Multi-source UAV-based object classification using CNN's and Data acquisition system for Robotic Skin

by

M.S. Raghavendra Sriram.

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2017

"But it's been no bed of roses, no pleasure cruise. I consider it a challenge before the whole human race, and I ain't gonna lose"

- *We Are the Champions/Freddie Mercury (1977)*

Acknowledgements

I am grateful to Dr. Frank Lewis for giving me an opportunity to work with him and mentoring my Master's thesis. I shall forever cherish that chance meeting of ours on December 1st, 2012 on the neederman hall bridge. It led to some of my most cherished years. Dr. Lewis provided an extraordinary level of attention and hands on guidance in, the shaping of my knowledge and skill base. I would also like to thank Dr. Dan Popa for letting me participate and contribute to some truly innovative research and his guidance throughout the years. I would like to thank them both for their critical assessment of my work. I would like to thank Dr. Yan Wan and Dr. William Dillon for serving on my committee.

I would also like to thank the various Research Assistants and employees at UTARI for their support and constant help. I would be remiss if I didn't thank the UTARI IT department especially Jared Beaty and Vinh, for continuously putting with a lot of frantic questions and issues and for finding solutions for various issues and most of all for being amazing friends and people during my time at UTARI. Thank you Norman Spayd, you will be missed.

I would like to thank my lab associates and friends, Patrik Kolaric, Rommel Alonzo, Sven Cremer, Sumit Das, Joe Sanford, Monica Beltran, JPaul and the other NGS group members over the years for their help during various projects. Cheers to all the great times at NH250!

A special thanks to Chaitanya Rani. If not for your help, a lot of things would not have been possible and as much fun. Mishka!

Kanishka, Rohit, Nayana, Soumitro, Sheetal, Shwetha, Dilip, Karthik,Patanjali, and all my

friends through the years, I must thank you for all the years of wild and crazy times. From the crazy times we had at House 140/141, to Bentleys and the parking lots across campus, lets bring them back soon!.

And finally, I would like to thank my Mom,Dad and my Sister, Lakshmi. If not for all the support and time and everything else, none of this and whats to come would be possible. From Phosphorus , 3 Dimensional vectors and to every problem ever, I love you all.

August 9th, 2017

Abstract

Autonomous robots are intelligent systems capable of performing tasks in the world by themselves, without explicit human control. Examples range from autonomous helicopters to Roomba, the robot vacuum cleaner to robotic manipulators. The numerous sensors on-board gather data related to the desired actions and this poses several challenges and design constraints in terms of computation and hardware design that can prove to be extremely difficult and expensive to avoid. The objective of this research is to study, develop and implement various Intelligent solutions to help solve several real-world problems with respect to multi agent configurations of unmanned systems. We shall see examples of integration of machine learning especially Deep learning capabilities in swarms.

We shall also discuss the development of robotic skin modules and the several constraints and design schemes adopted and tested to support the development of more social robots and help study and determine certain requirements and develop smart systems to help give robots a more natural sensory based interface with its surroundings.

Table of contents

Chapter 1

Introduction

Unmanned vehicle systems namely, unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs), have become a sought-after research area for industry as well and academics.

This thesis discusses two main topics - Autonomous systems and Assistive Robotics. In the first part, the potential of UAV-UGV collaboration and several design solutions and systems to work with each other in a distributed manner and with sensor based solutions for several well-known problems along with several human machine interface control mechanisms designed to provide a more intuitive and natural control interface is shown. A more distributed framework is also discussed.

Wireless mesh networks have been around for quite some time starting with the development of radio mesh networks to connect various clients as well as servers [1]. A mesh is an interconnected network of devices and systems that share information with each other. Mesh networks tend to have a fixed topology for fast and reliable communication. They can be dynamic in topology, but that comes at a cost of computation time of delivery route optimization. There are several implementations of wireless mesh networks already established and in use [2]. SMesh, an 802.11 multi-hop wireless mesh made it possible to allow real time applications such as VoIP[3]. Firetide[4] and Wave relay and systems that allow communication over multiple radio

bands using software defined radio. ZigBee, a low-cost radio module allows the creation of a network which allows small sensors and microcontroller systems to be easily integrated with each other. These are used in several smart home automation projects. Recently several consumer products have been increasingly introduced, that allows users to create a wireless WiFi based mesh networks at home.

Several protocols have been defined to allow better design and standardization of wireless mesh networks. Some of them are:

1. ZigBee: The protocol used by ZigBee wireless systems and several low-cost radios is based on the IEEE 802.15.4 standard. This protocol is best used in situations where the amount of data is very small and is transmitted over short distances and consumes very low power. This allows more abstraction of capabilities such as a self-healing network and is designed to utilize extremely low power when not being used for communication. The data is generally encrypted and has a size of 64k. Its maximum speed is 250 kBs [6].

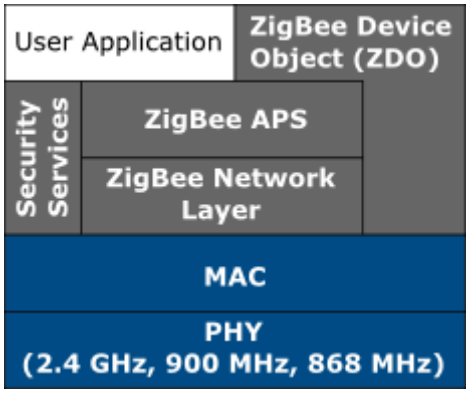

Fig 1. Zigbee Protocol architecture[6]

2. SNAP: SNAP is a high-performance stack that runs efficiently on 8-bit microprocessors. Along with wireless mesh capabilities it is extremely small in memory usage (~ 40 KB). Since SNAP includes a python virtual machine, its performance parameters can be changed and it does not require recompilation every time.

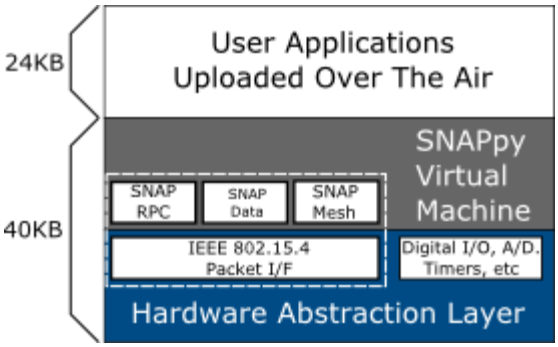

Fig 2. SNAP Protocol architecture [6]

3. WirelessHART: This is based on the HART protocol developed for use in industrial applications and automation. This protocol allows real time communication and uses a Time Synchronized Mesh Protocol (TSMP). Some key features are Industry standard 128-bit AES Encryption, Multi-tiered security, and data integrity and device authentication [6].

OSI Layer	HART
Application	Command Oriented. Predefined Data Types and Application Procedures
Transport	Auto-Segmented transfer of large data sets, reliable stream transport, Negotiated Segment sizes
Network	Power-Optimized Redundant Path, Mesh to the edge Network
Data Link	Secure, Time Synched TDMA/CSMA, Frequency Agile with ARQ
Physical	2.4 GHz Wireless 802.15.4 based radios, 10dBm Tx Power

Fig 3. WirelessHART Protocol architecture.

4. OpenWrt: OpenWRT is an open source writable file system management tool which allows easy configuration of Wi-Fi based devices. This allows application based selection and customization capabilities to suit user's needs. For developers, OpenWrt is a framework that allows reconfiguration of systems without the need to create separate firmware for devices. For users, this means the ability for full customization, to use the device in ways never envisioned. It supports several Wi-Fi protocols and acts as a separate configuration layer over these protocols [7].

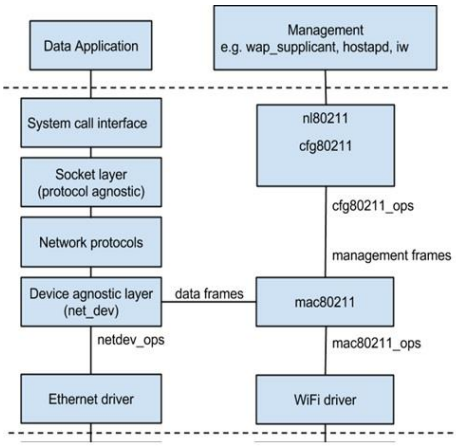

Fig 4. OpenWRT Protocol architecture

In this report, implementation of a multi-robot wireless mesh network where data can be shared amongst multiple systems and certain tasks can be performed even with constrains is studied.

An application of using the mesh network/system capable of behaving like a flying cluster of drones which can classify images using a convolutional neural network (CNN) trained using

the ImageNet dataset [8] and the inception CNN model [9] is shown. This is part of a robotic challenge that is developing an aerial robotic solution for complete autonomous farm mapping solutions.

A social robot is an autonomous robot that interacts and communicates with humans or other autonomous physical agents by behaviors attached to its design [10]. It also deals with study of methods to help reduce the gap between robots and human interaction and make the interaction more natural and intelligent. Robots that can interact and communicate among themselves, with humans, and with their immediate environment, within the social and cultural structure, need to appeal to the human in a manner like the user's expectations in various social interactions. This requires the need to make the robots not only appear more human-like but also feel more human and one such way is to provide the robot with a more human-like interface to interact with the environment physically. Several social robots have been used for research purposes such as the Phillip K Dick android [11] developed by Hanson Robotics. The first version was built in 2005 by Hanson Robotics with David Hanson, Andrew Olney with the Fedex Institute of Technology team, the University of Texas at Arlington Research Institute, UT Dallas and others. The robot interacted with various people orally and visually. However, it lacked a smart physical and human-like interaction system. This is one example of the gap between the robot and human. The skin developed for the android was a silicone based material which lacked any sensory capability [12].

Fig 5. Phillip K Dick Android developed by Hanson Robotics

In the second part of the thesis, the design of a data acquisition system for the design and implementation of artificial skin modules for use on social robotic systems to provide them with an additional natural interface between the robot and the immediate environment along with several sensory features such as pressure sensing, non-contact thermal sensing is discussed. It also addresses several design challenges such as scalability and communication.

Chapter 2

Ground-based Robotic Platforms

2.1 Platform design

Several robotic platforms were designed and modified according to their specific tasks. Some of them are detailed below with specifications.

1. DrRobot Jaguar 4x4 - The Jaguar 4x4 designed by DrRobot [13], is a 4-wheel Robotic Platform and is developed mainly for indoor and outdoor operation requiring higher ground clearance and fast maneuverability. It is driven by four powerful 80Watt motors. The Jaguar-4x4-wheel platform is rugged, weighing in at less than 40lbs, having a top speed of about 7 mph, compact, weather and water resistant. It is designed for rough environments and can run over vertical step up to 155mm and climbing up low-rise stairs (up to 110mm step). The Jaguar-4x4-wheel is WiFi enabled with two wireless routers. It was converted from a 2-channel drive system to a 4-channel drive system allowing for individual wheel control, allowing much faster responses in constrained situations. An on-board computer system running Ubuntu 14.04 and ROS indigo was also installed on the robot. This allows it to perform all the computing on-board the system instead of off-board and allowed integration of several sensors and planning algorithms on-board. A modular payload tray allows it to be used as a mobile landing or takeoff pad for UAVs using a ArTag, which is a 2D image similar to a QR code which can be designed for identification, etched on it. The UAV can land, take off or follow the system using this ArTag.

There is also a payload tray with a wireless charging pad, that allows a swarm to fly down to the charging pads, charge wirelessly and take off again. The design of each specific charging unit solves the main issue of the UAV landing uncertainty, i.e. error in final landing position. A motorized X-Y stage below each pad, detects the location of the landed UAV from a resistive touch film on it, and moved the secondary charging coil to the exact location below the UAV. Thus, giving more flexibility to the location and increasing the charging efficiency.

Fig 6. Jaguar 4x4 with various payloads (AR tags, Wireless charging)

2. Husky A200: The Clearpath Robotics Husky A200, as shown in Fig. 7 carries an Xsense MT700 IMU with integrated GPS and a piksy RTK GPS. It is equipped with a digital compass for heading information, as well as an optical encoder on each side of the differential drive for odometry measurements and velocity control. The modified system also includes a Kinect Camera and a Hokuyo LIDAR mounted on a set of specially fitted dynamixel motors which allows it to pan and tilt adding another dimension to the 2D LIDAR. The UGV can be

controlled by requesting linear and angular velocities, and is an ideal option for the kinematic control scheme developed and used via the ROS navigation stack along with the various localization methods.

Fig 7. Husky A200

It also consists of a long-range transmitter which allows for serial communication over an extended area of over 2 miles from the base station. An Intel NUC computer was added onto the system and connected to allow integration of various sensors and processing of data. This allows it to run various tasks such as mapping and navigation on-board. The on-board Wi-Fi router supports OpenWRT and hence can also be integrated into the Wi-Fi mesh set up in the lab. This allows the system to share data and maps with multiple robots and utilize the right maps when needed.

2.2 Software Architecture

Several open source packages were integrated and implemented on the system, allowing for various tasks:

1. Combined odometry using robot_pose_ekf: The Robot Pose EKF [14] package is used to estimate the 3D pose of a system, based on partial pose information coming from various sources. It uses an extended Kalman filter with a 6D model (3D position and 3D orientation) to combine measurements from wheel odometry, IMU sensor and visual odometry. Sensor fusion is a concept involving the combination of data from multiple sensors intelligently and use the result to improve the performance and estimates of certain states of a system. It allows us to identify several sources and reduce any errors that may be inherent in certain kinds of data. For estimating states, several methods of filtering and estimation are available. The most common technique is to use an Extended Kalman Filter. The basic idea is to offer loosely coupled integration with different sensors, where sensor signals are received as ROS messages. The package, allows improved odometry information using the GPS, RTK GPS, visual odometry from the RTAB mapping package using the Kinect along with the Xsense IMU used on the system. The stack is a multi-sensor fusion framework, that refers to one sensor as a fixed sensor and the others as update sensors. The input format is the 2D pose from the odometry (*x, y and yaw angle*), the 3D orientation from the IMU (*roll, pitch, yaw*) and the visual odometry from the camera which is the full 3D pose (*x,y,z and roll, pitch, yaw*). The *odom* message is converted into the a

vector3 of quaternions. The IMU is converted into an orientation vector with the *base_footprint* as reference and finally *visual odom* is converted into a pose vector. All the three are concatenated into the buffer and then passed into the Extended Kalman Filter node. More than one of each can be used and the data can be filtered or averaged before being passed to the node as well.

Basic Algorithm:

1. Collect data from sensors and verify
2. If valid, convert them with respect to the base frame of reference
3. Depending on the data rate, each sensor information is stored until information from all the sensors are available. Each of the data received has its own time stamp
4. Once all the data is consolidated, the EKF (as defined in the Orocos-BFL library) is updated for each sensor when all the sensor data is available (i.e, if the data from the odometry and the data from the IMU is received at time t_1(>t_0) and the data from the visual odometry is received at time t_2(>t_1), then the filter is run for all the three sets of data at time t_1).
5. This fused sensor data is converted into an odometry msg and published on the topic /odom_combined

The Extended Kalman Filter was developed mainly for nonlinear discrete time state estimation as a normal Kalman filter. The Kalman filter is a method based on recursive Bayesian filters where the noise is known and is assumed to be gaussian. The Extended Kalman filter is an extension of the Kalman filter where the system is first linearized around the initial states and then filtered. Owing to the inherent errors in linearization, the EKF, unlike the Kalman filter is not optimal. However, for most nonlinear cases it works much better than the regular filter. The EKF can be represented in standard state space form as shown below,

$$x_k = f(x_{k-1}, u_k, k) + w_{k+1}$$

$$y_k = h(x_k, u_k, k)$$

Where,

k denotes a discrete point in time (with *k-1* being the immediate past time point).
u_k is a vector of inputs.
x_k is a vector of the actual states, which may be observable but not measured.
y_k is a vector of the actual process outputs.
$\tilde{y}_k$ is a vector of the measured process outputs.
w_{k+1} and v_k are process and output noise respectively. They are assumed to be zero mean Gaussian with covariance Q_k and R_k respectively.
$f(.)$ and $h(.)$ are generic non-linear functions relating the past state, current input, and current time to the next state and current output respectively.

Given the inputs, and assuming the appropriate noise for the sensors, the estimated states, $\hat{x}_k$ and $\hat{y}_k$ are the estimated outputs. In a regular Kalman filter, the first step is the projection of the most recent state estimate and estimate of the error covariance forwards in time to predict the estimates of the current time. The next step is updating the predicted estimate with the most recent measurement to generate the state estimate. Due to the nonlinear nature of the states, the covariance prediction and the update equations first need to be linearized and thus their jacobians are used, which are,

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{(\hat{x}_k, u_{k,} k)} , \; H_k = \left.\frac{\partial f}{\partial y}\right|_{(\hat{y}_k, u_{k,} k)}$$

The Prediction step can be written as

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, k)$$

$$P_k^- = F_{k-1}P_{k-1}F_{k-1}^T + Q_k$$

and the measurement correction step is,

$$K_k^- = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(\tilde{y}_k - h(x_k^-, u_k, k))$$

$$P_k = (I - K_k H_k)P_k^-$$

Where P_k is an estimate of the covariance of the measurement error and K_k is the Kalman gain. With this information, the necessary states are obtained. Multiple sensors of the same type can also be used. The result of sensor fusion with an EKF running on one sensor and multiple sensors are shown in figure 8.

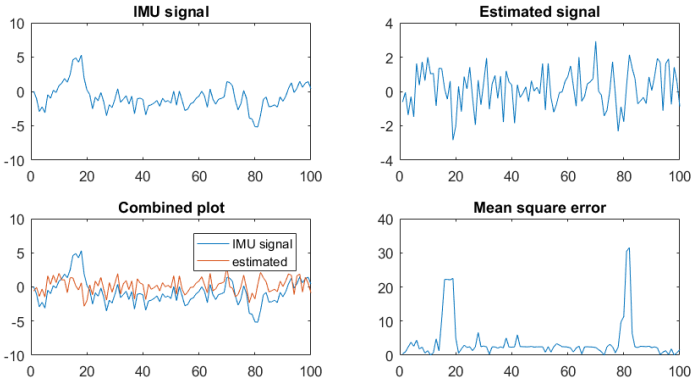

Fig 8. EKF with single input

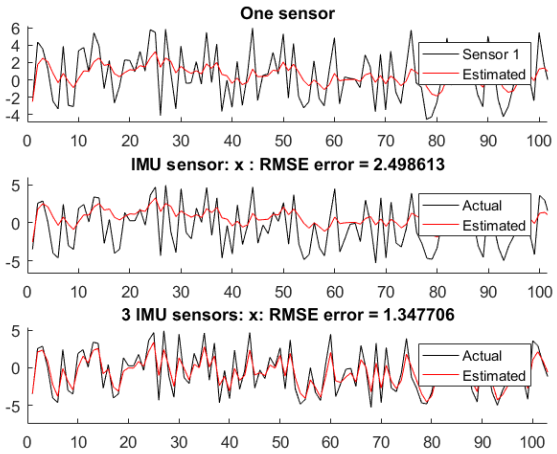

Fig 9. EKF with 1 IMU and with multiple IMU sources

From the above results, figure 9, the estimation gets better with three sensors rather than one and the mean squared error reduces from 2.49 to 1.34.

2. loam back and forth - This package, developed by Ji Zhang [15], is a real time odometry detection method that allows the use of a 2 axis LIDAR to localize itself and map its surroundings while moving in 6DOF. In the two algorithms developed, one algorithm allows localization using a point cloud and performs odometry at high frequency and low fidelity and another algorithm generates maps using scan matching and pose registration in the point cloud.

The package is made up of four nodes. The *scanRegistration* node saves all the laser data in a stack and publishes the data as a point cloud at the end of each sweep. The *laserOdometry* node estimates motion of the lidar between two sweeps, at a higher frame rate. By keeping

track of the odometry data, the *laserMapping* node creates and saves a map. It also consists of a transformation node that determines the pose of the laser and the states of the system. Two dynamixel motors, mounted on each other in a pan-and-tilt configuration are used. This allows a variety of ways to hold the LIDAR and use it. The benefit of this is the ability to choose the configuration of the LIDAR according to the situation.

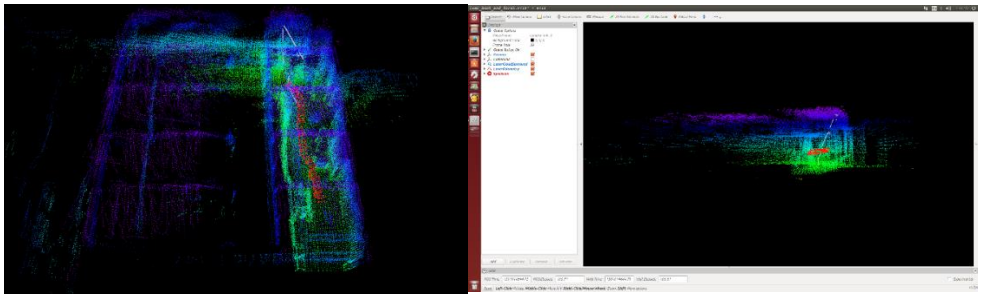

Fig 10. Loam running on nsh_dataset

3. RTAB mapping - Real-Time Appearance-Based Mapping is a RGB-D Graph-Based SLAM approach based on an iterative appearance-based loop closure detection algorithm. It utilizes a bag-of-words technique to identify features [17] present in various images and identify the probability of the features matching and determining loop closure. When a loop closure solution is accepted, the graph is optimized and the errors minimized. RTAB mapping is implemented using an Xbox Kinect camera mounted on the vehicle. These 3D maps are saved onto a single shared server and on the client system, and can be accessed by various robots [16].

Fig 11. Dimensional Pointcloud maps using real time appearance based mapping
(Left: UTARI Atrium, Right: Navigation within the 3D map and odometry in blue)

4. Gmapping – Gmapping is a package developed with the ROS navigation stack which allows the creation of grid-based maps. It is built on the OpenSLAM [19] software. It utilizes the odometry information to localize and traverse the maps generated. In conjunction with the ROS navigation stack, it can detect obstacles and simulate a potential field bubble around the obstacle allowing for obstacle avoidance during navigation. It reduces the amount of data to process by using a local costmap for immediate navigation goals such as obstacle avoidance and motion planning, while a global costmap, saves all the mapped data and generates the global map for localization. The global planner uses the global costmap to generate a long-term plan while the local planner uses the local costmap to generate a short-term plan. The package also can be used with preexisting maps to traverse.

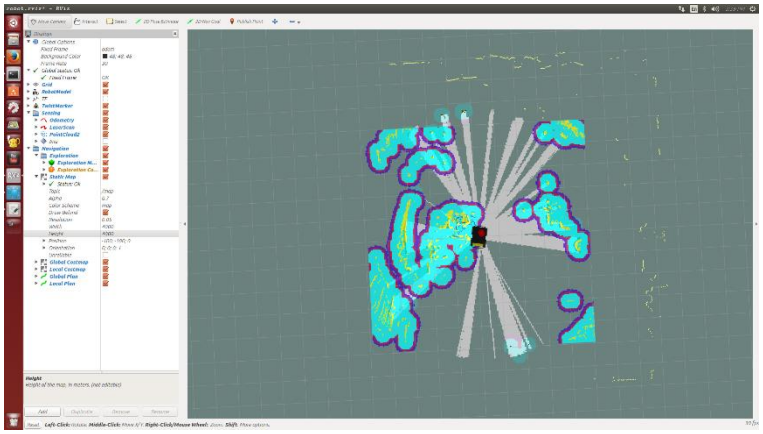

Fig 12. Navigation using gmapping (blue: Potential bubble around obstacles)

Chapter 3

Aerial-based Autonomous Platforms

3.1 Hardware design

Multi-rotor aerial vehicles have recently gained a lot of interest amongst enthusiasts and researchers. The simplicity in the design and the inherent benefits over single rotor and coaxial rotor systems allows for much easier control and stable flight responses. The reduction of mechanical linkages between the motors and the rotors allows them to be more robust, and maintenance free to a large degree. The design allows for fine flight control as well. These vehicles are being used regularly for photography and for research purposes to explore various intelligent algorithms such as localization, navigation and computer vision etc.

Unlike a helicopter with a single giant rotor that is unsafe to use by a civilian user, a multirotor can be made to be much safer, easier to use and can be designed to accommodate strong / uneven wind and loading conditions. It is also a purely electrical system allowing for extremely efficient flight and quick response.

Several multirotor platforms were designed and used for various tasks. Based on the design of the ‘Pelican’ by ascending technologies the Octorotor was designed to be a research platform capable of a variety of tasks and capabilities. The coaxial design of the propellers allows for lesser arms and reduced frame weight but also causes a reduction in motor efficiency. This can be improved by selecting the right propeller motor combination.

In our platform design, a multi system approach and a bottom-up architecture to handle various aspects of control and data acquisition and processing is implemented. The processing into critical flight controls and auxiliary control systems is divided. Different tasks such as flight stabilization and data acquisition and servo control of all the motors requires an embedded autopilot to be used with inertial measurement units and GPS and other sensors. For this purpose, a Pixhawk Autopilot was used, which is explained later. For the higher order navigation and decision control systems, LIDAR, camera and computer vision tasks, a separate Odroid U3 on-board computer was used.

The Pixhawk is an independent open source and open hardware project [20]. Low cost and availability enable its use in small autonomous flight systems and ground vehicles. The project started in 2009, and is being further developed and used at Computer Vision and Geometry Lab of ETH Zurich (Swiss Federal Institute of Technology) and supported by the Autonomous Systems Lab and the Automatic Control Laboratory.

The autopilot design includes an 32bit STM32F427 Cortex M4 core with FPU microprocessor with a 256 KB RAM and 2MB flash. It includes a ST Micro L3GD20H 16-bit gyroscope, a ST Micro LSM303D 14-bit accelerometer / magnetometer, Invensense MPU 6000 3-axis accelerometer/gyroscope and MEAS MS5611 barometer. The Pixhawk has several failsafe's built in.

One benefit of using the Pixhawk is that it makes the system modular and easily upgradable in terms of hardware and software.

The Pixhawk has various capabilities giving it an advantage over other systems available. Several companies, further developed several task specific sensor modules for the Pixhawk. For example, 3DRobotics developed (PX4FLOW), a camera coupled with an ultrasonic distance sensor to achieve accurate position estimation in GPS denied situations using optical flow [21]. PX4FLOW developed by 3D Robotics, offers a low-cost, low-power sensor with a 21-degree Field of View (FOV) and 752x480 resolution. The range sensor is used for distance calculation for scaling optical flow measurements to metric velocity. It also uses the gyroscope to reduce the error and improve estimation. An on-board STM32F4 processes the sensor data and makes it available via I2C. The result yields high dead-reckoning accuracy unavailable to similar autopilots.

Fig 13. Pixhawk Autopilot

The companion computer used with the Pixhawk is the Odroid u3 which is a single board ARM computer that runs a lightweight linux distribution called Linaro 14.04 and ROS Indigo. It connects to the Pixhawk via serial. Several sensors like the Hokuyo lidar, USB cameras, Xtion stereo camera and GPS system can be driven with this computer and the data processed on-board.

This allows the development of navigation algorithms and controllers to avoid obstacles and indoor mapping and navigation.

Fig 14. Odroid U3 with UPS

Several micro aerial platforms mainly the, Crazyflie 2.0 by Bitcraze have been used in several projects. The Crazyflie 2.0 is one of the smallest quadrotors available on the market. Its small size, robust design and safety features make it ideal for applications that require it to fly in close proximity to people and in constrained environments. It measures 92mm from propeller to propeller and weighs about 27 grams. It has a flight time of around 5 minutes. The small size of the Crazyflie presents many challenges in using it as a research platform. The controllers designed for it require extremely low latency to handle its small inertia. Its payload capacity is also limited (approx. 7 grams), making it challenging to add additional sensors. However, all the code for the system is completely open source and that gives us complete control over the firmware on the quadrotor, the firmware running on the radio as well as the client library running on the base-station.

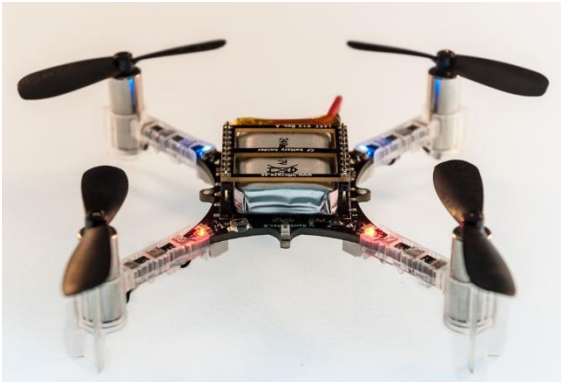

Fig 15. (A)Crazyflie 2.0 (B) Wireless charging coil

The Crazyflie 2.0 is equipped with two microcontrollers - an ARM Cortex-M4 embedded processor (STM32F405), is used to run the main system. The processor is a 32 bits architecture and can run at 168MHz. The ARM Cortex-M4 also has a floating-point unit that supports all ARM single-precision data-processing instructions and data types. The Crazyflie 2.0 also has a second microcontroller to handle power management and the radio. The microprocessor is an ARM CortexM0 (nRF51822) that runs at 32MHz. The Crazyflie Nano Quadcopter is equipped with an IMU, the MPU-9250. The IMU contains a 3-axis gyroscope and a 3-axis accelerometer.

The battery used by the Crazyflie 2.0 is a single cell Lithium-Polymer (LiPo) battery as it provides the best power-to-weight ratio. The supply voltage is 3.7V and has a capacity of 310 mAh. The battery also comes with a Protection Circuit Module (PCM) attached to it that prevents the user from under or overcharging the battery or from shorting it. The battery is easily removable from the quadrotor so connectors were built for it so that they could be easily replaced.

3.2 Software Architecture

The PX4 flight stack is an open source software package which includes several modules for various tasks such as guidance, navigation and control algorithms for autonomous systems. It includes controllers for fixed wing, multirotor and VTOL airframes as well as estimators for attitude and position. It also includes various estimators for local positioning systems (VICON) as well as global positioning systems (GPS). This allows indoor and outdoor use and appropriate sensor information can be selected. The diagram below, figure 16, shows some of the basic implementation of the various modules. These modules can be combined and others added depending on the application [22].

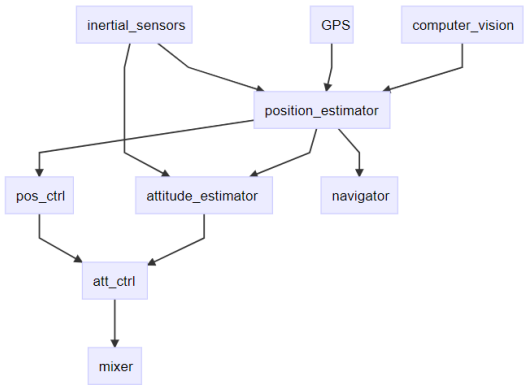

Fig. 16. Sensor and control modules

The system interacts with the ground control station (GCS) is through MAVLink packets and converts them into the on-board uORB data structures [23]. This allows some amount of abstraction within the system and reduces the dependencies. The MAVLink application also

consumes a lot of sensor data and state estimates and sends them to the ground control station.

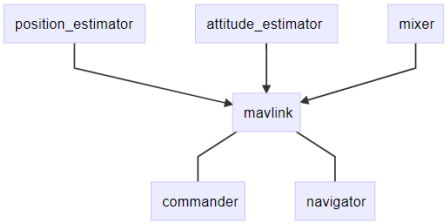

Fig 17. Mavlink protocol module

For communication with the base station the Odroid U3 computer on-board the UAV running ROS Indigo. ROS is a robotics framework providing many APIs and services such as hardware abstraction, low level device control and a graph architecture for message passing between multiple systems or nodes [24]. The ROS ecosystem can be defined into 3 groups [25],

1. Language and platform independence which allows for a distributed architecture.
2. Client and Server Library implementations
3. Open Source packages for various applications for specific client libraries.

ROS contains many open source packages developed for robotics purposes and are used to build and integrate various systems within our applications.

Some of the packages utilized for the robotic platforms and development are MAVROS developed by Vladmir Ermakov and distributed under the GPLv3, LGPLv3, BSD licenses. MAVROS is a MAVLink based communication node to allow for easier integration between ROS and MAVLink based autopilots such as the Pixhawk.

3.3 Sensor Integration

1. Px4flow - The PX4FLOW (Optical Flow) is an optical sensor consisting of a downward facing camera module and a 3-axis gyro . This high resolution camera module detects the ground texture and visible features and a separate built in sonar. The Maxbotix LZ-EZ4 sonar is used to determine the UAVs ground velocity and altitude. Unlike many other sensors, it also works indoors and in low outdoor light conditions without the need for an illumination LED. It can be freely reprogrammed to do any other basic, efficient low-level computer vision task.

Processor	168 MHz Cortex M4F CPU (128 + 64 KB RAM)
Sensor	752×480 MT9V034 image sensor, L3GD20 3D Gyro
Optics	16 mm M12 lens (IR block filter)
Dimensions	45.5 mm x 35mm
Power consumption	115mA / 5V

Table 1: Pxflow specifications

2. TP-Link TL-MR3020 – is one of the smallest wireless routers available and easy to use and supports OpenWRT technology. OpenWRT is an open source Linux distribution for embedded devices. Apart from creating a single, static firmware, OpenWrt provides a fully writable filesystem with package management. This allows us to select the necessary configuration and customize the device using packages to suit any application. For developers, OpenWrt is the framework to build an application without having to build a

complete firmware around it.

The router, being USB powered, allows us to create a portable Wi-Fi access point that can be utilized in creating a Wi-Fi mesh network. This allows all systems to communicate via a mesh network and provides several features such as a self-healing network where nodes damaged or not responding can be cut off from the system and data rerouted via other nodes, extending the range of a network and easy integration between multiple nodes.

Fig 18. TP link travel Router

Each of the ground and aerial vehicles have been fitted with the router allowing them to communicate with each other wirelessly over the same network. This allows global and local system control. It also allows unique address for each node to control or acquire data from a sensor.

The use of a wireless mesh also has the benefit of being able to choose our own network architecture such as a centralized architecture, a decentralized architecture where several nodes communicate with their nearest nodes and do not form a fully connected network, or a Distributed

graph, where all the systems are connected to the next node via different paths.

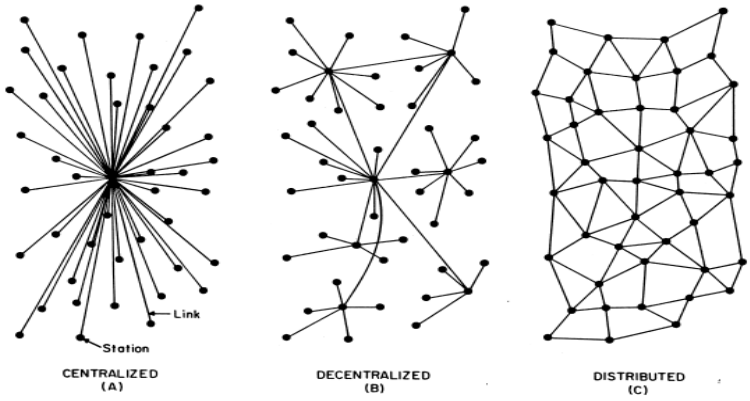

Fig 19. Mesh configurations

Chapter 4

Controller Designs, Simulations and Implementation

4.1. Skeletal Tracking

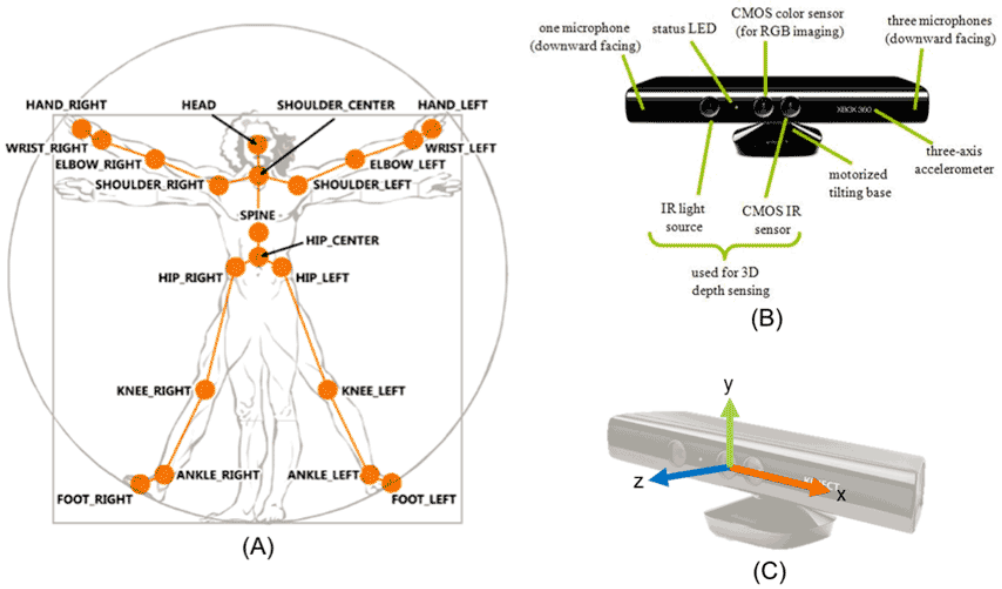

Fig. 20. (A) Skeletal joints, (B) Xbox Kinect, (C) Kinect coordinate frame

This project works based on the open source OpenNI ROS drivers for skeleton tracking. The package works with a Kinect camera connected to a PC and can track up to fifteen joints and body parts in human skeleton. These joints include hands, elbows, head, neck, shoulders, torso, left and right hips, knees, and feet. The 3D positions and rotation quaternions of all these joints in space are calculated with reference to the center of camera frame, */openni_depth_frame*. It also provides a set of transformations between the various joints (*/tf*). These joints are shown in figure 21.

This ROS node publishes the position and the orientation of joints as transforms, as shown in figure 21. All the frames of the human model are computed from the reference coordinate frame called *camera_rgb_optical_frame*, located on the RGB camera of the Kinect sensor. The sensor can identify multiple users; however, it becomes necessary to identify various users and map their structure data and calibrate the system to perform better.

A calibration sequence was designed to run initially, where the user moves the joints to the extremes and the system records the extreme transform and uses that to determine the thresholds for each user, this allows for clearer identification of gestures and control.

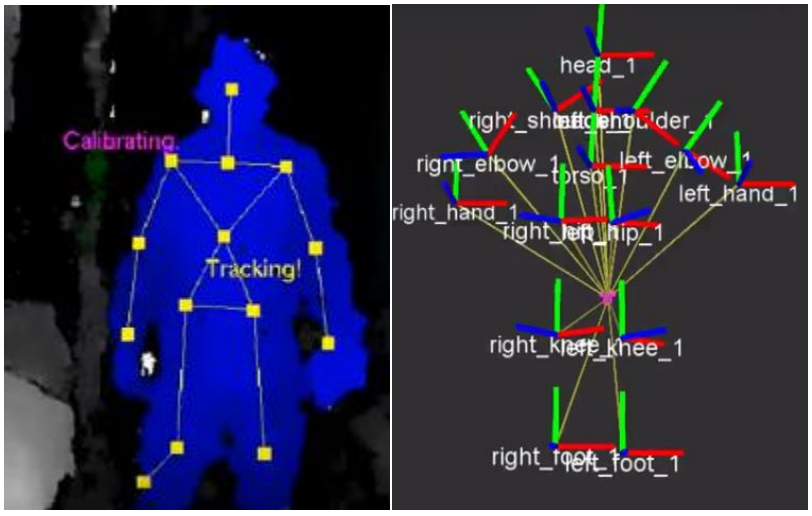

Fig 21. Joints and Tf being tracked by the Kinect

To calculate the transforms and rotations between various joints on the skeleton, the *tf* library was used. The Tf library is an open source library that provides a standard method to track multiple coordinate frames within a global frame. It allows the user to reliably obtain data of a frame without having to worry about transforming between multiple frames. The Tf library

provides functions to calculate the translation and rotation between multiple frames and calculate the pose and distances between the ends of different links.

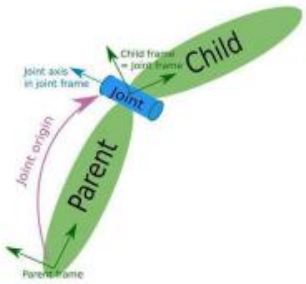

Fig 22. Example of Frames between parent link and child

Based on the transforms, the values of the coordinates and relation between multiple joins, the program can identify various actions and gestures. Some of the gestures are raising the right and left hand and lowering them or holding them at a fixed pose.

In the implementation, the Parrot AR Drone quadrotor was chosen to demonstrate the controller. The AR Drone connects to the system via WiFi and uses the *ardrone_autonomy* ros package. Different services and commands were mapped to different gestures using a state machine and tested [26].

The user must initially run through a calibration sequence where the user initializes the node by holding the psi calibration pose and once detected, rotates all tracked joints to the extremes and back, this allows the program to measure the complete range of values and identify the minimum and maximum values of all the frames. The program then decides the thresholding values based on these min and max values.

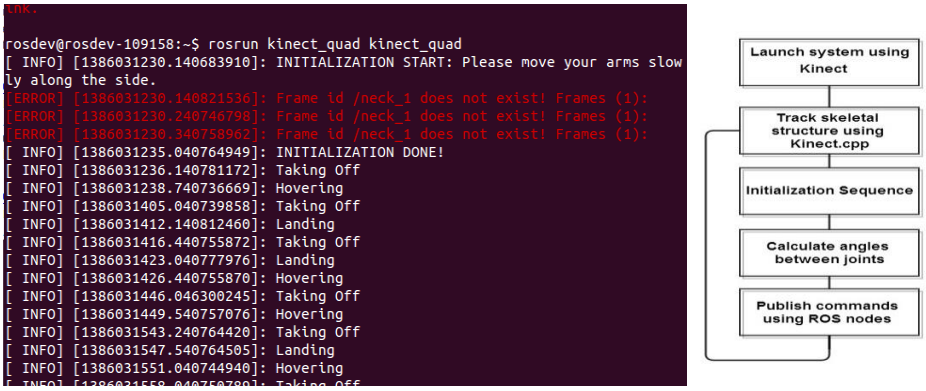

Fig 23. Changing states according to user gestures and controller Flowchart

Several gestures were used to control the UAV shown in figure 23.

Fig 23. Gestures recognized by the Kinect

Data from several instances with several users and calculated the errors in detection detailed below. The percentage of errors after the calibration sequence is lesser and is mainly present only when the gestures are similar.

Gesture	Command	Without calibration (approx.) Error %	After calibration (approx.) Error %
Pose 1	Take off	20	5
Pose 2	Hover	15	10
Pose 3	Land	10	2
Pose 4	Forward	15	5
Pose 5	Back	10	2

Table 2: Detected User Pose errors

Thus, with a simple calibration sequence, the pose and gesture detection can be improved quite well. The system could further be improved using a neural net classifier to correctly classify gestures.

4.2. Potential Field Navigation

In this section, the implementation of an Image-based local path planning algorithm based on potential field is discussed. The potential field algorithm uses distance based virtual forces to traverse from one location to another in the path of least resistance or repulsion.

The concept of potential fields is a natural phenomenon and is observed in many cases. Some examples are the effect of a magnetic field on a charged particle, flow of water down a hill. The difference between the potential of forces causes a force to be exerted on the object. It is possible to simulate the same effect, by introducing artificial potential forces based on distances between the object, the goal and the obstacles [27]. By defining a charge to the object, we can decide if it is attractive or repulsive in nature and thus influence the behavior of mobile robots.

At any point (x, y) in the plane, the net forces acting on a point based on the combination of individual forces are calculated. The net force as a function of position, is a vector force field that represents the resultant force obtained by the sum the attractive forces and repulsive forces which gives us a gradient in the direction of the minima.

$$\vec{F}(\vec{r}) = \vec{F}_{target}(\vec{r}) + \sum_{i=1}^{n} \vec{F}_{obs}(\vec{r}) = -\vec{\nabla}V(\overrightarrow{r})$$

where arrows denote vectors. In potential field motion control, it is necessary to calculate and keep track of the potentials at all points. With mobile ground robot applications, the potential fields are usually used in a 2D space.

Thus, the force gradient for $\vec{r} = [x \quad y]^T \in R^2$ is,

$$-\vec{\nabla}V(\overrightarrow{r}) = \left[\frac{\partial V}{\partial x} \quad \frac{\partial V}{\partial y}\right]^T = \left[F_x \quad F_y\right]^T \in R^2$$

Where the x and y Force components are

$$F_x(x,y) = -\frac{\partial V(x,y)}{\partial x} \qquad , \qquad F_y(x,y) = -\frac{\partial V(x,y)}{\partial y}$$

4.2.1. Image-based navigation

In this method, a combination of aerial systems and ground systems is used to help the ground vehicle navigate in real world terrain. This collaborative working technique can help a vehicle gain better situational awareness otherwise limited by its on-board capabilities.

In this algorithm, the images are taken by the camera and the following steps are executed to get the result:

Algorithm:
Step 1: Obtain image from camera
Step 2: Threshold and create an inverse binary map of the environments
Step 3: Calculate minimum distances from robot to goal point
Step 4: Create Potential field at all points detected in the binarized, threshold matrix
Step 6: Calculate Attractive and Repulsive forces caused by goal and obstacles respectively.
Step 7: Find the sum of all forces and calculate the next position of the robot based on previous position.
Step 8: Calculate the optimal path and calculate heading angle of the robot using the Pose estimate obtained using the earlier method.

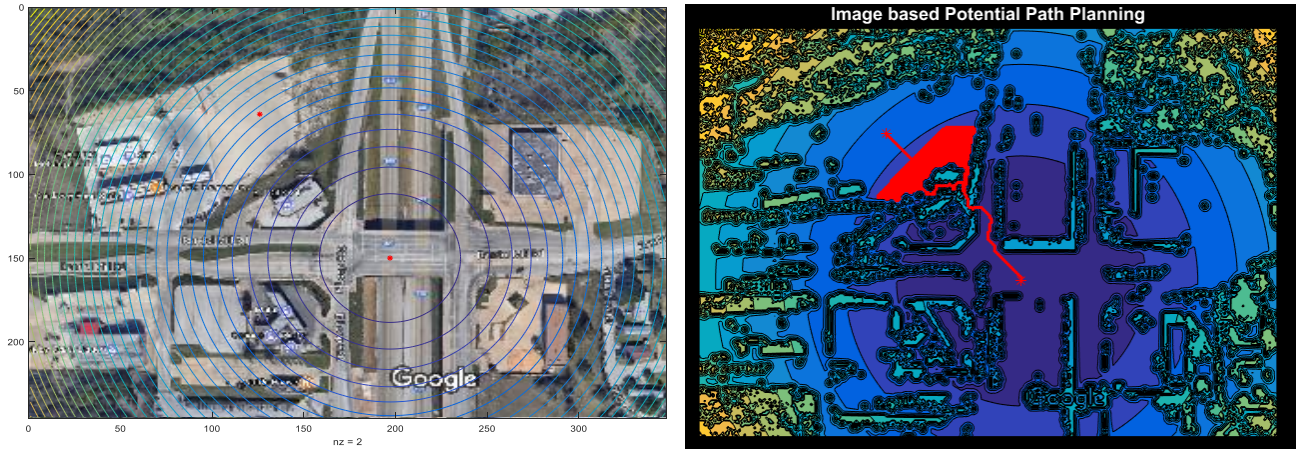

Fig 24. Input image taken from google maps to simulate outdoor images at high elevation. The red spots denote the initial point where the robot is located and the final goal point. The image on the right is the final path identified

However, it is evident that certain drawbacks exist in this method. The robot is easily and more likely to be stuck in local minima as the number of individual pixels having their own potential fields is high.This can be tackled by introducing various dither signals that cause small movements and can nudge the system out of the trap.

4.2.2 UAV based image classification using Convolutional Neural Networks

Machine Learning is an area which has exploded with interest and development in recent years and its integration with robotic systems has given rise to a whole new generation of intelligent systems capable of learning and making smart decisions based on data available. Traditional learning methods require the use of large datasets manually labelled and defined. This is extremely time-consuming and not viable in the long run for many applications. Deep learning is a subset of machine learning algorithms developed to reduce the development and learning time and to utilize unlabelled datasets which automatically allows a model to learn and obtain a good representation of the input from this data. It can be used in robotics to solve several tasks such as perception, navigation and control. Several studies have been performed on the application of deep learning algorithms and tasks with respect to UAVs and robotic systems and tasks such as security and surveillance.

One application is the implementation of Convoluted Neural Networks (CNN) in the use of classification and tracking applications on-board robotic platforms and the integration of this capability in a mesh network consisting of multiple sources and the ROS framework.

4.2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), are a specific type of neural networks designed to accept images as inputs. They perform several 2D convolution operations on the input data at

multiple levels within the network convoluting th 2D images with a 2D Kernel, whose depth more than the input layer. Multiple convolutions can be done to each layer with varying kernel dimensions and the final outputs of the layers are pooled together using a pooling function, which pools several local outputs obtained from a nonlinear activation function. This helps in representation of learned invariants from local outputs and results in a subset of the input data. Several convolutional layers and pooling layers are stacked together in a hierarchical structure to form a pyramid-like structure where each layer allows for a unique feature learning, thus providing more abstraction and generalization of learning and outputs. For example, the first layer can be used to identify certain edges and other low-level representations, the next layer can be used to identify higher level features like shapes and structures. Thus, as the layers progress, the amount of feature learning, progressively grows higher. Once the features required for the task (eg: faces, plants, fruits, etc.) has been learned, the output of the last layer is passed through an activation function and classified according to the input [28].

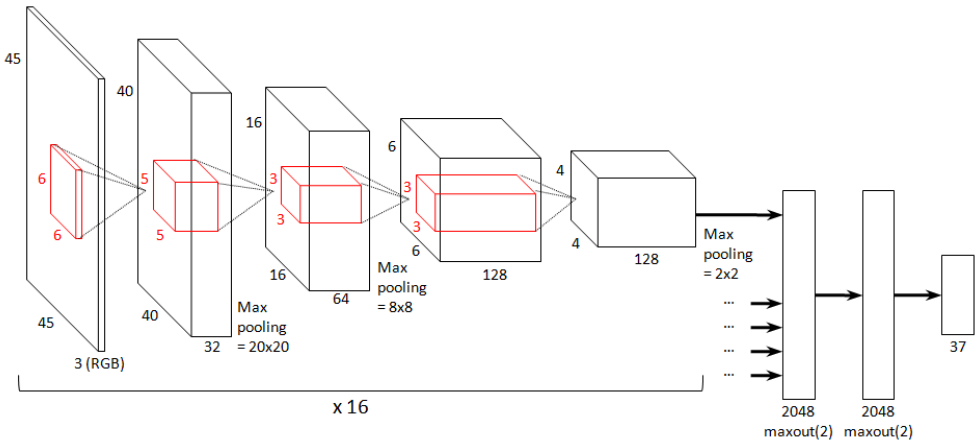

Fig 25. 16 layers channel CNN architecture

Image classification training data samples are simply images (usually a small image or patch containing a single object) labeled by class (typically integer class ID or a string class name). Object detection, on the other hand, requires more information for training. ImageNet training data samples are images of individual objects. The network is trained to identify objects within a single image. For this dataset, a naïve image label format works fine as each image has one object present within it. Other datasets are available such as DetectNet, where the network must detect not only multiple objects within an image, but also the location of objects within the images. This can be used to perform object identification and tracking in applications such as pedestrian tracking [29].

ImageNet model

Layer name	Layer description
input	Input 224×224 RGB image
conv1	11×11 conv. 96 ReLU units, stride 4
conv2	1×1 conv. 96 ReLU, stride 1
conv3	3×3 conv. 96 ReLU, stride 2
conv4	5×5 conv. 256 ReLU, stride 1
conv5	1×1 conv. 256 ReLU, stride 1
conv6	3×3 conv. 256 ReLU, stride 2
conv7	3×3 conv. 384 ReLU, stride 1
conv8	1×1 conv. 384 ReLU, stride 1
conv9	3×3 conv. 384 ReLU, stride 2, dropout 50 %
conv10	3×3 conv. 1024 ReLU, stride 1
conv11	1×1 conv. 1024 ReLU, stride 1
conv12	1×1 conv. 1000 ReLU, stride 1
global_pool	global average pooling (6×6)
softmax	1000-way softmax

Fig 26. Image net Architecture

Integrating CNN's trained with ImageNet and DetectNet with the existing UAV system framework available at the lab gives us the opportunity to create a mobile floating UAV cluster with one central system with high processing capable UAV and several UAVs with constrained

processing capabilities.

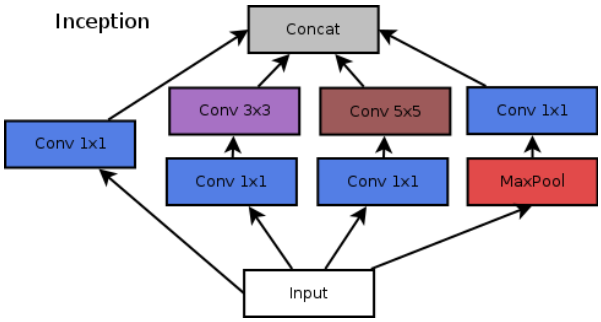

Fig 27. Convolution Filters for each layer

For the implementation of the model, Tensorflow, an open-source machine learning library released by Google, to run the inception based cnn model trained using the Imagenet dataset is used. The training data consists of nearly 14 million images of 1000 classes of objects. The Inception module [31] is based on a pattern recognition, in a manner similar to how the visual cortex works. After getting several input data images, the network learns minute details, middle sized features or almost whole images. Individual layers of the network reinforce some features and passes on to the next. If trained to recognize faces, the first layer detects edges, the second overall shape, the third eyes, mouth, nose, the fourth the face, the fifth the mood, etc.

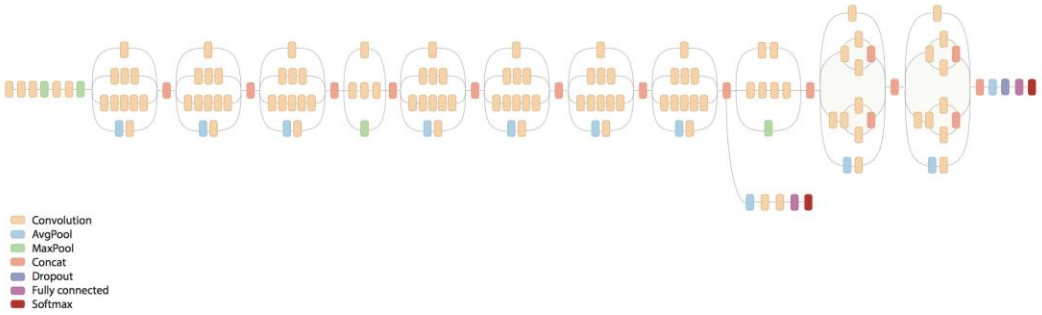

Fig 28. googLeNet architecture

For the application, an Nvidia Jetson TX2 module along running Tensorflow 0.9 and ROS Indigo on Ubuntu 14.04 was used. The ROS package consists of a python script which calls the tensorflow module and uses the inception network model. Since ROS is designed with a distributed architecture, multiple camera sources (UAV camera feeds) can be obtained on one system. Video streams from multiple AR Drone Quadrotor and USB cameras were used for testing. Several nodes are simultaneously run, which can process each video stream individually and identify various objects from each system. The image classifier requires an image size of 299x299 RGB which can be resized either using the *cv_camera*, package or by setting *the input_width* and *input_height* flag values. The program looks for training images to identify objects and classify. It can identify 1000 classes of objects. The data is downloaded from the ImageNet database *(inception-2015-12-05.tgz dataset)*. Using a softmax function at the outermost layer of the network, it identifies the object along with a calculated confidence score.

For this implementation, a custom-built Octocopter, a DJI s1000 was used. The system is

built with primarily for photography and cinematography. However, it has been modified and setup for use in aerial mapping and collecting field data for farmers. This enables precision farming and supplies a farmer with automated solutions for a variety of tasks. The system is designed to be scalable autonomous and allow a large amount of data processing on-board especially of images of farms and regions.

Fig 29. NVIDIA Jetson TX2

4.2.2.2 NVIDIA Jetson TX2

While dealing with image processing, on high resolution images, it becomes necessary to use specific graphically optimized computers. The s1000 has an on-board NVIDIA Jetson TX2 GPU computing module for this purpose. The Jetson TX2, released in March 2017, belongs to a family of GPU devices specifically designed for machine learning and vision processing. The specifications of the device are below:

CPU	ARM Cortex-A57 (quad-core) @ 2GHz + NVIDIA Denver2 (dual-core) @ 2GHz
GPU	256-core Pascal @ 1300MHz
Storage	32GB eMMC 5.1
Camera	12 lanes MIPI CSI-2 | 2.5 Gb/sec per lane | 1400 megapixels/sec ISP
Power	7.5W

Table 3: NVIDIA Jetson TX2 specifications

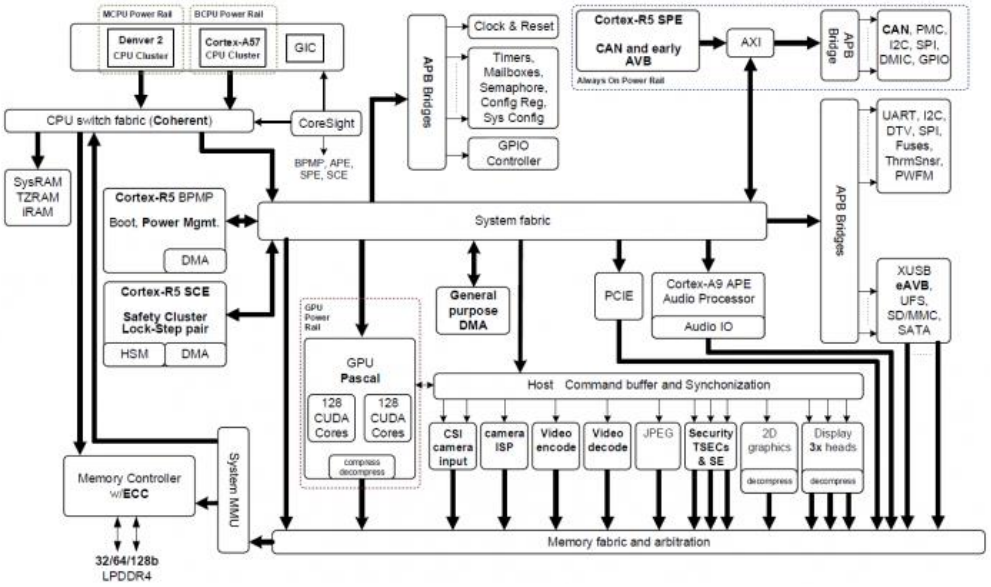

Fig 30. Internal Architecture of Jetson TX2

From figure 30, the Jetson is designed to handle several cameras together and is optimized to process multiple image streams together.

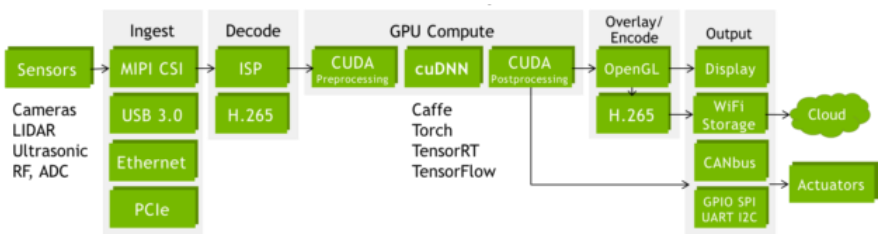

Fig 31.A.I. System Pipeline

The above figure(31) shows the Jetson TX2 platform machine learning pipeline. It is ideal for the end-to-end design of autonomous machines. It is designed specifically to handle high resolution images and can simultaneously process data from multiple sensors and media compression and extraction according to several encoding protocols, networking, and low-level command & control protocols after processing the data on the GPU. The image also shows the various sensors and the ways it can be interfaced with the system via USB, MIPI, PCIe and ethernet. It also supports serial and I2C communication as well.

4.2.2.3 Classification results

Using the Jetson as the main processing board running Tensorflow (ver 0.9), OpenCV a ROS package called *multi_uav_classifier, AR Drone autonomy* and *usb_cam* package to obtain the camera images. The classifier node, which uses the inception CNN model, then classifies the objects detected in the images. Once classified, it calculates a confidence value which is the probability score of the object being detected and outputs the score. The detection can be further filtered by adjusting certain thresholding parameters. The *usb_cam* package was used for testing however any size or source image can be used for classification. Multiple video sources from AR Drone parrots were also used and tested. The results are shown below in figure 32. The main limit on the number of sources is the amount of processing power available for the system.

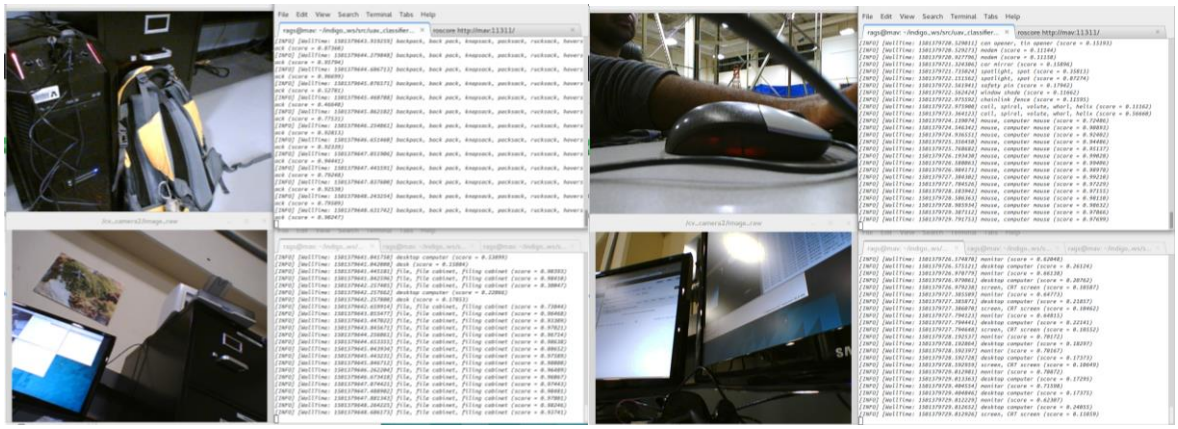

Fig 32. object detection using classifier on Multiple image sources

An implementation of a network to detect multiple objects and track objects in the image was also shown. Using the Detectnet model trained to identify pedestrians and a few other classes such as bags and chairs, the model runs on the jetson and makes use of the CUDA cores present on it. Pedestrians are identified and tracked along the image pixels as they move and that information can be sent back to the robot for navigation.

Fig 33. Pedestrian Tracking on the UAV

Chapter 5

Conclusion and Future work

From the results in the previous chapters, the development and design of such a mesh network of UAVs and platforms, along with systems capable of processing considerable amounts of data is demonstrated. It allows us to develop a simple and cheaper and effective solution of a floating UAV framework where small UAVs with simple cameras can stream the data to a central node, where it can then be processed and the real-time identification of objects can be done. This has the benefit of having few expensive processing systems, while working with several low-cost systems. The processing can also be done on multiple ground-based vehicles as well.

Another project which is currently being worked on is the implementation of the object tracking model for multi-source tracking and integration with image sources from multiple systems. The program returns the position of the object detected. This information can be used to control multiple systems and follow their targets independently. The object tracking algorithm has been successfully implemented and tested using an on-board camera and a high-resolution ZED stereo camera.

Chapter 6

Integration of multi-modal, multi-resolution, MEMS skin sensors to include tactile, thermal, pressure, acceleration, and distance IR sensing

6.1 Introduction

The development of artificial robotic skin is motivated by the necessity to provide robots with a more effective feedback of their interaction with themselves and the world[32]. Interactions of robotic systems is mainly grounded with the surface of the agent and the environment. As social robotics become more ubiquitous, it becomes necessary to address certain design challenges between these systems to create a more intuitive and natural sensing and feedback system which takes advantage of this surface interaction. There are several available methods such as resistive and capacitive touch methods, but these are not intuitive and natural. The objective of this research is to answer design questions for multi-functional robotic skin sensors, and develop various solutions for various design constraints such as sensor placement optimization, scalable data acquisition and processing, have the robot and human "learn" how to use the skin sensors efficiently, and quantitatively assess the impact of this assistive technology to humans. The approach is to design and fabricate integrated micro-scale sensors in conjunction with iterative simulation and experimental studies of the performance of physical human-robot interaction enabled by this technology.

This section presents the development of a Real-time data acquisition system to collect data from multiple sensors such as Infrared thermopile sensors namely the Omron D6T-44L,

Piezoelectric Flexiforce pressure sensors, accelerometers and to process the collected data in a deterministic manner and to generate a thermal map to allow the robot to obtain thermal data to identify obstacles namely humans and interact with the environment using the data.

Several attempts have been made to design robotic skin capable of replicating the various natural senses of human skin. Researchers from the University of Illinois, Urbana-Champaign designed a skin molded flexible fingertip sensor. The Hex-o-skin, one of the most sophisticated skin modules consists of several sensors such as accelerometers, proximity sensors and temperature sensors[32][33][34][35]. However, most of these sensors consist of only one sensor output or limited in some ways.

Using these systems as inspiration, Roboskin is developed using multiple types of sensors integrated within a soft silicone-based substrate that provides a more natural feeling and sensory response.

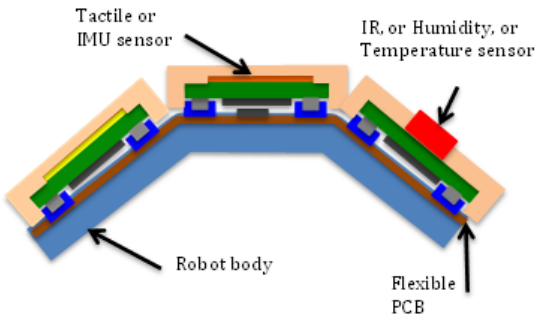

Fig 34. Proposed Design

6.2 Skin Prototypes

Multiple silicone skin patches were created at UTARI from a silicone based rubber from Silicones, Inc. The polymer is a 10:1 mixture of silicone to the curing agent. Several moulds coated with a releasing agent, were used to create skin patches. Patches of several thicknesses were made with various sensors embedded within. Once the silicone material is poured into the mould, the mixture is set in a vacuum chamber to remove any bubbles within, and then cured in an oven.

Several piezoelectric skin sensors were embedded within the skin patches with 3D printed buttons placed on top of the sensors.

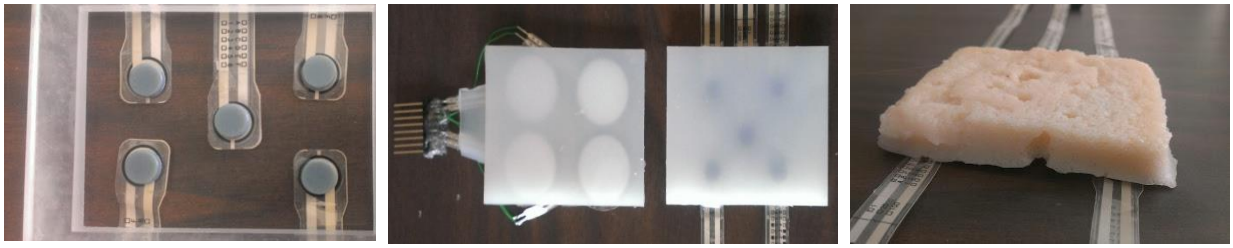

Fig 35. Skin Patch prototypes with embedded flexy force sensors

6.3. Temperature sensor

To incorporate a sense of temperature within the skin modules, a non-contact thermopile sensor was chosen. The Omron D6T – 44L [36] is a contactless based temperature sensor with a broad range and field of view. It consists of an array of 16 temperature sensing pixels, which output the raw data to an amplifier which is then processed and converted into digital data and transmitted

over I2C.

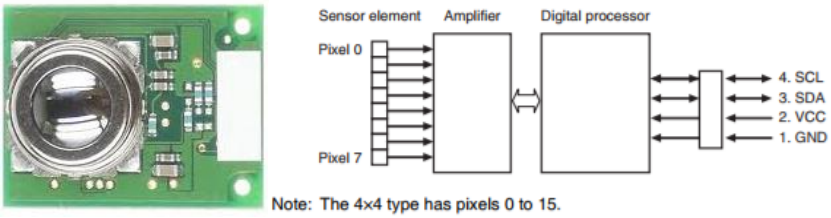

Fig 36. Omron D6T-44L Thermal Sensor

6.4. Sensor Integration

The first phase of testing and development was done using an open source microcontroller board running an ATmega2560 chip. The purpose of this was to understand the method of handling the data and the limitations and the speed of data acquisition that could be achieved. A speed of about 10 Hz was achieved due to limitations of the chip and the Omron sensors.

The second phase involved developing and designing a method to reduce the number of wires and power lines running through the system. This involved the development of the MicroBoard (µC Board). The board made use of a 16 channel 12-bit ADC (AMC7812) [38] by Texas Instruments and allowed us to gather data from up to 16 different Flexiforce sensors and 3 Omron sensors simultaneously. The Flexiforce also required several mini boards to be developed which were essentially voltage divider circuits which allowed us to condition the signal and reduce the errors. The µC board and the mini boards are covered in detail further below. Various versions of the board were made and revised until issues such as noise and power issues were resolved.

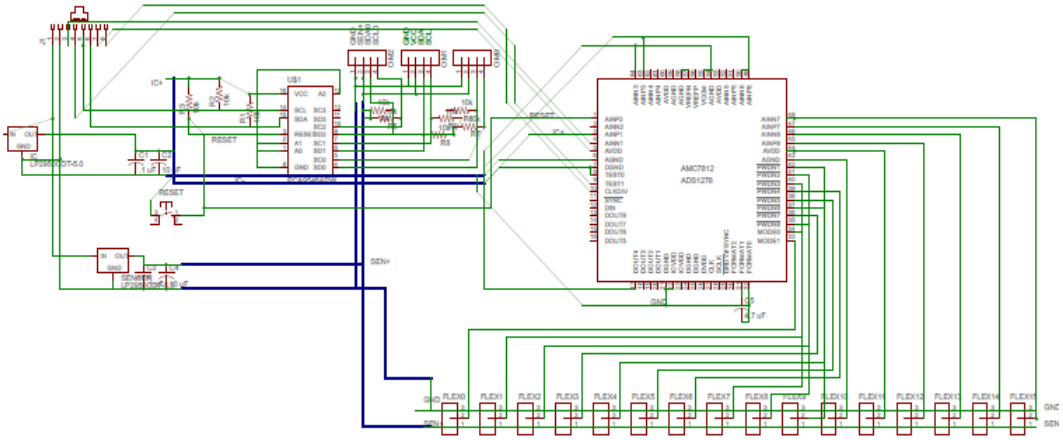

Fig 37. Electrical schematic of Data Acquisition board

The major work done in this phase was to set up the data acquisition system for acquiring the data and processing it. For this purpose, the National Instruments cRIO FPGA Real-time system was used. The cRIO [38] provides a flexibility and scalability, which allowed integration of several data protocols from various sensors over multiple communication methods and to sample the data simultaneously. It consists of a real-time FPGA module which was programmed using LabVIEW. The architecture allows the use of various modules which can be used for specific applications and sensor integration. Several of the modules are shown below. The cRIO initially was used with the Tangent blue to acquire data however due to constraints in the driver and the architecture of the module and incompatibility with the Omron sensors, this module was abandoned.

The following enlists most of the activities and major findings achieved during the various phases of the project:

1. Elastic effect of Skin:

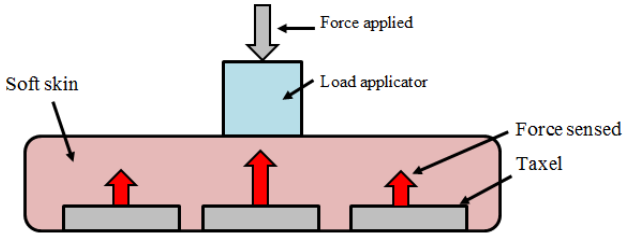

Fig 38. Flexi force sensor response without firm substrate below it

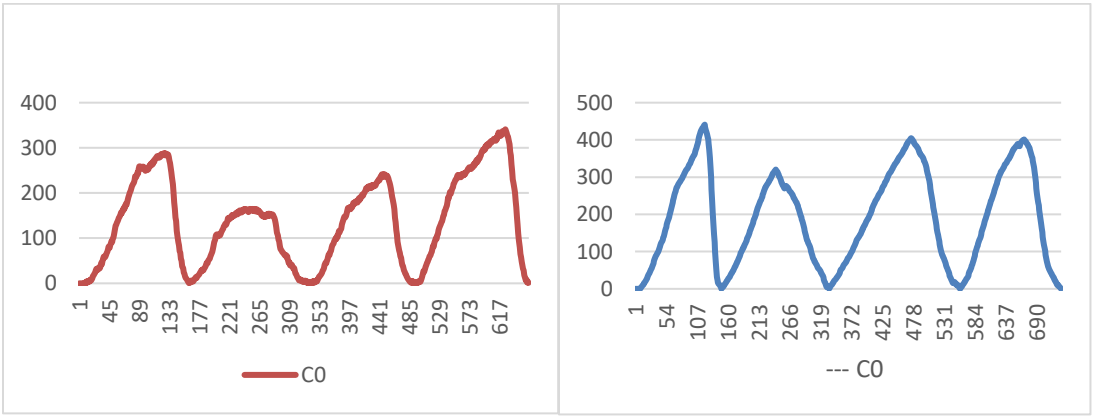

Fig 39. Single channel (sensor response) Left: Force response without a firm substrate Right: With hard base within the soft skin substrate

The uneven response observed is due to the viscoelastic property displayed by the material. This effect can be due to multiple reasons: friction between neighboring molecules, displacement of air from cells within the material causing deformation. The flexible and soft nature of the material provides a slow compressive and elastic response which distributes the force and leads to a reduced response. This was reduced by using a firm backing behind each sensor, within the silicone substrate. The compressive response

is still visible due to the soft surface action; however, a much better force detection and cleaner response is observed.

One of the outcomes of the project was SkinSim [37], a skin simulation environment based on the Gazebo simulator. It provides functionality for building robot models with robotic skin attached and near real-time realistic skin simulation. It allows testing of future skin models, simulating the pressure response similar to the test results.

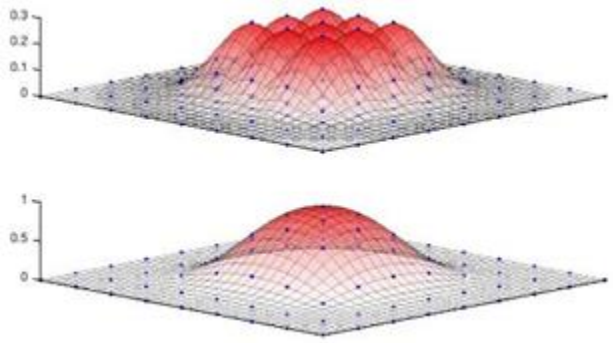

Fig 40. SkinSim simulation of pressure sensing

2. Omron Addressing: One major characteristic of the Omron is the fact that every single sensor utilizes a single common address which cannot be changed or reassigned. Thus, at any given time only one single Omron sensor can be addressed and if multiple sensors are used, they must be separately selected and read.

   This issue was tackled by adding 2 extra bytes of overhead to the address and adding a separate I2C bus switching IC which is used to separately select between different Omron sensors which have been connected to the IC.

1st Byte	2nd Byte	Omron Selected
0	0	Omron 0
0	1	Omron 1
1	0	Omron 2
1	1	Unassigned

Table 4: Omron Addressing

The following image (41), shows how the software uses the same bus and by changing the 1st 2 Bytes, it selects the Omrons.

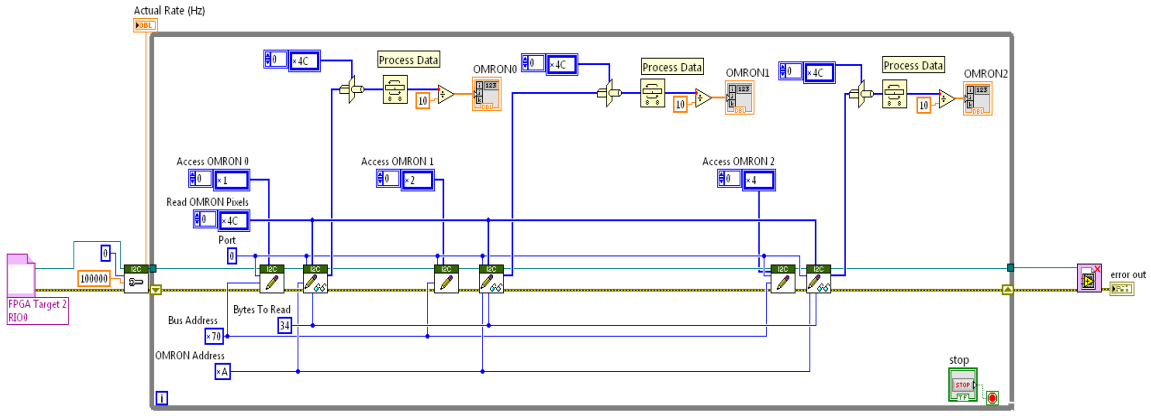

Fig 41. Selection process of the three Omron sensors

3. Clock Stretching: Another feature of the Omrons is that it utilizes a concept called clock stretching [39] and it appears that the I2C interface is a microcontroller acting as a slave device, it operates slower than a dedicated I2C peripheral. It has a maximum of 100KHz clock frequency and clock stretching is implemented because of the slowness of the interface. However, the total bandwidth of the shared bus might be significantly decreased. Hence while working with I2C buses shared by multiple devices, it is important to estimate the impacts of

clock-stretching. One method of reducing this task is to have separate I2C channels for each Omron sensor and to use modules that can handle the Clock stretching.

The I2C slave pulls the SCL line low on the 9th clock of every I2C data transfer (before the ACK stage). The clock is pulled low when the CPU is processing the I2C interrupt to evaluate either the address or process a data received from Master or to prepare the next data when Master is reading from the slave.

The time the clock is pulled low depends on the time the CPU takes to process the interrupt and hence is dependent on the CPU speed and not the I2C clock speed. This process is shown in figure 42.

Master	Slave (Sensor)
a) SCL drive to Lo for Ack.	
	Checking SCL status.(Lo)
	b) SCL drive to Lo for Wait.
c) SCL output change to Hi-Z.	Wait...
d) Checking SCL status.(Hi)	e) SCL output change to Hi-Z.(Wait finish)
f) Finish Detected.	
g) Next operation.	

Fig 42. Clock Stretching in the Omron sensor

Frequency	Frequency - Capable of 100 – 200 Hz individually,
Data	Data rate - Limited to 100 kbps on I2C Bus by device
Field of view	Field of view - 45 degree viewing angle
ADC	ADC - Using a TI AMC7812 16 Channel ADC – 12 bits (1.22 mV step on 0-5 VDC)
Interface	I2C or SPI
LDO	2 – 1 A, 5V 1 for IC, 1 for Sensors
8 Channel GP-I/O	8 Channel GP-I/O

I2C Switching Bus	I2C Switching Bus - Only adds 2 bytes per update per channel but allows up to 4 devices with same address
Single Ethernet connection for I/O	Single Ethernet connection for I/O
Temperature Sensor	Internal Temperature Sensor

Table 5: Micro Board Specifications

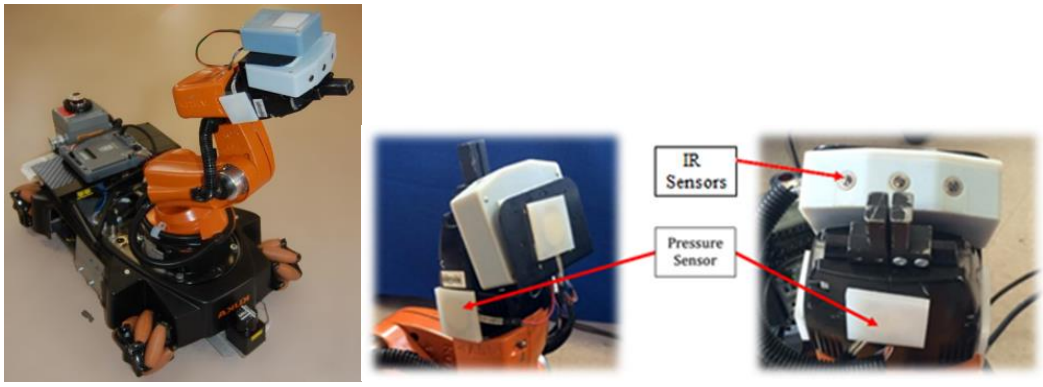

Fig 43. Kuka Youbot manipulator arm with sensor patches

4. Kuka Youbot: The Kuka Youbot [40] was chosen to be used as a responsive platform for the data acquisition system and controller. The pressure sensors are mounted at various locations along the arm and base as shown in image 43. The Kuka arm is outfitted with 16 different sensors while the base is fitted with 10 pressure sensors. The sensors on the arm are mounted in the form of sensor pads where several sensors are embedded in a patch of silicon with plastic buttons to concentrate the force incident on the buttons. The cables are snaked through the Kuka Youbot from within the arm to avoid exposing the cables. 30-gauge wires are used to connect all the sensors due to its thin size.

Fig 44. Top View of Kuka Youbot with sensors and cRIO installed

6.4 Outcomes

Some of the important milestones and achievements are :

- Temperature mapping using Omron
    - Three Omrons to achieve a 120 deg map.
    - Detect presence of obstacles in front of device and to be able to track movement over the entire map.
- Recognize heat signature: This task was quite difficult because of the constraints in resolution and clarity.
    - The lower pixel count does not allow the proper identification of humans and objects, but it is possible to track movements and follow objects.

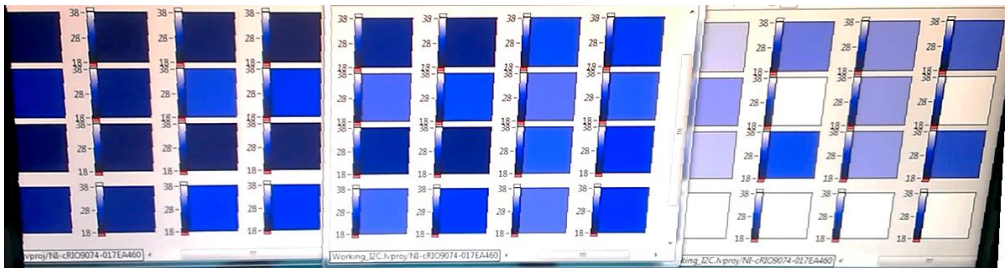

Fig 45. 4x16 pixel Thermal map to track object detected by all three Omrons

- Created Silicone-based pads with embedded pressure sensors to mimic the sense of touch with real skin.

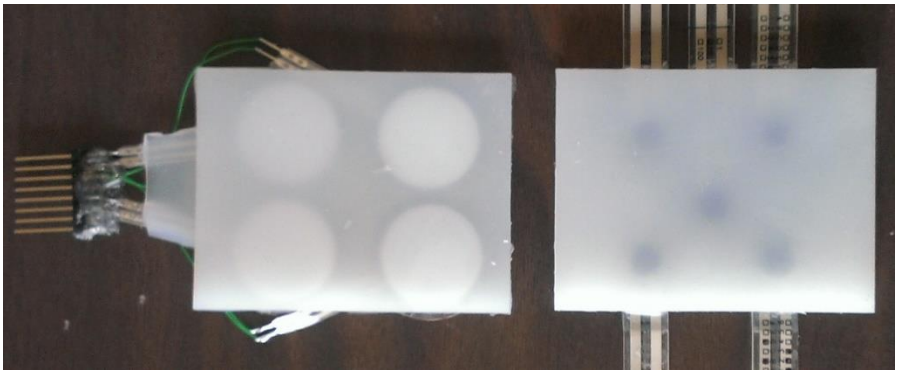

Fig 46. Skin cell pads with embedded pressure sensors

- Designed and developed single slave board to connect all sensor and then relay data to the data acquisition and processing system µC Board with LDO and ADC to gather and send data over a single cable.
- Small size of 3.3” x 2” allows for easy installation and small footprint. Further iterations increased capacity and improved performance and increased sensor numbers.
- Breakout board to connect to power and cRIO (power stepped down from 24V to 12V and from 12V to 5V for sensors)

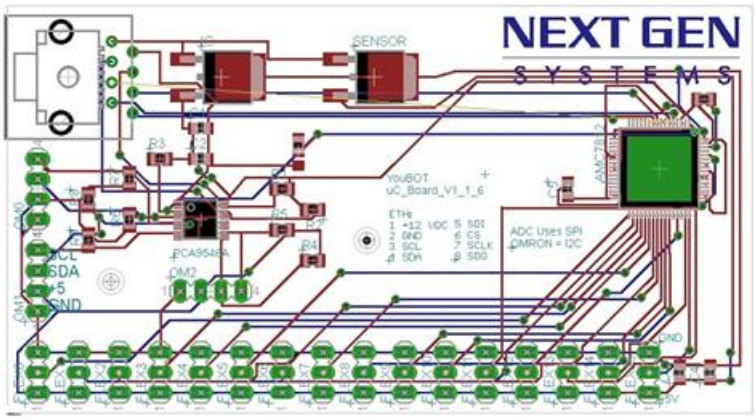

Fig 47. MicroBoard for data collection

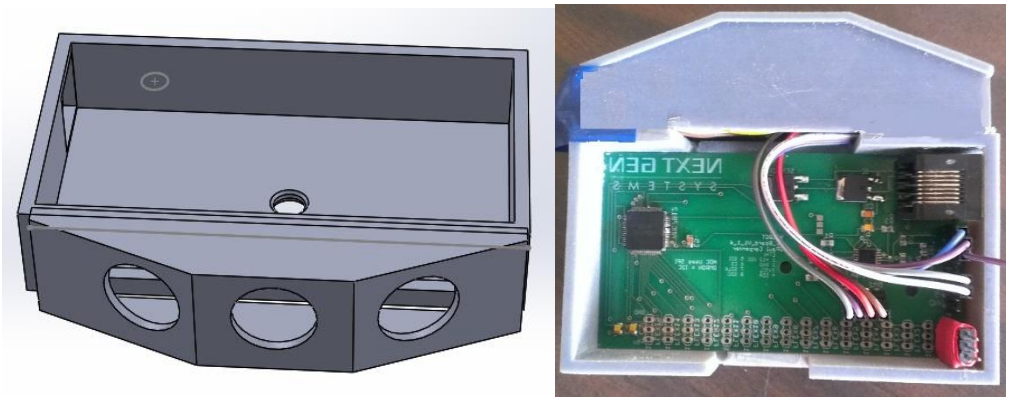

Fig 48. CAD drawing of final housing(left) and final 3D printed housing with sensors and boards fitted.(right)

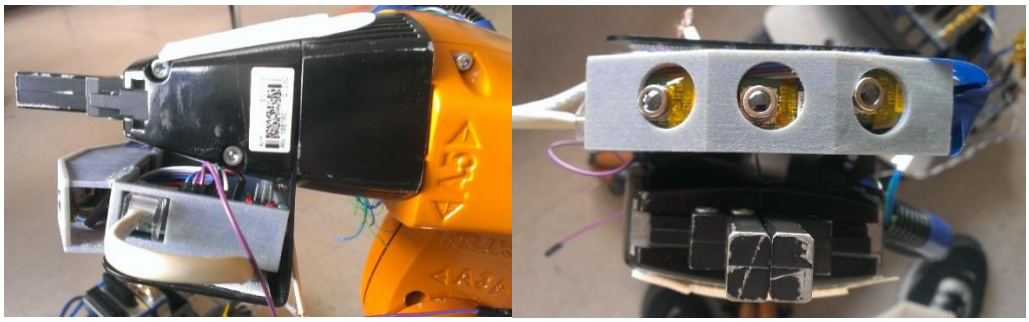

Fig 49. Kuka YouBot fitted with 3D printed box and mount for Omron sensors.

6.5 Data collection and processing

The process by which the data was collected and the different prototypes involved are detailed in this section.

The initial process involved acquiring data from 2-3 different sensors and using an Arduino Mega 2560 board to preprocess the raw data and then transmit the processed data for final processing and application.

The initial schematic of the prototype board is shown in figure (47).

The below figure 50, shows the method of addressing and reading data from the ADC and connection with the cRIO.

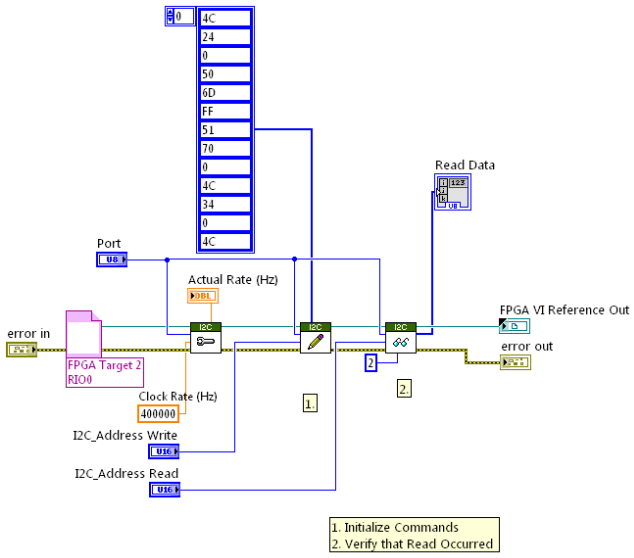

Fig 50. I2C addressing and Read and Write

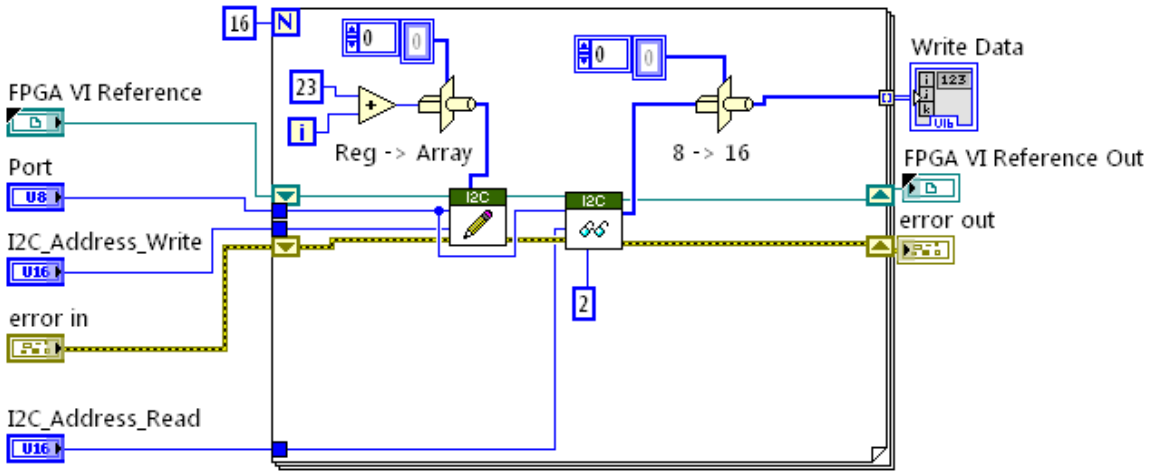

Fig 51. Reading and writing From the ADC to cRIO

Chapter 7

Conclusion

Design and development of a custom data acquisition board to collect the data from up to 19 different sensors in one centralized location and convert the analog data into digital data and transmit all the data via SPI and I2C protocols to the NI cRIO real time controller. Various methods to overcome certain default restrictions such as clock stretching in the Omron sensors, which limit the sampling rate and thus increase the sampling rate from 10Hz to 30Hz were found and implemented.

Mini signal conditioning boards were developed for the pressure sensors, thus allowing 0.023mV increments from 0-5V. These boards were basic voltage divider circuits which could be augmented with a wheatstone bridge circuit to linearize the outputs to a much better signal. Various other conditioning methods were considered. However, only the voltage divider method was implemented.

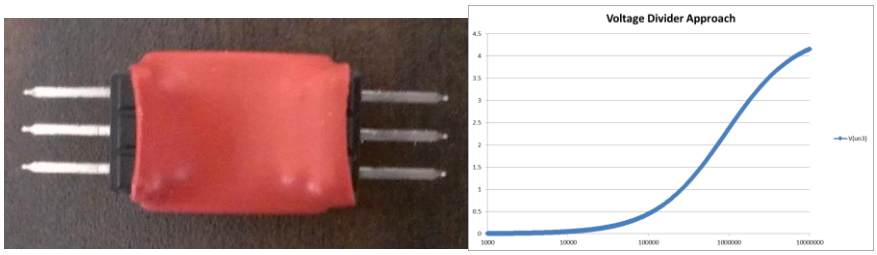

Fig 52. Signal Conditioning board with voltage divider circuit

Scalability of this system though restricted can be easily achieved by having systems which

can be addressed separately and thus made faster that the existing version (case in point: The Omron sensors).

Although the integration of the system to the robot controller was not implemented, majority of the data acquisition system was completed.

The current Micro Board prototype, due to limitations in the communication protocols, can be tackled by using multiple separate microcontrollers specifically allocated to collect and process the data from multiple sensors, thus allowing all the sensors to run and not be dependent or limited due to constraints imposed by the others. Factors such as clock stretching and too much data can be reduced or avoided and speed and precision can be increased. Separate I2C and SPI channels can be run in parallel and allowing better data transmission.

The RoboSkin patches can be modified by using custom pressure sensors and be designed to cover a larger area and make it more responsive and reduce the amount of unsensitized area on the patches.

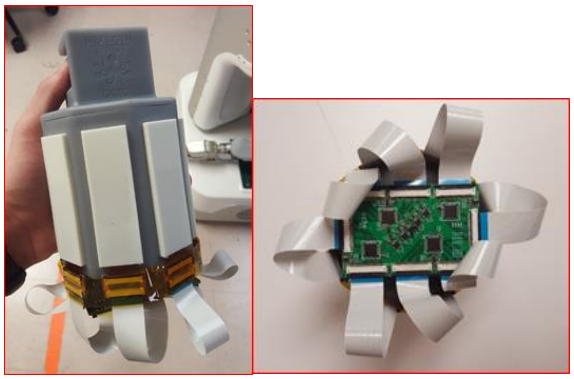

Fig 53. Subsequent testing skin patches and data acquisition board

References:

1. Chai Keong Toh Ad Hoc Mobile Wireless Networks, Prentice Hall Publishers, 2002. ISBN 978-0-13 007817-9
2. Wikipedia contributors, "Wireless mesh network," *Wikipedia,,The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Wireless_mesh_network&oldid=792086437 (accessed August 3, 2017).*
3. Amir, Yair, Claudiu Danilov, Raluca Musuăloiu-Elefteri, and Nilo Rivera. "The SMesh wireless mesh network." *ACM Transactions on Computer Systems (TOCS)* 28, no. 3 (2010): 6.
4. Firetide, I. N. C. "An introduction to wireless mesh networking." *2005-03-31)[2007-10-07]*. *http://www. firetide. com*.
5. Sun, Jing, and Xiaofen Zhang. "Study of ZigBee wireless mesh networks." In *Hybrid Intelligent Systems, 2009. HIS'09. Ninth International Conference on*, vol. 2, pp. 264-267. IEEE, 2009.
6. Wireless Mesh Protocols http://www.mouser.com/applications/wireless_mesh_networking_protocols
7. OpenWRT website: https://openwrt.org/
8. Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR*

2009. IEEE Conference on, pp. 248-255. IEEE, 2009.

9. Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248-255. IEEE, 2009.
10. "Social robot," *(*3, 2017) https://en.wikipedia.org/w/index.php?title=Social_robot&oldid=791927723
11. Hanson, David. "Expanding the aesthetic possibilities for humanoid robots." In *IEEE-RAS international conference on humanoid robots*, pp. 24-31. 2005.
12. "Frubber," (August2017) https://en.wikipedia.org/w/index.php?title=Frubber&oldid=722074884
13. DrRobot: http://www.drrobot.com/
14. Robot_pose_Ekf: http://wiki.ros.org/robot_pose_ekf
15. J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. Robotics: Science and Systems Conference (RSS). Berkeley, CA, July 2014.
16. Kejriwal, Nishant, Swagat Kumar, and Tomohiro Shibata. "High performance loop closure detection using bag of word pairs." *Robotics and Autonomous Systems* 77 (2016): 55-65.
17. Labbé, Mathieu, and François Michaud. "Memory management for real-time appearance-based loop closure detection." In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 1271-1276. IEEE, 2011.
18. Kostas J. Kyriakopoulos, Nikos Skounakis “Moving Obstacle Detection for a Skid-Steered

Vehicle Endowed with a Single 2-D Laser Scanner”. ICRA 2003: 7-12

19. Stachniss, Cyrill, Udo Frese, and Giorgio Grisetti. "OpenSLAM." *URL: http://www. openslam. org* (2007).
20. Meier, Lorenz, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. "Pixhawk: A system for autonomous flight using onboard computer vision." In Robotics and automation (ICRA), 2011 IEEE international conference on, pp. 2992-2997. IEEE, 2011.
21. Px4flow, website : https://pixhawk.org/modules/px4flow
22. Meier, Lorenz, Dominik Honegger, and Marc Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms." In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 6235-6240. IEEE, 2015.
23. Meier, Lorenz, J. Camacho, B. Godbolt, J. Goppert, L. Heng, and M. Lizarraga. "Mavlink: Micro air vehicle communication protocol." *Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22]*(2013).
24. Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. "ROS: an open-source Robot Operating System." In *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5. 2009.
25. Garage, Willow. "Robot operating system (ROS)." (2012).
26. Naseer, Tayyab, Jürgen Sturm, and Daniel Cremers. "Followme: Person following and gesture recognition with a quadrocopter." In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 624-630. IEEE, 2013.

27. Lewis, Frank L., Hongwei Zhang, Kristian Hengster-Movric, and Abhijit Das. Cooperative control of multi-agent systems: optimal and adaptive design approaches. Springer Science & Business Media, 2013.

28. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097-1105. 2012.

29. Berg, Alex, Jia Deng, and L. Fei-Fei. "Large scale visual recognition challenge 2010." (2010): 42-55.

30. Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

31. Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.

32. Wade, Joshua, Tapomayukh Bhattacharjee, Ryan D. Williams, and Charles C. Kemp. "A force and thermal sensing skin for robots in human environments." *Robotics and Autonomous Systems* (2017).

33. Mittendorfer, Philipp, and Gordon Cheng. "Uniform cellular design of artificial robotic skin." In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pp. 1-5. VDE, 2012.

34. Mittendorfer, Philipp, and Gordon Cheng. "Humanoid multimodal tactile-sensing modules." *IEEE Transactions on robotics* 27, no. 3 (2011): 401-410.

35. Engel, Jonathan, Jack Chen, Zhifang Fan, and Chang Liu. "Polymer micromachined multimodal tactile sensors." *Sensors and Actuators A: physical* 117, no. 1 (2005): 50-61.

36. VESELY, Michal, Aleksander CIESZCZYK, Yang ZHAO, and Wim ZEILER. "Low cost infrared array as a thermal comfort sensor." In *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*, no. EPFL-CONF-213353, pp. 393-398. LESO-PB, EPFL, 2015.

37. A. Habib, I. Ranatunga, K. Shook, and D. O. Popa (2014). *SkinSim: A Simulation Environment for Multimodal Robot Skin.* IEEE Conference on Automation Science and Engineering (CASE). Taipei, Taiwan.

38. Texas Instruments product website: http://www.ti.com/product/AMC7812

39. N. Instruments, "NI LabVIEW for CompactRIO Developer's Guide," 2013.

39. Leens, Frédéric. "An introduction to I 2 C and SPI protocols." *IEEE Instrumentation & Measurement Magazine* 12, no. 1 (2009): 8-13.

40. Bischoff, Rainer, Ulrich Huggenberger, and Erwin Prassler. "Kuka youbot-a mobile manipulator for research and education." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1-4. IEEE, 2011.

Biographical Information

Raghavendra Sriram is from Mumbai, India. He received his Bachelor in Engineering in Electrical and Electronics Engineering from Canara Engineering University, Mangalore. India in 2012. He earned his Master degree in Electrical Engineering from the University of Texas at Arlington and worked as a Graduate Research Assistant at the University of Texas at Arlington Research Institute. His current research interests include Autonomous Vehicle Systems, Control Systems, Distributed Control Systems, Automation, Embedded firmware, Robotics, Consumer Electronics and Real-time systems.