Object Detection and Recognition for a Pick and Place Robot

Conference Paper · November 2014

DOI: 10.13140/2.1.4379.2165

4 authors, including:

Rahul Kumar
University of Padova
17 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Sanjesh Kumar
University of the South Pacific
1 PUBLICATION **14** CITATIONS
SEE PROFILE

Praneel Chand
Waikato Institute of Technology
35 PUBLICATIONS **148** CITATIONS
SEE PROFILE

Object Detection and Recognition for a Pick and Place Robot

Rahul Kumar
The University of the South Pacific
Suva, Fiji
Email: rahul.kumar@usp.ac.fj

Sanjesh Kumar
The University of the South Pacific
Suva, Fiji
Email: s11065712@student.usp.ac.fj

Sunil Lal
The University of the South Pacific
Suva, Fiji
Email: sunil.lal@usp.ac.fj

Praneel Chand
The University of the South Pacific
Suva, Fiji
Email: chand_pc@usp.ac.fj

Abstract—Controlling a Robotic arm for applications such as object sorting with the use of vision sensors would need a robust image processing algorithm to recognize and detect the target object. This paper is directed towards the development of the image processing algorithm which is a pre-requisite for the full operation of a pick and place Robotic arm intended for object sorting task. For this type of task, first the objects are detected, and this is accomplished by feature extraction algorithm. Next, the extracted image (parameters in compliance with the classifier) is sent to the classifier to recognize what object it is and once this is finalized, the output would be the type of the object along with it's coordinates to be ready for the Robotic Arm to execute the pick and place task. The major challenge faced in developing this image processing algorithm was that upon making the test subjects in compliance with the classifier parameters, resizing of the images conceded in the loss of pixel data. Therefore, a centered image approach was taken. The accuracy of the classifier developed in this paper was 99.33% and for the feature extraction algorithm, the accuracy was 83.6443%. Finally, the overall system performance of the image processing algorithm developed after experimentation was 82.7162%.

Keywords **– Object Detection, Object Recognition, Feature Extraction, Classifier.**

I. INTRODUCTION

Vision based control of the robotic system is the use of the visual sensors as a feedback information to control the operation of the robot. Integration of the vision based algorithms can enhance the performance and the efficiency of the system. Vision based configurations have been implemented to mimic human visual sensors. Orienting towards robotic arms, object recognition is vital for the operation of arms for navigation and grasping tasks. Often it has been the case that image processing (IP) algorithms require huge processing time for the successful implementation of object recognition.

The work presented in [1] critically explains the basic algorithms to be addressed before applying image processing techniques. These techniques include; image enhancement, noise reduction and a visual loop algorithm (based on trial and error approach). Moreover, the works of [2], [3] and [4] presents the IP algorithms and approaches to reduce response time and increase in the efficiency for the object recognition tasks. In [8], discussion is based on the reduction of computation time using Trainarp algorithm (derived from ANN). It also presents the method to migrate from the statistical approach to Artificial Neural Networks (ANN). The author has stated the efficiency as 95% and response time of 94ms. Likewise, [3] has conversed on employing a parallel programming approach called object surface reconstruction method. Upon comparison with serial approach, parallel programming method is ten times faster. To reduce cost and improve on performance, [4] has presented the communication of vision system via USB. The vision system used was a webcam for which via MATLAB, the system is enabled to perceive environment through artificial vision via IP algorithms.

Along with the classification part, the concept of Feature Extraction (FE) is also studied. FE mostly acts as a pre-processing algorithm to furnish the dataset for the classifier to make important decisions/classification. The work of [5] elaborated on the usage of multi-stereo vision technique for the detection of 3D Object. Eliminating the background i.e. objects of least interest, using opening and closing morphological techniques, 3D detection of a particular object was achieved. Similarly, [6] conversed about one of the Robust Object detection algorithm. This algorithm is known as the Viola-Jones IP method, a state of the art face detector. The robustness of this algorithm was due to cascaded architecture of the strong classifiers arranged in the order of complexity. This approach was incorporated to reduce the processing time. Lastly, feature extraction via Contour matching is also one of the best methods to detect objects [7]. The trained shape is matched according to a probabilistically motivated distance measure which enhances the shape comparisons within the framework. The works in [7] also presented on the noise reduction and other image optimization via segmentation and other IP techniques.

The goal of this paper is to develop IP technique which will involve the FE and classification algorithms suitable for object sorting task. Additionally, the system to be developed needs to be robust as it will be tested on a real time basis. It is planned to use the developed IP technique on SCORBOT ER-4U (robotic arm platform) [9] which will be refurbished and utilized to sort electronic components such as resistors and capacitors for laboratory technicians. The remainder of this paper covers the algorithms of feature extraction and classification. Further discusses on the determination of object location and also portrays all the results carried out for the development of the algorithms. Finally, making the concluding remarks on the results and further recommendations on how to improve the developed model.

II. Conceptual Framework of the Entire System

The above figure shows how the image processing algorithm will work. The constant variable in this case is the x-y dimensions of the workspace. The image taken is first standardized according to the workspace dimensions. This is achieved by resizing the taken image according to the dimensions of the workspace.

$$\textit{width} \text{ of image (pixels)} = (37.875275591)\, w_S \quad (1)$$

$$\textit{height} \text{ of image (pixels)} = (37.875275591)\, h_S \quad (2)$$

Note : w_S and h_S are width and height of the workspace (where the components are) in centimeters

Next, via feature extraction algorithm, objects are detected, cropped and resized according to the classifier specifications. Then using the ANN classifier, the object detected is classified and the coordinates of the corresponding object is determined. The scope of this paper is limited up-till here, however, from here-on is the task of the Robotic arm to pick the object from the location (coordinates) specified and place it according to the user's discretion.

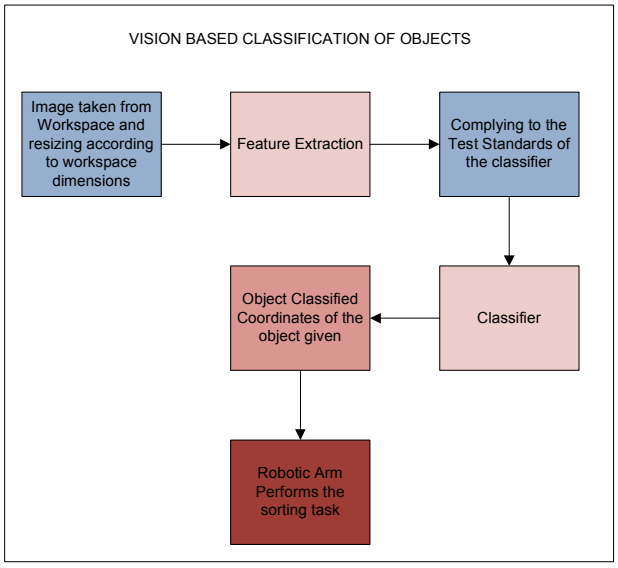

Figure 1: Conceptual Framework of the whole system

III. The Feature Extraction Algorithm

The feature extraction part in the development of this model plays a vital role as it furnishes the raw image and complies it according to the classifier's specifications.

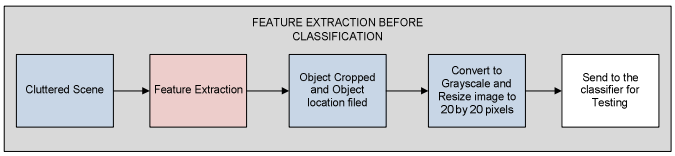

Figure 2: Feature Extraction before Classification

The above framework represents the images in the cluttered scene (test subjects) to be tested for correct classifications. For accurate classification of the objects, feature extraction algorithm needs to be considered prudently.

A. Algorithm

1. The image is read and converted to grayscale. The grayscale conversion is achieved by eliminating hue and saturation information while preserving the luminance. The colored image (RGB image) is 3 dimensional. To convert to a 2 dimensional grayscale image, the following equation represents the correct proportion of RED, GREEN and BLUE pixels to be taken into account:

$$0.2989R + 0.5870G + 0.1140B \quad (3)$$

Figure 3: The Original Image

2. Then the grayscale image is converted into a binary image. The method used to convert to binary image is known as Otsu's method. This method is a non-parametric and an unsupervised method of automatic threshold selection for picture segmentation. The threshold selected by Otsu's method will intend to minimize the intra-class variance of black and white pixels.
   *Even though the threshold value can be selected independently by the user, it is set to auto-mode as during the tests, images will vary.

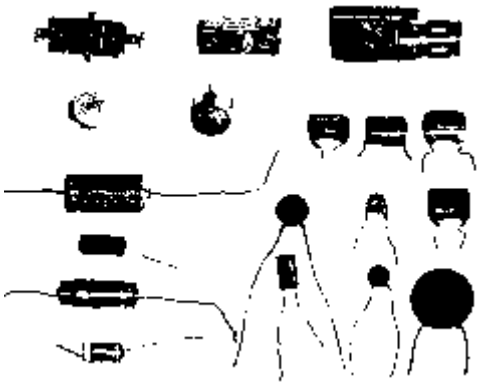

Figure 4: Binary Image

3. Edge Detection

Once the images are converted to binary, edge detection criterion is applied to detect edges in a given image. This part is very crucial as this criterion will act as a pre-requisite for BLOB's analysis in the later procedures. The Edge Detection algorithm developed is based on the "Canny method". This method finds edges by looking for the local maxima of the gradient of the image. This gradient is based on the Gaussian Filter.

Overview of the Canny Method

a. Smoothing: the effect of Gaussian smoothing is blurring of an image and this degree of blurring is entirely based on the standard deviation of the Gaussian distribution. The more the standard deviation, the more blurry the image. In this model a value of 1.41 is set as a standard deviation value.

b. Finding Gradients: the Canny method finds edges where intensity of the images changes the most. The gradient magnitudes are determined by the Euclidean distance. Hence, the gradients are calculated by:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (4)$$

$$|G| = |G_x| + |G_y| \quad (5)$$

Where G_x and G_y are the gradients in the x and y directions.

The direction of the gradient needs to be stored to keep track for the upcoming Non-maximum Suppression method. The direction is stored in:

$$\theta = \arctan\left(\frac{|G_y|}{|G_x|}\right) \quad (6)$$

c. *Non-Maximum Suppression*: at the above step, the edges were blurry, this method converts these blurry edges to sharp edges by preserving all the local-maxima in the gradient image and deleting the rest.

d. *Double Thresholding*: This is where edge selections are optimized, this step uses double Thresholding and is not likely be fooled by the noise/color variations due to rough surfaces. The pixels which are stronger than the high threshold value are preserved; however, the pixels which are weaker than the lower threshold value are suppressed.

e. *Edge tracking by Hysteresis*: Here, edges which are strong edges (with higher threshold values) and edges (weak ones if any) which are connected to the strong edges are finally preserved. By default, 8 connected pixels in the canny method are divided into BLOB's. The BLOB's containing at least one strong pixel are selected and preserved, while others are suppressed.

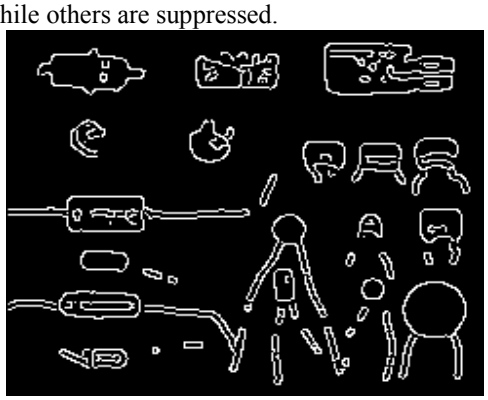

Figure 5: Edge Detection

4. *Image Dilation*: This procedure enlarges the edges created by the edge detection algorithm on the basis of a rectangular structuring element.

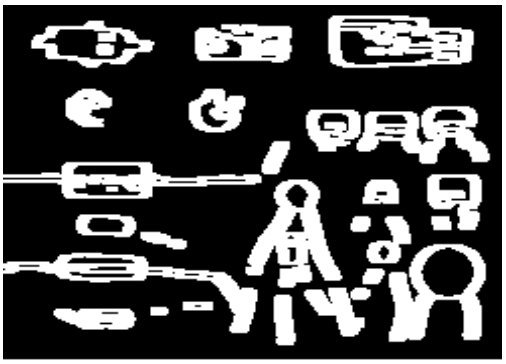

Figure 6: Image Dilation

5. *Image Filling*: The edges dilated are filled such that, outline of the edges are more clear and visible.

Figure 7: Image Filling

6. *BLOB's Analysis*: Since the image being analyzed is in the form of a cluttered scene, connected objects as per the strong filled edges will form a BLOB and are identified as an object. Likewise, each BLOB will form an object.

Once the objects are detected, a rectangular bounding box will be formed around the detected object and it will be cropped as outlined by the bounding box. Then that cropped image will be resized to 20 by 20 pixels and converted to grayscale to be ready for testing (for classification).

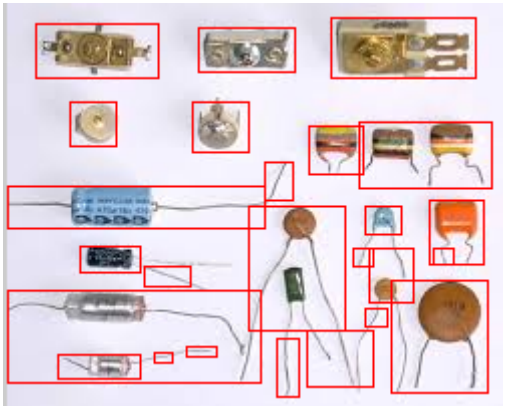

Figure 8: Object Detection

IV. Conceptual Framework for the Training of Classifier

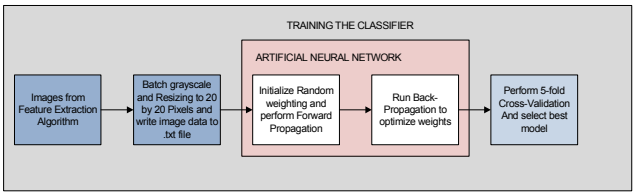

Figure 9: Conceptual Framework for the training phase

The above block diagram shows the process by which the training data is manipulated and trained by the classifier. The JPEG formatted images are the training data which consists of the images of Resistor and the capacitors equally weighted (same number). These images are first converted to grayscale and then resized into 20 by 20 pixels sizes. Thereafter, the data file (in .txt format) is trained by the classifier. Once, via Back-Propagation algorithm the weights are optimized, the values of weights are written to a file which will be used to perform tests (to determine the reliability of the classifier).

Software package used: MATLAB r2012b

V. Development and Training of the Classifier

The classifier is developed using the Artificial Neural Networks whereby the following describes the actual modelling oriented towards the object sorting task:

1. Organize and partitioning of data into test and training sets.
2. Assign a number to Capacitor images and Resistor images (e.g. Capacitor images are assigned the value of 1 and Resistor images are assigned the value of 2). This will be the output (y) for the training example, as this is a supervised learning algorithm.
3. Put the training output data (y) in form of a dimensional vector.
4. Decide on the number of layers (entirely based on the features of the images which are being manipulated).For the system proposed in the paper, total layers will be 3, i.e. Input layer, One Hidden layer and an output later.
5. Implement the Feed-Forward Propagation and Cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [-y_k^{(i)} \log(h_\theta(x^{(i)}))_k - (1-y_k^{(i)}) \log(1-h_\theta(x^{(i)}))_k \quad (7)$$

Regularizing the Cost function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [-y_k^{(i)} \log(h_\theta(x^{(i)}))_k - (1-y_k^{(i)}) \log(1-h_\theta(x^{(i)}))_k + \ldots$$

$$\ldots + \frac{\lambda}{2m} \sum_{j=1}^{H} \sum_{k=1}^{400} (\theta_{j,k}^{(1)})^2 + \sum_{j=1}^{2} \sum_{k=1}^{H} (\theta_{j,k}^{(2)})^2 \quad (8)$$

Regularization of the parameters depends on the number of layers, and the learning rate, λ. The value of K=2 as there are only two possible outputs. *M* stands for the number of training examples, $y_k^{(i)}$ is the output of the *ith* training example and $h_\theta(x^{(i)})$ is the hypothesis function whereby:

$$h_\theta(x) = \frac{1}{1+e^{\theta^T x}} \quad (9)$$

The hypothesis function is a sigmoid function which has 1 as its upper bound and 0 as its lower bound. In addition to that the sigmoid function is differentiable at all points.

6. The above process computes the cost of the feed forward propagation. To determine the weights of the model, first

a random run must be carried out to start off with the optimization via Back-propagation. Therefore, in this initial stage, theta values (weights) are randomized, for the input and hidden layers.

7. Implement Back-Propagation for optimization of theta values: Once the hypothesis/prediction is made according to the initialized random weights. The Back-propagation starts off with the output layer; it measures the difference between the networks activation value and the true target value and further goes towards in the direction of input layer assigning the error to each neuron.

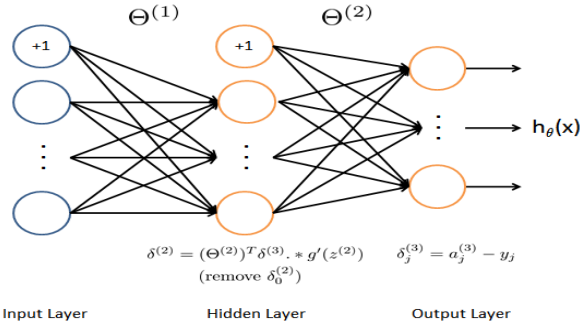

Figure 10: Back-Propagation (weights adjustment)

The above block diagram represents how the error is assigned to each term for a particular layer.

8. For higher accuracies, the Neural Network is trained using higher number of iterations. For the development of this model 20000 iterations were run to denote the best values of the weights (theta values).

VI. Models

For the object sorting task, in-terms of image processing, two models of the classifier were developed. The differences in the model are specified below:

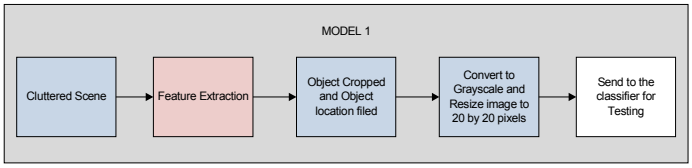

Figure 11: Model 1 Block Diagram

The Model 1 technique converts the cropped image to grayscale and resizes it to comply with the testing standards (20 by 20 pixels images used for testing). However, in Model 2, considering that resizing of an image detoriates the quality of the image, the cropped image is placed on the center of a white background.

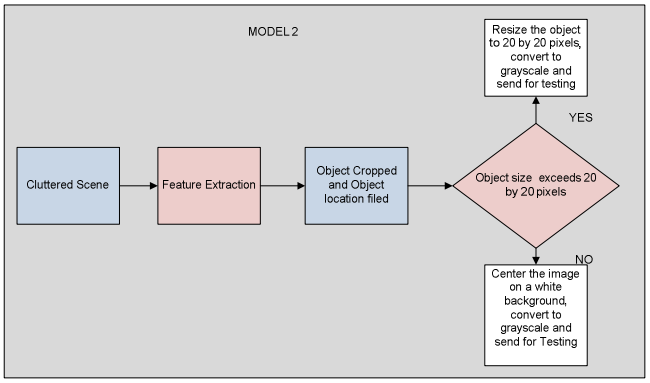

Figure 12: Model 2 Block Diagram

Note: For the images which exceed the boundary of 20 by 20 pixels, the respective dimension is only resized. (e.g. if an image is 35 by 18 pixels, 35 will be resized to 20 and 18 will remain the same)

VII. Cross Validation of the Classifier

The total training data contained 312 images and the classifier accuracy was determined by carrying out n-fold cross validation. The number of folds created was 5. For comparison purpose the following models were developed and cross validated to determine the best classifier. The output (2 neurons) and the input (400 neurons) number of layers were constant and the numbers of hidden layers were varied.

Model	No. of Neurons	% Accuracy
1	**25**	**99.33333**
1	30	99.66667
1	40	99.00000
1	50	98.66667
1	70	99.00000
2	25	99.00000
2	30	98.00000
2	40	98.66667
2	50	98.66667
2	70	98.66667

Table 1: Cross Validation Results

The Bold numerics in the table represent the best model.
Best Model: Model 1

- 400 Input Layers
- 25 Hidden Layers
- 2 Output Layers

VIII. Location of the Extracted Image (Coordinates)

The location of the object is a very important parameter because without the coordinate of the classified object, the object sorting task would be impossible. Before applying feature extraction, the whole image was resized as per the workspace dimensions (converting the metric dimensions to pixels and this will become the size of the image). From the scene, during the feature extraction, the locations of the objects

were filed and upon testing (classifier) the program gave out the center coordinates of the object also assuming the robotic arm is confined within the boundary of the scene.

The locations of the objects were obtained in form of a bounding box. In MATLAB this was in form of [xmin, ymin, width, height]. The (xmin, ymin) were actually the coordinate of the top-left most edge of the rectangular bounding box and the (width,height) were actually the width in x direction and height in y direction.

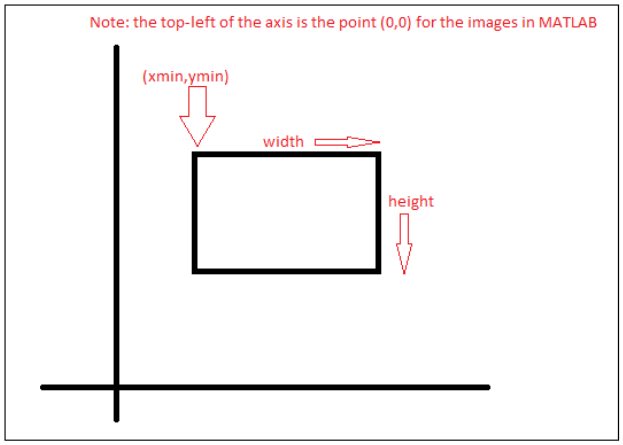

Figure 13: Location for the bounding box

Center of the object is given by:

$$xc = x\min + \frac{1}{2} width \quad (10)$$

$$yc = y\min + \frac{1}{2} height \quad (11)$$

whereby xc and yc are in pixels

Converting from pixels to centimeters:

$$[xn, yn] = 0.0264583333(xc,yc) \quad (12)$$

where 0.026458333 is the conversion factor to obtain cm values from pixels.

IX. Experimentation

A. Training Dataset Description

The training data consisted of 312 images which comprised of 156 individual capacitor images and 156 individual non-capacitor images. The capacitor images were taken on a white background and the positioning of the images were varied. The positioning were not only centered but also cornered in the training images. The training data was converted to grayscale and resized to 20 by 20 pixels and then sent in for testing.

B. Test Dataset Description

The test data had 32 images with cluttered scene. The test data was converted to gray scale, complied with the classifier specifications and then sent for testing.

C. Iterations and Learning Rate Used for training

For training of the dataset, 20000 iterations were first ran with a randomized values of weights, once via Back-Propagation new weights (theta) are determined, the iterations (20000) were rerun to get optimized values of weights (theta). Moreover, to avoid large deviations in the gradient descend algorithm, the value of learning rate was kept to 0.1.

X. Results

A. Feature Extraction Accuracy

The feature extraction algorithm was tested using a separate set of test data and its accuracy was 83.6443%.

B. Classifier Accuracy

$$\frac{No.\ \text{True Positive} + \text{No. True Negative}}{Total\ \text{Samples}} \times \frac{100}{1} \quad (13)$$

The classifier accuracy was determined using equation 13.

C. Final Results

Below is the result table of 32 scenes which altogether has a total of 448 objects inclusive of both capacitor and non-capacitor images. The operations regarding the testing starts from the feature extraction to the classification task and this test images are a separate set of data from the training examples.

Scene	Capacitors	Non Capacitors	FE Accuracy	TP	TN	Overall Accuracy
1	6	4	1	4	2	0.6
2	3	0	1	2	1	1
3	0	6	1	0	6	1
4	12	13	0.36	6	3	1
5	25	3	0.57	16	0	1
6	7	8	0.8	4	7	0.916667
7	9	0	1	1	3	0.444444
8	13	5	0.83	11	0	0.733333
9	1	6	0.57	1	3	1
10	6	12	0.5	3	5	0.888889
11	8	1	1	4	1	0.555556
12	5	0	1	3	2	1
13	10	0	0.9	8	0	0.888889
14	7	1	1	5	3	1

15	6	0	1	3	0	0.5
16	58	0	1	40	0	0.689655
17	0	13	0.38	0	5	1
18	0	6	0.5	0	2	0.666667
19	10	7	0.59	6	4	1
20	4	3	0.86	3	2	0.833333
21	12	0	0.83	10	0	1
22	24	0	0.67	14	2	1
23	2	1	1	1	2	1
24	18	17	0.6	10	11	1
25	4	0	1	3	0	0.75
26	8	9	1	6	5	0.647059
27	1	0	1	1	0	1
28	6	7	1	4	6	0.769231
29	6	5	1	1	5	0.545455
30	10	15	1	6	10	0.64
31	0	5	0.8	0	2	0.5
32	8	12	1	6	12	0.9
Average	-	-	0.83644			0.82716

Table 2: Final Results

The above test results are from the test data described in section IX (Experimentation). For this particular test data, the FE Accuracy is 83.6443% and the Overall System Performance (OSP) as a function of FE and Classification is said to be 82.7162%.

XI. Conclusion

The modeling and implementation of feature extraction algorithm and two classifiers for object recognition and detection were presented in this paper. The feature extraction algorithm yielded an accuracy of 83.6443%. The classifier yielded an accuracy of 99.33% (upon Cross validation) and 82.7162% (the OSP as it also takes into account the FE accuracy) upon final testing given a cluttered scene. For the selection of the classifier, the accuracies of the models developed were very close. However, the best one chosen from those two models was Model 1, which had 25 neurons in its hidden layer. This is because upon inspecting the accuracies for individual CV piles, the majority of the piles in Model 1 had their accuracies ranging towards 100%. The choice of the best model was not only made according to the accuracy but the selection was made upon going over the following criterions: least cost to attain favorable results (i.e. less number of neurons), processing and execution time and simplicity of the model. Once the model is classified, the MATLAB program also outputs the coordinate of the classified object to be ready for the robotic arm to execute the sorting task.

XII. Recommendation

The major challenge when developing the models was that upon resizing of the images, there's loss in the pixel data as the images were in raster formats. The potential solution to this predicament would be to invoke the concept of Scalable Vector Graphics (SVG) formatting of the images. However, since MATLAB is not able to process the vector graphics file, the first proposition would be to write a function file which enables MATLAB to read and modify the .svg/ or any other vector files. Having formed this foundation would solve many problems in terms of scaling the images.

References

[1] T. P. Cabre, M. T. Cairol, D. F. Calafell, M. T. Ribes, and J. P. Roca, "Project-Based Learning Example: Controlling an Educational Robotic Arm With Computer Vision," *Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de,* vol. 8, pp. 135-142, 2013.

[2] P. J. Sanz, R. Marin, and J. S. Sánchez, "Including efficient object recognition capabilities in online robots: from a statistical to a Neural-network classifier," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 35, pp. 87-96, 2005.

[3] V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani, "Visual grasp planning for unknown objects using a multifingered robotic hand," Mechatronics, IEEE/ASME Transactions on, vol. 18, pp. 1050-1059, 2013.

[4] A. C. Bernal and G. M. Aguilar, "Vision System via USB for Object Recognition and Manipulation with Scorbot-ER 4U," International Journal of Computer Applications, vol. 56, 2012.

[5] Farahmand, Fazel, Mahsa T. Pourazad, and Zahra Moussavi. "An intelligent assistive robotic manipulator." In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pp. 5028-5031. IEEE, 2006.

[6] A. Ellgammal, "Object Detection and Recognition" *Spring 2005*, Rugters University, Dept of Computer Science.

[7] Schindler, Konrad, and David Suter. "Object detection by global contour shape." *Pattern Recognition* 41, no. 12 (2008): 3736-3748.

[8] P. J. Sanz, R. Marin, and J. S. Sánchez, "Including efficient object recognition capabilities in online robots: from a statistical to a Neural-network classifier," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 35, pp. 87-96, 2005.

[9] *User Manual by Intelitek: Catalog #100342 Rev. E*, SCORBASE version 4.9 and higher for SCORBOT-ER 4u, SCORBOT-ER 2u, 2006.