International Journal of Engineering & Technology, 7(4.16) (2018) 208-213

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Autonomous Spherical Surveillance Robot with Vision-Based Human Recognition and Tracking

Miguel Richney D. De Guzman[1], Carlo Luigi G. Ocampo[1], Jonathan E. Subong[1], Aeisha Dominique P. Zamudio[1], John Anthony C. Jose and Melvin K. Cabatuan[1*]

[1]Electronics and Communications Engineering Department
De La Salle University
Manila, Philippines
**Corresponding author E-mail: melvin.cabatuan@dlsu.edu.ph*

Abstract

Managing and securing university grounds poses quite a challenge. This research mainly focused on assisting security personnel in identifying specific persons-of-interest within a given area through the construction of a spherical mobile robot. This structure was conceptualized to protect the mechanical parts and electronics used for the robot's movement. Serial communication using Bluetooth modules imposed effective communication between the electronics that were both inside and outside of the sphere. Different sensors processed different signals inside an Arduino module to achieve the robot's autonomous state. Additionally, Open Source Computer Vision (OpenCV) was used on a Raspberry Pi 3 module and machine learning on a laptop, for facial detection and recognition, respectively. Whenever faces were detected, the robot-captured images were sent to a base station via file transfer protocol (FTP) through a virtual private network (VPN) over the Internet. A selected image is then compared to a trained set of images within the system's database to identify if that specific individual is a person-of-interest. If the identity matches, then the operator will alert security personnel. All in all, the researchers successfully constructed an autonomous surveillance robot that identified specific persons-of-interest and scouted a specific area inside De La Salle University (DLSU).

Keywords: *facial detection and recognition; image transfer; line-following; obstacle detection; spherical surveillance robot*

1. Introduction

De La Salle University (DLSU) is an educational institution that caters to a substantial number of people each day. Consequently, to ensure the safety of these people, it is undeniably essential to establish a reliable security system within the institution's premises. The enormous task of handling and monitoring large places where numerous people meet is quite challenging. The group's proposal is to construct a mobile robot that is capable of surveying areas around DLSU. The group has noticed that some of the guards around school roam around the campus to check up on students. With this proposal, the amount of work for the security personnel would be reduced and they would be able to cover more ground. The technical difficulty lies in three key aspects: (1) navigation, (2) mobility, and (3) processing. The robot is expected to navigate its way in a specified path inside the university. Furthermore, the design will be spherical in nature. This means that the prime mover must come from inside of the sphere; it is expected to be capable of forward, backward, right, and left movements. Likewise, the autonomous feature of the robot is implemented such that only sensors, such as a line tracker sensor and ultrasonic sensor, relay its proximity. Lastly, the challenge in processing the facial data is encountered when the robot is stationary, wherein it will locate a person's face and the data is later cross-referenced on a specific database.

Overall, the general objective of this research is to design an autonomous spherical robot that will aid the security personnel in monitoring and surveying some of the key grounds of De La Salle University, Manila. The specific objectives are as follows:

(1) To design an autonomous point-to-point surveying and monitoring feature for a specific area wherein the robot will follow a distinct line on a flat concrete terrain inside DLSU Manila; (2) To develop an obstacle detection safety mechanism during the span of the robot's operation with a delay of not more than 2 second with an accuracy of at least 80%; (3) To transmit the image signals via wireless local area network to a base station with a latency of less than 2 seconds with an accuracy of at least 80% and; (4) To develop a face recognition algorithm that will cross-reference gathered data to a database that is limited to only the four members of the group with an accuracy of at least 80%.

2. Related Literature

One of the inspirations of the spherical design was from SPHEMO [1]. That was a spherical robot that scanned and surveyed a specific area. It was capable of multidirectional movement.

Going to the main discussion, facial recognition is the focal point of this research. As such, different studies were taken into consider-ation. Ultimately, facial recognition follows this general process: Input Source (image/video), Face Detection, Face Tracking, Feature Extraction, only then will Facial Recognition happens [2].

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

That being said, the one heavily relied upon on in this research was deep metric learning. Famous implementations of this are the Sia-mese [3] and triple loss networks. This function measures the simi-larities between images. The notion of metric learning is to perform a function/metric that quantifies the distance among sets of images.

In comparison, here are some of the commonly used facial recogni-tion schemes that were also considered in this research: Local Bina-ry Patterns (LBPH), Deep learning using OpenCV, and Convolu-tional Neural Network (CNN).

LBPH labels a group of pixels with a certain threshold amongst its neighboring pixels and classifies the result as a binary number [4] [5]. The LBPH operation is known for its robustness to various grayscale changes such as illumination. In facial recognition, this approach divides the image into several local regions, and LBPH texture descriptors are extracted independently per region. The descriptors are then concatenated into a histogram to form a global description of the image.

Deep learning using OpenCV, on the other hand, outputs a real-valued feature vector [6]. In this method, the user tweaks the weights of the neural network of the test images so that it is far from the measured weights of the picture in question.

And finally, CNN [7] which makes use of four main operations: convolution, non-linearity, pooling or sub-sampling, and classifica-tion. Convolution is where the feature extraction happens. Like LBPH, this is where the features are represented by binary data (0-255 depending on the intensity of the image region), Non-linearity or Rectified Linear Unit (ReLU) is where the all the negative pixel values with a zero. This is performed every after convolution opera-tion. Pooling or subsampling is reduction of each feature in a given matrix to its highest value. Lastly, the image classification connects the most important feature from each operation and matches it with the nearest value from a given dataset.

The application of this research was mainly for surveillance. That is why different papers on the topic of surveillance were looked upon as well. Different methods were used such as different papers on Principal Component Analysis (PCA) [8] [9], and moving object detection [10].

3. Methodology

3.1. The Internal Wheeled Mechanism

Fig. 1: Internal wheeled mechanism

For the internal wheeled mechanism, shown in Fig. 1, the body was made of wood and had a length of 25 cm and a width of 20 cm. The wheels used were also made of wood, which had a diameter of 17 cm. These dimensions were enough to fit inside the spherical shell. The motors used were DC geared motors that were capable of moving the spherical shell. Consequently, the motors had a torque of 31.7, which contributed to the successful movement of the shell. The DC motors and wheels were placed on the lower end of the body. Furthermore, the robot could move in such a way that the weight of the dc motors shifted the shell to move forward.

3.2. The Spherical Shell

Fig. 2: Spherical shell made of fiber glass material

The group decided to invest in the most reliable option they could fathom - a spherical shell made of fiberglass material. Since the spherical shell had to be hollow, a spherical mold with a diameter of 40.5 cm needed to be created. This mold allowed the fabrication of the spherical shell. The spherical shell design is very sturdy and the most durable among all the designs conceptualized and constructed by the group. It can be reverted to its original shape when deformation occurs. Additionally, it is strong enough to push the support mechanism. The only drawbacks of this design are: (1) it took time for the overall spherical shell to be constructed and (2) it is heavy. Amidst such difficulties, the group was able to resolve the issues encountered and produce a fully functioning prototype that exhibited the group's desired movement and yielded appropriate results.

3.3. The Support Mechanism

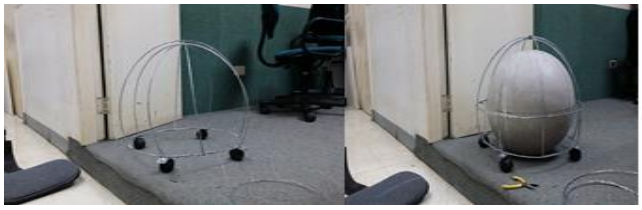

Fig. 3: Support mechanism design

The group decided to utilize thick metal wires to mold the support mechanism into a semi-spherical shaped structure. Thick wires were bent in a circular form around the spherical shell of the robot, while thin wires were used to keep the thick wires bundled together. The process involved the repetitive measuring, bending, and connecting of the metal wires together to form the support mechanism. Four rollers were attached to the bottom circular base so that the support mechanism may move around with the spherical robot. Additionally, a base for the robot's head was attached on top of the metal structure for majority of the electronic components to be nested upon. This design, Fig. 3, was stable yet lightweight at the same time, which made it an appropriate support mechanism for the robot.

3.4. Arduino to Arduino Communication

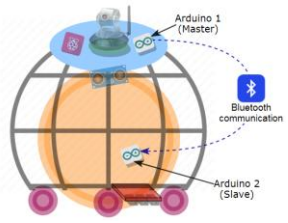

Fig. 4: Microcontroller schematic diagram

The group decided to prioritize the implementation of the Arduino to Arduino communication through a bluetooth connection between the two microcontrollers, because it had been considered as of vital importance to make the overall robot work. Each of the

Arduinos had HC-05 Bluetooth communications modules connected to them, wherein the two HC-05 Bluetooth communication modules were programmed to have a Master-Slave setup; hence, one is tasked to keep sending commands while the other is tasked to receive and implement the aforementioned commands. Arduino 1 had been placed on the base of the robot's head, while Arduino 2 had been placed on the internal wheeled mechanism within the spherical shell of the robot. The line tracker sensors and the ultrasonic sensors had been connected to Arduino 1 as shown in Fig. 4.

3.5. Line-following and Obstacle Avoidance

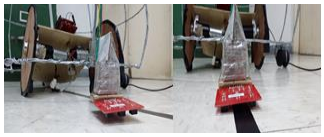

Fig. 5: Placement of line-tracking sensors

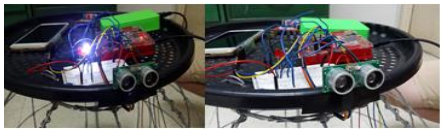

Fig. 6: Placement of ultrasonic sensors with obstacle detected (left) and no obstacle detected (right)

The three-channel line-tracker sensor component was mainly responsible for the line-following capability of the robot. It had been placed at the bottom base of the support mechanism, as shown in Fig. 5, to keep it as close as possible to the ground, which is needed for it to function properly. In testing the line-following capability of the robot, the group used electric tape to place lines on the ground. The microcontroller was programmed in such a way that the robot follows a black line. Furthermore, the group made variations on the line taped on the ground to affect the robot's movement.

An ultrasonic sensor was placed on the base of the robot's head, and it was set up in such a way that when something is obstructing the robot's intended path, the white LED would light up and the robot would stop its movement. When no obstructions are present to hinder the robot's movement, the LED is off and the robot begins to move again. Fig. 6 is a sample implementation.

3.6. Arduino to Raspberry Pi Communication

Fig. 7: Wired connection of Arduino and RPi

The wired connection between Arduino 1 and the Raspberry Pi, as shown in Fig. 7, made it possible for the camera to capture images at appropriate intervals, since the movement and the current state of the spherical robot affected the facial detection and recognition of the overall system.

3.7. Raspberry Pi Server and Image Transfer

Fig. 8: Sample test images collected by the Raspberry Pi 3

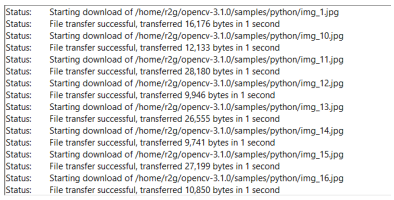

Fig. 9: Sample Filezilla download status

For facial detection, the Raspberry Pi 3 was set up and placed at the height of the robot. The webcam was positioned the same way that it will act as the "eye" of the robot. As discussed in the theoretical consideration, Haar cascades were used for this. The data gathering began when the webcam was turned on and was left to capture images of faces of the test subjects. It is important to take note that these test subjects were verbally and briefly informed about this research and agreed to be part of the data collection. The images collected from the test subjects can be seen in Fig. 8. These generated images (test images) will then be stored in the SD card on a specific folder inside the Raspberry pi 3. Furthermore, the same images will also be the ones to be transferred via ftp. They will later be used as the input images for facial recognition, which will be operated in the base station.

For image transfer, the generated images (Test images) will be sent via ftp through the internet. It is important to take note that the Raspberry Pi 3 and the base station should be connected before the ftp transfer begins. This was done via checking both the haguichi, in the Raspberry Pi 3, and the Hamachi, in the base station. Once they are connected, the ftp can begin. This will be done and will be operated by the person in the base station. He/she will download all the images in the specified folder in the Raspberry Pi 3 via Filezilla. Each image transfer will be recorded via the output presented in Filezilla. A sample image can be seen in Fig. 9.

3.8. Dataset Preparation

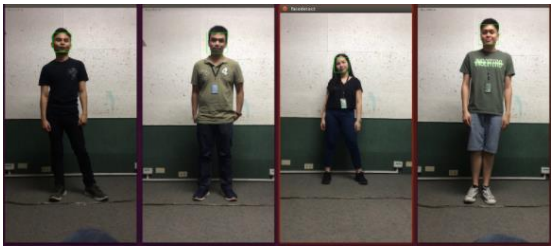

Fig. 10: Implementation of face detection and cropping

Fig. 11: Sample dataset of the 4 classes

Before the implementation of facial recognition training, a dataset per member, limited to 4, was prepared through video. Face extraction was used to crop the detected faces of each member in their corresponding videos. Fig. 10 is a sample implementation of face extraction. The motive behind a video shoot for data gathering is to analyze and save, frame by frame, all the possible facial expressions, angles, focus and lighting. Fig. 11 presents a sample dataset for each member in the group. All of the images used in the dataset were converted to grayscale for faster image processing. This is because grayscale images would only have 1 channel whereas RGB images would have 3 channels, thus in-

creasing processing time. Another reason for grayscale conversion is to detect the edges of each face with ease.

3.9. Facial Recognition

As of today, there are several ways of implementing a facial recognition software. It can be as simple as using the prebuilt facial recognition libraries provided by OpenCV to as complex as using deep learning and neural networks. In this research, 3 methods were used and compared amongst each other to reach the specified objective. These methods are (a) Convolutional Neural Networks (CNN), (b) Local Binary Pattern Histogram, and (c) Deep Metric Learning (DML)

The convolutional neural network model used for training was a sequential model. A dataset of 8000 images per class was used for the neural network training. Table 1-4 shows the sequential network models used with various changes in parameters. After one trial, a few set of images were used to test out the newly generated model. The limitation for the model used was that it only classified new images as one of the 4 members of the group. The problem became more complex when a 5th class, regarded as the unknown class had to be added. The unknown class models are from Tables 2-4.

Table 1: Sequential Model 1

Layer	Output Shape	Parameters
Convolutional 2D	150 x 150 x 32	320
Activation	150 x 150 x 32	0
Convolutional 2D	148 x 148 x 32	9,248
Activation	148 x 148 x 32	0
Max Pooling 2D	74 x 74 x 32	0
Convolutional 2D	72 x 72 x 16	4,624
Activation	72 x 72 x 16	0
Convolutional 2D	70 x 70 x 16	2,320
Activation	70 x 70 x 16	0
Max Pooling 2D	35 x 35 x 16	0
Flatten	19600	0
Dense	64	1,254,464
Activation	64	0
Dropout	64	0
Dense	4	260
Activation	4	0
Total	-	1,271,236

Table 2: Sequential Model 2

Layer	Output Shape	Parameters
Convolutional 2D	150 x 150 x 32	320
Activation	150 x 150 x 32	0
Convolutional 2D	148 x 148 x 32	9,248
Activation	148 x 148 x 32	0
Max Pooling 2D	74 x 74 x 32	0
Convolutional 2D	72 x 72 x 16	4,624
Activation	72 x 72 x 16	0
Convolutional 2D	70 x 70 x 16	2,320
Activation	70 x 70 x 16	0
Max Pooling 2D	35 x 35 x 16	0
Flatten	19600	0
Dense	64	1,254,464
Activation	64	0
Dropout	64	0
Dense	5	325
Activation	5	0
Total	-	1,271,301

Table 3: Sequential Model 3

Layer	Output Shape	Parameters
Convolutional 2D	150 x 150 x 32	320
Activation	150 x 150 x 32	0
Convolutional 2D	148 x 148 x 32	9,248
Activation	148 x 148 x 32	0
Max Pooling 2D	74 x 74 x 32	0
Dropout	74 x 74 x 32	0
Convolutional 2D	72 x 72 x 64	18,496
Activation	72 x 72 x 64	0
Convolutional 2D	70 x 70 x 64	36,298
Activation	70 x 70 x 64	0
Max Pooling 2D	35 x 35 x 64	0
Dropout	35 x 35 x 64	0
Flatten	78400	0
Dense	64	5,017,664
Activation	64	0
Dropout	64	0
Dense	5	325
Activation	5	0
Total	-	5,082,981

Table 4: Sequential Model 4

Layer	Output Shape	Parameters
Convolutional 2D	148 x 148 x 32	320
Activation	148 x 148 x 32	0
Convolutional 2D	146 x 146 x 32	9248
Activation	146 x 146 x 32	0
Max Pooling 2D	73, 73, 32	0
Flatten	170528	0
Dense	128	21,827,712
Activation	128	0
Dropout	128	0
Dense	5	645
Activation	5	0
Total	-	21,837,925

Fig. 12: Sample dataset of the unknown class

Fig. 12 presents a sample dataset of the unknown class. The unknown class is a set of images of people who willingly volunteered to become part of the research. Additionally, the group asked for verbal consent from them before proceeding.

Due to the nature and complexity of neural networks, the group opted to a simpler method for implementing a facial recognition software, which is through local binary pattern histograms. One issue encountered with the recognizer file is that it would take several minutes, approximately 20 minutes and depending on the dataset used, before it could load and recognize faces, which may seem to be inefficient. This is why the group cut down the training dataset to 200 images per class, including the unknown.

Although method (b) presents a simpler and faster way of implementing facial recognition, the reliability rate has not made the required objective. For the third and final approach, the group used deep metric learning using triplet loss. For this method, the group used 250 images per class, including the unknown.

3.10. Tracking

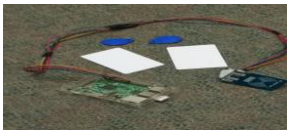

Fig. 13: RFID tags installed with the Raspberry Pi 3 module

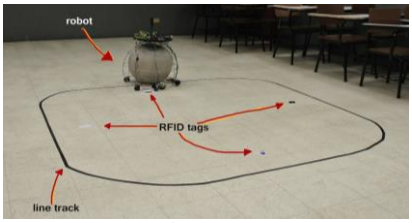

Fig. 14: Testing the movement of the sphere inside Velasco building classroom

To track the position of the robot, MFRC522 was used. It is a 13.56 MHz Radio-frequency identification (RFID) module. It has an antenna that can read and write ISO/IEC 14443 A/MIFARE cards. The MFRC522 was attached to the Raspberry Pi and a python script was used to display the UID of the RFID tags. The setup can be seen in Fig. 13. A sample testing of the robot's movement can be seen in Fig. 14 with the implementation of the RFID system.

4. Results and Discussion

4.1. Line-following

The line-following capability of the robot had been tested several times on a track constructed by the group. The robot had been programmed to follow a black line. Even though the robot had a tendency to divert its path from the line, it was still capable of finding its way back towards the intended direction. Moreover, the robot had been tested inside a Velasco building classroom. This area has also been the venue for testing the robot's speed, since the group wanted to record the average speed of the robot.

4.2. Obstacle Detection

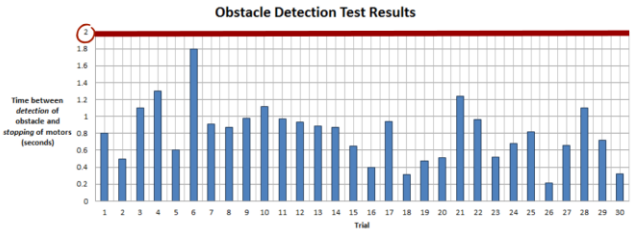

Fig. 15: Graphical representation of results obtained from 30 trials, wherein all satisfied the objective (delay was below 2 seconds)

The figure above exhibits the visual representation of the obstacle detection results gathered by the group. One hundred percent (100%) or all of the tests conducted yielded successful obstacle detection; when obstructions became present in front of the robot, it was able to halt its movement. Additionally, the time between the detection of an obstacle and the stopping of the robot's movement were all measured as less than two seconds, which satisfied one of the objectives of this research. Consequently, the average stopping time of the robot was 0.805 seconds, wherein 30 out of 30 trials yielded the desired results.

4.3. Image Transfer

Fig. 16: Graphical representation of image transfer data

For image transfer, there were 172 test images collected. As seen in Fig. 15, there is no correlation of byte sizes versus transfer time from the Raspberry Pi 3 to the base station. One reason for this is that the test images were used were quite small, as the largest is only 34560 bytes or 34.56KB. This is to be expected as the test images were all converted into grayscale. Instead, one suspected cause for some sudden spike in transfer time is the unpredictable internet traffic. Meanwhile, the average image transfer from Raspberry Pi 3 to the base station over the 172 tests was 1.069 seconds. This is with 160/172 image transfer of less than 2 seconds.

4.4. Facial Recognition

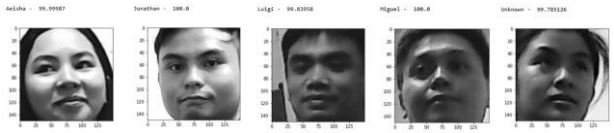

Fig. 17: Sample test results using convolutional neural networks

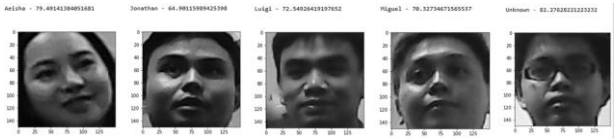

Fig. 18: Sample test results using local binary pattern histogram

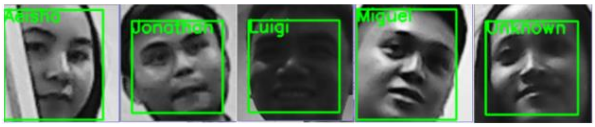

Fig. 19: Sample test results using deep metric learning

From the results shown in Table 5, it can be seen that the highest reliability rate came from method (c), which was deep metric learning with eighty-four percent (84%). Method (a), using convolutional neural networks, only reached a peak of approximately seventy percent (70%), given that this was strictly limited to only the 4 members of the group. The reliability rate greatly decreased when the unknown class was added, which dropped to almost half of its previous rate. The same is true for method (b), using local binary pattern histogram.

The major drawback encountered by the group using these methods is that even though a number of negative images have already been added, the recognizer still has the tendency to recognize unknown personalities as one of the 4 members of the group. Fig. 17-19 show sample test results using each of the three methods: convolutional neural networks (CNN), local binary pattern histogram (LBPH), and (deep metric learning (DML), respectively.

In Table V, shown below, it can be seen that deep metric learning has the highest reliability rate amongst all the trials performed. One particular reason is that this method utilizes the idea of facial landmarks whereas the methods (a) and (b) rely more on the structure of the image rather than its features. The first sequential model was shown to have a high accuracy over the other 3 sequential models due to the fact that it was only limited to strictly the 4 members of the group, that is, no unknown classification.

Table 5: Summary of methods used

Method	Number of Correct Responses	Total Number of Test Images	Reliability Rate
Sequential Model 1	52	75	69.33%
Sequential Model 2	36	100	36.0%
Sequential Model 3	38	100	38.0%
Sequential Model 4	33	100	33.0%
Local Binary Pattern Histogram (LBPH)	41	100	41.0%
Deep Metric Learning (DML)	84	100	84.0%

Although numerous facial recognition models have reached high accuracy ratings, these models are only limited to a number of selected people. Only a few models have tackled the problem of introducing the unknown class.

5. Conclusion

The researchers were able to create a fully self-made spherical surveillance robot. The test runs on line following proved that the robot is capable of following the intended track and a safety mechanism was successfully implemented. The 30 tests conducted resulted a mean stopping time of 0.805 seconds, with 30/30 successful trials.

For the facial recognition part of this research, data collection of faces yielded 172 test images and each one was transferred to the base station. This resulted a mean image transfer time of 1.069 seconds with 160/172 or approximately ninety-three percent (93.02%) of the trials successfully complied with the third specific objective. Furthermore, different facial recognition trainings were conducted. The addition of an unknown class has been proven to be difficult with the reliability rate dropping to as low as thirty-three percent (33%). Although using the prebuilt face recognizer provided by OpenCV did not reach the required reliability, it was still proven to be a bit more effective than neural networks, reaching a reliability rate of forty-one percent (41%), with the addition of its simplicity. Lastly, a method introduced in between basic computer vision and neural networks was also proven to have been the most effective among the 3 methods with a reliability rate of eighty-four percent (84%), which made it to the required objective.

Acknowledgement

We would like to extend our thanks to our colleagues, professors, the security personnel of DLSU, and the Lasallian community for all the assistance, especially for willingly and enthusiastically participating during our data gathering phase.

Furthermore, we would like to give special thanks to our parents, who have always given us unconditional love and unending support throughout all our academic endeavors.

References

[1] Abad, A., Gomez, C.B.C., Gonzales, P.J.M., Mallari, N., Opulencia, A.F.M., & Soriano, A.C. (2016). SPHEMO: A Teleoperated SPHErical Mobile Robot with Video-streaming Capability. 4th DLSU Innovation and Technology Fair, November 2016

[2] Dahake, R., Kharat, M., Lahane, P. International Journal of Advanced Trends in Computer Science and Engineering, Volume 5, No.6, December 2016

[3] Gupta, H. One shot learning with Siamese networks in PyTorch. Retrieved from https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e, 2017

[4] Pietikäinen, M. Local binary patterns. Retrieved from http://www.scholarpedia.org/article/Local_Binary_Patterns, 2010

[5] Devi, Y., Kumar, M., &Nagaraju C. (2014). Face Detection and Classification based on Local Binary Patterns. International Journal of Advanced Trends in Computer Science and Engineering, Volume 3, No. 6., 2014

[6] Rosebrock, A. Face recognition with OpenCV, Python and deep learning. Retrieved from https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/, 2018

[7] The Data Science Blog. An intuitive explanation of convolutional neural networks. Retrieved from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/, 2016

[8] CA, A., Jose, H., T, J., Wilson, A. Security Alert Using Face Recognition. International Journal of Advances in Computer Science and Technology. Volume 5, No. 12, December 2016

[9] Makwana, K., A Survey on Face Recognition Eigen face and PCA method. International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 2, February 2014

[10] Mishra, S., Bhagat, K. Human Motion Detection and Video Surveillance using MATLAB, International Journal of Scientific Engineering and Research, 2015