**Student Name: Cashin Green Martin Clement Joe Student ID: 200653343**

**Project Supervisor: Dr. Francis Franklin**

**NEWCASTLE UNIVERSITY**

**School of Engineering**

**Electrical and Electronic Engineering**

**Student Declaration**

I, Cashin Green Martin Clement Joe, confirm that this report and the work presented in it are my own achievement.

I have read and understand the penalties associated with plagiarism.

Signed: CASHIN GREEN MARTIN CLEMENT JOE Date: 06-09-2021

Camera based Indoor positioning system using Raspberry Pi

**MODULE CODE : EEE8097**

Author : Cashin Green Martin Clement Joe

Project Supervisor : Dr. Francis Franklin

# ABSTRACT

The aim of the project is to determine the coordinates of an object, or a position in a closed environment using cameras. An object is tracked using the technology called computer vision and its bounding box dimensions is obtained in pixels. When the bounding box size is compared to the actual object size, the distance between the item and the camera may be determined. If three cameras from three sides picks the image of an object, the distance from the three cameras to the object is calculated. By using the concept of trilateration the coordinates of the object are determined. This technique of finding the position of an object can also be considered as an indoor positioning system using a camera.

# INTRODUCTION

Indoor positioning particularly in the manufacturing domain plays an important role, maybe it may not have a significant value in the industry revolution 3.0 or older. But it will be a huge advantage for the current industrial paradigm that is industry 4.0. Tracking of an object in a workspace is done using many methods. Currently, the tracking of an object indoors is done using a chain of wireless technologies like WI-FI, Bluetooth or using plain radio waves that tracks objects when they are in a certain anchor range[1]. For example, the recent AirTag technology [18] is also a positional finder that can track objects by connecting them to the nearest iPhone (Anchor) and sends a signal to the owner's device with its coordinates.

But the most supreme technology that stands at the apex today is computer vision and object detection. By using this technology an object is tracked in a closed environment without a transmitter and a receiver. Three cameras are used in total for this project. This could be a new technology of its class because cameras have the ability to measure the distance of an object. Though classic, the concept of Trilateration remains undisputed and it helps in solving the coordinates of the intersection of three circles and the coordinates can be plotted in a graph for further use.

These technologies can be bind to design an indoor positioning system and can finally produce a prototype out of commercial embedded components available in the market. In this project raspberry pi is sufficient enough to do the described innovation. The system is clearly explained in this paper. The algorithms and design are explained, as well as the performed experiments. The results are discussed and further development is proposed.

# HARDWARE REQUIREMENTS

The whole project relies on the computation power and the level of accuracy used to detect the object. Though the project is done in Raspberry Pi 4 Model B with 2GB of RAM, Machine learning is wholly related to the computational power of the hardware. Especially Raspberry Pi doesn’t support any GPU to run Machine learning effectively to rely on it Solely. An external computational power with a decent GPU is required to handle the project effectively. Raspberry Pi Camera V2 is an additional module that is commercially available which is portable to the Raspberry Pi is used for detecting an object in the project. The communication protocol is the Ethernet cable used to connect the Pi to the computer.

# SOFTWARE REQUIREMENTS

Since this study requires high computational power to handle neural network operation which is utilized in this research various software are used. The programming language used in this project is python version 3.6.1 [15] and to handle neural networks TensorFlow version 2.5.1 library [20] is imported. The key challenge for this object detection is the collection of data to create the neural network architecture. The data set is created manually in real-time by using OpenCv contrib version 4.5.3 library [6] and the metadata of the image is extracted using an open-source module called labelimg [7].

In terms of connecting Raspberry Pi to the computer, Putty is used as a terminal emulator for Raspberry Pi to connect with the Windows platform. The server is created between the raspberry pi and the PC using SSH, a secure network connection that helps in accessing the Raspberry Pi remotely using another device. Raspberry Pi runs on Raspbian OS which is a open-source software available on its website [19]. The software is etched in a memory card using Balena etcher software [17] and the memory card is removed and connected in its slot available in Raspberry Pi.

# THEORETICAL BACKGROUND

The indoor positioning system was performed in various technologies, and their aim was not to provide the accurate location of anybody. Those efforts described the position of the object in a certain range and furthermore, those are designed to help people to locate in a massive closed environment like airports, Shopping mall etc…

Though myriad technologies were used to perform the indoor positioning like Wi-Fi, Ultra-wideband system, Infrared systems and radio wave system, the only necessity for the technology change is to find the distance of the object precisely in a shorter time. The base concept in finding the position for most of the positioning system are the same. Even the concept used in the GPS system has the same principle currently, that is the trilateration and triangulation method. It cannot be superseded using the current paradigm, But the technology used can be replaced to get better results in IPS(Internal positioning system) [2].

When it comes to the positioning system the time of signal transfer and receiving is very accurately noted to find the distance. Most signal transfer happens through radio waves at a speed of 3x106 metres per second, then the era of signal transmission using light was began [3] with the data transfer speed of 3x108 meters per second.

In all the above methods, measuring of the distance is erratic and further fine-tuning is required to make it more accurate [1]. Theoretically, trilateration is the point at which three spheres meet, But practically this is impossible and inaccurate. The classical method derives two points in special cases which requires a further fourth circle to calculate the valid point which may be unnecessary. These hurdles are boycotted by using a simple computer vision technology in a positional system and enables implementing it in a Raspberry Pi. This paper finds the approach for that.

## *Computer Vision*

The most persuasive tech to spot an object is by using computer vision, artificial intelligence and neural networks or collectively this could narrowly lead us to a domain called CNN. It has the ability to detect an object in an environment using the data that it has pre-trained or by comparing other images/ frames and predicts the output result as an object. The output of the object detection can be in boxes/ values. The boxes are the bounding box of the acquired output and can be plotted in the frame to detect the object [4].

A typical structure of a CNN will look like as shown in Fig. 1.

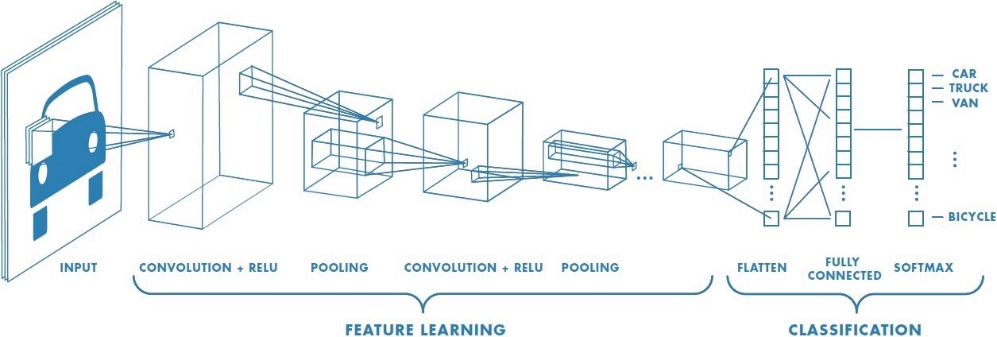


Fig. 1. Structure of a neural network

The convolutional network, often known as CNN or ConvNet, is one type of deep neural network in deep learning. Multi-layer perceptrons (MLPs) are another name for feed-forward neural networks, which are the most common deep learning models. The models are referred to as "feed-forward" because data flows freely through them. There are no feedback connections, therefore the model's outputs do not feed back into itself. The biological visual cortex is the inspiration for CNN. Hubel and Wiesel came up with the concept of CNN in 1962 [16]. They discovered that all of these neurons were well-ordered in a columnar pattern and that when they worked together, they could produce visual perception. This concept of specialised components inside a system doing specific functions is also used by machines, and it can be found in CNNs. These neural networks have shown to be effective in a variety of real-world case studies and applications including Image classification, computer vision, object detection, segmentation, face recognition and many more.

CNN requires a lot of memory and computing power, thus it's best to execute it on GPUs or in the cloud. The model may not be compatible with CPUs. Furthermore, to ensure that the algorithm works effectively, the batch size must be lowered.

## *Object Detection*

The crucial computer vision technique used in this project is Object detection. It is a branch of computer science that deals with detecting occurrences of objects in photos and videos. It is related to computer vision, image processing, and deep learning. Typical models like face detection and vehicle classifications are predefined in a library called Harr classification, which cannot be utilised for detecting objects for a specific purpose in a standard object detection library like OpenCV. So conventional machine learning of objects to detect has to be studied to use it for a specific purpose.

## *Trilateration and Triangulation*

Trilateration is a mathematical approach for calculating the position of a point in space using distances between that point and a sequence of known geometrical structures, such as a sphere or a circle.

It is a point at which three circles intersect in a three-dimension space by knowing the centre position and radii of those circles. The location of the point is narrowed to be less than two. This could be explained by using an equation of the circle which is centred at (*xa, ya, za*). Equation (1) could be taken as a reference

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Applying the analytical approach for the general case would require the equations for the three spheres to be formulated and solved for the three unknowns (*x,y,z*), i.e., the location of the intersection point; however, two assumptions can be made in order to simplify this problem:

* The anchor points are the centre of the circle, this is the points where the circle is fastened in the x-y plane with a suitable z-axis value if required. It would be better to consider all the circle meets at the origin of the z-axis or simply Fang’s method.
* Usually, the first point is considered to pass through the origin and the following points are allowed to set based on the required field of view.

|  |  |  |
| --- | --- | --- |
|  | A1(0, 0, 0) | (2) |
|  | A2(*x1*, 0, 0) | (3) |
|  | A2(*x2, y2*, 0) | (4) |

The circle equation is rearranged to the above *x* and *y* values mentioned in (1), (2) and (3). This is simply described below

|  |  |  |
| --- | --- | --- |
|  |  | (5) |
|  |  | (6) |
|  |  | (7) |

So the equation for finding the coordinates with the known radii (*r1, r2, r3*) can be defined as

|  |  |  |
| --- | --- | --- |
|  |  | (8) |
|  |  | (9) |
|  |  | (10) |

## *Distance Measurement*

The distance measurement is the novel creation for this project. Finding distance by detecting the object and by extracting the pixel value from the bounding box and using the triangulation algorithm, the focal length of the camera could be easily found by knowing the value of *P* which is the width or height of the image in pixel, W is the real width/height of the object and D is the real-time distance. By substituting the value of D, W, P in the formula (11)

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

By knowing the value of the focal length *F*, width/height of the object *W* and pixel value *P*  which is obtained by constantly changing the location of the object, we can calculate the distance using the formula (11).

# DESCRIPTION

After the required modules are installed as per the versions given in the software requirements, the data has to be collected in real-time for object detection. These data are in the form of images either in *.jpg* or in *.png* format. Bear in mind before handling the images *it is advantageous to know the significance of the image type to be used*. For generating the bounding boxes, metadata has to be extracted from the images which are in the file format .*xml*, used for extracting the pixel value, file name, file size, length and breadth of the image. This is done with the tool called labelimg [7] as described in the above paragraph. These data will help us to form the YOLO (bounding box) over the required object.

The model from the TensorFlow model zoo is downloaded based on the requirements. There are different modules available for different purposes. The output to be extracted should be in the form of boxes or YOLO, So that it will be very easy for us to derive the dimensions of the YOLO. Additionally, to reduce processing time and improve accuracy, it is preferable to use the quick reaction time model. The chosen model in this project is *SSD MobileNet V2 FPN lite 340 x 340* [8]*,* which has good reaction time and accuracy. This model utilized for object detection uses Region convolution Neural network (R-CNN) architecture which has more significance when compared to the standard CNN. This technology extracts randomized blobs as patches from an image and the pixel values are machine learned by the neural network to form the output. An Image’s aspect ratio is split into six different layers and each layer is trained in the model with its standard parameters [10].

Program setup involves the reading of *.xml* files for creating a model in neural networks. The data from the xml files are to be extracted using the pipeline configuration file provided by IBM [9]. This configures the creation of the model. The model which is used in this project will transform the image to a 340 x 340 resolution. As mentioned earlier, image processing requires high computation power and it handles multi-dimension arrays in high-level programming, the GPU is activated to make the process fast and effective. Most GPUs will have a tensor core to handle machine learning. In order to activate it, based on the version of python used, the relevant CUDA[12] and cudNN [11] are installed.

The output detections will be in the form of a bounding box which highlights the object from the field. *These bounding box will have the values of its dimensions* which is used for estimating the distance. The value is extracted by the output of the program by comparing the size of the image used for detections and the output tensor obtained to create the bounding box. Normally a tensor is a multi-dimension matrix of uneven size and length, So it is mandatory to squeeze the output values of the tensor into an ideal matrix. Additionally, a tensor has its positive image in any dimension, So by squeezing it, the tensor becomes a 2D array that represents the output results. It will be more efficient to know the values of the tensor when the image is processed. As previously mentioned in an early paragraph, an image is segmented into six different layers each layer will have a bunch of positive values. So these values are filtered by using an acute range of thresh hold to avoid false predictions and precision in dimensions of the bounding layer. Henceforth, the output dimensions of the bounding box are obtained and by multiplying its value to the full image dimension, the length and breadth of the bounding box are determined. To find the distance between the object and the camera lens, the focal length of the camera is calculated using equation (11) and the distance between the lens and the object is found. Three cameras are connected in parallel to capture the object simultaneously and the distances are taken as an output list to manipulate in the raspberry pi. The model is then exported to raspberry pi to detect the object and to find the distances.

# RESULTS

In order to achieve the trilateration to determine the position of an object, this project demands a number of milestones to be achieved, in which the preliminary objective is object detection. The *SSD MobileNet V2 FPN lite 340 x 340* [8] model architecture is used to perform object detection which has shown a promising result. The dataset is collected with 35 images for training and 6 images for validation. In order to achieve better accuracy, the model is trained on 2000 epochs and the outcomes are clearly plotted as localization loss and total loss which are highly essential in this project to consider.

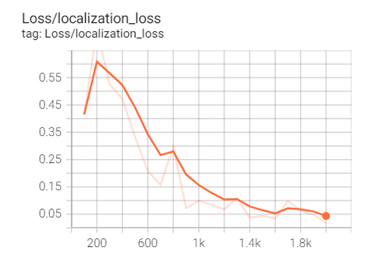
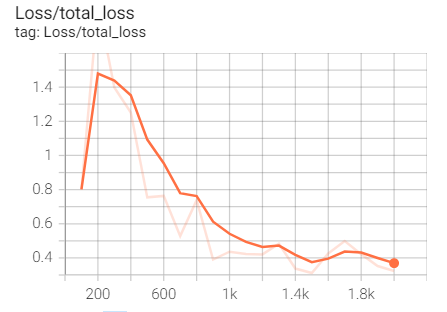


Fig 2.(a) Localization loss of the model (b) Total loss of the model

From Fig. 2(a), it clearly shows that the localization loss continuously plummets after the 200th epoch which depicts that the trained image from the datasets are noise proof. Similarly, the total loss reduces over time which reflects that the model becomes accurate for higher epochs. Below is the detections did by the camera using the R-CNN. The images in the dataset consist of a coke can which is supposed to be detected by the model. Fig. 3. shows the image with detections which is highlighted by a bounding box with prediction accuracy.



Fig. 3. the model SSD MobileNet V2 FPN lite 340 x 340 detects an object with a hundred per cent accuracy.

The coke can is placed at a distance of 193cm away from the camera lens and has a height of 11.5cm. The dimensions of the bounding box is extracted based on the image taken with the can place in that position and the focal length is determined. The height obtained in the camera in pixel is 53.9059 and by using equation (11) the focal length is 867.1824. The values obtained are just a multiplication factor used to determine the distance.

To validate the program output, readings of the cameras are taken with detections and their corresponding distance is compared with the manual distance taken using a laser distance measurer and the outputs are plotted. This reading is taken by using the dataset taken from three different distances of different locations. The number of epochs ran is 2000 and the size of the dataset is 35 images. The results are plotted to compare the distance measure manually to the program output as shown in Fig. 4(a).

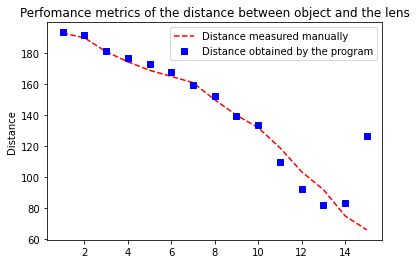
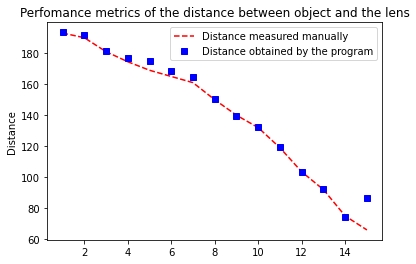


Fig 4. Comparison of Distance values measured in real-world and the values obtained from the model.

(a). Dataset images taken from three different distances (b) Dataset images taken from seven different distances.

It is seen that many deviations occur where the readings are taken much closer to the object. This is where the camera started giving false recognitions with the object and the deviations are maximum. the standard deviations of the value obtained are 38.4682 which is lower than the value of measured value 41.4222. From this scenario, it is easily understood that the values obtained from the program is much precise at the distances where the dataset images are taken and the distraction starts in the intermediate distances between the points at which data collection is done. This is further analyzed to tune the model to give precise results. The same amount of data is machine learned further with seven dataset image capturing spots between the camera and the object. And the results were very good compared to the previous results as shown in Fig. 4(b). But False predictions were higher compared to the previous one. This could be minimized by decreasing the threshold value, which is the pixel tolerance value of the object detection.

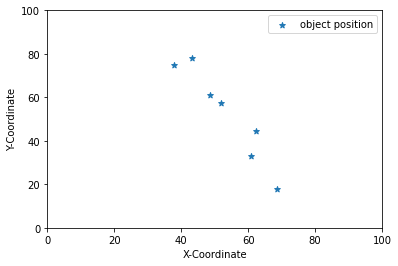


Fig. 5. Position of an object calculated using trilateration by the object detection

Provided, this accuracy in the R-CNN results, the trilateration is performed using the three distances obtained from each camera and the outputs are plotted in a graph. This project sets the anchor points of the camera at the coordinates of (0, 0, 10cm), (0, 100cm, 10cm) and (100cm, 100cm, 0). It is good to choose a low-resolution camera for faster image rendering, So, *Trust model XLIS*  camera was used which has a decent resolution of 640x480. The object is placed in random places to get the coordinates of it and the results are plotted in a graph shown in Fig.5. From Fig.5 it is seen that the measurements were taken only at the centre of the x, y plane and this is because of the field of view of the camera. It is very essential to choose the right camera for the best field of view.

Trilateration using the three cameras fixed in different anchor points mentioned above is done and the outcomes are tabulated in below table I

TABLE I

Values of the Coordinates X and Y were calculated using trilateration, by the distances measured using the three cameras

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CAMERA 1 | CAMERA 2 | CAMERA 3 | X | Y |
| 82.10 | 79.90 | 67.33 | 51.78 | 57.46 |
| 91.11 | 67.62 | 85 | 68.64 | 18 |
| 94.79 | 80.71 | 71.46 | 62.35 | 44.67 |
| 77.39 | 67.93 | 79.31 | 48.49 | 61.17 |
| 66.31 | 82.78 | 66 | 37.72 | 74.75 |
| 93.22 | 80.83 | 88.13 | 60.78 | 33.04 |
| 79.91 | 88 | 59.31 | 43.21 | 77.91 |

\*All values provided in the table are in centimetre

The same results are expected doing it in the Raspberry Pi but were unable to find the results of the object accurately as the rendering of the image took so long to give results and the object detection was not accurate enough to perform trilateration in it. But the object was detected when the video rendering has little noise in it. Due to this problem in the object detection performed in the raspberry pi, false predictions were noted even the number of predictions to predict is set to be one per frame. The same model was deployed in raspberry pi and its performance was analyzed. The Raspberry Pi does not perform optimally for the model provided. When tested, each frame takes 500 milliseconds to render. Unfortunately, it was unable to perform the distance estimation properly. The frame rate drops very low and it takes a longer time to process object detection. Many false predictions are identified though the model works perfectly in the computer system.

# DISCUSSION

The results are fairly accurate based on the calculation and the algorithm used in this work. Since these results were calculated and compared with real-time measurements taken manually, the accuracy may be affected due to *measurement location errors* and *parallax error.* The possible errors in output measurement sensed by the camera occurs in the manual labelling of the image done while creating the data sets. Since the height of the bounding box is used to measure the distance of the object, the root means square error value is taken in the heights of the bounding box to minimise the possible errors created manually while labelling [7]. The deviations are plotted as shown in Fig. 6. The errors are minimised and the tuning is done. In Fig.7 it is seen that the object is highlighted manually and these highlighted box dimensions are taken as the bounding box values. This masking is done more closely and tactfully to avoid false predictions and to get the dimensions of the detected object more accurate. Additionally, the object should be selected with high colour contrast compared to the environment, this allows the neural network to detect objects by identifying the perfect edges of the images within the bounding box.

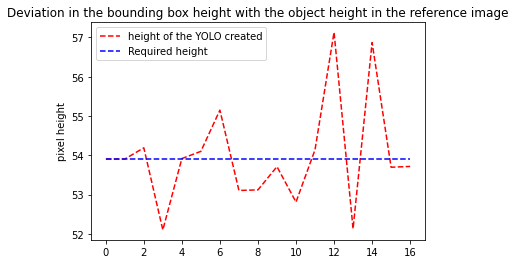


Fig. 6. Deviation in the height of the object measured in training images vs standard image

It is crucial to capture the object in an environment that differentiates the positive image from the negative images or else the colour of the object is picked from the environment and tested to segregate a particular colour from the environment.

The model used in this Object detection uses R-CNN deep learning architecture, as already mentioned this model takes random blobs in an image for machine learning. When time and accuracy comes to the picture, the model has 22 milliseconds as the processing time for each frame to detect, which is fairly low when it provides objection for a single camera. In this project, Trilateration is performed using three cameras which consolidates to 66 milliseconds for a single frame in each camera. If the faster models are selected from the TensorFlow 2 model zoo [8] the accuracy decreases, which is not recommended.



Fig. 7. region to be detected is masked for machine learning.

The accuracy of object detection is increased by creating a sampling dataset of images with different distances and angles of the object to perform machine learning. Especially, in this project two datasets are created, one with images collected at three different distances and the other one with seven. The datasets with different distances gives good results compared to the lower ones. So, the more diverse the datasets are, the more predictions the model does to identify the object. So, the distance measurement is based on the accuracy of the dataset created.

# CONCLUSION

The paper presents a novel indoor positioning technique using object detection. Several works on the programming part and the design was contributed to determine the distance between the camera lens and the object and to perform the trilateration algorithm. Their results were proven to be reasonably good.

Since the idea of the project is proposed to deploy in an industrial environment using Raspberry pi, the deployment is executed in different hardware specifications in which hardware with high computation power turns out to show significant performance improvement when compared to Raspberry Pi model 4B+. It's apparent that precise interior positioning has a long way to go, but more advancements appear to be on the road, with some very intriguing potential.

This idea can four see a good future scope in an industrial environment. By plotting the position of an object using trilateration, the values can be machine learned with some tolerances and the coordinates can be given as an input to an automated guided vehicle(AGV) to move in a certain path.

In addition, to deploy this project effectively in Raspberry Pi, the Pi is overclocked [14] above its operating voltage to give the best results for object detection or an additional module to be used as a GPU. The overclocking in raspberry pi has a lot of limitations and is not recommended. The versatile computation power in a miniaturized embedded system will provide the desired output in future for this project.

# ACKNOWLEDGEMENT

I would like to thank Dr Francis Franklin for providing me with enough knowledge for programming in python and for guiding me throughout the project. I would also like to thank all the lecturers from the MSc Mechanical Engineering department and the university for providing me with knowledge for completing the project.

REFERENCES

1. [Pablo Cotera](https://journals.sagepub.com/doi/full/10.5772/63246), [Miguel Velazquez](https://journals.sagepub.com/doi/full/10.5772/63246), [David Cruz](https://journals.sagepub.com/doi/full/10.5772/63246), [Luis Medina](https://journals.sagepub.com/doi/full/10.5772/63246), [Manuel Bandala](https://journals.sagepub.com/doi/full/10.5772/63246)[\*](javascript:popRef('corresp1-63246')), “Indoor Robot Positioning Using an Enhanced Trilateration Algorithm”, *International system for advanced robotic system*, CIDESI, Querétaro, Querétaro, Mexico: sage journals, Jan. 2016**.**
2. Curran, Kevin; Furey, Eoghan; Lunney, Tom; Santos, Jose; Woods, Derek; McCaughey, Aiden (2011). ["AnEvaluation of Indoor Location Determination Technologies*"*](https://semanticscholar.org/paper/c214477f7e82ff398c3badbd0cf716b8e2240c0b),  *Journal of Location Based Services,* June 2011.
3. Ray, Brian. “How an indoor positioning system works”, Aug 16, 2018. [Online]. Available: <https://www.airfinder.com/blog/indoor-positioning-system>
4. Sezilska, Richard, “Computer Vision: Algorithms and Applications”, 2011 Edition, Springer publications vol 1, 2011, pp 2-21
5. Sharma, Adithaya, “Convolution Neural network with python and keras”, Dec 5, 2017, [Online]. Available: <https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python>
6. https://pypi.org/project/opencv-contrib-python
7. tzutalin, Darrenl, “Labelimg”, 2017, Github, [Online], Available: <https://github.com/tzutalin/labelImg>
8. Tensorflow, “Tensorflow 2 model zoo” , 2015, [Online], Available: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md>
9. IBM, “Pipeline configration files ”, IBM technologies, [Online], Available : <https://www.ibm.com/docs/en/cics-ts/5.2?topic=infrastructure-pipeline-configuration-files>
10. Tensorflow, “Tensorflow 2 object detection using R-CNN”, Github, [Online], Available, <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md>
11. cudNN, “NVIDIA cudNN”, Nvidia Technologies, [Online], Available : <https://developer.nvidia.com/cudnn>
12. CUDA, “NVIDIA CUDA”, Nvidia technologies, [Online], Available: https://developer.nvidia.com/cuda-downloads
13. Tensorflow, “Tensorflow build from source windows”, Tensorflow, Nvidia Technologies, [Online], Available: <https://www.tensorflow.org/install/source_windows>
14. Lucy hattersly, “How to overclock Raspberry Pi 4”, [Online], Available: <https://magpi.raspberrypi.org/articles/how-to-overclock-raspberry-pi-4>
15. Python programming language, “Python 3.6.1”,Python Software Foundation, [Online], Available: <https://www.python.org/downloads/release/python-361/>
16. Felhaber, Kate, “Hubel and Wiesel & the Neural Basis of Visual Perception”, Knowing Neurons, [Online], Available: <https://knowingneurons.com/2014/10/29/hubel-and-wiesel-the-neural-basis-of-visual-perception/?source=post_page>
17. Balena etcher, “Burning operating system in SD card”, Balena Inc., [Online], Available: <https://balena-etcher.eu/>
18. AirTag, “How AirTag works?”, Apple Explained, [Online], Available: <https://www.youtube.com/watch?v=lXxWh2tkKgk>
19. RaspbianOS, “Download Raspbian OS for Raspberry Pi”, Raspberry Pi Foundation, [Online], Available: <https://www.raspberrypi.org/software/>
20. TensorFlow 2, “Download and install TensorFlow 2”, Google Inc., [Online], Available: https://www.tensorflow.org/install/

|  |  |
| --- | --- |
|  |  |