2018 Intel Cup Undergraduate Electronic Design Contest
- Embedded System Design Invitational Contest

Final Report

Intel Cup Embedded System Design Contest

Project Name: Intelligent Rescue Operation Bot

Students: Abhavya Chandra
Abhinav Narayan Harish
Sagar Gupta

Faculty: Prof. Joycee Mekie
University: IIT Gandhinagar

2018 Intel Cup Undergraduate Electronic Design Contest
- Embedded System Design Invitational Contest

Declaration of Originality

We hereby declare that this thesis and the work reported herein was composed and originated entirely by ourselves. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given in the references.

Team Members Signature: _____

Name (in Block Letters): Abhavya Chandra,
Abhinav Narayan Harish
Sagar Gupta

Date: 09-07-2018

Intelligent Rescue Operations Bot

ABSTRACT

We propose an Intelligent Rescue Operations Bot, to assist the rescue team to plan and prioritize their rescue operations by providing them real time data from the rescue site. The details includes various live feedback of the processed rescue site image, real time 3D/2D map of the area along with localization of the bot in that region. The processing of the live feed image involves Dehazing, person detection and classifying potential alive person on the rescue site. The bot is meant for assistance and requires an active user interaction. All the streaming of the feed from the rescue site is done on a local area network.

Keywords: Person Detection, SLAM, Live Feedback, Motion Magnification, Indoor Localization, Dehazing, Wireless bot control

Chapter 1:
Introduction

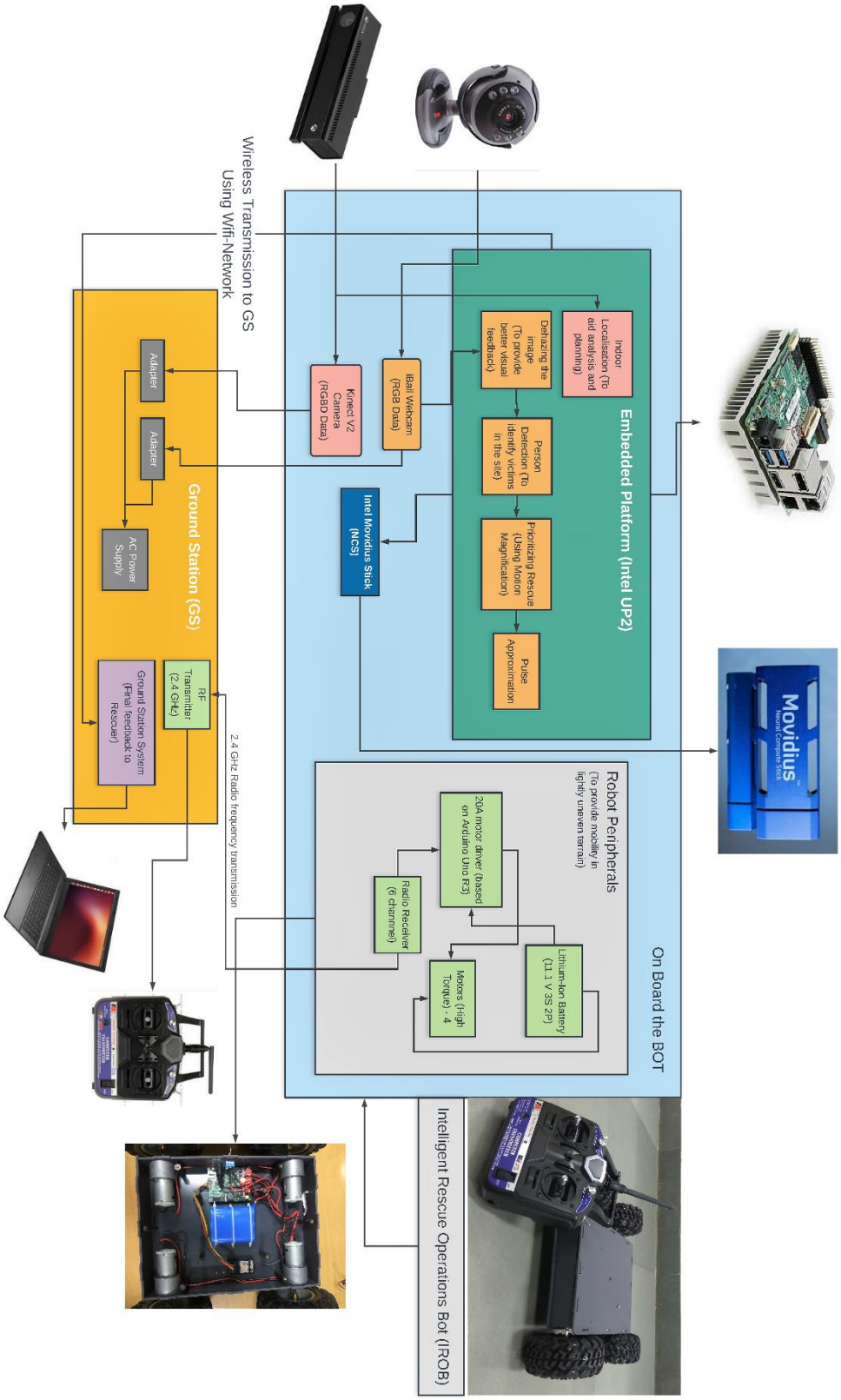
Motivation:

Fire accidents and fire related injuries claim about 120,000 victims every three years. The Indian National Fire Protection Association claims that majority of the deaths that occur due to fire are a result of smoke inhalation rather than burns. Research shows that, oxygen level of less than 9% can lead to unconsciousness and less than 6% can lead to death. Despite knowing the complete layout of their homes, victims are unable to escape to safety.
On a similar note, the Bhopal Gas tragedy of 1984, which involved the accidental release of a gas *Methyl Isocyanide* led to the death of about 8,000 people in under a span of 2 weeks. Considering the lethal nature of this gas, rescue operations were ill planned and unorganized. Several rescuers faced severe health complications which last even today.
Considering the health hazard that rescuers face in rescue operations from disasters involving smoke, haze or poisonous leakages, we propose our solution, **'The Intelligent Rescue Operations Bot'**

Domain of Operation:

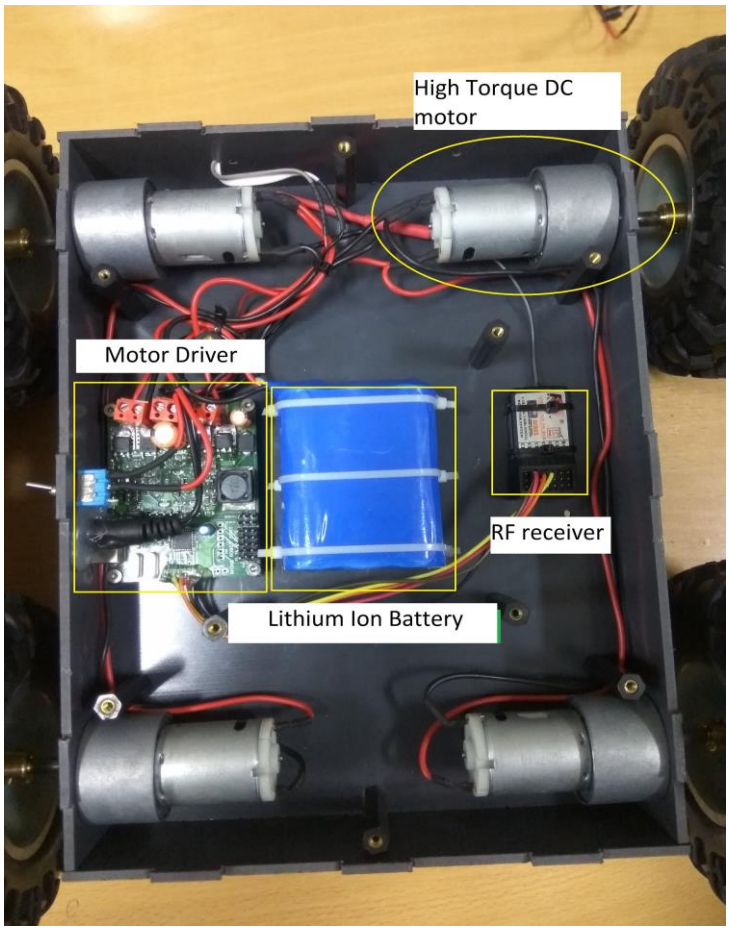
We propose an Intelligent Rescue Operations bot to provide clear visual feedback, detect possible victims and help rescuers prioritize their rescue operation.
Our bot can work in situations involving haze or light smoke(translucent), which can arise after a fire or discharge of a poisonous gas where direct human intervention is not feasible. However, our bot cannot function effectively in a full-fledged fire considering very high temperatures which may cause our embedded platforms which include UP^2 and camera to malfunction.
The bot will be able to provide clear visual feedback in mild to moderate haze, but not thick dense smoke which is opaque in nature. Since all the real time feed related data transfer is done using TCP/IP a local area network is required. The wireless controlling of bot can only be done through radio frequency upto a range of 50-60 meters.

System Schema:



Chapter -2
Individual Hardware Components:

a. Manually Controlled Robot:



1. **High Torque DC motors:**

High torque DC motors having a no-load speed of 300 rpm are being used in the project. It has a torque rating of 30kg - cm. The maximum current that it can handle is 7.5A.

2. Radio Frequency RX/TX module:

The Radio Frequency transmitter and receiver module, is used for transmitting and receiving the control signal. It operates in 2.4Ghz frequency and has a 1km line of sight range. The radial range is about 500m. It has a maximum current rating of 20A. The larger range would enable the bot to be effectively controlled from the ground station.

3. Rechargeable Lithium Ion battery:

We are using a rechargeable Lithium ion battery rated at 11.1V consisting of 3 cells in series and two such combinations in parallel. We are using higher voltage rechargeable batteries to supply the high torque DC motors and to simplify the charging procedure.

4. Arduino Uno R3:

We are using a 20A motor driver based on Arduino UNO R3. It has terminal to connect two motors and can give an output of 20A at either.

b. Hardware on the bot:

- **UP^2 development board**

  All major computations are performed on the Intel UP^2 development board. The UP^2 runs on Ubuntu 16.04-LTS. The major packages installed on UP^2 includes- Opencv-Python, Caffe, NCS API, ROS.

- **Movidus NCS**

  The Neural Compute stick is a deep learning edge device. It is the first device that tried to bring deep learning to edge instead of cloud, as most of practices are followed today. The NCS runs on Myriad 2 VPU present on the device itself. It is an inference device and can only be used for testing purpose.

- **Kinect V2 Xbox One:**

  The Kinect Xbox One is RGB D sensor which uses the Time of Flight mechanism to find the depth of a point wrt to the centre origin of kinetic. We use kinetic to generate 3D reconstruction of rescue site using point cloud data.

- **Webcam:**

  We are using an Iball webcam for Dehazing, Person detection and Motion Magnification.

Chapter 3:
The Features of our Project

Section 3.1 : The All Terrain Bot for Navigation:

We are using an all-terrain bot for navigation within the affected area. As mentioned earlier the bot is controlled by a radio frequency using a transmitter and receiver module using a RF controller.

RF Transmitter

RF Receiver

The specified range of the Radio frequency module 1 km line of sight. The wheels are 125 mm in diameter, and provide suspension, owing to its flexibility.

It is capable of navigating on uneven terrains that may appear during a Disaster scenario.

Section 3.2 Providing Clear Visual Feedback:

In this section we discuss how we can improve the quality of user feedback, providing better visual aid, by dehazing the input video feed by non-traditional technique of image reconstruction via deep-learning. Till now three deep learning approaches has been used to reconstruct dehaze image out of haze image, namely DehazeNet, MSCNN (multi-scale Convolution Neural Network) and AOD-Net (All-in-One-Network). We tried DehazeNet and later switched to AOD-Net.

Implementation:

To prototype our idea we implemented the pre-trained model of AOD-Net first on my ASUS laptop with Intel(R) Core(TM) i5-6200U CPU with 2.30GHz clock speed and later deployed on UP Square with Intel(R) Celeron(R) CPU N3350 with 1.10GHz clock speed. The time taken to perform dehaze CNN on a 300x300 image took 0.20658s on laptop while on UP Square it took 0.47023s. The CPU usage involved in both the cases were close to 95% which left no room for any other process to perform simultaneously on UP Square.

AOD-Net is implemented using caffe model, trained on synthetic dataset [3] created by atmospheric scattering model [5] in which haze was added using the atmospheric scattering model.

Prior to AOD-Net we tried pre-trained DehazeNet model, the first deep learning approach for single image dehazing.

Results:

The Results for both are highlighted in the following paragraph:

- IROB has high chances to face both Haze and Haze-Free conditions. As seen in image as shown below, Dehaze Net leads to loss of feature of region of slightly higher brightness, while AOD-Net diminishes the brightness but keeps tries to retrieve more features.

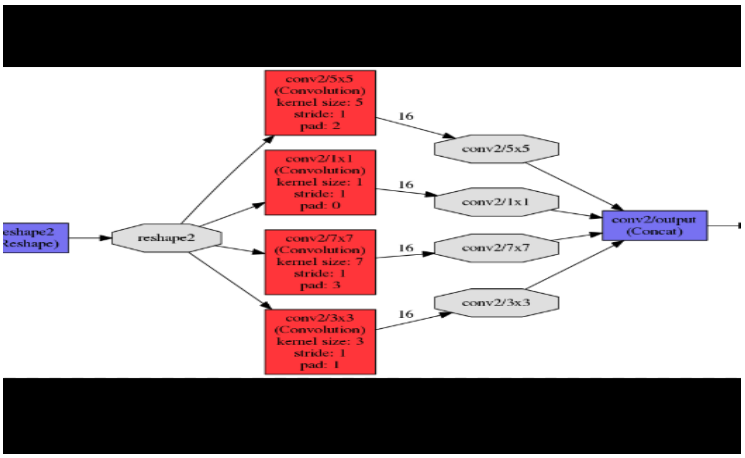
AOD-Net DehazeNet Original

- Result comparison of original image, image from AOD-Net and DehazeNet. (images are in order AOD-Net, DehazeNet and original)

- Standard metric scale comparison like PSNR and SSIM as stated by [3] on the synthetic and real data sets proves AOD-Net to be better than other state-of-the-art dehazing methods like, DehazeNet and MSCNN.

Metrics	ATM [33]	BCCR [20]	FVR [35]	NLD [1], [2]	DCP [9]	MSCNN [27]	DehazeNet [3]	CAP [45]	**AOD-Net**
PSNR	14.1475	15.7606	16.0362	16.7653	18.5385	19.1116	18.9613	19.6364	**19.6954**
SSIM	0.7141	0.7711	0.7452	0.7356	0.8337	0.8295	0.7753	0.8374	**0.8478**

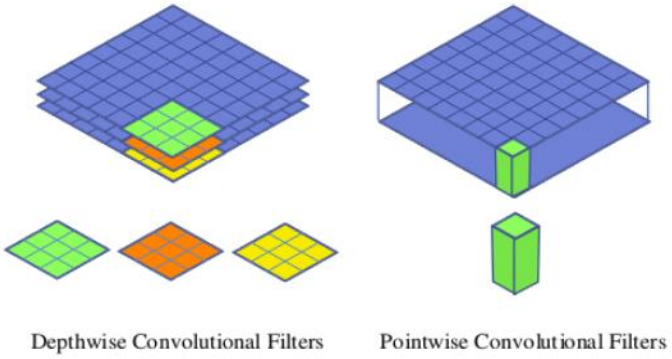
- In case of DehazeNet parallel convolution takes place in its second layer, which makes it computationally expensive. Below is the caffe second layer of DehazeNet.



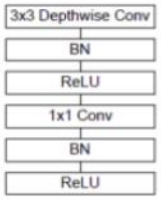
Conclusion : DehazeNet gave unfavorable results as shown above. AOD-Net is an end-to-end, light weight and involves less latency while in case of DehazeNet final dehazed image is output after estimation of two independent parameters that AOD-Net predicts as one parameter, which makes it suitable for IROB.

Section 3.3: Detecting victims on the rescue site

After Dehazing the input video stream, we propose person detection for identifying possible victims, which is not clearly visible to the naked eye.
We apply Object Detection and then threshold it to only detect person in streaming frame. Convolution Neural Networks have outperformed other traditional methods of Object Detection. This has been possible due to high and low-level feature extraction capabilities, multi-scale feature detection and its performance.
Person detection is a computationally expensive task as it involves both localization and classification tasks. When dealing with embedded platforms, the constrained resources on the board restricts the performance of object detection. Due to immense research going on in this field, everyday new CNN approaches are introduced. Few state of art algorithms being YOLO (You Look Only Once), Faster R-CNN (Regional - Convolutional Neural Network), MobileNet and SSD(Single Shot Detection) .
During the research phase we tried variant pre-trained models of YOLO and MobileNet on UP^2 . We came with a conclusion that combination of MobileNet and SSD suit our purpose well.
MobileNet architecture was designed by Google Inc. in 2017 to narrow the gap between computer vision and embedded platform. Keeping in mind the limited computational capability on embedded platform the model is designed to be small size with less latency. To reduce the computations involved the regular convolutional layer are replaced by *depthwise separable convolution* blocks.
Image - [11]



Depthwise Convolutional Filters Pointwise Convolutional Filters



Depthwise Separable Convolution

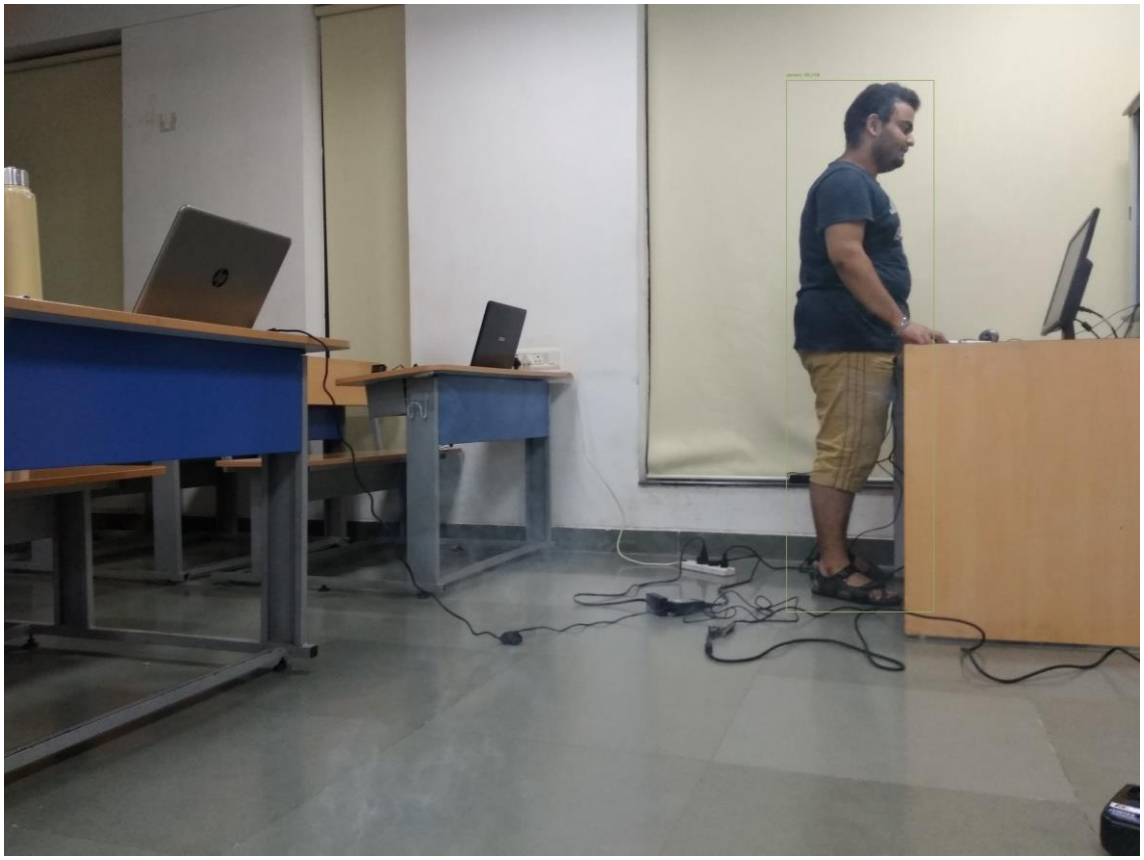
In recent times SSD are found to be more accurate than YOLO and can be easily concatenated with other networks, therefore fusion of MobileNet and SSD performs well in embedded object detection problem.

Implementation:

We used pre-trained model of MobileNet-SSD. The model is conversion of model originally implemented on tensorflow framework. The Model was trained on COCO dataset [12] but since we mainly focus on finding person we choose the model that was fine-tuned on PASCAL VOC dataset [13], which has only 20 different object classes with one background class. Initially we implemented the caffe model on my ASUS laptop with Intel(R) Core(TM) i5-6200U CPU with 2.30GHz clock speed and later deployed on UP^2 with Intel(R) Celeron(R) CPU N3350 with 1.10GHz clock speed.

Results:

Sample Output:



On the UP^2, the frame rate we obtained for the MobileNet SSD network 1.25 to 1.5 fps.
The two algorithm, MobileNet SSD + AOD-Net stream at a rate of 1.0 to 1.3 fps.

Conclusion:
The results obtained proved that just using cpu for the computer vision tasks was not suitable for IROB. Apart from this the cpu usage in all most of the cases reached 100% which left very less room for any process to run. The only option we were left with was to employ an external hardware like Movidus NCS that could reduce the load on UP^2.

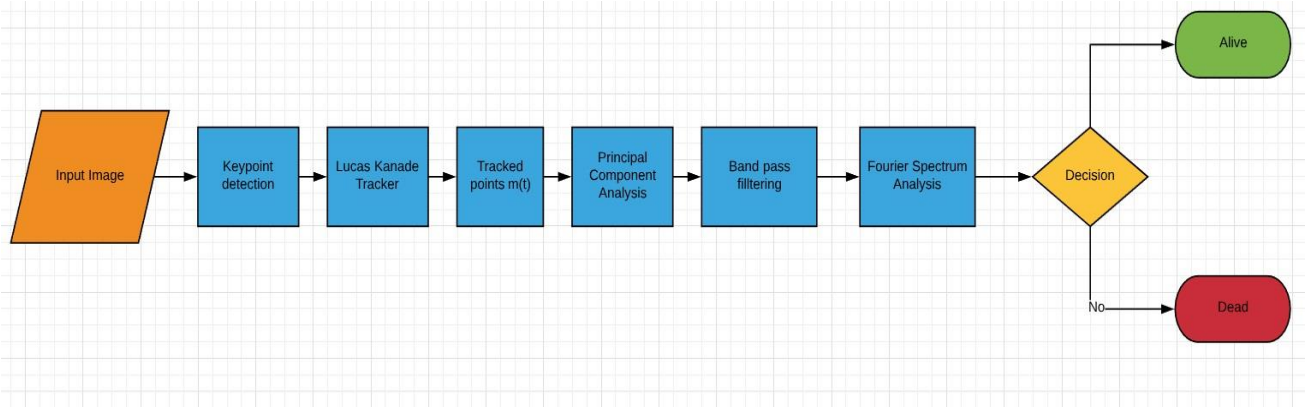
Section 3.4: Prioritizing the Rescue Operation:

In situations involving life and death, where time is of essence, planning of the rescue operation enables rescuers to save maximal number of victims most efficiently.

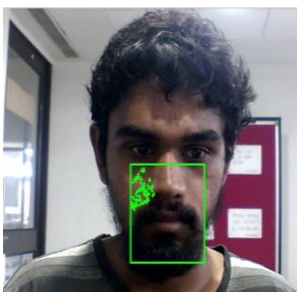
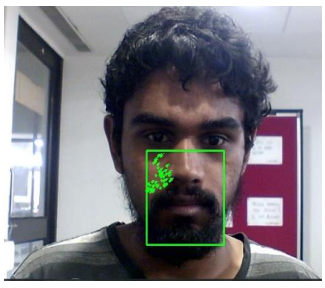
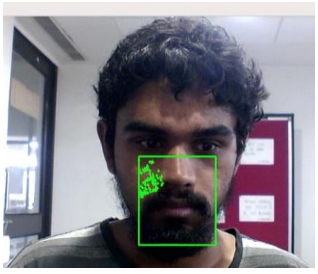
Assumption:
In this implementation we make one assumption, that the body of a person who is dead, shows negligible motion or change in position. However, this may not always be the case, tiny motions may still be observed when the person is dead. Our bot only takes into account only this one parameter to carry out the classification.

Our bot has a high false positive rate and a low false negative rate. This means if the person is alive he would most probably be classified correctly. However, if the person may be dead, it is possible he is classified incorrectly.

Implementation:
To classify the victims as alive or dead by analyzing the local motion of tracked keypoints on the person. Currently, the algorithm is demonstrated for the person's face, however, we plan to modify our algorithm to work for the abdomen region.



First we obtain the input image from the camera, apply a key point detection to identify points on the person's face to track over multiple frames. We are currently using Harris Corner detection in our application.

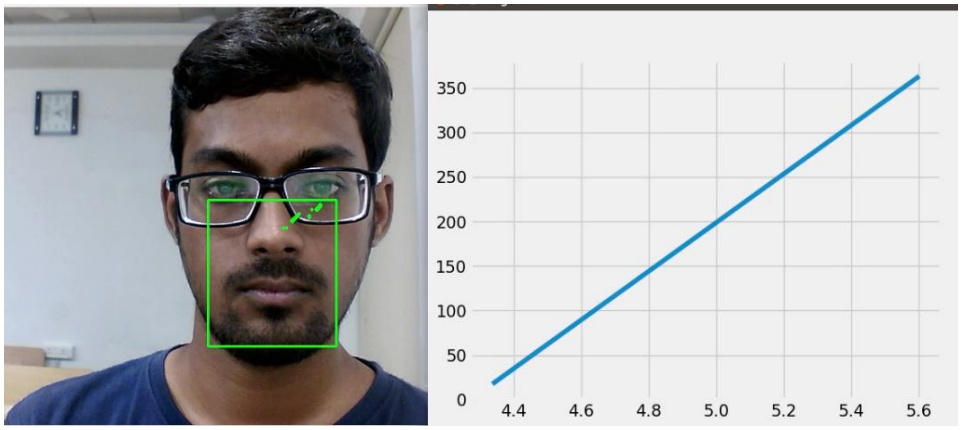
The Key Point Detection algorithm for multiple frames

The Lucas Kanade tracker keeps track of those identified points over multiple frames in a video feed. These tracked points are stored in a list. Using, principal component analysis we reduce the dimensionality of the data and identify the maximum variance component of the set of tracked points.
The band pass filtering is applied to remove high frequency components and isolate the motion components corresponding to a person's breath rate.
We plot the sum of the magnitude of the square of the magnitude of all frequencies in the fourier spectrum of maximum variance component. This is plotted with time.

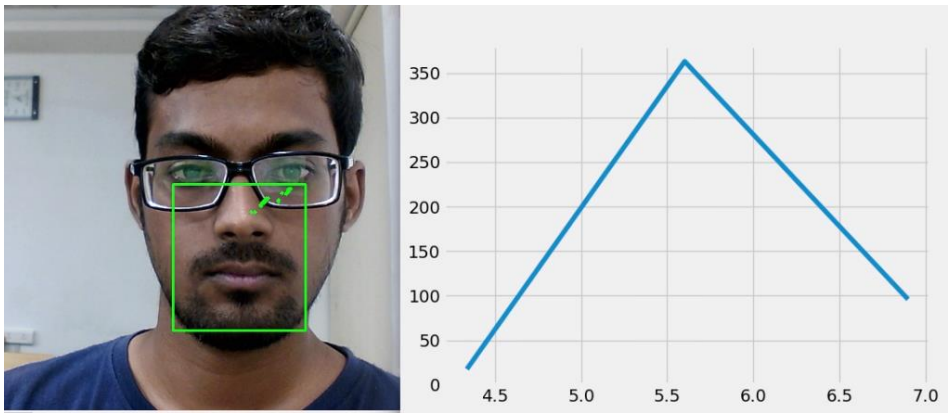
To further improve the classification we used motion magnification to amplify a video feed before applying this algorithm. This would produce better classification output.
We used the Eulerian Motion magnification considering that it is the fastest and thus most suitable for an embedded platform(UP^2)

Results and Testing:

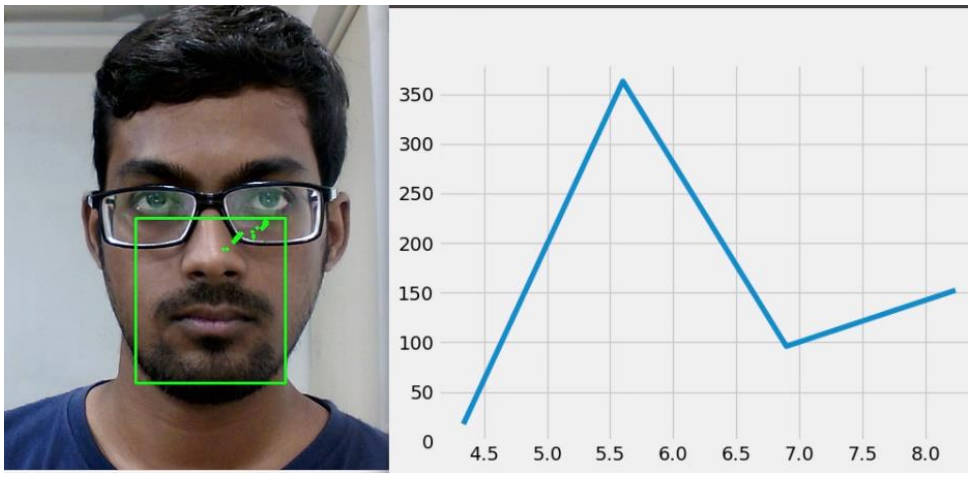
a. Ordinary Video Stream:



Identified Motion Feature(t = 5.6, y(t) = 352 units)



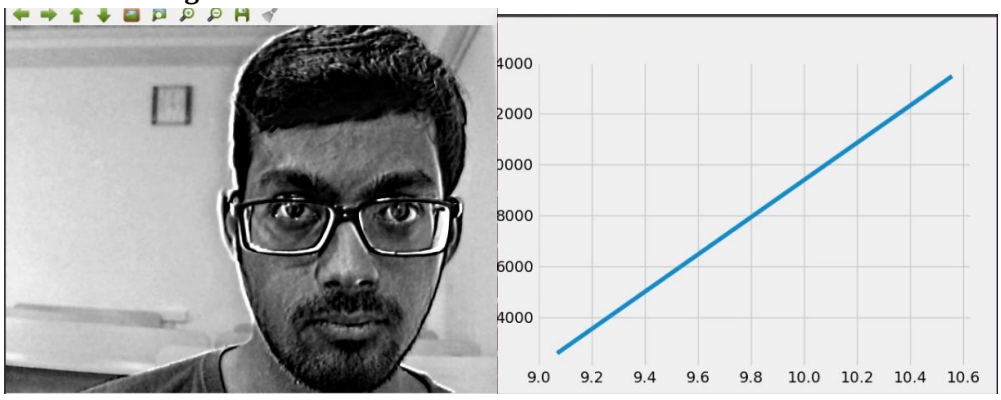
Identified Motion Feature(t = 6.9s, y(t) = 98 units)



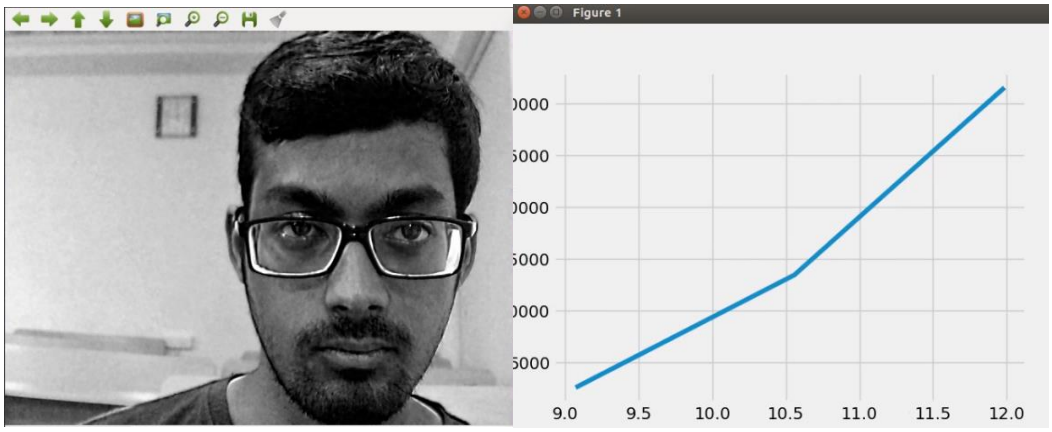
Identified Motion Feature(t = 8.2s, y(t) = 150 units)

Maximum : 352 units , **Minimum :** 98 units **Average :** 220 units
Approximate fps = 10

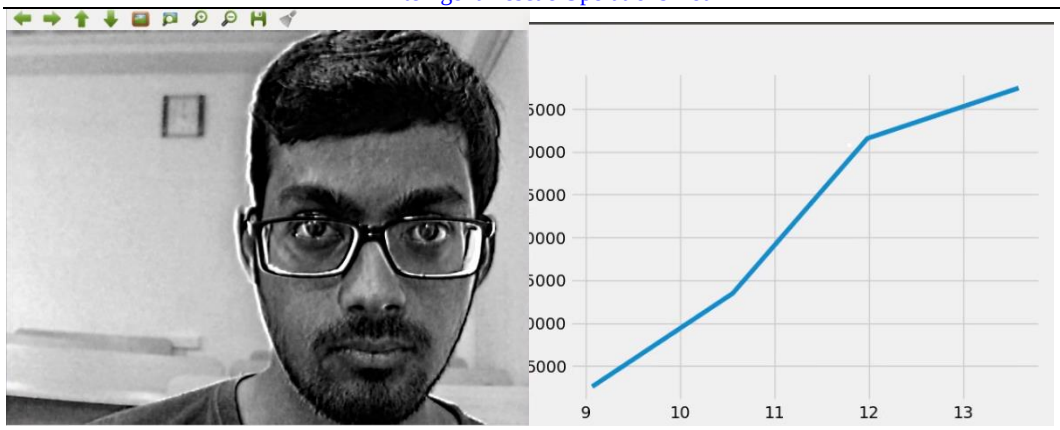
b. Motion Magnified Video Stream.



Motion Point identified(t = 10.57, y(t) = 13468.62)



Motion Point identified(t =11. 9s y(t) = 31570 units)



Identified Motion Feature(t = 13.587, y(t) = 37440 units)

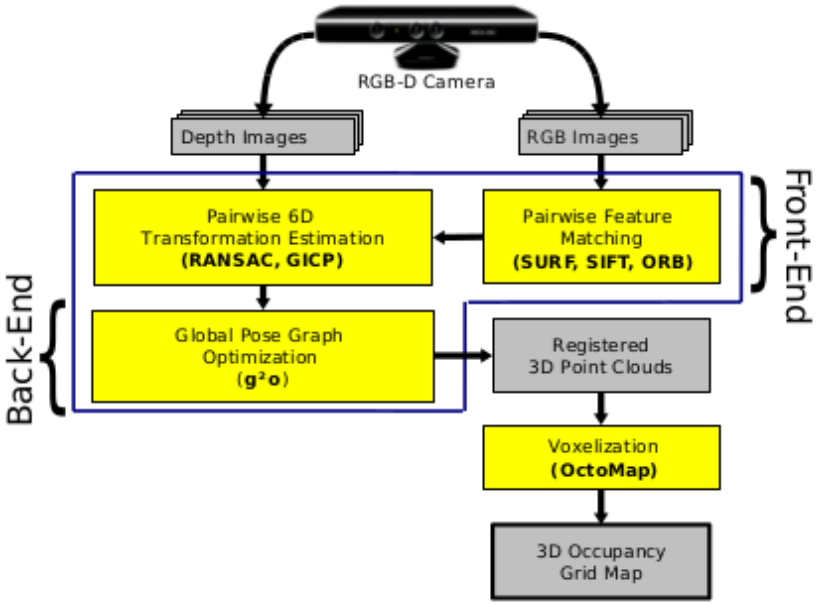
Maximum = 37440 units Minimum = 134682 units Average = 18,020 units

Considering the detected motion for motion magnified stream is more we can use it to classify a person as dead or alive.

3.4. Mapping and Localization:

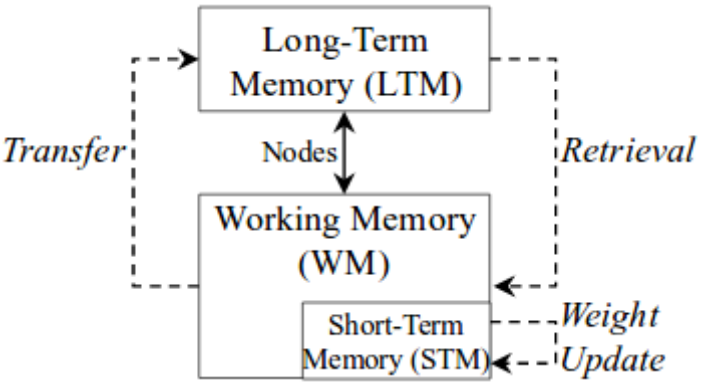
While performing rescue operations, situations when the victims get trapped in elusive places are quite common. In such scenarios being able to remotely monitor the location of the victims and be able to get a visual overview of the prescribed area would help in a better analysis of the situation and hence device a more robust rescue plan. In order to serve this purpose IROB makes use of SLAM algorithms.
Implementation of SLAM involves the processing of odometry, pose and depth data These data can be obtained by using sensors like depth/ stereo cameras, laser scanners (LIDAR), IMU's and motor encoders. IROB is equipped with a Kinect V2 (XBOX One) which has a RGB sensor, and an Infrared (IR) emitter and detector. We have implemented RTAB-Mapping on the UP^2 board using the same sensor.

Working Principle: Real-Time Appearance-Based Mapping (RTAB-Mapping)
RTAB-Mapping is a RGBD graph based SLAM. It is based on global Bayesian loop closure detection. The detector uses a bag-of-words (BoW) approach to determine how likely a new image comes from a previous location or a new location. Upon acceptance of a new loop closure hypothesis, a new constraint is added to the map's graph and then a graph optimizer minimizes the errors in the map.
Considering the possibility that the areas to be mapped during rescue operations may be of varied sizes and that the online embedded platform has to make most out of its computational resources available, RTAP-Map makes use of a memory management approach which limits the number of frames for loop closure detection by classifying the available memory into three sections.



RGBD-SLAM

Formulation of Graph based SLAM and Tree based netwORk Optimizer (TORO):
RTAB-Map makes use of the graph/network based formulation of SLAM. Under this formulation the poses of the robot along with its odometry data and RGBD images, etc. are modeled as the nodes of a graph. The geometrical transforms and loop closure detection between these nodes are stored in the form of edges of the graph. These transforms and the poses stored in the nodes act as the constraints for map optimisation.



Memory Management:
In order to be able to map larger areas over longer terms, multi-session mapping is implemented. This is achieved by the following approach:

1. The new nodes initially flow into the STM. Here no loop closure detection is implemented since the incoming frames are very much similar.
2. When the newer incoming nodes have a similarity in the visual words, above a threshold Y, they are added in the same node. This is known as weight update.
3. Upon reaching a threshold S the oldest node in STM is then dumped into the WM. Here loop closure detection is performed.
4. The size of WM is based on a fixed time T. When processing time of the nodes in the WM exceeds T, the oldest nodes in WM are sent to the LTM.

5. The nodes in LTM are not considered for loop closure detection. However in case of detection, the neighbors of that node in LTM can be sent back to WM. This process is called retrieval.
6. To ensure that older mapping sessions are not forgotten completely (i.e. all nodes sent to LTM), the heuristic that the bot should incrementally start remembering the area of the map that it has spent most time in is used. Meaning, among the older nodes in WM the one that is lightest will be sent to LTM.

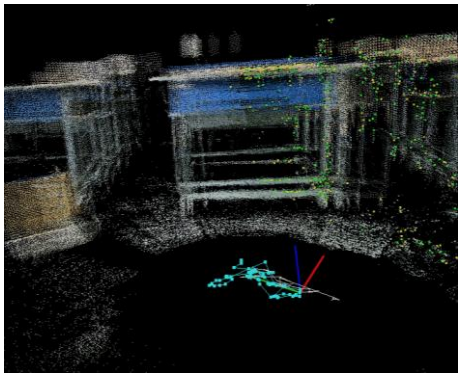
Implementation:

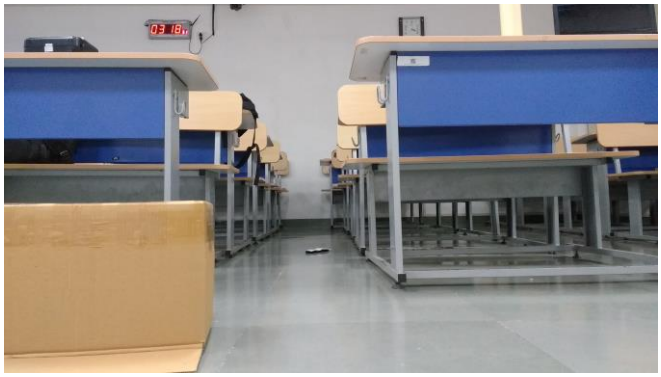
We used a ROS package of rtabmap_ros and implemented it on ROS-Kinetic/Ubuntu 16.04. It was observed that at the time of running the algorithm it made use of 100% CPU usage and about 70% memory usage. In order to not lose odometry data, the kinect needs to be rotated slowly, and the below results were acquired by rotating th ebot at an approximate speed of 8 degrees/second.

We are providing both a 3D map with rgb data overlayed over the point cloud and a 2D projection map which is made using projec_map function wherein the depth data from Kinect is converted to fake laser scan data using depth_to_laser package. It is able to provide a decent data about the top overview of the area to be mapped.

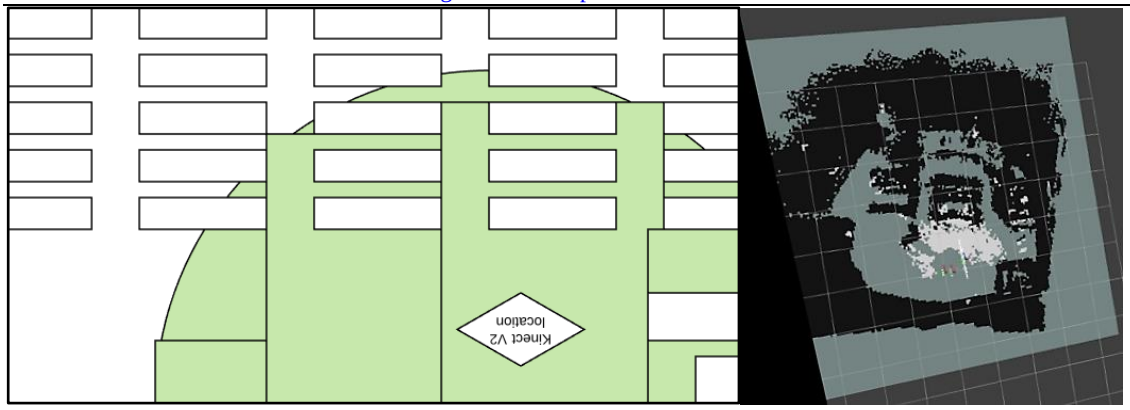
Testing and Results:

The following images show the 3D mapping of a confined space and an RGB image of the same area. Upon comparison, we can infer that the point cloud generated gives a decent visualisation of the area.





When viewing the 2D projected map it can be seen that the camera is able to make an approximate laser scan data in a circular region around the location of Kinect. The left image denotes the floor plan of the test room and the right image is the corresponding 2D projected map. The location of the camera was unchanged in either scenarios (2D and 3D). In the 2D map the grey area denotes the area traversable and the part with the grid shown correspond to blocked regions. In this test the camera was able to reach a depth of about 5m from its origin.



The above were examples of single session mapping. However, in order to be able to cover more area multiple sessions can be conducted and then the loop closure detection and graph optimisation will create a corrected global map of all sessions conducted.

3.5 Real time live video feed back -

For the live feedback purpose we are using flask, a python microframework. The server is set on the UP^2 on the bot. The network protocol used here is TCP/IP.

Chapter 4
Further application

We further confidently say that different components of IROB can be used in other domains as well. One such module is Dehazing. Today smog has covered huge area of megacities. This not only reduces visibility but also propose a big problem for self driving cars.

The dehazing module can be used to reduce the haze in the atmosphere for higher level computer vision tasks.

The dehazing module of IROB can be used in other areas where Haze is a big issue.

References -

[1] -https://arxiv.org/abs/1601.07661
[2] -https://arxiv.org/pdf/1607.07155.pdf
[3] -https://arxiv.org/pdf/1707.06543.pdf
[4] -https://github.com/Boyiliee/AOD-Net
[5] https://ieeexplore.ieee.org/document/7976290/
[6] -https://github.com/zlinker/DehazeNet
[7] -https://pjreddie.com/darknet/yolo/?utm_source=next.36kr.com
[8] -https://arxiv.org/abs/1506.01497
[9] -https://arxiv.org/pdf/1704.04861.pdf
[10] -https://arxiv.org/abs/1512.02325
[11] -https://github.com/Zehaos/MobileNet
[12] -http://cocodataset.org/#home
[13] -http://host.robots.ox.ac.uk/pascal/VOC/
[14]-https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/
[15]-http://msue.anr.msu.edu/news/smoke_inhalation_is_the_most_common_cause_of_death_in_house_fires
[16] -https://aquibjk.wordpress.com/2016/10/19/installing-caffe-on-ubuntu-16-04/
[17] -https://pypi.org/project/opencv-python/
[18]-https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad
[19]-https://www.coursera.org/lecture/convolutional-neural-networks/object-detection-VgyWR
[20]-http://cs231n.github.io/
[21]-https://www.hindustantimes.com/bhopal/no-help-for-helping-hands-of-bhopal-gas-tragedy/story-YFGGr7w9T69tM4bc7653pM.html
[22]-http://flask.pocoo.org/static/logo/flask.png
[23]https://robokits.co.in/wireless-solutions/ir-rf-remote-control/6ch-2.4ghz-trasmitter-reciever-1km-for-2-dc-motors-20a-drive?cPath=7
[24]-https://www.lucidchart.com
[25]TWiedemeyer “IAIKinect2,”https://github.com/code-iai/iai_kinect Institute for Artificial Intelligence, University Bremen, 2014 – 2015,accessed June 12, 2015
[26]https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6459608&tag=1
[27]https://introlab.3it.usherbrooke.ca/mediawiki-introlab/images/e/eb/Labbe14-IROS.pdf
[28]http://www2.informatik.uni-freiburg.de/~stachnis/pdf/grisetti07iros.pdf
[29]http://www2.informatik.uni-freiburg.de/%7Eendres/files/publications/felix-endres-phd-thesis.pdf
[30]https://pdfs.semanticscholar.org/59fc/0378d941a6caa8450207d6de01bd131c972a.pdf
[31]http://introlab.github.io/rtabmap/
[32]https://github.com/introlab/rtabmap_ros#rtabmap_ros
[33]https://github.com/OpenKinect/libfreenect2
[34]https://github.com/ros-drivers/freenect_stack
[35]http://www2.informatik.uni-freiburg.de/~stachnis/toro/
[36]https://ieeexplore.ieee.org/document/6619284/
[37]https://github.com/trachelr/pulse_video
[38]http://people.csail.mit.edu/mrub/vidmag/
[39]https://github.com/miguelgrinberg/flask-video-streaming
[40]google image