B.Tech Project Report

# Semi-Autonomous Fire Extinguishing Robot

Anukul Kumar
B18ME013

Oswal Vinit Jitendra
B18ME040

# Table of Contents

1. Abstract
2. Introduction
3. PyBullet Implementation
    3.1. Design
    3.2 Path Planning Algorithm
    3.3 Implementation
4. ROS Implementation
    4.1 Robot Specification
    4.2 Environment and Robot Setup
    4.3 Algorithms
    4.4 Implementation on Turtlebot3 Waffle Pi
5. Conclusion
6. References

# 1. Abstract

In the last decade the robotics industry has boomed with constant improvement in technology and new milestones are reached everyday . In short, the very next decade robotics will become vital components in a number of applications and robots paired with AI will be able to perform complex actions that are capable of learning from humans, driving the intelligent automat-ion phenomenon. Robots can be very useful in performing tasks which are dangerous for humans and there is risk to life.  In this paper, we try to solve one such problem, industrial fires. We propose a robot model to effectively detect and extinguish fires with minimum human intervention.

## 2. Introduction

In today's world of industrialization and economic growth, there has been a drastic rise in the number of industries and factories around the world. Today's industries are highly automated and operate using very heavy machinery. The fires encountered in the buildings and facilities of industrial areas make up quite a small proportion of all fire cases. However, in terms of the heat release rate, size of the burned area, damage, and impact zone, such fires have a large impact as compared to other fire cases. All fires have risk in terms of propagation. However, since fires in industrial structures and plants have rather high levels, qualitatively and quantitatively, compared to residential fires and other types of fires,[1] These machines are operated at full capacity and sometimes even in harsh climatic conditions. Even with the sophistication of equipment and fail safe mechanisms, the risk of malfunctions still remains. There is also a possibility of human errors in these conditions. Even though the chances of these disasters happening have been minimised to a great extent, there is still a need for a mechanism to minimise the damage even further.

In the next decade robotics will become vital components in a number of applications and robots paired with AI will be able to perform complex actions that are capable of learning from humans, driving the intelligent automation phenomenon[2]. Robots can be very useful in performing tasks which are dangerous for humans and there is risk to life. Robots would be especially useful in performing tasks of high risk where the normal human life is endangered. Detecting fire and extinguishing is a hazardous job for a fire extinguisher, it often risks the life of that person[3].

Nowadays, the fire departments around the world have started taking the help of robotic technologies for fire extinguishing purposes. The current robots are generally manually controlled and need a trained person to operate them. Moreover, these robots can only perform certain tasks and are not equipped to work in a dynamic environment. The amount of research done in the field of semi and fully automated fire extinguishing robots is limited.

In this paper, we try to introduce a new way to implement robotics for efficient fire detection and extinguishing. We present a mobile robot design that can be installed in an industrial setting. We have designed a simulation of a fire extinguishing robot in PyBullet Physics SDK. We then implement the ideas simulated in pybullet on a Turtlebot3 Waffle Pi robot with a OpenCR Manipulator and Raspberry Pi camera using Robot Operating System (ROS). As soon as a fire breaks out, the robot can react with immediate effect, scan the area for potential fires and extinguish them as soon as possible. Also, the data collected by the robot about the victims and fire can be used by the fire department afterwards to investigate the cause and severity of the fire.

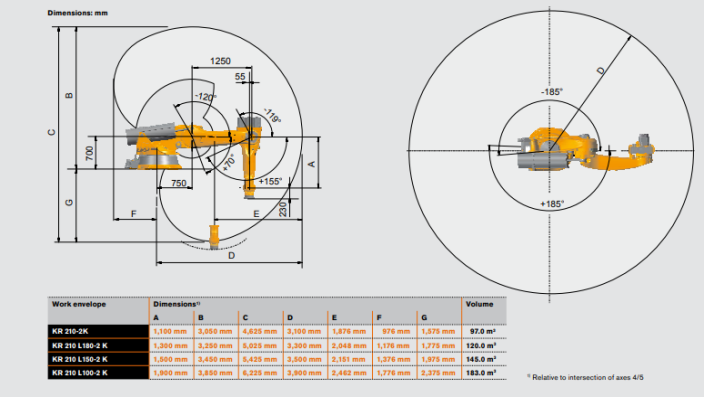
# 3. PyBullet Implementation

## 3.1 Design

**Husky base**- It is a rugged, outdoor-ready unmanned ground vehicle (UGV), suitable for research and rapid prototyping applications.

Image Source- http://wiki.ros.org/Robots/Husky#:~:text=Husky%20A200%20is%20a%20rugged,that%20you%20have%20Fuerte%20installed.

**Kuka KR210 arm -** The KUKA KR 210 industrial robot arm is a high payload solution for serious industrial applications. With a high payload of 210 kg and a massive reach of 2700 mm, the KR 210 KR C2 robot is ideal for a foundry setting




| Work envelope | Dimensions[1] | | | | | | | Volume |
|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | |
| KR 210-2K | 1,100 mm | 3,050 mm | 4,625 mm | 3,100 mm | 1,876 mm | 976 mm | 1,575 mm | 97.0 m² |
| KR 210 L180-2 K | 1,300 mm | 3,250 mm | 5,025 mm | 3,300 mm | 2,048 mm | 1,176 mm | 1,775 mm | 120.0 m² |
| KR 210 L150-2 K | 1,500 mm | 3,450 mm | 5,425 mm | 3,500 mm | 2,151 mm | 1,376 mm | 1,975 mm | 145.0 m² |
| KR 210 L100-2 K | 1,900 mm | 3,850 mm | 6,225 mm | 3,900 mm | 2,462 mm | 1,776 mm | 2,375 mm | 183.0 m² |

Source-https://www.kuka.com/-/media/kuka-downloads/imported/6b77eecacfe542d3b736af377562ecaa/pf0031_kr_2102_k_en.pdf

## 3.2 Path planning Algorithm

**A-star search** - A graph traversal and path search algorithm, which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency[10].

Getting the location of the robot and next red block, the algorithm can calculate the heuristics to find the best path for the robot, provides shortcuts, solve problems and makes judgement more efficiently and quickly.

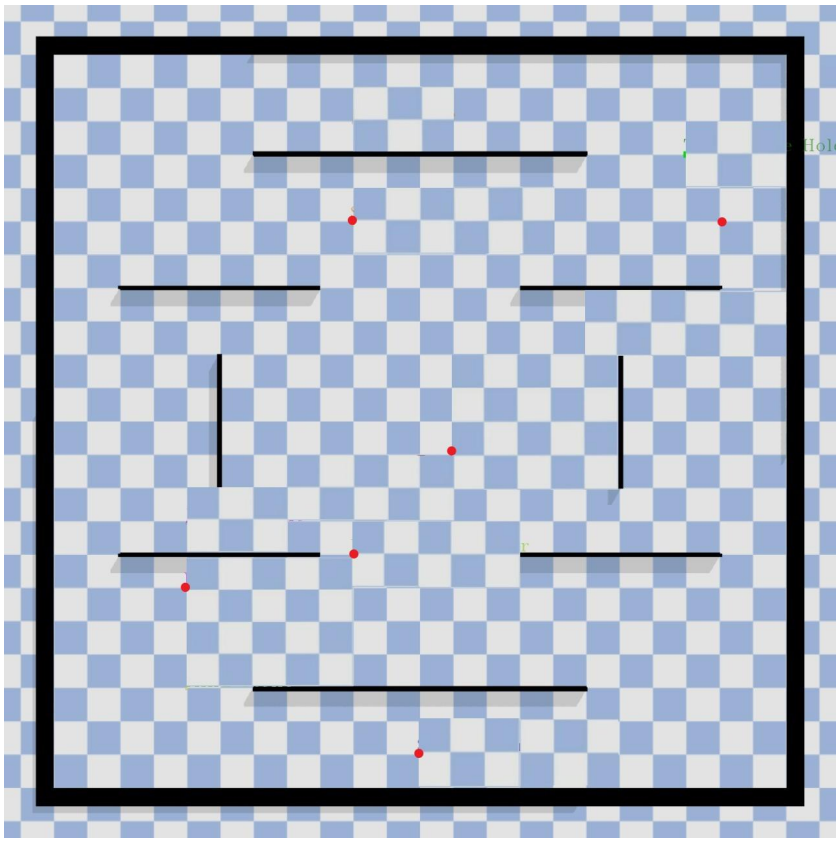


Fig: Map of the PyBullet environment

## 3.3 Implementation

In this implementation, the robot model is first loaded in the PyBullet environment along with the environment variables. The environment is preconfigured to include blocks of red colour at some positions on the map as shown in the above figure. The robot is spawned on the (0,0,0) co-ordinates. In this implementation, the red boxes are used to represent

fire. In a physical model implementation, the red color detector model can be changed to the fire detection model.

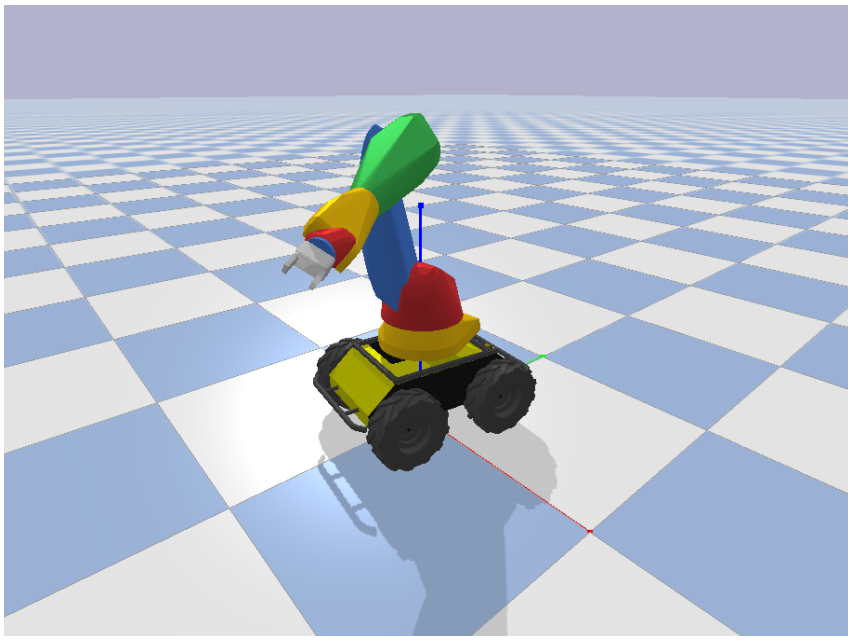


Fig: Pybullet model of Husky mobile robot with a KUKA KR210 mounted on it.

The fire detection program then starts and the following steps are implemented by the program as long as the robot doesn't reach all the six red blocks placed in the map.

1. Get the coordinates of the red patches stored in the map.
2. Calculate heuristics and determine the best path to reach the goal using A star search.
3. Move the husky base on the path returned by the A star algorithm.
4. Pass the coordinates of the block to the kuka arm and calculate the inverse kinematics needed for the robot to move to the desired point in space.
5. Pass the calculated path instructions to the KUKA arm and execute them.
6. Once the arm goal is reached, send the current location of the robot and location of the next block to A star algorithm.

# 4. Real time Implementation using ROS

## 4.1 Robot Specifications

Fig: Turtlebot3 Waffle Pi with OpenCR manipulator and Raspberry Pi camera

| TurtleBot3 Waffle Pi Specifications | |
|---|---|
| **Maximum translational velocity** | 0.26 m/s |
| **Maximum rotational velocity** | 1.82 rad/s (104.27 deg/s) |
| **Maximum payload** | 30kg |
| **Size (L x W x H)** | 281mm x 306mm x 141mm |
| **Weight (+ SBC + Battery + Sensors)** | 1.8kg |
| **Threshold of climbing** | 10 mm or lower |
| **Expected operating time** | 2h |
| **Expected charging time** | 2h 30m |
| **SBC (Single Board Computers)** | Raspberry Pi 3 Model B and B+ |
| **MCU** | 32-bit ARM Cortex-M7 with FPU (216 MHz, 462 DMIPS) |
| **Remote Controller** | RC-100B + BT-410 Set (Bluetooth 4, BLE) |
| **Actuator** | XL430-W210 |
| **LDS (Laser Distance Sensor)** | Raspberry Pi 3<br>360° Laser Distance Sensor LDS-01 (360°LiDAR) |
| **Camera** | Raspberry Pi Camera Module v2.1 |
| **IMU** | Gyroscope 3 Axis<br>Accelerometer 3 Axis<br>Magnetometer 3 Axis |
| **Power connectors** | 3.3V / 800mA<br>5V / 4A<br>12V / 1A |
| **Expansion pins** | GPIO 18 pins<br>Arduino 32 pin |
| **Peripheral** | UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4 |
| **DYNAMIXEL ports** | RS485 x 3, TTL x 3 |
| **Audio** | Several programmable beep sequences |
| **Programmable LEDs** | User LED x 4 |
| **Status LEDs** | Board status LED x 1<br>Arduino LED x 1<br>Power LED x 1 |
| **Buttons and Switches** | Push buttons x 2, Reset button x 1, Dip switch x 2 |
| **Battery** | Lithium polymer 11.1V 1800mAh / 19.98Wh 5C |
| **PC connection** | USB |
| **Firmware upgrade** | via USB / via JTAG |
| **Power adapter (SMPS)** | Input : 100-240V, AC 50/60Hz, 1.5A @max<br>Output : 12V DC, 5A |

Fig: TURTLEBOT3 Waffle Pi Specs chart

Source: ROBOTICS Turtlebot3 Waffle Pi Documentation (https://www.smartrobotworks.com/TurtleBot3-Waffle-Pi.asp)

## 4.2 Environment and robot setup

### a. Robot Model Setup

Turtlebot3 Waffle Pi with ROS Kinetic is used in this project.

The OpenCR manipulator is installed on the top surface mesh of the robot.

A Raspberry Pi is connected to the Raspberry Pi and installed in the camera holder at the front of the robot.

A Raspberry Pi 3 360 degree Laser Distance Sensor (LIDAR) is installed on the top mesh of the Waffle Pi in front of the OpenCR manipulator.

### b. PC Setup

We use a PC installed with Ubuntu 18.04 Operating System

ROS Melodic is installed in the PC along with MoveIt 1

The following Turtlebot3 packages are also installed in the ROS workspace

1. Turtlebot3
2. Turtlebot3 description
3. Turtlebot3 navigation
4. Turtlebot3 bringup
5. Turtlebot3 msgs
6. Turtlebot3 slam
7. Turtlebot3 teleop
8. Turtlebot3 manipulation
9. Turtlebot3 manipulation simulations
10. Open Manipulator dependencies
11. ROS Navigation stack
12. ROS geometry stack
13. ROS MoveIt
14. Fire extinguisher package

## 4.3 Algorithms

### Adaptive Monte Carlo Localization (AMCL) Algorithm

The Monte Carlo localization algorithm is a probabilistic localization algorithm applied to a two-dimensional occupation grid map [7], which uses the particle filter algorithm [8]

Traditional AMCL algorithm has a limited pose accuracy because of the nonconvexity of the laser sensor model, but building a laser radar-based robot localization system in a complex and unstructured environment, with a LIDAR point cloud scan-matching process enhances the localization accuracy of mobile robots.

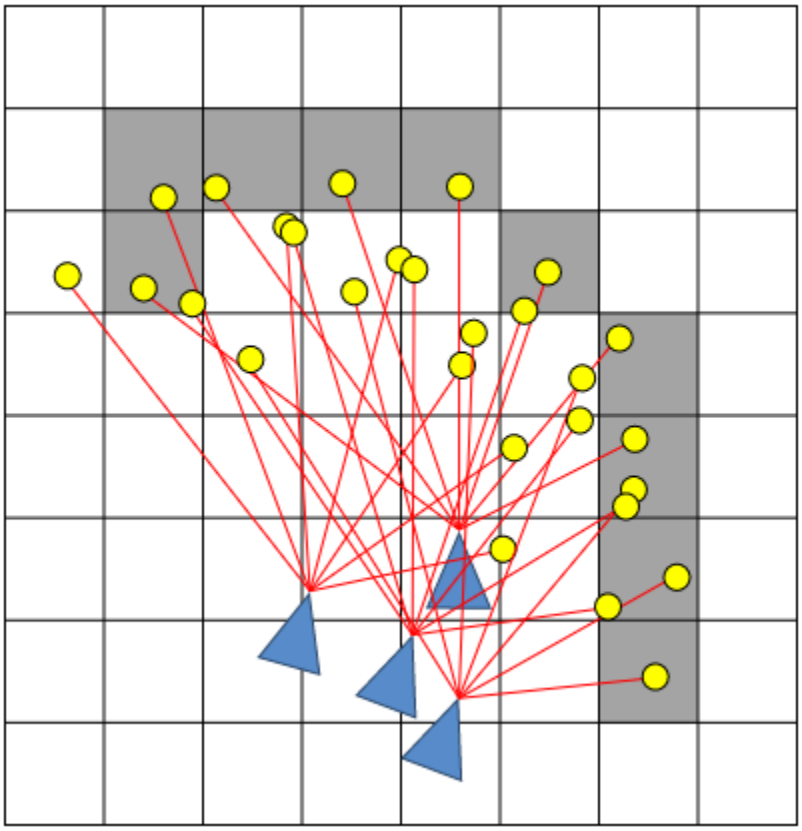


Fig: Matching particle swarms to grid maps.

Image source- https://www.hindawi.com/journals/complexity/2018/2327637/

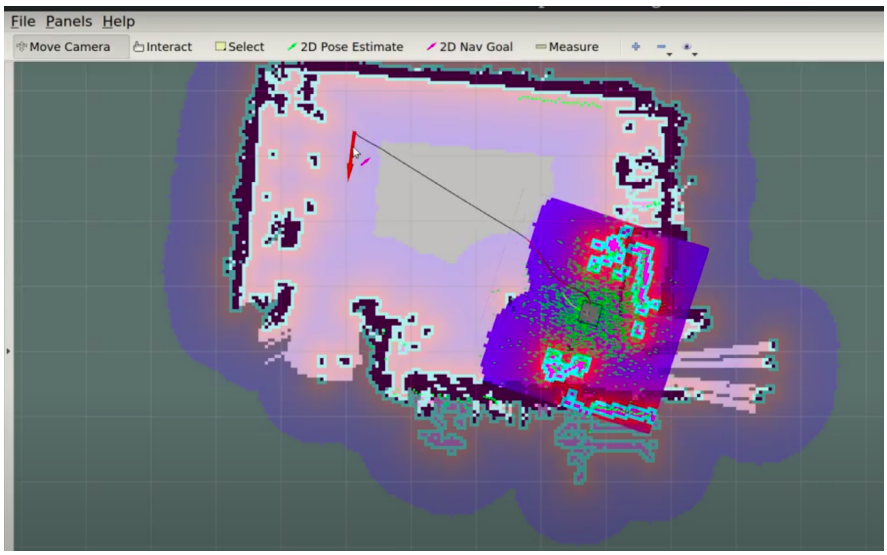



Fig: Path planned on RViz by the AMCL node

When a laser sensor is used to locate a robot on a 2D grid map, if the robot pose is given, it is very easy to calculate the agreement between the laser beams and the occupied grid. Therefore, the MCL algorithm can be used, which represents the pose of the robot with many particles [9], as shown in Figure .

## Fire Detection Algorithm

The fire detection algorithm is implemented in Python using OpenCV and deep learning. The author of [7] created a customised Inception V3 model for fi. They trained the model with a total 2000 images, 1800 were included in the training dataset and 200 images as cross validation set.

The InceptionV3 model is then imported from keras API and layers are added on the top of the model, global spatial average pooling layer followed by 2 dense layers and 2 dropout layers to ensure that our model does not overfit,and softmax activated dense layer for 2 classes.Top layers are trained for 20 epochs, we get a training accuracy of 98.04% and a validation accuracy of 96.43%. The training and validation losses are 0.063 and 0.118 respectively.

We used this pretrained model using OpenCV, python and tensorflow to detect if a frame has fire in the real time scenario. The information about a potential fire is then sent to the fire extinguishing node which takes the corresponding actions

Fig: fire detection using Inception V3 model

## Victim Detection Algorithm

The victim detection algorithm helps the robot determine if there are victims trapped in the fire affected area. The algorithm processes the image captured by the live feed of the Raspberry Pi camera installed on the robot and checks if victims are trapped in the area of interest. It then counts the number of people trapped and sends the information to the controller.

For this detection, we use the MobileNets SSD model to detect humans in an image. The model is imported from [8]. This model is used by the authors for detecting 20 classes of objects including humans. We fine tune this model to detect only humans in the images. It was originally developed for infrared images. We implement it on RGB visible images and get a mean average precision mAP of 87.08% on a cross validation dataset of 200 images. The model is also able to detect the total number of people in a particular image. A sample is shown below.

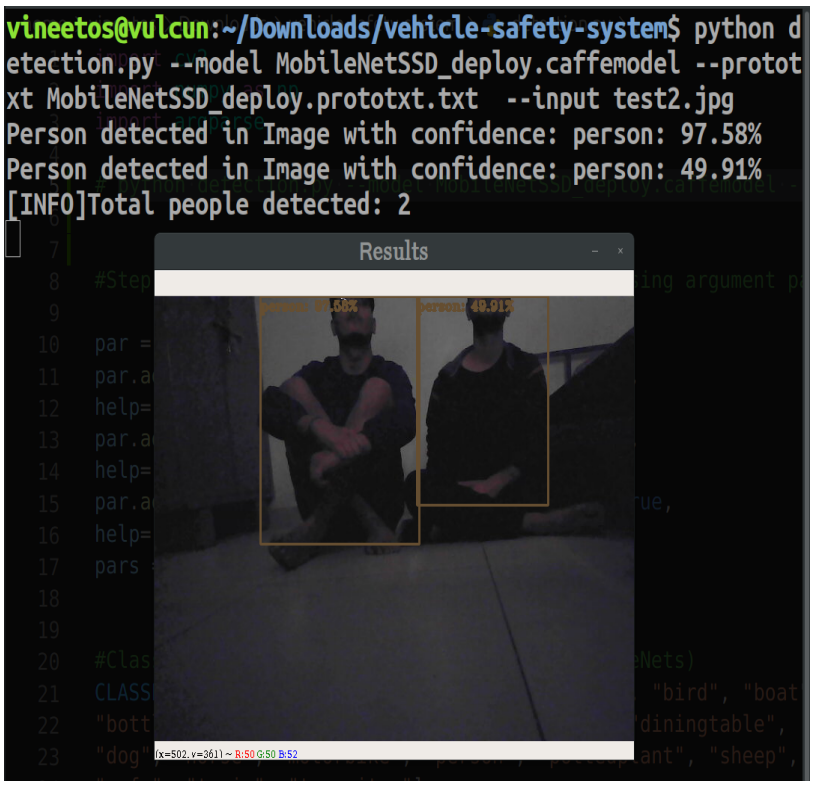



Fig: Victim detection with corresponding confidences

An infrared image of the same region is taken and passed through the model for infrared object detection. The results of the two models are then compared and a weighted sum of the confidence of the two models is calculated to determine the final accuracy of the detection. The formula of the final confidence is :

$$c_f = w_1 * c_1 + w_2 * c_2$$

cf = final confidence

w1 = weight of visible image model (default = 0.5)

w2 = weight of infrared image model (default = 0.5)

c1 = confidence of the visible image model

c2 = confidence of the infrared image model

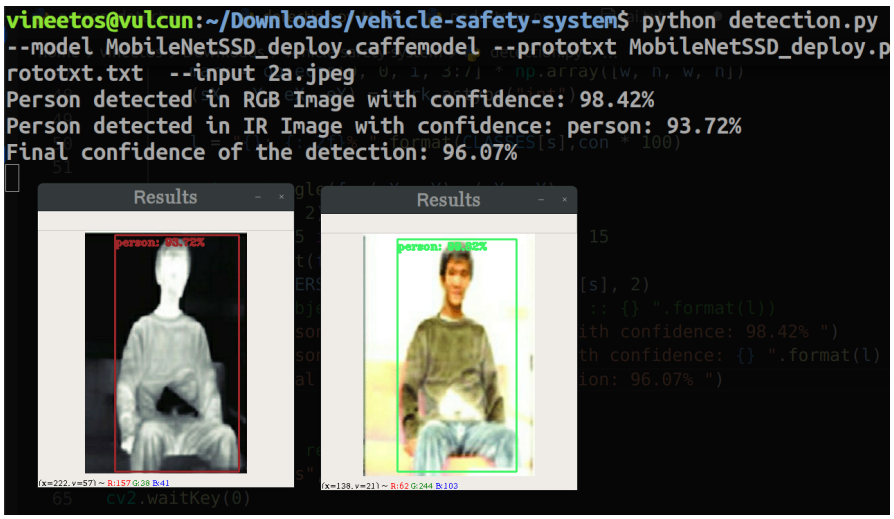



Fig: Victim detection in IR and RGB image.

The weights are by default kept at 0.5 each i.e. both models have equal priority. This is for cases where the visibility in the area is still good. In conditions when the visibility is reduced, the weight of the infrared image increases since it will be hard for the model to detect victims in low lit conditions using the visible image.The model will return that a victim has been if the final confidence is more than 75%.

## 4.4 Implementation on Turtlebot3 Waffle Pi

The following steps are taken to implement the fire extinguishing robot using turtlebot3 and ROS:

1. Start the turtlebot3 waffler pi and connect it with the master machine.
2. Run the turtlebot3 bringup launch file on the robot.
3. Run the turtlebot3 slam-gmapping launch files on the remote PC to start the RViz environment.
4. Generate a map of the area of interest using turtlebot3 slam-gmapping pre configured package by manually moving the robot using turtlebot3 teleop key package.
5. Load the generated map and run the turtlebot3 amcl nodes. In the RViz environment, set the initial pose of the robot as close as possible to the actual location of the real environment.
6. Run the camera node to collect the image data from the Raspberry Pi camera mounted on the robot.
7. Run the fire extinguishing node.

8. On the RViz map, point out the location where the fire is suspected

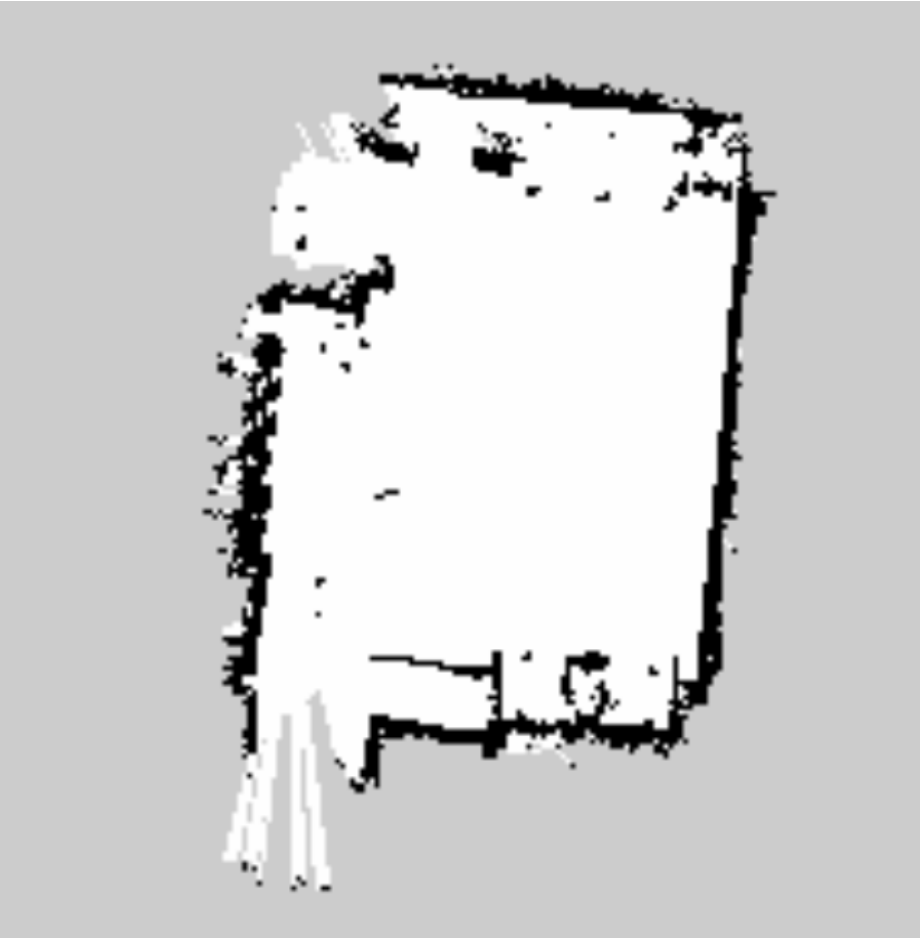


Fig: Map of room generated using SLAM-Gmapping

As soon as the above steps are completed, the robot will start the AMCL and generate a path keeping in mind the static as well as dynamic obstacles. It then starts moving towards the goal and keeps updating its path according to the dynamic changes in the environment. The fire extinguisher node collects the images captured from the camera and then sends them to the victim detection model and fire detection model simultaneously.

If a victim is detected, then the node sends a message to the controller and also updates the victim counter. In case a fire is detected, the robot is stopped. Then the manipulator moves to simulate an action of extinguishing the fire. Then after these tasks are completed the robot moves forward on its path.

## 5. Conclusion

Industrial fires are one of the most dangerous disasters and lead to heavy loss of lives and property if not taken care of properly and in time. In this paper, we propose a fire extinguishing robot model which can be installed in industries itself. The robot can autonomously detect and extinguish fires and help in solving this problem to some extent. The conceptual robot structure and working is simulated using a KUKA KR210 arm (scaled to 75%) mounted on a husky mobile robot in PyBullet Physics simulator. The simulation uses red color blocks as a symbol of fire and the robot is programmed to detect red color and move the arm towards it.

The ideas are then implemented physically on a Turtlebot3 Waffle Pi robot with a OpenCR manipulator and a Raspberry Pi camera mounted on it. The robot uses SLAM gmapping to map the environment. This map is then used for navigation using AMCL and the victim detection algorithms send messages about the number of victims detected in the room. The fire detection algorithm sends a command to stop the robot as soon as fire is detected. Then a command is sent to start the OpenCR manipulator and extinguish the fire.

The ideas and algorithms presented in this paper can be implemented on more sophisticated robot models to develop a prototype that can effectively perform the tasks mentioned and work as a completely autonomous fire extinguishing robot.

# 6. References


1. Alkış S., Aksoy E., Akpınar K. Risk Assessment of Industrial Fires for Surrounding Vulnerable Facilities Using a Multi-Criteria Decision Support Approach and GIS. Fire 2021,4,53.
2. Pagliarini, Luigi & Lund, Henrik. (2017). The future of Robotics Technology. Journal of Robotics, Networking and Artificial Life. 3. 270. 10.2991/jrnal.2017.3.4.12.
3. M S, Nagesh & T V, Deepika & Michahial, Stafford & M, Dr. (2016). Fire Extinguishing Robot. IJARCCE. 5. 200-202. 10.17148/IJARCCE.2016.51244.
4. Saravanan, P Design and Development of Integrated Semi - Autonomous Fire Fighting Mobile Robot International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 4, Issue 2, March 2015.
5. Stanford Artificial Intelligence Laboratory et al. Robotic Operating System [Internet]. 2018. Available from: https://www.ros.org
6. Gang Peng, Wei Zheng, Zezao Lu, Jinhu Liao, Lu Hu, Gongyue Zhang, Dingxin He, "An Improved AMCL Algorithm Based on Laser Scanning Match in a Complex and Unstructured Environment", *Complexity*, vol. 2018, Article ID 2327637, 11 pages, 2018. https://doi.org/10.1155/2018/2327637
7. Dhruvil Shah, Early Fire detection system using deep learning and OpenCV
8. Anukul Kumar, Vinit Oswal, Object detection in IR images using Python-opencv and deep learning, B.Tech Project 2020
9. S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT press, 2005.
10. F. Zhou, W. Jiang, S. Q. Li, Y. H. Zhang, X. Zeng, and Y. Wu, “Moving target localization and tracking algorithms: a particle filter based method,” *Journal of Software*, vol. 24, no. 8, pp. 2196–2213, 2013.
11. Y. Wang, B. Ba, W. J. Cui, and Z. Y. Lu, “Indoor positioning algorithm based on Markov Monte Carlo,” *Journal of Xidian University*, vol. 43, no. 2, pp. 145–149, 2016.
12. Russell, Stuart J. (2018). Artificial intelligence in  a modern approach. Norvig, Peter (4th ed.). Boston: Pearson