




Inductive Machine Learning with Image Processing for Objects Detection of a Robotic Arm with Raspberry PI

Mao Queen Garzón Quiroz^{1,2} 

¹ Universidad Católica de Santiago de Guayaquil,
Av. Carlos J. Arosemana Km 1 1/2, Guayaquil, Ecuador
mao.garzon@cu.ucsg.edu.ec

² Facultad de ingeniería industrial, Universidad de Guayaquil,
Cdda. Universitaria “Salvador Allende”, Guayaquil, Ecuador
mao.garzonq@ug.edu.ec
<http://www.ucsg.edu.ec/>

Abstract. *Goals.* The present study was designed to build a prototyping and develop algorithms that allow the detection, classification, and movement of objects of a robotic arm of 4 DOF with the following technologies: ArmUno arm structure, Raspberry Pi 3 B+, PiCam 2.1, driver PCA9685 for servomotors, Opencv3, and python. Another goal was to measure the effectiveness of prediction and classification of objects photographed by the robotic arm, using machine learning with the KNN classifier method.

Methodology. The generation of a dataset of 800 photographic images was proposed, in 4 categories: volumetric geometric shapes conformed by 200 images each one of them. With this, processing techniques were applied to the image captured by the camera to detect the object in the image: Grayscale filtering, Gaussian filtering, and threshold.

Then, the characteristics of the object were obtained through the first two invariant moments of HU, and finally, the machine learning method KNN was applied to predict, that the image captured by the robotic arm belongs or not to a certain category. In this way, the robotic arm decides to move the object or not.

Results. According to the plot of the obtained data described in the results section; the level of correct answers increases markedly by using the techniques described above. The prediction and classification using KNN were remarkable, For all the tests carried out The average effectiveness of KNN method was 95.42%. Once the scripts were integrated, the operation of the robotic arm was satisfactory.

Keywords: Opencv3 · Python · Machine learning · KNN · Robotics Raspberry Pi

1 Introduction

The following research deals with an analysis of the algorithms necessary for the optimal processing and classification of images captured by a camera integrated into a robotic arm of 4 DOF (Degree of Freedom), with the aim of being able to detect and select objects through artificial mink by computer and, also applying the machine learning methodology with the purpose of predicting percentage to which category the geometric image captured by the camera belongs.

The main problem that surrounds the present study, is to determine if the Raspberry Pi platform has the capacity to control, manipulate and process images, and at the same time grant the ability to select and move geometric objects to a robotic arm of 4 DOF (Degree of Freedom), also using the KNN classifier method as machine learning methodology.

For which, the KNN classifier method and image processing are used to clean, equalize and detect objects within the photographic image, which is taken by the PiCam 2.1 built into the Raspberry Pi 3 model B [2], device robotic arm driver. The PiCam has a resolution of $2,592 \times 1,944$ pixels (approximately represents 5 megapixels), but for this project the configuration of 640×480 pixels was used, for reasons of standardization of the study.

The mechanism used is the following: First, the robotic arm takes a picture with the PiCam 2.1 mentioned above, the same one that passes through several methods for image processing. Then, the characteristics of the object are obtained through the first two invariant moments of HU, and finally, the KNN method is applied to predict that the image captured by the robotic arm belongs to a specific category (Geometric Shape Categorized).

Regarding the robotic arms, are very versatile and can be used for a series of applications in real life. The design of a robotic arm depends on the number of parameters among which the Degree of Freedom (DOF) is the most basic. Each DOF is a joint on which it can be bent or rotated; this project uses 4 DOF. The DOF number will be the number of actuators of the robotic arm [15].

Robotic arms are used for their ability to perform tasks very similar to a human, wrapped by several characteristics such as dexterity, flexibility, space saving, less complexity of tools and gripping devices. The robotic arm of this study has a clamp as a gripper, see Fig. 3 [13, 14].

Besides, robotic arms are very flexible for their design and configuration even in 3D virtual environments.

Another important aspect is to define the movement of the robot, the proposed approach allows the automatic generation of robot movements, requesting a small number of critical positions, for example, the location of grabbing the piece, avoiding instructing positions to avoid collisions with accessories, see Table 1 [13, 14].

The start and rotation positions of each DOF are fixed for this investigation; this is because the robotic arm is static, the same would happen in industrial conditions, that is, a robotic arm that selects objects in a band, it would be located in a fixed position.

By including sensors and internet connection, the robotic arms can even be operated remotely. There is a wide range of applications such as: in medicine for human operations or training of new surgeons, in the academy for the teaching of physics, in primary sciences as motivators for small students, and in the industry all in assembly lines [13, 14, 16].

This project intends to use Raspberry Pi as a control platform and to determine an introduction with machine learning and image processing, to create a prototype of a robotic arm that can detect objects with a high predictive level.

2 Prototype Construction

2.1 Control of Devices with Raspberry Pi

For the control of the robotic arm, Raspberry Pi was used (Figs. 1 and 2), in recent years this reduced plate computer (SBC) has motivated research in colleges and universities not only in Europe, but throughout the world, has a small size which favors mobility, has the characteristics of a standard computer, that is, has USB ports, a micro-SD that is used as a hard disk at low cost has a memory of 1 GB, and its ARM Cortex A53 processors run to 1.2 GHz 64-bits [11].



Fig. 1. Raspberry Pi B+ 3 (source: Raspberry Pi Foundation) [1, 11]

The current project uses the Raspberry Pi 3 Model B+ (Fig. 1). One of the most favorable aspects of the Raspberry [2], is that it has 40 pins of connection and interactivity with external hardware to the devices, as shown in Fig. 2, in the particular case of this project, these pins served to control the 4 servo motors (DOF) of the robotic arm, plus an advantageous feature of the Raspberry is that it has an operating system, among the most used, are Raspbian (Stretch with Desktop April-2018 used in this project), Ubuntu Mate, Windows 10 IOT, Openelec, OSMC, among others.

In this way, you can load some platforms and applications to the Raspberry device, for this research project, the following software packages were installed: Python 3.6, OpenCV 3, and the following drivers: Python SMBus (I2C), Adafruit PCA9685 [11].

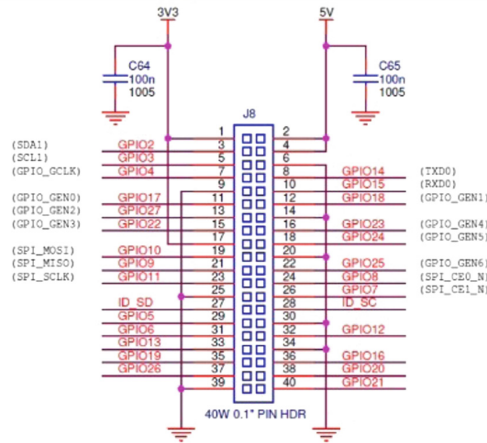


Fig. 2. Raspberry Pi GPIO model 3 (source: Raspberry Pi Foundation) [1,11]

2.2 Raspberry Pi for Robotics, Advantages

Starting from the fact that Raspberry in all its models is an SBC (Single Board Computer), that is to say, it is a miniature computer, it has all the elementary and essential characteristics of a machine, and also its GPIO (Fig. 2), allows interaction with other hardware, constructed or embedded.

From the above, it can be mentioned that it is a computer with a great ability to add functionality to other hardware, this allows, for example, adding sensors, servomotors, among other modules to this device, that is why, and that the Raspberry PI is one of the leaders of the mini-open source computer market, for which reason it was chosen as the base platform for this research project [1,5,11].

One of the most flexible and secure features is its ability to run on various operating systems, such as those of the Linux family: Raspbian, Ubuntu Mate, Pidora, Minibian, ArchLinux ARM, Openelec, among others. These platforms favor the loading of packages that facilitate the development and support to work in various applications on Raspberry PI [11].

It should also be noted that its low power consumption, full performance model B consumes 330 ma, this is full capacity including video playback and internet connection, low consumption favors the creation of portable applications with built-in rechargeable batteries. For this research, we used python, OpenCV 3, libraries that were installed on the device under the Raspbian operating system [5].

2.3 Robotic Arm Movement with PWM on Raspberry Pi

Within the applications or uses of robotics most used, are the robotic arms, these can be applied in different instances and contexts of the life of the human being.

The development of this prototype, has the purpose of introducing a robotic arm application of 4 DOF (Servomotors SG90) that has the ability to detect the following volumetric geometrical shapes: triangular prism, pentagonal prism, rectangular prism, and cube; with certain characteristics and colors, the ability to recognize ways is described within the feature vector and image processing section below.

For control of the robotic arm, a PCA9685 driver is used, which works correctly with PWM or pulse width modulation, a scheme of the type of signal is shown in Fig. 4. PWM is a type of voltage signal used to send information or to modify the amount of energy that is sent to a load. This type of signals is widely used in digital circuits that need to emulate an analog signal [6, 11].

In Raspberry PI, there is a channel for PWM, this is GPIO 18 (Not used for this project), for the prototype we used GPIO 1 (3.3 V), GPIO 2 for data, and GPIO 3 clock handling, this is the configuration of the Raspberry Pi with PCA9685, this driver or controller that can simultaneously handle up to 16 channels.

The motors used in the robotic arm are SG90 servomotors. This type of motor is controlled by the Raspberry PI and maneuvers the position through PWM pulses. The servos, short form to call them, have a nominal control voltage between 4.8 and 7.2 V with an electric current between 0.2 and 5 A. The value of the voltage rating will depend on how many servos will be connected at the same time. The servomotors, present an angular movement between 0° and 180° normally, If a more extensive range is desired in the position, a continuous rotating servo is required. On the contrary, these motors are controlled by speed, the servomotors that are controlled by their position [3, 7, 8].

The servos can be analog or digital, although most are analog, like the ones used (DOF) in the robotic arm. The digital servomotors are more precise and tend to be of better quality and with longer service life. They can also operate at higher frequencies (on the order of 300 Hz), but they can work correctly for simple, low-power applications. The servos only run at a frequency of 50 Hz and have a low power consumption [3, 4, 11].

To use the PCA9685 driver, the Adafruit driver was used for 16 channels; however the robotic arm is 4 DOF, so only 4 of the 16 channels were used. For the robotic arm, we worked with the channels 0,1,2,3 of the PCA9685; It is worth mentioning that the Adafruit controller needs 3 V power, the same ones that are charged through the 3.3V GPIO Output Pin [3, 4, 11].

Modulation Amplitude Strategy. By working with the PCA9685 driver, the optimal use of the Raspberry GPIOs is guaranteed, since we will only occupy two pins, all thanks to the fact that the PCA9685 uses the I2C protocol, which also ensures that several devices are connected through a serial bus of data like teachers, and a bi-directional communication takes place in the case of our project, but at the same time if more connected devices are demanded also it could be implemented.

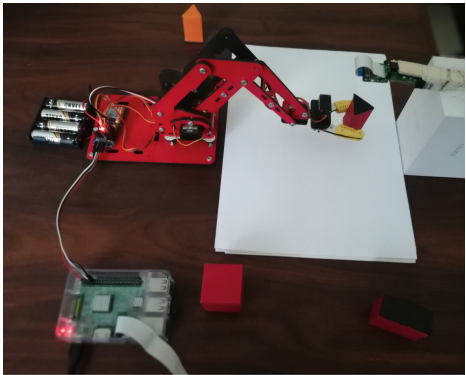


Fig. 3. ArmUno arm structure with Raspberry Pi of this project

With the PCA9685 and the I2C protocol, pulses can be generated through software, so the following Table 1 is implemented with Python and the Adafruit Library for PCA9685, explains the pulses used for each servomotor, the angular limits and the name that was used with these within the robotic arm controller script.

Table 1. Servomotors values of the project

Name	Programming name	Angular limits []	Pulse limits	Canal at PCA9685
Rotation	RotationArm	[0,180]	[140,550]	0
Up down right	UpDown1	[80,120]	[220,350]	1
Up down left	UpDown2	[90,160]	[300,450]	2
Pincers	OpenClosePincers	[0,150]	[140,300]	3

3 Vector of Features and Image Processing

As mentioned in the introduction to the present study, the images are captured at a resolution of 640×480 pixels at 32 bits of color by the Picam 2.1 installed in the Raspberry Pi driver device. For the application of the robotic arm, geometric objects are detected and classified.

For the processing of the images, the color space of the image is changed to grayscale. To eliminate the noise and soften the image a Gaussian filter function (GaussianBlur) is used. Then, thresholding is applied to segment the image and obtain the object to be identified. The thresholding returns a binary image, where the pixels with value ‘1’ represent the object and the pixels with value ‘0’ the background.

To obtain the characteristics that identify each geometric object the invariant moments of Hu were used, which are seven invariant descriptors that quantify the shape of an object. The first two moments of Hu are obtained, and with these descriptors, the vector of characteristics that serves as an input for the classification of the objects is formed by the KNN classification method of the nearest neighbor [9,10].

3.1 Conversion of BGR Color Struct to Grayscale in OpenCV

OpenCV has an information structure to manipulate the pixels of an image when loading the picture, it is structured according to coordinates where the pixels that make up the image itself, in this way, when loading a color image in BGR it would have the following matrix:

$$\mathcal{I}_{m,n} = \begin{bmatrix} a_{1_1} a_{1_2} a_{1_3} \dots \\ a_{2_1} a_{2_2} a_{2_3} \dots \\ a_{3_1} a_{3_2} a_{3_3} \dots \\ a_{i_j} a_{i_j} a_{i_j} \dots \end{bmatrix}$$

OpenCV has an information structure to manipulate the pixels of an image when loading the picture each element of the matrix has a particular value. If it is a BGR image, the element $a_{1_1} = [B, G, R]$ [12], that is, that coordinate would contain a triad of values to represent a pixel of color, that is B represents the blue channel, G the green, and R the red. When converting the image captured by the PiCam 2.1 to grayscale format, instead, each pixel is represented by a unique value between 0 and 255 (0 represents black and 255 white). The implementation with OpenCV is as follows:

```
$imagegray = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)$
```

3.2 Smoothing

The next step is to apply the Gaussian Blur function to eliminate the noise and soften the edges. This function allows the elimination of high-frequency noise by convolving the image with a Gaussian kernel. The kernel is a square matrix, which must be odd. A kernel of size $[5 \times 5]$ and a standard deviation of 1 is applied for the x-axis, and for the y-axis, as shown in the implementation with Python and Opencv [9]. The mathematical application of the Gaussian kernel will be as follows: Since H_1 is the coordinate matrix of the image captured by the PiCam 2.1 and H_2 the transpose of the said matrix and Sigma with 1, the K kernel is defined by:

$$K = \rho^{-(H12+H22)/(2*2)}$$

$$KNORMAL = K_{i,j} / \sum K$$

For the Kernel (K) standardization, the scale of the data is basically standardized, as shown in the KNORMAL equation, each Kernel element given by K_i, j is divided for the sum of all the elements given by K, in this way each part of K is normalized. Next, the implementation with python and OpenCV with the same mathematical base is shown:

```
$imageblur = cv2.GaussianBlur(imagegray, (5, 5), 1, 1)$
```

3.3 Thresholding

For the identification of geometric objects, it is necessary to separate the pixels that interest us from others. This separation is based on the variation of the intensity between the object pixels and the background pixels. A comparison of each pixel with the established threshold is made. In this way, a binary image is obtained, where the pixels that are in the range of the threshold take the value of 1 representing the object and the remaining pixels take the value of 0. According to the tests carried out, the threshold value was determined in 87 [10].

```
ret,thr = cv2.threshold(imageblur, 87, 255, cv2.THRESH_BINARY)
```

3.4 Feature Extraction (Hu Moments)

After processing the image, we obtain the characteristics that will allow us to differentiate and classify each object. To obtain these characteristics, the first two invariant moments Hu were used. The moments enable quantifying the characteristic shape of an object using the way in which the pixels of the object are distributed on the plane of an image.

The moments of Hu are invariant to the geometric transformations that the object can undergo (translation, rotation, and scaling); that is, very similar values are obtained for objects of the same type and at the same time they are discriminant since different values are obtained according to the type of object [12].

The moments of Hu are calculated based on the geometric moments of the object. The geometric moments of the order $(p + q)$ are calculated in the following way:

$$m_{p,q} = \sum_x \sum_y x^p y^q I(x, y)$$

where $I(x, y)$ is a pixel of the object with coordinates (x, y) . From these moments we can obtain the second order normalized central moments:

$$\mu_{p,q} = m_{p,q} / \mu_{00}^{1+(p+q)/2}$$

Based on $\mu_{p,q}$ Hu, he obtained the seven invariant moments, the first two being:

$$\theta_1 = n_{20} + n_{02}$$

$$\theta_2 = (n_{20} + n_{02})^2 + 4n_{11}^2$$

In OpenCV two functions are used: `cv2.moments` with which the moments of the object are obtained and `cv2.HuMoments` that is applied to the moments of the object.

```
momentosHu = cv2.HuMoments(cv2.moments(thr))
for i in range(len(momentos)):
    for j in range(len(momentos[i])):
        if i == 0:
            Hu1 = momentos[i][j]
        elif i == 1:
            Hu2 = momentos[i][j]
```

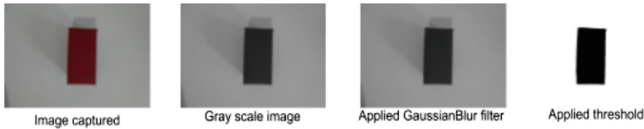


Fig. 4. Application of filters to the image captured by the PiCam 2.1

4 Construction of Data Training

The test dataset was made taking different photographs of the objects to be classified. The pictures were taken with the PiCam 2.1 at a resolution of 640×480 pixels with 32 bits of color. Each type of object has the same dimensions and different colors. The photographs taken were taken with natural light and artificial light generating some kinds of shade. The dataset consists of 800 elements, having 200 items for each type of object.

In the construction of the data training, the images of the dataset are processed, and each object is characterized by a vector that contains the characteristics that identify it, called pattern, which is associated to a class tag that identifies it among the different classes of objects. For this investigation, each object is represented by a vector of two characteristics (Hu1 and Hu2) and is associated with one of the four object classes: 1 = triangular prism, 2 = pentagonal prism, 3 = rectangular prism, 4 = cube. For training and testing the KNN method, the dataset was divided: 70% (760 images) for training and 30% (240 images) for the test.

For standardization of image processing, the resolution was used 640×480 , with an optimal resolution, it was also decided to use natural lighting, inside the

laboratory where the experiments were processed artificial light did not provide the necessary conditions, to assemble a dataset with good results when applying both the processing techniques and the KNN.

Under this context, in the first instance, the color histogram used, to obtain one of the characteristics that would form part of the vector of parameters for the data training, standardization of the size of images. Therefore the expected results were not obtained, these processing techniques were left aside, they consumed a lot of processing time in the Raspberry Pi, and the KNN did not reflect the best predictions.

On the other hand, the background chosen to capture the photographic images of the robotic arm was the white color, although some trials were done with a black background, but did not provide the necessary conditions to improve the results in the image processing to build the dataset and the KNN, it is also included, that the shadow that is projected with artificial and natural light also affects the result in the vector of characteristics.

To reduce overfitting, photographs were taken with different color scales to vary the intensity and add a shadow factor to obtain results closer to the real world. For schemes of use of the robotic arm in industrial applications or factories, the brightness levels should be adjusted. Improve the camera resolution to obtain clearer results, in this way the algorithm used for image processing would work optimally.

5 Classification Method (KNN) and Inductive Learning

Inductive Learning allows creating models of concepts from generalizing simple examples, primarily derived from the search for common patterns that somehow explain the cases, in other words with inductive learning you get general conclusions of specific information. Inductive reasoning, on the other hand, allows you to make statements based on the evidence you have collected so far, a scientific experiment for example. However, the evidence is not the same as a fact: the solid proof of something that is only suggested, albeit strongly, that it is a fact in itself [10].

The KNN classifier method is based on the fact that a new sample is classified according to the class of the closest neighbors of the training pattern. The recognition of objects is done by comparing the object to be classified with a reference sample of each type of object; assigning him the class of his closest neighbors. The algorithm makes this assignment according to the nearest K neighbors using the Euclidean distance between each of the samples [10].

The learning process in this project consists in storing in a vector the characteristics of the training set together with the class label associated to each sample forming the training data (data training), the tests carried out applying this methodology will be shown in the next section. To select the optimal k value; a test process was carried out in several instances, as shown in Fig. 5, the best result was with $k = 3$.

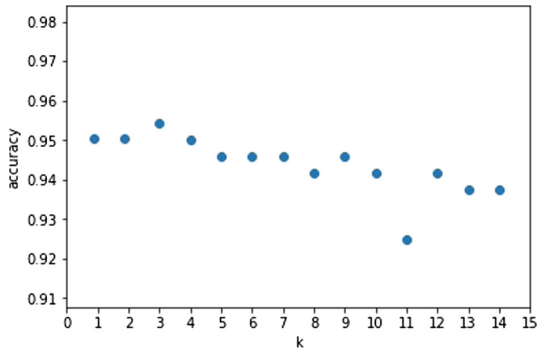


Fig. 5. Accuracy of K value

6 Results

For all tests, it was determined that the optimal number of neighbors was determined, $k = 3$, as explained in the previous section on Classification method KNN. Then the test was carried out with 240 images of the different types of objects to be classified (53 triangular prisms, 67 pentagonal prisms, 66 rectangular prisms and 54 cubes).

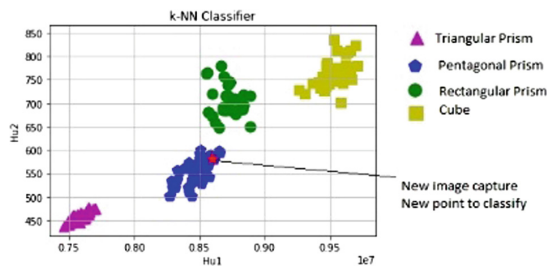


Fig. 6. Pentagonal prism test

In the test plotted in Fig. 6, which represents the trial with the pentagonal prism, that is, in front of the robotic arm, the volumetric geometric figure is placed, and this is captured with the PiCam, the photograph captured by the robotic arm falls on the nearest neighbor [2.2.2.], whose values represent precisely the category Pentagonal Prism, with a Euclidean distance of $1.3566319e-14$ for each of the neighbors with slight variations. Of the 67 data to classify as a pentagonal prism, it ranked 8 as a rectangular prism and scored 59, having a precision of 88%, Table 2 (Classification Report) details the results.

On the other hand, in the test plotted in Fig. 7, which represents the test with the rectangular prism, the photograph captured by the robotic arm falls on the

Table 2. KNN Classification Report

Geometric objects	Precision	Recall	F1-score	Support
Triangular prims	1.00	1.00	1.00	53
Pentagonal prims	0.95	0.88	0.91	67
Rectangular prims	0.89	0.95	0.92	54
Cube	1.00	1.00	1.00	27
Avg./Total	0.96	0.95	0.95	240

nearest neighbors [3.3.3.], whose values represent exactly the Pentagonal Prism category, with values for the Euclidean distance of $[0.0000000e+000.0000000e+002.7098367e-13]$ for each of the close neighbors. The Euclidean distance, is the distance calculated between two points within a Euclidean space, obtained from the Pythagorean theorem. [9] the 66 data to classify as a rectangular prism, it classified 3 as a pentagonal prism and scored 63, obtaining a precision of 95%, Table 2 (Classification Report) details the results.

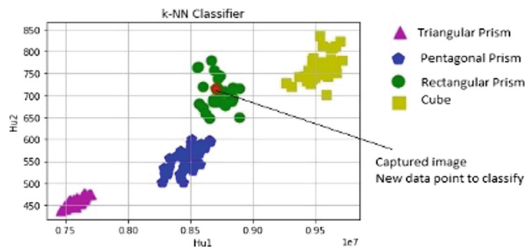


Fig. 7. Rectangular prism test

Finally, the test plotted in Fig. 8, which represents the test with the cube, the photograph captured by the robotic arm falls on the nearest neighbors given by [4.4.4.], whose values represent exactly the Pentagonal Prism category, with values for the Euclidean distance of $[0.0000000e+001.2050940e-156.5864266e-14]$ for each of the close neighbors.

The confusion matrix is a square matrix whose order is the number of classes. The real classes are presented in the columns, while the categories assigned by the classifier are shown in the rows [10].

The description of the confusion matrix for KNN is as follows: 53 true positives for triangular prism, for pentagonal prism 59 true positives and 3 false positives were obtained, while for rectangular prism 63 true positives and 8 false positives were collected and for the cube, the true positives were 54.

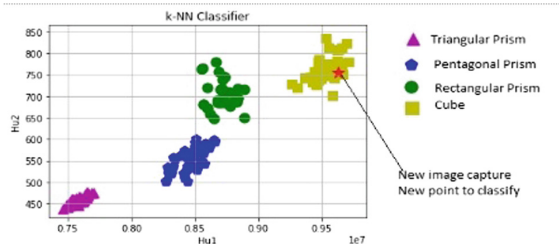


Fig. 8. Cube test

$$\text{KNN Confusion Matrix} = \begin{bmatrix} 53 & 0 & 0 & 0 \\ 0 & 59 & 8 & 0 \\ 0 & 3 & 63 & 0 \\ 0 & 0 & 0 & 54 \end{bmatrix}$$

For the KNN test, the respective matrix confusion and the classification report shown in Table 2 are shown. For all the tests carried out The **average effectiveness** of KNN was 95.42%

7 Discussion

The results achieved by the tests carried out suggest the acceptable performance of the robotic arm. First, according to the results of the confusion matrix, an average accuracy of 95.42% was achieved. The confusion matrix presents information on the prediction made by the method. It is a square matrix whose order is the number of classes. The real classes are shown in the columns, while the categories assigned by the classifier are presented in the rows, see Table 2.

For trained data, the robotic arm can take classification and movement activities with a greater than 89% accuracy for the case of the rectangular prism, but in the case of the cube, and the triangular prism is 100% accurate (see Table 2). On the other hand, for the processing of 800 images of the dataset and form the vector of characteristics, which is the input of the data training and data test, it takes 32 s in the Raspberry Pi 3 B+, previously an experiment was performed with a datasets of 400 images, earning a time of 20 s. For the final examinations, the dataset was left with 800 illustrations, whose construction was specified in the construction section of the data training.

After this initial process, the robotic arm takes two seconds to take the picture, do the image processing and classify it, which is a good time of action for the Raspberry Pi. For an industrial robotic arm, it may not be a good time, but we should consider better hardware features and the use of better deep learning methods for future tests.

8 Conclusions

By applying these techniques and the machine learning methods, it was demonstrated that the optimal use of the Raspberry Pi as a controller. On the one hand, worked controlling the movements of the robot arm, its ability to contain software solutions empower prototypes, the handling of 4 servomotors with PCA9685, high-level coding was carried out using Python, due to the Adafruit library as an interface between Python and Raspberry Pi.

In the same way, Raspberry Pi worked at a high level with the processing of the images necessary for the data training characteristics vector, approximately 32s to capture and process each image, an excellent process time. This led to an appropriate construction of the data formation, and to very good executions of the presented method.

The reliability of the results was determined by the confusion matrix, and the sci-kit learn classification report, where the magnitude of the application of the KNN method was predicted.

It is estimated that the use of more efficient algorithms in classification will be used for future works, such as neural networks, all of this to improve the productivity and automatic learning of the robotic arm. One of the objectives of this research project is to introduce in the medium term the use of robotic arms for industrial applications and education.

References

1. Sharma, J., et al.: IOP Conference Series: Materials Science and Engineering, vol. 263, p. 052049 (2017)
2. Anandhalli, M., Baligar, V.P.: A novel approach in real-time vehicle detection and tracking using Raspberry Pi. *Alexandria Eng. J.* **57**, 1597–1607 (2017). <https://doi.org/10.1016/j.aej.2017.06.008>. ISSN 1110-0168
3. Vazquez Navarro, D.: Control of a robotic arm using an Omega 2+ module. Thesis, Universitat Politècnica de Catalunya (2018)
4. Rahman, M.F., Patterson, D., Cheok, A., Betz, R.: 30 - motor drives. In: Rashid, M.H. (ed.) *Power Electronics Handbook*, 4th edn, pp. 945–1021. Butterworth-Heinemann, Oxford (2018). <https://doi.org/10.1016/B978-0-12-811407-0.00034-9>. ISBN 978-0-12-811407-0
5. Perumal, V.S.A., Baskaran, K., Rai, S.K.: Implementation of effective and low-cost building monitoring system (BMS) using Raspberry PI. *Energy Proc.* **143**, 179–185 (2017). <https://doi.org/10.1016/j.egypro.2017.12.668>. ISSN 1876-6102
6. Soriano, A., Marn, L., Valls, M., Valera, A., Albertos, P.: Low cost platform for automatic control education based on open hardware. *IFAC Proc. Vol.* **47**(3), 9044–9050 (2014). <https://doi.org/10.3182/20140824-6-ZA-1003.01909>. ISSN 1474-6670
7. Di Piazza, M.C., Pucci, M.: Techniques for efficiency improvement in PWM motor drives. *Electr. Power Syst. Res.* **136**, 270–280 (2016). ISSN 0378-7796
8. Soriano, A., Marin, L., Valles, M., Valera, A., Albertos, P.: Low cost platform for automatic control education based on open hardware. *IFAC Proc. Vol.* **47**(3), 9044–9050 (2014). ISSN 1474-6670, ISBN 9783902823625
9. Beyeler, M.: *Machine Learning for OpenCV. Intelligent Image Processing with Python*, 1st edn. Packt Publishing, Birmingham (2017). ISBN 978-1-78398-028-4

10. Muller, A.C., Guido, S.: Introduction to Machine Learning with Python, Kindle edn. O'Reilly Media, Sebastopol (2017). ISBN 978-1-449-36941-5
11. Monk, S.: Raspberry Pi CookBook, 2nd edn. O'Reilly Media, Sebastopol (2016). ISBN 978-1-491-93910-9
12. Yoon, D.C., Mol, A., Benn, D.K., Benavides, E.: Digital radio-graphic image processing and analysis. *Dent. Clin. North Am.* **62**(3), 341–359 (2018). <https://doi.org/10.1016/j.cden.2018.03.001>. ISSN 0011-8532, ISBN 9780323610766
13. Tlach, V., Kuric, I., Kumericakova, D., Rengevic, A.: Possibilities of a robotic end of arm tooling control within the software platform ROS. *Proc. Eng.* **192**, 875–880 (2017). <https://doi.org/10.1016/j.proeng.2017.06.151>. 12th International Scientific Conference of Young Scientists on Sustainable, Modern and Safe Transport. ISSN 1877-7058
14. Makris, S., et al.: Dual arm robot in cooperation with humans for flexible assembly. *CIRP Ann.* **66**(1), 13–16 (2017). <https://doi.org/10.1016/j.cirp.2017.04.097>. ISSN 0007-8506
15. Sunny, T.D., Aparna, T., Neethu, P., Venkateswaran, J., Vishnupriya, V., Vyas, P.S.: Robotic arm with brain. Computer interfacing. *Proc. Technol.* **24**, 1089–1096 (2016). <https://doi.org/10.1016/j.protec.2016.05.241>. International Conference on Emerging Trends in Engineering, Science and Technology, ICETEST 2015. ISSN 2212-0173
16. Kadir, W.M.H.W., Samin, R.E., Ibrahim, B.S.K.: Internet controlled robotic arm. *Proc. Eng.* **41**, 1065–1071 (2012). <https://doi.org/10.1016/j.proeng.2012.07.284>. International Symposium on Robotics and Intelligent Sensors, IRIS 2012. ISSN 1877-7058