confidence of the localization.

Although GPS is used to localize in three dimensions (latitude, longitude and altitude), it is not the best solution when it comes to measuring the altitude due to thermal dependent variations in readings that can reach up to 20 meters in normal conditions. This accuracy of the GPS is dependent on the temperature of the Ionosphere since it is the medium in which the signals are travelling. The variation could be even more dramatic at higher altitudes [97]. Therefore, most of IMU modules come with pressure sensors (barometers).

Pressure is another method to measure the altitude. By using Eq. 16, the altitude above the considered sea level can be measured [98]. Similar to GPS, pressure depends on the temperature as well, since the change of temperature causes the air to expand or to shrink.

$Altitudebarometer$=44330∗(1−(PP0)15.255)

(16)

Therefore, the average altitude can be taken from GPS measurement and barometer measurement to enhance the accuracy of the read results.

55

3.2.3. Proposed solution. This section will discuss the improvement that has been done to enhance the localization process. The section is divided into three main sub-sections. It starts by reviewing the problem and then it will discuss the solution that has been selected. Finally, it will show the diagram of which the process of localization is taking place.

In the previous sections the fact that IMUs are not reliable due to drift errors has been discussed, which makes them not accurate enough to meet the requirement of navigation applications. In this section, the proposed solution to overcome this problem will be discussed. Since multiple sensors are used to estimate the position, these sensors can be fused to assist each other to develop a more reliable solution.

As discussed, the camera is a reliable source. However, it is not as fast the as the IMU, and it does not work in all conditions. In other words, it can lose availability when it is too close to the wall, or when a severe swerve occurs by the robot. Therefore, fusing the both mentioned sensors would enhance the overall localization process and would decrease the dependency on a certain localization methodology.

There is much literature which discusses sensor fusion. When selecting the method of fusion, the time of convergence is important, since this is real-time navigation. On the top of that, the complexity of the system model becomes a burden on the controlling unit, since the system is highly nonlinear.

Moreover, filters such as Kalman filter has many limitations such as the need of knowing the covariance values and statistical properties in advance. Many limitations of Kalman Filter were discussed in [99-105]. Thus, this makes neural networks a good solution, since it does not require the model of the system, making it easily implementable on any hardware.

Determining the best topology when applying neural network creates a challenge when designing the artificial neural network (ANN), which also includes the hidden neurons and hidden layers in the network. Multilayer Feedforward Neural Networks (MFNN) is a possible neural network solution that provides the best accuracy with the appropriate number of hidden neurons and hidden neural layers. Many ways were discussed to determine the best size (hidden neurons) and depth (hidden layers) for the neural networks in [106] and [107], which indicate that the size and depth are

mainly dependent on the applications itself. Another dilemma is the time needed for the MFNN to learn the pattern.

The chosen fusion method is constructive neural networks (CNNs). Since the navigation setup can be extended to be on a single-computer chip, the number of layers and neurons that the neural network has is vital. The advantage that constructive neural network has is its ability to adjust its layers based on the needs. In other words, it starts constructing the layer one by one, while keep checking if the number of layers is enough.

Many methods for constructive neural network were proposed [108]. Since cascade-correlation network (CCN) [109] was the algorithm that attracted the most attention in the past two decades under the umbrella of CNN, it has been chosen to be the algorithm for the selected CNN. By using MATLAB Neural Network toolbox, a decent comparison was made between ANN and the CCN. As will be shown later, the errors associated with the CNN were much lower than ANN for the same path and the same number of hidden layers.

In the beginning, the topology of the CCN starts as shown in Figure 34. The weights linking the input neurons to the output ones are started to be trained. This is the minimal topology which will be extended adaptively based on the application needs. In this case, the input neurons represent the IMU localization data in the global frame (XIMU_global, YIMU_global and ZIMU_global). It is worth mentioning that the heading is directly fed by the gyroscope. The output neurons (desired outputs) represent the depth camera localization data (XDepth, YDepth and ZDepth).

The data will be trained in the initial topology until no further improvement can be done. The data are trained through Quickprop algorithm which does not require any backpropagation to adjust the weights [109]. As per [107], Quickprop is relatively fast and simple compared to the other algorithms.

Figure 34. Initial Topology for CCN (no hidden layers).

57

Table 9. The explanation of results shown in this section.

Blue rectangle

Input neurons layer

Orange rectangle

Output neurons layer

Gray rectangle

Added neurons layer

Grey rectangle

Winner neurons layer

Neuron

The next phase is to recruit a new pool of candidate neurons as shown in Figure 35. The new set of the candidate neurons will have their own initial weights which will be trained from the inputs initially. In contrast, the connections to output neurons have an equal residual error. Therefore, the candidate neurons are firstly trained based on their connections to input neurons, while the connections to the output neurons have not been made yet. The training is done to ultimately maximize the correlation among all connections going from the candidate neurons to the output neurons. Thus, a pseudo connection is made to deliver information about residual error that was initially randomized. Resulting in a maximization of the correlation in the connections from the candidate neurons and the residual error at the output neurons. S in Eq. 17 represents maximization formula, explained in Table 10.

Figure 35. Second phase of CCN construction when adding hidden layer.

S=Σ|Σ(Vp−$\overline{V}$

p)(Ep,o−$E\overline{\overline{o}}$)|o

(17)

58

Table 10. Parameters of Eq. 17.

o

Network output where the error (Ep,o) is measured

p

Training pattern

$\bar{V}$

The average of (V) over all training patterns

$E\bar{\bar{o}}$

The average of (Eo) over all training patterns

V

The output of candidate neurons

Eo

The residual error at (o)

Maximizing S can be done through differentiation which uses the Quickprop algorithm to adjust concerned weights. Afterwards, the winner neurons layer based on the incoming weights will be frozen, resulting in the insertion of the winner layer into the active network once the training is completed. This engages the second step of recruitment, in which the weights of the connections from the hidden neutrons to the output neurons become adjustable and start to be trained. In addition, the input neurons connections to the output neurons are retrained as shown in Figure 36.

Figure 36. Winner neurons are being trained to match the output neurons.

When another layer of hidden neurons is added, the whole process starts again as shown in Figure 37 and Figure 38. This process repeats itself and the layers of hidden neurons are added until no further improvement can be done, or the number of training passes reaches its maximum. Finally, the network can be represented with (n) hidden neurons and layers modified in feedforwarded manner as shown in Figure 39.

Figure 37. Third phase of CCN construction when adding the second hidden layer.

59

As explained, the advantages of using the CCN are many; the backpropagation is replaced by feedforward algorithm which reduces the complex interaction between the hidden neurons. Moreover, the feedforward algorithm can always be incremented, hence, the trained layers will stay and will not need to be retrained again as shown in Figure 38. CCN topology is decided automatically depending on the application complexity, so the need of performing additional trials to re-decide the hidden neurons and hidden layers does not exist.

Figure 38. Second set of winner neurons are being trained to match the output neurons.

Figure 39. Full Architecture with N hidden layers (neurons).

The diagram shown in Figure 40 illustrates the process of the localization. The IMU gives the position in addition to the heading, then the position data (XIMU_global, YIMU_global and ZIMU_global) is fed to

the neural network as input neurons. The trained network (by a previous mission) is used in the next missions at different training

60

percentages. The idea of using a previous mission is to use a pre-trained data to ensure convergence from the beginning.

However, the learning process could be continuous, and the network can keep on training from a later mission. This method was only feasible because of the use of the CNN. Whenever the availability of the camera localization is compromised, the learning process stops, and network used is based on the last architecture created when the camera localization was available.

Figure 40. Full process of localization.

As has been implied, the process of the neural network construction is like enhancing the Kalman filter covariance values since new information is always added. Based on the error histogram among 60%, 70%, 80% 90% and 100% training data, 80% was observed to be the best training percentage among the other percentages.

Once the first mission is done, and no availability is there for the camera, then the training stops, and the position estimated by the IMU becomes the testing set. Once the availability recovers, the training process engages from where it left over.

61

3.3. Control

Although Kobuki is a non-holonomic robot, wheel odometry in Kobuki is linked to its heading. In other words, sensing of encoders considers the X and Y axes. However, Kobuki wheels can only actuate in its local X axis. Therefore, to move in Y direction, it needs to rotate first, then it can move in the global Y axis, taking into consideration that the controller is controlling the X linear velocity. PI controller was used to control the speed of the wheels using Eq. 18.

v=$Kp\int ev\ dt$+$Ki\int\int \dot{e}v\ dtT0T0T0$

(18)

Where, ev is the error in linear speed. The gains of PI controller were obtained using the process reaction curve through Cohen-Coon method ($\boldsymbol{Kp}$=$\mathbf{9}.\mathbf{465}$,$\boldsymbol{K}i$=$\mathbf{0}.\mathbf{642}$). The accepted error range was 0.01 meters before moving onto the next waypoint.

Once the waypoints are fed to the user-defined navigation program, the robot will divide the waypoints into two main categories (linear and twist). The linear category is responsible for linear movement of the Kobuki considering that the orientation has been achieved already. The twist category is responsible for the rotation of the Kobuki to adjust and achieve the needed orientation.

In indoor environments, the program will keep using the odometry coming from the Kobuki for localization. However, in outdoor environments, the program replaces the Kobuki odometry with the GPS data in ECEF representation (which converts the longitude and latitude system to metric system). The GPS or odometry data is being compared to the depth localization, INS localization and localization

data from the CCN fusion. As shown earlier in Figure 21, waypoints are given to the localization block, where different sensors are used to localize. Intel d435 depth camera, XSens IMU, and the Kobuki odometry are used to localize in indoor environments. The XSens GPS replaces the Kobuki odometry in outdoor environments. However, the CCN fusion is working independently of the environment type, since it depends on the depth camera and the XSens IMU only.

62

Chapter 4. Experimental Setup

In this section the used Hardware and Software is going to be introduced. Due to the link between the software and the hardware, the section will discuss the two things simultaneously.

4.1. Robotic Operating System (ROS)

The medium of all the used sensors and actuators is called ROS. ROS, which stands for Robotic Operating System [110], provides services to be used by interchangeable topics that are initiated by running nodes on different platforms.

ROS is not an operating system but the architecture of how it works and the way it passes messages between processes make it a unique framework that combines multiple frameworks.

ROS eases the communication between different sensors and actuators by initiating nodes that handle pieces of messages called topics.

ROS 1 is the most used platform and it requires the open source Linux operating system to get the administrator privileges. Recently, ROS 2 has been releases as a new version of ROS, which is planning to include Microsoft Windows 10 as an additional compatible operating system.

The architecture of ROS is shown in Figure 41. ROS works with nodes that publish or subscribe to different topics and deal with them accordingly. Every topic must have a certain message type, which is like a protocol, so that the publisher and receiver can communicate understandably. Topics can have multiple subscribers or multiple publishers.

Each topic has its own frequency, which is independent of the sensor frequency. For instance, a sensor can be modified to be slower than the publishing topic by showing the same reading multiple times consecutively. ROS Master is needed to register all the nodes under the same server to be able to communicate together.

The main advantage of ROS is that it provides a domain for all the sensors and actuators to communicate with each other in different patterns, different types and different locations.

63

Figure 41. An example for an architecture of ROS.

4.2. Kobuki

The robot used is Kobuki as mentioned previously. From software point of view, Kobuki is a ROS-enabled robot, which makes it integrable with variety of sensors and other actuators. It can also work with

MATLAB. In all cases, Kobuki is also available in simulation environments such as Gazebo. Table 11 shows the specifications of the Kobuki used.

Table 11: Specifications of Kobuki [88]. Maximum translational velocity 70 cm/s Maximum rotational velocity 180 deg/s (>110 deg/s gyro performance will degrade) Payload 5 kg (hard floor), 4 kg (carpet) Expected Operating Time 3/7 hours (small/large battery) Expected Charging Time 1.5/2.6 hours (small/large battery) Odometry 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm Gyro factory calibrated, 1 axis (110 deg/s) Sensor Data Rate 50Hz

ROS

ROS MasterMaster

Node

Node 1

Topic

Topic 1

Node

Node 2

topic

topic 2.1

topic

topic 2.22.2

Node

Node 3

Topic

Topic 3MessageMessage

64

These pieces of information (Table 11) are important to consider when using Kobuki. For example, the payload capacity is important, since the processing unit (laptop) is going to be mounted in addition to the sensors. This makes it important to know the maximum current that the motors of the Kobuki can allow.

Kobuki documentation availability is wide due to its ease of use and its adoption by a wide range of scholars and researchers, who are constantly developing and building their own algorithms. The ready used node in ROS is the one responsible for getting the state of Kobuki. It is also worth mentioning that there are ready navigation nodes for Kobuki. However, the used algorithm was developed to match the path-planning method that was discussed earlier. Since ROS is where Kobuki and other sensors are

communicating, the designed algorithm can be written in any language that ROS can support, and it is not limited to one language. However, for direct programming, Kobuki can only understand C++.

4.3. IMU

The used IMU shown in Figure 42 is called XSens MTi-G-710 GNSS. It is a 10 DOF IMU that consist of 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, and barometer.

Figure 42. The XSens IMU used along with GPS receiver [111].

As mentioned before, the magnetometer is not used due to disturbances that may arise in the environment. However, the other IMU sensors are used for the localization. The associated driver/node that comes for the IMU is ROS-enabled. It is also adjustable,

to give different type of results, such as switching between Euler angles and quaternions. The GPS mounted on the IMU is automatically enabled when there is signal. However, it can be switched off manually. Table 12 shows the specifications of the used IMU.

Table 12: Specifications of the IMU [111].

Input voltage	4.5-34V or 3V3
Typical power consumption	450-950 mW
IP-rating	IP 67 (encased)
Sampling frequency	10 kHz/channel (60 kS/s)
Output frequency	Up to 2 kHz
Standard full range gyro	450º/s (1000 º/s optionally)
Standard full range acc	200m/s2
In-run bias stability gyro	10º/h
Bandwidth gyro	

415 Hz

Bandwidth acc

375 Hz

4.4. Depth Module/Camera

The used depth camera is manufactured by Intel and called Intel d435. This camera as explained earlier supports two types of depth localization (Structured Light technology and Stereo technology). Figure 43 shows the layers of Intel d435 followed by Figure 44 which shows the locations of the visual and depth sensor on the module. Moreover, the specifications of the intel d435 module is shown in Table 13.

Figure 43. Layers of the Intel d435 module [112].

66

Figure 44. Sensors locations of the Intel d435 module [112].

Table 13: Specifications of Intel d435 module [112].

Depth Technology

Active IR stereo

Main Intel® RealSense™ component

-Intel® RealSense™ Vision Processor D4

- Intel® RealSense™ module D430

Depth Field of View (FOV)

(Horizontal × Vertical × Diagonal)

87°±3° x 58°±1° x 95°±3°

Depth Stream Output Resolution

Up to 1280 x 720

Depth Stream Output Frame Rate

Up to 90 fps

Minimum Depth Distance (Min-Z)

0.1 m

Maximum Range

Approx. 10 meters; Varies depending on calibration, scene, and lighting condition

RGB Sensor Resolution and Frame Rate

1920 x 1080 at 30 fps

Intel d435 is a ROS-enabled camera, and it has its own user interface that is compatible with all operating systems. However, as explained earlier, the built-in driver along with RTAB-MAP [85] is used to extract the location from the module in ROS only.

4.5. Miscellaneous

The processing unit has the operating system Linux installed on in, since ROS can only be installed fully on Linux as mentioned earlier. The version of the operating system is Linux 14.04, where it has ROS Indigo installed on it. This version of ROS is reliable and has many documentations.

However, some items have been updated in it to make it compatible with the used hardware. It is also worth mentioning that the specifications of the processing unit are important for the performance of the localization process, especially the one for the Intel d435 module (depth localization). The specifications of used laptop are shown in Table 14.

Table 14. Specifications of the used processing unit.

Processor	Intel Core I7 7th gen
VGA Card	Nvidia 1050 4 GB
RAM	12 GB
USB Ports	3xUSB 3.1 + 1 thunderbolt
Storage	1 TB HDD + 128 GB SSD

The presented specifications indicate how many sensors and actuators can be connected at once. In addition, they show the capacity and speed of which the results can be recorded. Figure 45 show the all the sensors in addition the laptop mounted on the Kobuki.

Figure 45. Experimental Setup (all sensors are mounted on the Kobuki).

Chapter 5. Results

5.1. Testing the devices

In this section, each sensor will be tested alone, and its performance will be evaluated. To make sure that all the devices are well calibrated, the results of each device was recorded in outdoor and indoor environments.

5.1.1. Outdoor testing. In this section, the recorded data will be compared to real images from google maps, which will indicate the performance difference among the sensors and will show the difference in frames from a sensor to another.

In this test, the localization data from the Intel d435 depth camera and the XSens GPS module are going to be discussed. The first test was taken in the morning at 10:00 in the university area for a better vision for the depth camera.

The maximum speed was 10 km/h recorded by google maps application, and all sensor were mounted on the vehicle. Figure 46 shows the actual image of the used path taken from google maps compared to the image taken from the GPS module.

Figure 46. The actual path taken from google maps compared to the image taken from GPS module.

The stopping of yellow part continuity, shown in Figure 47, was the time at which it started raining, which affected the camera vison adversely, and resulted in frames losses which led to failure in the depth localization, as shown in Figure 48.

69

Figure 47. The time at which the rain started and depth camera lost vision.

Figure 48. The recorded data from the depth camera.

70

It can be inferred that the depth camera is sensitive to the environment noises and conditions. In other words, the camera requires the light to be directly reflected of the environment to be able to capture the depth image, process it and localize the frame taken with respect to the previous taken frames.

Conditions such as raining (water drops), low light exposure or even tough jerks and vibrations would affect the camera and distort the frame taken.

The second test was done in a sunny day near the university city on Sunday at 11:00. The place was chosen due to the lack of traffic. The consideration of preventing the previous distortion sources was taken. Figure 49 shows the image taken from google maps for the second chosen path.

Figure 49. The image taken from google maps.

Firstly, the test was interrupted when a quick turn has been made by the vehicle, this caused a loss in frames in the camera, which interrupted the localization process. Figures 50 and 51 show the data for the depth camera and the GPS module respectively. To prevent such error shown from happening, rotation must be slow enough to allow the camera to process the new frames since the features in every next frame will dramatically change.

71

Figure 50. Interrupted recorded data test (depth sensor).

Figure 51. Interrupted recorded data test (GPS module).

72

In linear movement the camera has access to the previous features from the same objects in the environment easily. However, in rotational movement the objects are quickly disappearing from the frame, which makes it very sensitive to the speed at which the camera is rotating.

Therefore, to fix this, the same test was done again while considering reducing the speed while the vehicle (or depth camera) is rotating. In addition, to forcing the camera to repeat the localization process from where it left of was attempted. Figures 52 and 53 show the results for the second trial.

Figure 52. Shows the second trial for recorded depth localization data.

As shown the camera was able to process all the frames even when some frames were dropped, resulting in full localization for the shown path. Figure 54 shows the recorded localization data from the GPS and depth camera on the top of the actual path.

73

Figure 53. The second trial for recorded GPS data.

Figure 54. The sensors data and the actual path on the top of each other.

The difference shown in Figure 54 between the GPS and the depth camera paths

is due to the difference in mounting location on the vehicle. However, the constructed

Actual

d435

GPS

74

map by the depth camera is saved as a database file, that can construct the registered map at any time, as shown in Figure 55.

Figure 55. Constructed map by the depth camera.

5.1.2. Indoor testing. Since GPS is not feasible in indoor environments, depth camera was used for the indoor testing. The test was done in the old MTR lab in EB1. The recorded map is shown in Figure 56, where the localization path is shown in Figure 57.

Figure 56. The recorded map including the path taken while recording the map.

75

Figure 57. The path taken from the depth camera.

5.2. Navigation

In this section the steps of navigation are going to be discussed. Figure 58 shows the sequence of navigation for the mobile robot. The navigation has taken place in the western part of the mechatronics laboratory in AUS.

Figure 58. The navigation process for the selected task.

5.2.1. Mapping, path-planning and waypoints. As a first step of navigation, the map of the environment, which is the mechatronics lab is predefined using MATLAB. Precisely, the western part of the MTR lab, where the all the desks are distributed. ‘O’ Represents the starting point of the robot, where ‘x’ represents the destination. Once the starting point and destination are defined, the cost-map can be generated as illustrated in Figure 59. As shown, the obstacles are considered to have infinite cost, where the gradual increase of the darkness of the yellow color indicates

Mapping

Mapping

path

path-planningplanning

Generating

Generating waypointswaypoints

Localization

Localization (CCN) (CCN) -Proposed Proposed SolutionSolution

Control

Control

an increase in the cost. Then, the A* algorithm can start planning the path based on the generated cost-map as shown in Figure 59.

Figure 59. The predefined map along with the path needed to reach destination.

The generation of waypoints will be done once the path is defined. Although the path will have very repetitive – unnecessary – waypoints, it will be optimized to confirm points at which there is a change in the orientation. Table 15 shows the waypoints generated from the above path.

Table 15. The waypoints generated of the shown path.

The above waypoints are to be fed to the robot through ROS, which will engage the control program and will use the sensors to know the location of the robot as it is moving.

X (m)

Y (m)

YAW (rad)

2.5

2.5

π/4

5

2.5

0

5

5

π/2

77

5.2.2. Localization. In this section different approaches of localization will be shown and discussed. Outdoor localization will be firstly discussed followed by indoor localization, then a small comparison will be made between them. Different methodologies of neural networks are going to be discussed and compared, leading to decide on the best methodology. Although localization was done in real time, the use of neural networks was firstly used in offline mode.

Outdoor localization

In this section, an example will be explained in which localization using GPS module, IMU, Depth Camera and the fusion of the last two will be shown. Sensors were mounted on the Kobuki along with the laptop.

GPS data, IMU raw data and depth camera data are recorded and saved in MATLAB workspace. Using the neural networks toolbox, IMU raw data (INPUT) were trained to the depth camera data (OUTPUT) as shown in Figure 60, while GPS data were used as a reference.

Figure 60. The OUTPUT and INPUT data for the neural networks training.

While setting the hidden layers and hidden neurons values constant, two approaches of neural networks were used (ANN, CCN). The hidden neurons values for both approaches were 10, 15 and 20 in one hidden layer, where the training percentage varies from 60% to 100% with 10% increment.

78

The recorded data of the different sensors are shown in Figures 61 – 63. Firstly,

the data are shown on the top of each other. Then, the data is plotted with respect to

time along with a zoomed portion of it.

Figure 61. The recorded path by Depth camera, GPS and IMU on top of each other.

Figure 62. The change in x and y with respect to time in the three different sensors.

79

Figure 63. Zoomed view on recorded x and y localization data with respect to time.

Figures 64 – 67 show the estimated path for (x and y) resulted from the fusion

between the depth camera and the IMU data using CCN and ANN. Each figure is marked with its training percentage, the NN methodology and its performance (the lower the value the better). Figures 68 – 71 show error associated with the previous estimated path figures.

Figure 64. The resulted path at different training percentages while using CCN methodology for the x axis.

80

Figure 65. The resulted path at different training percentages while using ANN methodology for the x axis.

Figure 66. The resulted path at different training percentages while using CCN methodology for the y axis.

81

Figure 67. The resulted path at different training percentages while using ANN methodology for the y axis.

Figure 68. The associated error for the previous results while using CCN methodology for x axis.

82

Figure 69. The associated error for the previous results while using ANN methodology for x axis.

Figure 70. The associated error for the previous results while using CCN methodology for y axis.

83

Figure 71. The associated error for the previous results while using ANN methodology for y axis.

As can be inferred, best performance happened when the training rate was 70% – 80%. It can be more obvious by plotting the error histogram for the three different network sizes (10, 15 and 20) as shown in Figures 72-74 for different hidden neurons.

Figure 72. Error histogram for different training percentages at hidden neurons = 10.

84

Figure 73. Error histogram for different training percentages at hidden neurons = 15.

Figure 74. Error histogram for different training percentages at hidden neurons = 20.

Moreover, as shown in Table 16, most of the error values were at their minimum when the used methodology was CCN and the training percentage was 80%. At 100%, data was overfitted, since the errors were relatively higher and very unstable at some point, even when a similar path was used. Errors at 60%, 70% and 90% were very close to the errors at 80%. However, this is the average error, therefore, the accumulative error would show a significant difference among the training percentages.

Table 16. Average errors for different network sizes at different training percentages.

Hidden neurons

NN Methodology

Axis

Training percentage

60%

70%

80%

90%

100%

10

CCN

x

0.031

0.025

0.015

0.022

0.055

y

0.023

0.037

0.014

0.018

0.104

ANN

x

0.036

0.024

0.014

0.021

0.086

y

0.021

0.021

0.014

0.020

0.124

15

CCN

x

0.033

0.027

0.015

0.023

0.067

y

0.026

0.022

0.016

0.023

0.203

ANN

x

0.036

0.022

0.014

0.021

0.091

y

0.032

0.020

0.016

0.021

0.137

20

CCN

x

0.035

0.022

0.017

0.023

5.204

y

0.039

0.030

0.024

0.021

0.206

ANN

x

0.058

0.026

0.014

0.022

0.094

y

0.033

0.025

0.017

0.020

0.114

Indoor localization

In this section, the same test is applied, but with replacing the GPS with the Kobuki odometry (which uses the wheel encoders) as a reference for the localized data. Therefore, the difference of errors between the outdoor and the indoor tests is expected to be very low, since no change was done on the input and output modules of the neural network. Figures 75 – 77 show the recorded localization data from the depth camera, IMU and Kobuki odometry in indoor environment for the same path. Figure 78 – 80 show the error histogram for the indoor test. Table 17 shows the error values obtained from the error histogram.

86

Figure 75. The recorded path by Depth camera, Kobuki odometry and IMU on top of

each other.

Figure 76. The change in x and y with respect to time in the three different sensors.

87

Figure 77. Zoomed view on the recorded x and y localization data with respect to

time.

Figure 78. Error histogram for different training percentages at hidden neurons = 10.

88

Figure 79. Error histogram for different training percentages at hidden neurons = 15.

Figure 80. Error histogram for different training percentages at hidden neurons = 20.

89

Table 17. Average errors for different network sizes at different training percentages.

Hidden neurons	NN Methodology	Axis	Training percentage			
			60%	70%	80%	90%	100%
10	CCN	x	0.041	0.25	0.018	0.026	0.072
		y	0.026	0.019	0.014	0.020	0.099
	ANN	x	0.046	0.029	0.020		

0.027

0.106

y

0.027

0.022

0.014

0.022

0.082

15

CCN

x

0.039

0.026

0.019

0.028

0.084

y

0.027

0.020

0.015

0.019

0.088

ANN

x

0.047

0.029

0.022

0.028

0.112

y

0.025

0.017

0.018

0.023

0.321

20

CCN

x

0.042

0.025

0.020

0.025

0.143

y

0.025

0.022

0.018

0.026

0.212

ANN

x

0.054

0.028

0.019

0.027

0.119

y

0.030

0.020

0.016

0.022

0.151

According to the previous table, the errors were very close to each other, however, the minimum values were at training percentage = 80%. As discussed earlier, the difference in errors between the indoor test and outdoor test is negligible. This is due to the similar used path while keeping the same neural networks arguments. Figures 81 – 83 show the errors of the outdoor and indoor test compared to each other.

Figure 81. Comparison for error for indoor and outdoor tests at hidden neurons = 10.

90

Figure 82. Comparison for error for indoor and outdoor tests at hidden neurons = 15.

Figure 83. Comparison for error for indoor and outdoor tests at hidden neurons = 20.

As shown the error difference between the indoor test and the outdoor test is

independent of the environment type, since the used sensors are independent as well.

As far as concerned, the depth camera needs to get enough light to work properly,

otherwise, no error difference is expected.

Localization using an S-shaped path

To verify the performance of the algorithm in different paths, an S-shaped path

was tested in indoor environment. Figures 84 – 86 show the data recorded for the Sshaped

path in indoor environment.

91

Figure 84. Recorded path by Depth camera, Kobuki odometry and IMU.

Figure 85. Change in x and y with respect to time in the three different sensors.

92

Figure 86. Zoomed view on recorded x and y localization data with respect to time.

As shown, the path is different than the trained path to some extent. It still has the behavior of the previous paths with more data. In other words, when ‘x’ axis changes, ‘y’ axis remains constant and vice versa.

Afterwards, the algorithm was applied resulting in the Figures 87-88 for 10 hidden neurons, followed by the associated errors shown in Figure 89. Table 18 summarizes the average errors for different training sets percentages at different hidden neurons.

The errors – as shown in Table 18 – have increased slightly, but the algorithm can still give very close results to the reference. The test mission was conducted for almost 800 seconds, where each sensor had its own frequency. Therefore, as shown in the previous figures, the raw IMU data is faster than the depth camera. The average errors shown in Figure 86 are with a range of 1 meter approximately.

93

Figure 87. Shows the resulted path at different training percentages while using CCN

methodology for the x axis at layers = 10.

Figure 88. Shows the resulted path at different training percentages while using CCN

methodology for the y axis at layers = 10.

94

Figure 89. Error histogram for different training percentages at hidden neurons = 10.

Table 18. The average errors for different network sizes at different training percentages.

Hidden neurons

NN Methodology

Axis

Training percentage

60%

70%

80%

90%

100%

10

CCN

x

5.005

0.793

0.429

0.445

6.261

y

2.057

0.646

0.1682

0.665

0.807

ANN

x

2.794

0.646

0.428

0.667

3.313

y

1.221

1.200

0.7236

1.237

1.503

15

CCN

x

5.405

0.855

0.4627

0.479

6.755

y

2.219

0.696

0.181

0.718

0.877

ANN

x

3.014

0.665

0.462

0.718

3.574

y

1.317

1.245

0.780

1.334

1.612

20

CCN

x

4.900

0.776

0.420

0.435

6.130

y

2.014

0.632

0.165

0.652

0.790

ANN

x

2.735

0.632

0.420

0.652

3.243

y

1.195

1.175

0.708

1.211

1.469

95

Localization using a different/planned path

Following the previous path-planning section, the shown path (new MTR lab – western part) was used to validate the performance of the neural network when used at a different path. Using the shown waypoints in Table 4, the following results were obtained as shown in Figures 90 – 93.

Figure 90. Depth camera localization data.

Figure 91. Localization data obtained from Kobuki odometry and raw IMU data.

96

Figure 92. Axes (x and y) with respect to time obtained from Kobuki odometry and raw IMU data.

Figure 93. Zoomed view on the recorded x and y localization data with respect to time.

97

The pervious data were trained without and with normalization which led to a

significant difference in the errors as will be discussed shortly.

Without normalization

At first results were trained using the original results, which led to higher error values as shown in Figures 94 – 96. Table 19 shows the error values obtained from the error histogram.

Figure 94. Error histogram for different training percentages at hidden neurons = 10

Figure 95. Error histogram for different training percentages at hidden neurons = 15.

98

Figure 96. Error histogram for different training percentages at hidden neurons = 20.

Table 19. The average errors for different network sizes at different training percentages.

Hidden neurons	NN Methodology	Axis	Training percentage				
			60%	70%	80%	90%	100%
10	CCN	x	1.085	0.5406	0.3636	0.1591	2.890
		y	0.4289				

0.3172

0.2213

0.1411

2.551

ANN

x

2.372

1.726

1.098

1.380

4.940

y

0.8572

1.098

0.0680

0.675

2.466

15

CCN

x

2.374

1.012

2.411

1.111

310.6

y

1.109

0.623

1.550

0.612

276.4

ANN

x

2.621

1.515

1.003

1.641

6.575

y

1.012

0.893

0.589

0.904

4.077

20

CCN

x

1.831

1.401

0.860

0.953

3.965

y

0.816

0.979

0.534

0.528

3.693

ANN

x

2.452

1.611

0.932

1.356

6.632

y

0.907

0.983

0.516

0.652

4.118

99

With normalization

As shown in Figures 97 – 99, after normalizing the data over their range

(Maximum – Minimum), the error has dropped significantly, as shown Table 20.

Figure 97. Error histogram for different training percentages at hidden neurons = 10.

Figure 98. Error histogram for different training percentages at hidden neurons = 15.

100

Figure 99. Error histogram for different training percentages at hidden neurons = 20.

Table 20. The average errors percentages for different network sizes at different training percentages.

Hidden neurons

NN Methodology

Axis

Training percentage

60%

70%

80%

90%

100%

10

CCN

x

0.101

0.063

0.044

0.035

0.228

y

0.038

0.051

0.024

0.022

0.174

ANN

x

0.133

0.132

0.040

0.068

0.273

y

0.042

0.059

0.021

0.031

0.149

15

CCN

x

0.202

0.061

0.031

0.087

0.328

y

0.078

0.049

0.022

0.041

0.272

ANN

x

0.064

0.056

0.049

0.074

0.655

y

0.048

0.033

0.033

0.044

0.254

20

CCN

x

0.044

0.117

0.023

0.026

0.352

y

0.028

0.068

0.023

0.034

0.659

ANN

x

0.109

0.069

0.062

0.099

1.872

y

0.038

0.044

0.036

0.049

0.093

101

In the previous tests, two main conclusions were reached. Firstly, 80% of training set was observed to provide the lowest amount of errors. Secondly, CCN was observed to give better performance and lower errors than ANN, while keeping the other parameters constant. This conclusion would verify the theoretical advantage of using CCN over ANN.

All MATLAB results shown above had two layers, the difference was in the hidden neurons. So, the hidden neurons were 10, 15 and 20 for the first layer, where the second layer had only one hidden neuron. However, the second layer is just a bias layer. Therefore, it can be dropped, since no bias was added. Figure 100 shows the topology of an ANN generated network by MATLAB for x axis.

Figure 100. Topology of ANN generated network used in MATLAB for x axis.

As shown, the previous figure had 10 neurons in the first layer and one neuron in the second layer. In other words, the network topology was 1-10-1, since each axis was separately trained and fed to the network.

Using feed-forward CCN

Based on the previous data, the feed forward CCN program used in [113] was modified to be used for the purpose of 2D localization. Unlike the NN created by MATLAB, feed-forward CCN had 12 layers, each layer had its own neuron (2-1-1-1…-1-2). This is dictated by the nature of the feed-forward CCN. In other words, after each layer is done, it gets shorted to one hidden neuron, then another layer is added if needed. Using the previously introduced Table 9, Figure 101 shows the topology generated by the feed-forward neural networks that was used by python.

102

Figure 101. Topology of CCN neural network used by python.

As shown in the previous figure, the number of hidden layers is 12, where each

layer had its own neuron. The main difference here is that all layers are interconnected

to each other. Moreover, the network has both x and y data combined which makes

them correlated to each other. The results of the depth camera, the CCN fusion and the

raw IMU data are shown in Figures 102 – 104, followed by Table 21 that shows the

performance details, where the training percentage was chosen to be 80%.

Figure 102. The localization data obtained from the depth camera and the sensor

fusion.

103

Figure 103. Axes (x and y) with respect to time obtained from camera and sensor

fusion.

Figure 104. Zoomed view on recorded x and y localization data with respect to time.

104

Figures 105-106 show the error and the error percentage, respectively. Errors

were calculated by subtracting Kobuki odometry data from the corrected IMU data.

Figure 105. The error in meters for x and y.

Figure 106. The error percentages for x and y.

105

The below results (shown in Table 21) show the performance achieved after fusing the depth camera data with the IMU data using CCN for 80% training set. As illustrated, the associated errors were low enough to consider this method as a substitute for the depth camera solution only or the IMU solution only.

Since it is not always feasible to get all the frames in the navigation process from the depth camera, – while the IMU data cannot be considered reliable – this method proves to utilize both sensors to enhance the localization performance. Therefore, the localization is more precise and always available. Comparatively, the best performance that was recorded by the ANN is shown in Table 22.

Table 21. Details of the feed forward CCN for the navigated path in the new MTR lab (western part).

Hidden layers
12
Hidden neurons
1 hidden neuron in each layer
Training percentage
80%
Average error in x (m)
0.069 m
Average error in y (m)
0.059 m
Error percentage in x
1.4%
Error percentage in y
1.2%

Table 22. Details of the feed forward CCN for the navigated path in the new MTR lab (western part).

Hidden layers
1
Hidden neurons
10 hidden neurons

Training percentage

80%

Average error in x (m)

0.113 m

Average error in y (m)

0.113 m

Error percentage in x

2.3%

Error percentage in y

2.3%

5.2.3. Comparison to literature. The obtained results were compared to the literature and observed to be among the lowest errors. Table 23 shows the average errors for selected literature that have used INS or depth camera sensor fusion, along with their fusion method.

106

Table 23. Results obtained in the literature using similar methodologies.

Literature

Average error (axis)

x

y

Robust INS/GPS Sensor Fusion for UAV Localization Using SDRE Nonlinear Filtering [114]: Nonlinear GPS/INS sensor fusion based on State-Dependent Riccati Equation for 3D localization. The method tries to solve linearization problem in other filters such EKF.

4.0513m

1.2580m

Robust Vision-Based Relative-Localization Approach Using an RGB-Depth Camera and LiDAR Sensor Fusion [115]: Vision-based localization using RGB-Depth camera and lidar for 2D and 3D localization. The fusion is done through adaptive color-based particle filter and interacting multiple model estimator.

Within 0.5 meters

Indoor Positioning using Sensor-fusion in Android Devices [116]: Wi-Fi-IMU-based localization for 2D environment to replace the GPS when it is unavailable. The algorithm averages the RSSI signal with estimated positions by the IMU unit in an android device using different algorithms.

Less than 2 meters

Indoor robot localization by RSSI/IMU sensor fusion [117]: The paper uses the Extended Kalman Filter (EKF) to fuse the IMU raw data and the RSSI/Wi-Fi based data to localize a mobile robot.

Approximately one meter

Sensor Fusion for Augmented Reality [118]: Kalman filter was used to fuse computer vision and inertial sensors data. The main of aim of the paper is to use this for augmented reality in a 3D indoor environment.

A range of -2 to 2 meters, MAE was less than 0.02 m

Visual–Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration [119]: Unscented Kalman Filter (UKF) was used to self-calibrate the transform between Visual sensor and Inertial sensor readings to remove the biases in 3D environment.

Within 0.2 meters

Hybrid tracking of human operators using IMU/UWB data fusion by a Kalman filter [120]: Kalman Filter is used to fuse UWB data and IMU data in indoor environment to track human operators more accurately.

Within 0.14m

Pedestrian Dead Reckoning-Assisted Visual Inertial Odometry Integrity Monitoring [121]: The authors have used the depth sensor in addition to IMU head reckoning to switch between visual Inertial odometers to pedestrian dead-reckoning to reduce the effect of environment to the depth camera.

2.13m

107

Chapter 6. Conclusion and Future Work

6.1. Conclusion

To enhance the process of navigation of mobile robots, three main tasks needed to be considered, which are mapping, localization and path-planning. The focus of this work was on the localization part.

Mapping was performed in advance and was fed to the path-planning algorithm as a cost map. The path-planning algorithm which is based on A* was able to reach the destination with the lowest possible cost, resulting in the planned path. To optimize the path, minimizing the waypoints was needed to reach the destination. This is performed by changing the orientation points in the path to be the new waypoints while keeping the distances unchanged.

Localization is another important part of the whole navigation process. A neural network method, called feed-forward cascade correlation, was selected to enhance the localization part of the navigation. The input of the neural network is the raw data from the Inertial Measurement Unit (IMU), and the output is the depth camera data. The performance of the neural network was compared to the GPS data in the outdoor environment and to the Kobuki odometry in the indoor environment.

Based on the errors shown in the results section, the CCN type of the neural network was proved to be better than ANN type. Moreover, the training set percentage of the network was selected based on the

lowest error given, as explained in the results chapter. The results showed that the error highly increases when the data is overfitted (100%) or underfitted (60%). Therefore, having the training set from 70%-90% was comparatively more stable for the given task than the 100% and 60% training sets. The training set percentage was set to 80%, since it had the lowest associated errors in the results section.

According to the results, the performance of localization using the feed forward CCN was best compared to ANN, with a network with 12 layers and no improvement was noticed with more layers.

6.2. Future Work

This section discusses the potential improvements to the presented work. The improvements are divided into hardware, software and combined improvements.

6.2.1. Hardware improvements. The Kobuki is a good platform, however, the requirement of having a laptop on top of it, increases the weight, which led to some difficulties in controlling the motion of the robot and excessive vibrations during the motion.

Another improvement that could be considered is to use wireless beacons for the localization and to compare the results with the proposed method, in this work. As wireless beacons have higher range than infrared cameras, yet they are less accurate, the comparison could show the tradeoff between the two methods.

6.2.2. Software improvements. The use of SLAM to simultaneously localize and map the environment can also be a software improvement. This would change the path-planning algorithm as well, since it needs to be updated in real-time as the map is being constructed. One of the limitations for this method is the used depth camera, since its SLAM node is not compatible with the Kobuki node.

This work can be extended to other platforms, e.g. drones. Indeed, the presented sensor fusion method, can easily be extended to 3D motion. For the hardware, there is a challenge to stabilize the camera, due to the tilt motion of the drone. This problem could be solved using a gimbal.

References

[1] L. Zhu, A. Yang, D. Wu and L. Liu, "Survey of Indoor Positioning Technologies and Systems", Communications in Computer and Information Science, vol. 461, pp. 400-409, 2014.

[2] H. Liu, H. Darabi, P. Banerjee and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems", IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 6, pp. 1067-1080, 2007.

[3] A. Alarifi et al., "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances", Sensors, vol. 16, no. 5, pp. 707, 2016.

[4] Blaz, "Basics", Indoor Positioning Systems based on BLE Beacons, 2015. .

[5] “Real Time Location Systems”, Clarinox Technologies Pty Ltd, [Online]. Available: http://www.clarinox.com/docs/whitepapers/RealTime_main.pdf,2009. [Accessed: 27- Dec- 2018].

[6] Y. Xiao, S. Zhang, J. Cao, H. Wang and J. Wang, "Exploiting distribution of channel state information for accurate wireless indoor localization", Computer Communications, vol. 114, pp. 73-83, 2017.

[7] G. Wurms, "Technologies for Server-Based Indoor Positioning Compared: Wi-Fi vs. BLE vs. UWB vs. RFID", positioning technology, 2017. .

[8] Peter Hauff and Michael Reinhardt and Stuart Foster. Ultrasound Basics. Global Drug Discovery Bayer Schering Pharma AG, Berlin, Germany, 2008, pp. 91-107.

[9] Y. Jin, Y. Pennec, Y. Pan and B. Djafari-Rouhani, "Phononic Crystal Plate with Hollow Pillars Actively Controlled by Fluid Filling", Crystals, vol. 6, no. 6, pp. 64, 2016.

[10] J. WONG, "How Do Ultrasonic Sensors Work", Ultrasonic techonoloiges, 2017. .

[11] N. Couronneau, “Performance analysis of assisted-GNSS receivers” Ph.D. dissertation, University of Cambridge, England, 2013. [Accessed: 27-Dec-2018]. Available: https://doi.org/10.17863/CAM.16638

[12] Van Diggelen, F. “A-GPS.” Journal of Computer and Communications, vol. 5 no. 6, April 28, 2017.

[13] Y. Xiao, S. Zhang, J. Cao, H. Wang and J. Wang, "Exploiting distribution of channel state information for accurate wireless indoor localization", Computer Communications, vol. 114, pp. 73-83, 2017.

[14] J. Vcelak, A. Zikmund and J. Kral, "Long range magnetic localization- accuracy and range study", Journal of Physics: Conference Series, vol. 450, pp. 012023, 2013.

[15] C. Lukianto, C. Hönniger, H. Sternberg, "Pedestrian smartphone-based indoor navigation using ultra portable sensory equipment", Indoor Positioning and Indoor Navigation (IPIN) 2010 International Conference, 2010, pp. 1-5.

[16] J. O. Nilsson, A. K. Gupta, P. Händel, "Foot-mounted inertial navigation made easy", Proc. Int. Conf. Indoor Position. Indoor Navig. (IPIN), 2014, pp. 24-29.

[17] "SparkFun IMU Breakout (MPU9250)", Canada Robotix, [Online]. Available: https://www.canadarobotix.com/products/2193. [Accessed: 28- Dec- 2018].

[18] T. Karamat, R. Lins, S. Givigi and A. Noureldin, "Novel EKF-Based Vision/Inertial System Integration for Improved Navigation", IEEE Transactions on Instrumentation and Measurement, vol. 67, no. 1, pp. 116-125, 2018.

[19] J. Seitz, T. Vaupel, S. Meyer, J. G. Boronat and J. Thielecke, "A Hidden Markov Model for pedestrian navigation," 2010 7th Workshop on Positioning, Navigation and Communication, Dresden, 2010, pp. 120-127.

[20] Renawi, M. Enhancement of Mobile Robot Navigation and Localization. [online] Available at: https://dspace.aus.edu/xmlui/handle/11073/9152 , 2017[Accessed 18 Oct. 2018].

[21] A. Valada, N. Radwan, and W. Burgard, “Deep auxiliary learning for visual localization and odometry,” IEEE International Conference on Robotics and Automation (ICRA 2018), 2018, pp. 6939-6946.

[22] T. Than, G. Alici, H. Zhou and W. Li, "A Review of Localization Systems for Robotic Endoscopic Capsules", IEEE Transactions on Biomedical Engineering, vol. 59, no. 9, pp. 2387-2399, 2012.

[23] I. Gőzse, "Optical Indoor Positioning System Based on TFT Technology", Sensors, vol. 16, no. 1, pp. 19, 2015.

[24] H.-r. Song, W.-s. Choi, S.-m. Lim, H.-d. Kim, "Target localization using rgb-d camera and lidar sensor fusion for relative navigation", Automatic Control Conference (CACS) 2014 CACS International, 2014, pp. 144-149.

[25] A. Censi, J. Strubel, C. Brandli, T. Delbruck, D. Scaramuzza, "Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 891-898.

[26] "OpenCV: Detection of ArUco Boards", Docs.opencv.org, [Online]. Available: https://docs.opencv.org/3.4/db/da9/tutorial_aruco_board_detection.html , [Accessed: 18- Apr- 2019].

[27] K. Kamarudin et al., "Method to convert Kinect's 3D depth data to a 2D map for indoor SLAM", 2013 IEEE 9th International Colloquium on Signal Processing and its Applications, 2013, pp. 247-251.

[28] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo", 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 2181-2188.

[29] "Ir Sensor Module For Arduino / Rasberry Pi", indiamart.com, [Online]. Available: https://www.indiamart.com/proddetail/ir-sensor-module-for-arduino-rasberry-pi-16016512830.html, [Accessed: 12- Nov- 2018].

[30] "Edge avoiding arduino robot", robotshop, [Online]. Available: https://www.robotshop.com/community/robots/show/edge-avoiding-arduino-robot, [Accessed: 14- Jan- 2019].

[31] H. Fransson, and G. Ehrenborg, “Sensor Fusion based Indoor Positioning with iBeacons,” MSc Thesis, Chalmers University, Sweden, 2015. [Accessed 2 Oct. 2018]. Available: https://hdl.handle.net/20.500.12380/248965

[32] M. Rainer, “Indoor positioning technologies.” Habilitation Thesis, Department of Civil, Environmental and Geomatic Engineering, Institute of Geodesy and Photogrammetry, ETH Zurich, switzerland, 2012. [Accessed 10 Oct. 2018]. Available: https://doi.org/10.3929/ethz-a-007313554

[33] T. Taylor, "Kinect for Robotics", Microsoft Robotics, 2011. .

[34] "How The Kinect Works", Depthbiomechanics.co.uk, [Online]. Available: http://www.depthbiomechanics.co.uk/?p=100, [Accessed: 14- Jan- 2019].

[35] N. Vesdapunt, and U. Sinha. Class Lecture, Topic: “Structured Light 3D Scanning” Carnegie Mellon University, Pittsburgh, PA 15213, United States, 2015.

[36] O. Wasenmüller and D. Stricker, "Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision", Computer Vision – ACCV 2016 Workshops, pp. 34-45, 2017.

[37] "Xtion 2 | 3D Sensor | ASUS Global", ASUS Global, [Online]. Available: https://www.asus.com/3D-Sensor/Xtion-2/specifications/, [Accessed: 13- Oct- 2018].

[38] M. Kalantari and M. Nechifor, "Accuracy and utility of the Structure Sensor for collecting 3D indoor information", Geo-spatial Information Science, vol. 19, no. 3, pp. 202-209, 2016.

[39] "Astra Series", Orbbec, [Online]. Available: https://orbbec3d.com/product-astra/, [Accessed: 13- Oct- 2018].

[40] intel REALSENSE, “Intel® RealSense™ Camera SR300 MM#943228”, H89061-XXX datasheet, June. 2016.

[41] "Time-of-flight camera", En.wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Time-of-flight_camera, [Accessed: 11- Nov- 2018].

[42] Sony SoftKinetic, “DepthSense 325”, datasheet, Sep. 2012.

[43] Katsumi Kimoto, Norihiro Asada, Takayoshi Mori, Yoshitaka Hara, Akihisa Ohya, Shin'ichi Yuta, "Development of small size 3d lidar", Robotics and Automation (ICRA) 2014 IEEE International Conference on, 2014, pp. 4620-4626.

[44] "YouTube", Youtube.com, [Online]. Available: https://www.youtube.com/watch?v=Qbo9mGAFkqc,2017 [Accessed: 17- Oct- 2018].

[45] Bebis. (2007). Stereo Camera. [ebook] 1(1). Available at: https://www.cse.unr.edu/~bebis/CS791E/Notes/StereoCamera.pdf, [Accessed 17 Oct. 2018].

[46] "ZED Stereo Camera | Stereolabs", Stereolabs.com, [Online]. Available: https://www.stereolabs.com/zed/, [Accessed: 18- Oct- 2018].

[47] S. Kang, “An Active Multibaseline Stereo System with Real Time Image Aqusition”, Carnegie Mellon University Technical Report CMU-CS-94-167, September, 1994. [accessed Dec 05 2018].

[48] Intel REALSENSE, “Intel® RealSense™ Camera D400-Series”, datasheet, Sep. 2017. [Accessed 25 Oct. 2018].

[49] "How does DGPS (Differential GPS) work?", [Online]. Available: https://racelogic.support/@api/deki/pages/488/pdf/How%2bdoes%2bDGPS%2b(Differential%2bGPS)%2bwork%253F.pdf?stylesheet=default, [Accessed 19 Nov. 2018].

[50] "How Does DGPS (Differential GPS) Work?", RACELOGIC Support Centre, [Online]. Available: https://racelogic.support/01VBOX_Automotive/01General_Information/Knowledge_Base/How_Does_DGPS_(Differential_GPS)_Work%3F, [Accessed 19 Nov. 2018].

[51] L. Yang, J. Qi, D. Song, J. Xiao, J. Han and Y. Xia, "Survey of Robot 3D Path Planning Algorithms", Journal of Control Science and Engineering, vol. 2016, pp. 1-22, 2016.

[52] H. Durrant-Whyte; N. Roy; P. Abbeel, "A Linear Approximation for Graph-Based Simultaneous Localization and Mapping," in Robotics: Science and Systems VII (MITP), pp. 41-48, 2012.

[53] S. M. LaValle, “Rapidly-exploring random trees a new tool for path planning,” Tech. Rep. 98-11, Computer Science Department, Iowa State University, Ames, Iowa, USA. 1998.

[54] Joon, "Improving the Optimality of RRT: RRT*", Joon's Lectures, 2011. .

[55] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, “Dynamicdomain RRTs: efficient exploration by controlling the sampling domain” Proceedings of the IEEE International Conference on Robotics and Automation, 2005, pp. 3856–3861.

[56] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods” Proceedings of the 49th IEEE Conference on Decision and Control (CDC ’10), 2010, pp. 7681–7687.

[57] H. Safadi, “Local Path Planning Using Virtual Potential Field”, Project, Spatial Representation and Mobile Robotics, McGill University, Canada, 2007.

[58] L. Kavraki, P. Svestka, J. Latombe and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566-580, 1996.

[59] C. Taylor, Online lecture, "Probabilistic Road Maps", Computational Motion Planning, University of Pennsylvania, USA, 2016.

[60] F. Zhan and C. Noon, "Shortest Path Algorithms: An Evaluation Using Real Road Networks", Transportation Science, vol. 32, no. 1, pp. 65-73, 1998.

[61] E. Dijkstra, "A note on two problems in connexion with graphs", Numerische Mathematik, vol. 1, no. 1, pp. 269-271, 1959.

[62] P. Hart, N. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968.

[63] "A* search algorithm", En.wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/A*_search_algorithm, [Accessed: 15- Nov- 2018].

[64] A. Stentz, "Optimal and Efficient Path Planning for Partially Known Environments", Intelligent Unmanned Ground Vehicles, vol. 338, pp. 203-220, 1997.

[65] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain", IEEE Transactions on Robotics, vol. 21, no. 3, pp. 354-363, 2005.

[66] R. Bhattacharya, "OPTRAGEN: A MATLAB Toolbox for Optimal Trajectory Generation," Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 6832-6836.

[67] D. Jia and J. Vagners, "Parallel Evolutionary Algorithms for UAV Path Planning", AIAA 1st Intelligent Systems Technical Conference, 2004, pp. 6230.

[68] C. Taylor, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Complex Adaptive Systems.John H. Holland", The Quarterly Review of Biology, vol. 69, no. 1, pp. 88-89, 1994.

[69] Yanrong Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04, 2004, pp. 4350-4355.

[70] A. Viseras, R. Losada and L. Merino, "Planning with ants", International Journal of Advanced Robotic Systems, vol. 13, no. 5, pp. 172988141666407, 2016.

[71] R. Glasius, A. Komoda and S. Gielen, "Neural Network Dynamics for Path Planning and Obstacle Avoidance", Neural Networks, vol. 8, no. 1, pp. 125-133, 1995.

[72] S. Li, M. Ding, C. Cai and L. Jiang, "Efficient Path Planning Method Based on Genetic Algorithm Combining Path Network," 2010 Fourth International Conference on Genetic and Evolutionary Computing, 2010, pp. 194-197.

[73] A. Viseras, R. Losada and L. Merino, "Planning with ants", International Journal of Advanced Robotic Systems, vol. 13, no. 5, pp. 172988141666407, 2016.

[74] C. Xiao-dong, Z. De-yun and Z. Ruo-nan, "New method for UAV online path planning," 2013 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2013), 2013, pp. 1-5.

[75] "Visibility graph", En.wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Visibility_graph, [Accessed: 07- Sep- 2018].

[76] L. Freda, F. Loiudice and G. Oriolo, "A Randomized Method for Integrated Exploration," 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 2457-2464.

[77] Z. Yiping, G. Jian, Z. Ruilei and C. Qingwei, "A SRT-based path planning algorithm in unknown complex environment," The 26th Chinese Control and Decision Conference (2014 CCDC), 2014, pp. 3857-3862.

[78] W. Nie, Q. Li, G. Zhong and H. Deng, "Indoor 3D path planning using a Kinect V2 sensor," 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), 2017, pp. 527-531.

[79] Shifei Liu, Yanhui Wei and Yanbin Gao, "3D path planning for AUV using fuzzy logic," 2012 International Conference on Computer Science and Information Processing (CSIP), 2012, pp. 599-603.

[80] S. Majumder and M. S. Prasad, "Three dimensional D* algorithm for incremental path planning in uncooperative environment," 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), 2016, pp. 431-435.

[81] Yao Cui and Guofeng Qin, "Intelligent path planning in 3D scene," 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), 2010, pp. V3-579-V3-583.

[82] L. Yang, Z. Wei-guo, S. Jing-ping and L. Guang-wen, "A path planning method based on improved RRT," Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference, 2014, pp. 564-567.

[83] J. Ni, L. Wu, S. Wang and K. Wang, "3D real-time path planning for AUV based on improved bio-inspired neural network," 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2016, pp. 1-2.

[84] B. Song, Z. Wang and L. Sheng, "A new genetic algorithm approach to smooth path planning for mobile robots", Assembly Automation, vol. 36, no. 2, pp. 138-145, 2016.

[85] "RTAB-Map", RTAB-Map, [Online]. Available: http://introlab.github.io/rtabmap/2013, [Accessed: 18- Oct- 2018].

[86] M. Labbé and F. Michaud, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation," in IEEE Transactions on Robotics, vol. 29, no. 3, pp. 734-745, 2013.

[87] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2661-2666.

[88] "About | KOBUKI", KOBUKI. [Online]. Available: http://kobuki.yujinrobot.com/about2/, [Accessed 25 Apr. 2019].

[89] R. Siegwart et al., Introduction to autonomous mobile robots. MIT press, 2011.

[90] J. Doscher, Class Lecture, Topic: "Accelerometer Design and Applications", Puig Castellar Institute, Spain, 2008.

[91] N. Yazdi, F. Ayazi and K. Najafi, "Micromachined inertial sensors," in Proceedings of the IEEE, vol. 86, no. 8, pp. 1640-1659, 1998.

[92] "Gyroscope", En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Gyroscope, [Accessed: 22- Jan- 2019].

[93] W. Tao, T. Liu, R. Zheng and H. Feng, "Gait Analysis Using Wearable Sensors", Sensors, vol. 12, no. 2, pp. 2255-2283, 2012.

[94] "Accelerometer & Gyroscope Sensor Fusion", Digikey.com, [Online]. Available: https://www.digikey.com/en/articles/techzone/2018/jan/apply-sensor-fusion-to-accelerometers-and-gyroscopes, [Accessed: 15- Feb- 2019].

[95] M. Tsai, Y. Liu, C. Sun, C. Wang, C. Cheng and W. Fang, "A 400×400μm2 3-axis CMOS-MEMS accelerometer with vertically integrated fully-differential sensing electrodes," 2011 16th International Solid-State Sensors, Actuators and Microsystems Conference, 2011, pp. 811-814.

[96] W. Premerlani, P. Bizard, "Direction cosine matrix IMU: theory." DIY Drone: USA, 2009, pp. 13-15.

[97] M. Borgelt, “Altitude vs Pressure Altitude.” Available: http://www.borgeltinstruments.com/GPSvsPressurealtitude.pdf 2011. [Accessed: 18- Feb- 2019].

[98] M. Ibrahim and O. Moselhi, "Inertial measurement unit based indoor localization for construction applications", Automation in Construction, vol. 71, pp. 13-20, 2016.

[99] M. Grewal and A. Andrews, "Kalman Filtering", 2008. Available: 10.1002/9780470377819.

[100] R. W. Longman, T. Kwon, and P. A. LeVoci, “Making the Learning Control Problem Well Posed – Stabilizing Intersample Error,” Advances in the Astronautical Sciences, vol. 123, pp. 1143-1162, 2006.

[101] K. Takanishi, M. Q. Phan, and R. W. Longman, "Multiple-Model Probabilistic Design of Robust Iterative Learning Controllers," Transactions of the North American Manufacturing Research Institution, Society of Manufacturing Engineers, vol. 33, pp. 533-540, May 2005,

[102] A. Gelb, “Applied Optimal Estimation. Cambridge,” England: MIT Press, 1974.

[103] R. Brown and P. Hwang, Introduction to Random Signals and Applied Kalman Filtering. Hoboken, NJ: John Wiley and Sons, 2012.

[104] P. Vanicek, and M. Omerbasic, “Does a navigation algorithm have to use Kalman filter?” Canadian Aeronautics and Space Journal, vol. 45, pp. 262-296, 1999.

[105] Mohamed, A. H. “Optimizing the estimation procedure in INS/GPS integration for kinematic application,” UCGE Reports 20127, Department of Geomatics Engineering, The University of Calgary, 1999.

[106] S. Haykin, “Neural Networks: A Comprehensive Foundation (2nd ed.),” Upper Saddle River, NJ: Prentice-Hall, 1999.

[107] C. Bishop, Natural networks for pattern recognition. Oxford: Oxford Univ. Press, 1995.

[108] E. Alpaydin, " GAL: Networks That Grow When They Learn and Shrink When They Forget.", International Journal of Pattern Recognition and Artificial Intelligence, vol. 08, no. 01, pp. 391-414, 1994.

[109] S. Fahlman and L. Christian, “The Cascade-Correlation Learning Architecture.” Advances in neural information processing systems, 1990, pp. 524-532.

[110] "Robot Operating System", En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Robot_Operating_System, [Accessed: 15- Jan- 2018].

[111] "MTi 100-series", Xsens.com, [Online]. Available: https://www.xsens.com/products/mti-100-series, [Accessed: 06- Mar- 2019].

[112] "Depth Camera d435 – Intel® RealSense™ Depth and Tracking Cameras", [Online]. Available: https://www.intelrealsense.com/depth-camera-d435, [Accessed: 25- Nov- 2018].

[113] “AndreeaMusat/Cascade-Correlation-Neural-Network.” GitHub. [online]. Available: https://github.com/AndreeaMusat/Cascade-Correlation-Neural-Network, 2018 [Accessed 6 May 2019].

[114] A. Nemra and N. Aouf, "Robust INS/GPS Sensor Fusion for UAV Localization Using SDRE Nonlinear Filtering", IEEE Sensors Journal, vol. 10, no. 4, pp. 789-798, 2010.

[115] H. Song, W. Choi and H. Kim, "Robust Vision-Based Relative-Localization Approach Using an RGB-Depth Camera and LiDAR Sensor Fusion", IEEE Transactions on Industrial Electronics, vol. 63, no. 6, pp. 3725-3736, 2016.

[116] U. Shala and A. Rodriguez, Indoor Positioning using Sensor-fusion in Android Devices, 1st ed. Kristianstad: Kristianstad University, 2011.

[117] V. Malyavej, W. Kumkeaw and M. Aorpimai, "Indoor robot localization by RSSI/IMU sensor fusion," 2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2013, pp. 1-6.

[118] J. D. Hol, T. B. Schon, F. Gustafsson and P. J. Slycke, "Sensor Fusion for Augmented Reality," 2006 9th International Conference on Information Fusion, 2006, pp. 1-6.

[119] J. Kelly and G. Sukhatme, "Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration", The International Journal of Robotics Research, vol. 30, no. 1, pp. 56-79, 2010.

[120] J. A. Corrales, F. A. Candelas and F. Torres, "Hybrid tracking of human operators using IMU/UWB data fusion by a Kalman filter," 2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2008, pp. 193-200.

[121] Y. Wang, A. Peng, Z. Lin, L. Zheng and H. Zheng, "Pedestrian Dead Reckoning-Assisted Visual Inertial Odometry Integrity Monitoring", Sensors, vol. 19, no. 24, pp. 5577, 2019.

Vita

Fares Amin Alkhawaja was born in 1996, in Dubai, UAE. He earned his High School Certificate from Al Hikmah School in Ajman, UAE. He was ranked the top graduate in his High School in 2013. He graduated from University of Sharjah in 2017 in Electrical and Electronics Engineering. He received the “First Honors” Academic Standing in bachelor’s degree.

Mr. Alkhawaja worked at schlumberger as an Electrical and Computer intern in Wireline segment, then he went to Portugal for a long internship in Ramos Ferreira. He started his master’s program in Mechatronics Engineering in 2017 after receiving a graduate assistantship from the American University of Sharjah.