ACTUAL-PERIOD FACE MASK AND TEMPARATURE RECOGNITION USING DEEP LEARNING ON RASPBERRY PI

T RAVI CHANDRA, SRIKANTH MYDAPALLY, MUNAVATH VASANTHA RAO
4.N.MOUNIKA, 5.T.SANDEEP
Assistant Professor ,ECE,
Chandra.torthi@gmail.com

ABSTRACT

The corona virus COVID-19 pandemic is causing a global health crisis. World Health Organization (WHO) has stated that there are two ways in which the spread of COVID 19 virus takes place that are respiratory droplets and physical contact. So, an effective strategies to restrain COVID-19 pandemic need high attention. Regardless of discourse on medical resources and diversities in masks, all countries are mandating coverings over the nose and mouth in public. To contribute towards communal health, this paper aims to devise a highly accurate and real-time technique that can efficiently detect non-mask faces in public and thus, enforcing to wear mask. A hybrid model using deep and classical machine learning for face mask detection will be presented. A face mask detection dataset consists of with mask and without mask images, we are going to use OpenCV to do real-time face detection from a live stream via Pi Camera Module. We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, OpenCV, and Tensor Flow and Keras. Our goal is to identify whether the person on image/video stream is wearing a face mask or not with the help of computer vision and deep learning.The creation of dataset, Training of model and Detection of face mask is carried out on Raspberry pi 4 to make it cost efficient.

KEYWORDS:

Dataset, Training set, Testing set, Face mask, COVID-19, Camera module.

INTRODUCTION

On the brink of new year 2020, the world witnessed a new and deeply concerning situation that only a few have anticipated to have such serious impact on a world-wide scale. What started out as a local outbreak in Wuhan, China, quickly became an uncontrolled, world-wide pandemic, main-ly due to slow precautionary measures taken by local governments and large underestimation of the emerging threat in the form of the COVID-19 disease. Since then, however, preventive measures have been implemented, in order to at least slow down the already devastating impacts – both social and economic, across the whole world [1]. To this end, devices either for treatment (e.g. ventilators) or for autom-ated, proactive detection and monitoring of possible virus carriers – have become essential tools of trade in the fight against the pandemic and further spread prevention. While reverse transcription polymerase chain reaction or antibody testing is still relatively time consuming and costly, a simple evaluation of the body temperature at various checkpoints using medical-grade infrared sensors and control of proper hygienic standards, coupled with personalized, indoor location data, could be a quick and effective way of identifying and controlling the spread of the disease within closed environments, such as workplaces or factories.

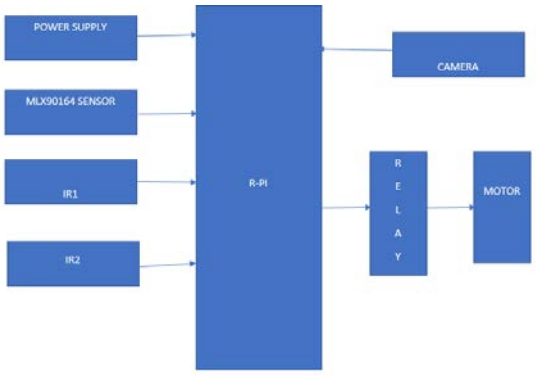
REVIEW IN "YANG, SHUO AND LUO, PING AND LOY, CHEN CHANGE AND TANG, XIAOOU, IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGN-ITION (CVPR). WIDER FACE: A FACE DETECTION BENCHMARK, 2016" Face detection is one of the most studied topics in the computer vision community. Much of the progresses have been made by the

availability of face detection benchmark datasets. We show that there is a gap between current face detection performance and the real world requirements. To facilitate future face detection research, we introduce the WIDER FACE dataset1, which is 10 times larger than existing datasets. The dataset contains rich annotations, including occlusions, poses, event categories, and face bounding boxes **IN "GE, SHIMING AND LI, JIA AND YE, QITING AND LUO, ZHAO, DETECTING MASKED FACES IN THE WILD WITH LLE-CNNS, THE IEEE CONFERENCE ON COMPUTER VISION AND PATT-ERN RECOGNITION (CVPR), 2017**" Detecting faces with occlusions is a challenging task due to two main reasons: 1) the absence of large datasets of masked faces, and 2) the absence of facial cues from the masked regions. To address these two issues, this paper first introduces a dataset, denoted as MAFA, with 30, 811 Internet images and 35, 806 masked faces. Faces in the dataset have various orientations and occlusion degrees, while at least one part of each face is occluded by mask. Based on this dataset, we further propose LLE-CNNs for masked face detection, which consist of three major modules. The Proposal module first combines two pre-trained CNNs to extract candidate facial regions from the input image and represent them with high dimensional descriptors. After that, the Embedding module is incorporated to turn such descriptors into a similarity-based descriptor by using locally linear embedding (LLE) algorithm and the dictionaries trained on a large pool of synthesized normal faces, masked faces and non-faces. In this manner, many missing facial cues can be largely recovered and the influences of noisy cues introduced by diversified masks can be greatly alleviated. Finally, the Verification module is incorporated to identify candidate facial regions and refine their positions by jointly performing the classification and regression tasks within a unified CNN. Experimental results on the MAFA dataset show that the proposed approach remarkably outper -forms 6 state-of-the-arts by at least 15.6%

PROPOSED SYSTEM:

A work in progress prototype of the proposed automated temperature and hygienic compliance testing device is shown in Fig. 1. The device is mounted on a fixed stand before the main entrance to the workplace or at various checkpoints, e.g. entrances between halls.

BLOCK DIAGRAM



Raspberry Pi 4 Model B w/4GB RAM [2] – core of the whole system, enables real-time processing of all image and sensor data.

Raspberry Pi Camera Module v2 – acquisition of image data (1640 x 1232 pixels) for detection of the face mask and on-screen face navigation;

- MLX90614 infrared thermometer in TO-39 package design, with medical-grade accuracy (±0.1 ℃) in human body temperature range [3];

- VL53L0X Time-of-Flight Distance Sensor for presence detection [4];

- 7" capacitive Raspberry Pi LCD module with 800x480px resolution;

The device prototype is connected either via LAN or WiFi connection to a local area network in order to write necessary information to the SQL database server, which stores acquired measurement data and enables correlation with positional data from the independently-operated real-time locating system (RTLS) database. The same LAN subnet is also used to operate the opening of locks once safe conditions are met by sending specially formulated HTTP post requests to specific door lock nodes with self-integrated Arduino relay devices. The independent RTLS system, used primarily for other monitoring purposes (e.g. productivity assessment or optimization of work processes), is based on commercial devices available from Sewio and utilize ultra-wideband (UWB) technology which enables very precise

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-6, 2017
DOI: 10.21276/ijcesr.2017.4.6.6

localization – up to several centimeters in optimal conditions. Information is processed in real-time using RTLS studio and positional data is stored in a MySQL database. Anchors are placed throughout the building and personal tags, each assigned to specific person, are tracked and recorded in real time – and can also be visualized and further processed. More information about the commercial RTLS system from Sewio can be found in [5]

DEEP LEARNING

Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers.

Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs.

Since deep learning has been evolved by the machine learning, which itself is a subset of artificial intelligence and as the idea behind the artificial intelligence is to mimic the human behavior, so same is "the idea of deep learning to build such algorithm that can mimic the brain".

Deep learning is implemented with the help of Neural Networks, and the idea behind the motivation of Neural Network is the biological neurons, which is nothing but a brain cell.

Deep learning is a collection of statistical techniques of machine learning for learning feature hierarchies that are actually based on artificial neural networks.

So basically, deep learning is implemented by the help of deep networks, which are nothing but neural networks with multiple hidden layers.

Tensorflow and LibrarieS Used:

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-ofthe-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, Video Surf, and Zeitera, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching street view images together, detecting intrusions in survey-llance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects atWillow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-6, 2017
DOI: 10.21276/ijcesr.2017.4.6.6

interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers

TENSORFLOW

TensorFlow is an open-source software library. Tensor Flow was originally deve-lopped by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well!

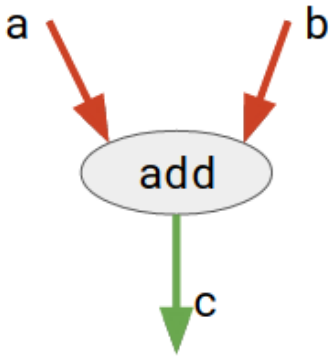
Let us first try to understand what the word TensorFlow actually mean!

TensorFlow is basically a software library for numerical computation using data flow graphs where:

• nodes in the graph represent mathematical operations.

• edges in the graph represent the multidimensional data arrays (called tensors) communicated between them. (Please note that tensor is the central unit of data in TensorFlow).

Consider the diagram given below:



Here, add is a node which represents addition operation. a and b are input tensors and c is the resultant

This flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API!

TensorFlow APIs

TensorFlow provides multiple APIs (Application Programming Interfaces). These can be classified into 2 major categories:

1. Low level API:
   a. complete programming control
   b. recommended for machine learning researchers
   c. provides fine levels of control over the models
   d. TensorFlow Core is the low level API of TensorFlow.
2. High level API:
   a. built on top of TensorFlow Core
   b. easier to learn and use than TensorFlow Core
   c. make repetitive tasks easier and more consistent between different users
   d. tf.contrib.learn is an example of a high level API.

KERAS

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neuralnetwork building blocks such as layers, objectives, active-tion functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-6, 2017
DOI: 10.21276/ijcesr.2017.4.6.6

Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:

• Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

• Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.

• Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

• Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend

MobileNetV2

MobileNetV2 builds upon the ideas from MobileNetV1, using depth wise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture: 1) Linear bottlenecks between the layers, and 2) Shortcut connections between the bottlenecks. The basic structure is shown below channels. MobileNetV2 was chosen as an algorithm to build a model that could be deployed on a mobile device. A customized fully connected layer which contains four sequential layers on top of the MobileNetV2 model was developed. The layers are

1 Average Pooling layer with 7×7 weights.

2 Linear layer with ReLu activation function.

3 Dropout Layer.

4 Linear layer with Softmax activation function with the result of 2 values

Dependencies Installation on Raspberry pi for Face Mask Detection

Setting up Raspberry pi

- OpenCV is an open source software library for processing real-time image and video with machine learning capabilities.
- Imutils is a series of convenience functions to expedite OpenCV computing on the Raspberry Pi.
- Tensorflow is an open source machine learning platform.

FOUR PHASE PROCESS

1. Data Gathering
2. Training the Model
3. Face Mask Detection
4. Practical Implementation

Data Gathering

In the first phase of the project, we are going to create a Dataset to store the faces with masks and without masks.

Training the Model

After we have collected the face samples, we can pass them on to the neural network and start the training process to automatically detect whether a person is wearing a mask or not. Start the training process importing the packages and Initialize the initial learning rate, number of epochs to train for and batch size. Load the Image sets that are saved and initializing the data and label lists. Now Loop over the imagePaths and load all the images to python script so that we can begin the training. Pre-processing steps include resizing the images to 224×224 pixels, converting them to array format, and using the preprocess input convenience function to scale the pixel intensities in the input image to the range [-1, 1]. After that perform the one-hot encoding on the labels. One hot encoding is used to represent categorical variables as binary vectors. Then split the data into training and testing sets. 80% of the data will be used for training and the remaining 20% will be used for testing. Load the MobileNet model with pre-trained ImageNet weights, leaving off the head of the network. construct the head of the model that will be placed on top of the base model.The AveragePooling2D layer calculates the average output of each feature map in the previous layer. In order to prevent over-

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-6, 2017
DOI: 10.21276/ijcesr.2017.4.6.6

feeding, we have a 50% dropout rate. After constructing the head of the model, compile the model with the Adam optimizer. After this, start training the model and once training is finished save the model

Face Mask Detection

In this phase we will use the trainer data to classify each face as with mask or without mask from the live video feed. So, open up previously programmed file in the mask detector directory. Import the required packages grab the dimensions of the frame and then construct a blob from it. Then pass the blob through the network and obtain the face detections using the **faceNet** model. Then loop over the face detections and extract the confidence associated with the detection. Filter out the detections whose confidence is less than 0.5. Compute the x, y-coordinates of the bounding box for the objects. Convert the image to RGB from BGR and then resize it to 224x224, and preprocess it. Then loop over the frames from the video and resize them. Detect the faces in the frame and determine if they are wearing a face mask or not, then loop over the predictions and compare the predictions for both the labels and display the label and bounding box rectangle on the output frame.

Testing the Face Mask Detection

To see real-time COVID-19 face mask detector in action, connect Raspberry Pi Camera module with Pi.

Launch the programmed file for detecting mask . After a few seconds, you should see your camera view pop-up window. If the mask is detected, you will see a green box with the label 'Mask Detected' and a red box with the label 'No Mask Detected' if no mask detected

Practical Implementation

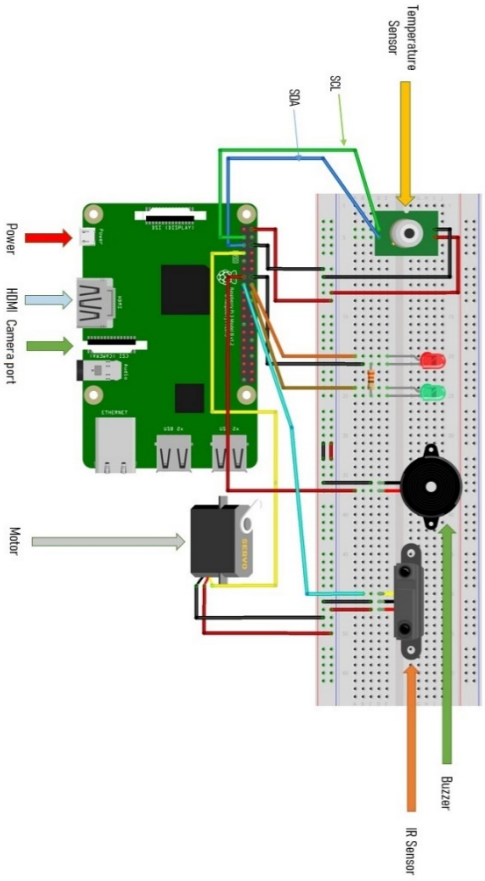
Now a practical implementation is carried out for detecting if a person is wearing face mask or not using camera module and his/her temperature is checked subsequently to check if that person is having fever or not using temperature sensor. When a person approaches near the device the IR sensor module activates making camera module to detect the face mask, when the person has worn the mask then the temperature checks the temperature if everything is good then Gates/door open. If either the person is detected of not wearing mask or if the person is having fever then buzzer goes on.

Hardware components:

1. Raspberry Pi 4 B (2GB)
2. Temperature sensor - MLX90614
3. IR sensor module
4. Servo motor
5. Buzzer
6. LEDs

Connections

Wire the LEDs and buzzer as shown in the diagram below. (Always add a resistor between the positive terminal of your LED and your GPIO pin on your Pi.).



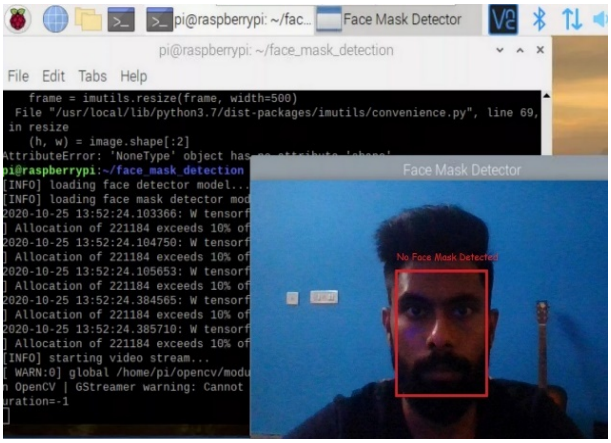
- Red LED will be controlled by GPIO14.
- Green LED will be controlled by GPIO15.
- Buzzer will be activated by GPIO 21
- Connect GND to GND on your Pi

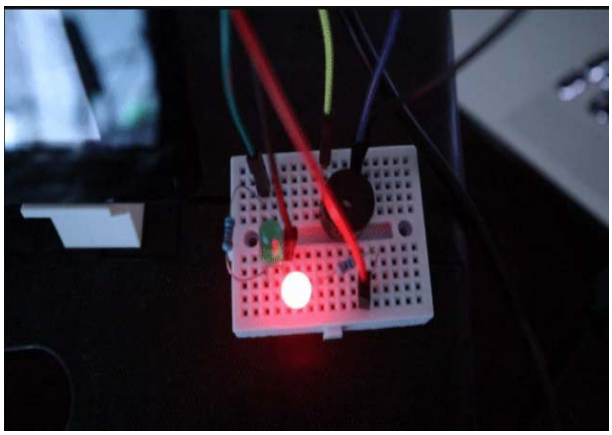
ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-6, 2017
DOI: 10.21276/ijcesr.2017.4.6.6

- Connect the serial clock and serial data of Temperature sensor to GPIO 2 serial data and GPIO 3 serial clock
- Connect 5 v and ground of temperature sensor to Raspberry pi 5v and Ground pins.
- IR sensor Signal pin is connected to GPIO 22 and Vcc,Ground pins to 5v and Ground pins of raspberry pi board
- Servo motor signl pin is connected to GPIO 27 and Vcc,Gnd to 5v and Ground pins of raspberry pi board.

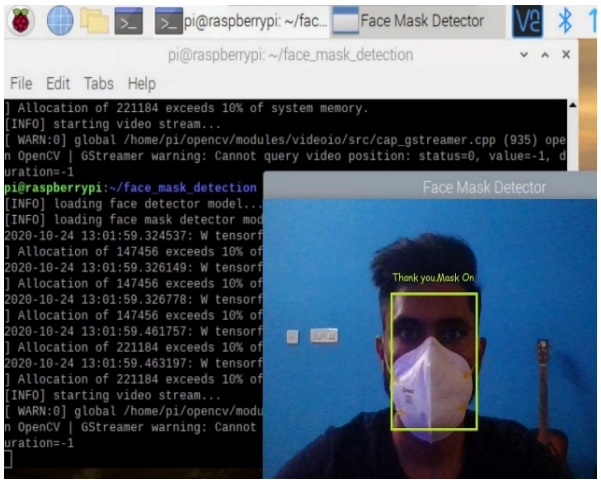
RESULTS

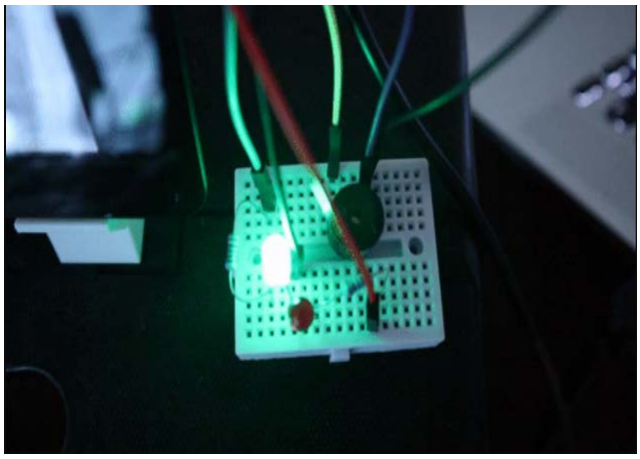
Result of the project is that, when a person without wear a mask is detected then a red box appears on the face in screen as well Red led turns on





when a person with wear a mask is detected then a green box appears on the face in screen as well Green led turns on





CONCLUSION:

Given the current dire epidemiological situation and enormous socio-economic impacts, devices that can either help identify or prevent further spread of COVID-19 disease are much sought for. Here the system implementation is such that when a person is wearing face mask or not using camera module and his/her temperature is checked subsequently to check if that person is having fever or not using temperature sensor. When a person approaches near the device the IR sensor module activates making camera module to detect the face mask, when the person has worn the mask then the temperature checks the temperature if everything is good then Gates/door open. If either the person is detected of not wearing mask or if the person is having fever then buzzer goes on. The proposed device prototype described in this paper can help in two ways – via automatic evaluation of bodily temperature at various checkpoints and by enforcing proper hygiene standards related to face masks.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-6, 2017
DOI: 10.21276/ijcesr.2017.4.6.6

REFERENCES

[1] "World Health Organization Coronavirus disease (COVID-2019) situation reports website", available online [10.4.2020] at https://www.who.int/emergencies/diseases/novel-coronavirus2019/situation-reports

[2] "Raspberry Pi 4 Model B website", available online [20.3.2020] at https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[3] "Datasheet for Melexis MLX90614", available online [20.3.2020] at https://www.melexis.com/-/media/files/documents/datasheets/mlx90614-datasheet-melexis.pdf

[4] "Datasheet for - VL53L0X", available online [20.3.2020] at https://www.st.com/resource/en/datasheet/vl53l0x.pdf

[5] "SEWIO UWB Real-Time Location System (RTLS)", available online [5.4.2020] at https://www.sewio.net/real-time-location-system-rtls-onuwb/

[6] "OpenCV packages for Python", available online [10.4.2020] at https://pypi.org/project/opencv-python/

[7] "Kivy – Cross-platform Python Framework for NUI development", available online [10.4.2020] at https://kivy.org

[8] C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic. 300 faces In-the-wild challenge: Database and results. Image and Vision Computing (IMAVIS), Special Issue on Facial Landmark Localisation "In-The-Wild". 2016.

[9] Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou, IEEE Conference on Computer Vision and Pattern Recognition (CVPR). WIDER FACE: A Face Detection Benchmark, 2016.

[10] Ge, Shiming and Li, Jia and Ye, Qiting and Luo, Zhao, Detecting Masked Faces in the Wild With LLE-CNNs, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[11] "ROCK Pi N10 website", available online [28.5.2020] at https://wiki.radxa.com/RockpiN10

[12] "Jetson Nano Developer Kit website", available online [28.5.2020] at https://developer.nvidia.com/embedded/jetson-nano-developer-kit.