**Engineering and Technology Journal**
**Journal homepage:** engtechjournal.org

# Pose Estimation of Objects Using Digital Image Processing for Pick-and-Place Applications of Robotic Arms

**Firas S. Hameed [a*], Hasan M. Alwan [b], Qasim A. Ateia [c]**

[a] University of Technology, Assina'a Street, Baghdad, Iraq, firassubhyhameed@gmail.com

[b] University of Technology, Assina'a Street, Baghdad, Iraq, 20071@uotechnology.edu.iq

[c] University of Technology, Assina'a Street, Baghdad, Iraq, 20044@uotechnology.edu.iq

*Corresponding author.

Submitted: 17/08/2019 Accepted: 10/02/2020 Published: 25/05/2020

K E Y W O R D S

Robot Vision,
Pose Estimation,
Image Processing,
Edge Detection,
Eigenvector Line Fitting,
Scatter Matrix,

A B S T R A C T

*Robot Vision is one of the most important applications in Image processing. Visual interaction with the environment is a much better way for the robot to gather information and react more intelligently to the variations of the parameters in that environment. A common example of an application that depends on robot vision is that of Pick-And-Place objects by a robotic arm. This work presents a method for identifying an object in a scene and determines its orientation. The method presented enables the robot to choose the best-suited pair of points on the object at which the two-finger gripper can successfully pick the object. The scene is taken by a camera attached to the arm's end effector which gives 2D images for analysis. The edge detection operation was used to extract a 2D edge image for all the objects in the scene to reduce the time needed for processing. The methods proposed showed accurate object identification which enabled the robotic to successfully identify and pick an object of interest in the scene.*

**How to cite this article**: F. S. Hameed, H. M. Alwan, and Q. A. Ateia, "Pose estimation of objects using digital image processing for pick-and-place applications of robotic arms," Engineering and Technology Journal, Vol. 38, Part A, No. 05, pp. 707-718, 2020.

DOI: https://doi.org/10.30684/etj.v38i5A.518

## 1. Introduction

Robot vision is the field of image processing that enables a robotic system to visualize the environment in which it is working. Having this visual ability gives the robotic system more advantages related to performance and task accomplishment. Welding, polishing, assembling and many other applications a robotic system can do better when it has the ability to visualize the region of interest. One of the most commonly known tasks is that of pick-and-place of objects. The present work aims to develop an efficient picking and placing algorithm for an object regardless of its shape

and size (within the gripper stroke length). The gripper is a two-finger type and the algorithm of picking and placing can be assigned to the *RBX1* 3D Printed Robotic Arm shown in Figure 1.

**Figure 1: RBX1 Robotic Arm**

The proposed algorithm consists of a sequence of steps that lead to successful task accomplishment. When a robotic arm is ordered to pick an object of interest in a scene it must:
1- Determine the pose of each object in the scene so that it can rotate the end effector to the correct angle once the object of interest is detected.
2- Determine the geometry for all the objects in the scene and compare each of them with the geometry of the required object to *recognize* that object.
3- Decide where the suitable gripping points are on the circumference of the object and make sure that the distance between these points is within the size of the gripper opening.
The only way through which the robotic system can make the right decision regarding the above-mentioned points is the use of image processing techniques.

## 2. Related Work

As mentioned earlier, Pick-and-place is a common task assigned to robotic arms, which made many researchers work on finding the best possible approaches and methods to make the robotic arm successfully accomplish the task. Fernando Casado et al. [1] used a template matching method to detect the orientation and size of an object in a scene. Different versions of the same template (in scale and angle) were used to adopt the different possibilities of the existence of the object in the scene and the best match among them are the best chance that a specific object exists. With two depth cameras, Yu-Kai Chen et al. [2] captured a 3D scene and proposed a CAD-based multi-view pose estimation algorithm. Andy Zeng et al. [3] classified the expected objects into categories and compared them with reference images to decide the type and pose of the object to be grasped. Aitor Aldoma et al. [4] adopted a matching method between the scene and a set of 3D models by detecting a set of key points on the object and compare them with those in the model. Colin Rennie et al. [5] acquires the image of an object by an RGB-D sensor and compare the scene with a huge dataset that contains thousands of images, each accompanied with the corresponding information about all of its possible 3D poses. Max Shcwarz et al. [6] tried artificial neural networks to accomplish identifying and pose estimation of an object. They used a large number of images as their training set and the decision is made after passing an object image as an input to this neural network. Kai-Tai Song et al. [7] used a 3D Model database as a reference and an RGBD camera to capture the point cloud of the objects and a pose estimation matching algorithm. Yu Xiang et al. [8] used a Convolutional Neural Network for a 6D object pose estimation. They trained the CNN by a dataset that provided them with accurate 6D poses of 21 objects observed in 92 videos each with 133,827 frames. There are many other researchers worked in this field each with their own techniques and approaches. Some of them are included in the list of references at the end of this paper-like Le Duc Hanh and Le Minh Duc [10], Wu et al. [11] and Myint et al. [12].

## 3. Description of the Algorithm

The algorithm proposed works in a sequence of steps:
*I. Object Isolation.*
*II. Object Geometry Extraction.*
*III. Object Pose Estimation.*
*IV. Object Identification.*

*V. Suitable Picking Points Selection.*

The algorithm was applied on the 12 cm by 12 cm test scene shown in Figure 2.

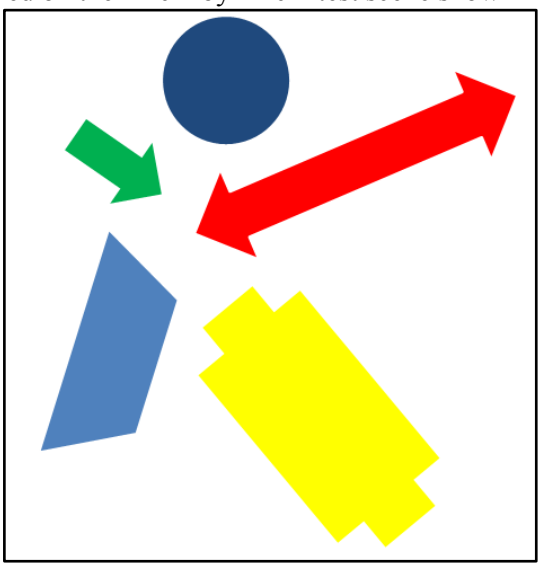

**Figure 2: Set of Objects in a Sample Image**

The steps of the algorithm are discussed in details below:

*I. Isolation of objects in the scene*

Before applying the object identification algorithm, it is important to isolate each object in the scene from the others. This is done by turning the original image into a multi-layer array each layer contains the pixels of only one object. This allows each object to be processed alone.

The first step in the object isolation process is subtracting the background from the scene. The background is well known in advance and it can be either a pattern or just a simple uniform color distribution. This process turns the original colored image into a binary one in which all of the pixels have either value of "0" (background black pixels) or the value of "1" (white pixels for the objects). Figure 3 shows the binary image version of the original image.

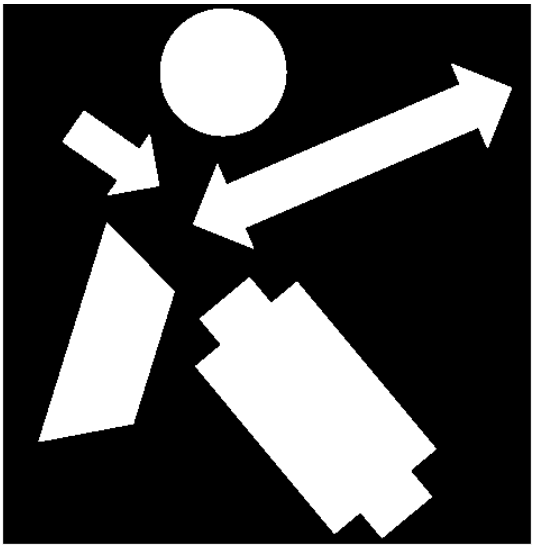

**Figure 3: The Binary Version of the Original Image**

The binary image that resulted from the previous process (Figure 3) is composed of a number of connected regions each region is an object. The isolation process of these objects is accomplished by applying the *Region Growing* technique on the binary image. This technique will assign the pixels that belong to a certain connected region a specific mark, which is different for each of the objects in the original binary image, and saves each object in a separate layer. After applying the region growing technique the binary image will be converted into a multi-layer array of numbers each layer

contains a set of "zeros" (as a background) and another set of "ones" (representing the isolated object). Figure 4 shows five sub-images for five isolated objects.

**Figure 4: Sub Images for Five Isolated Objects.**

*II. Extracting the geometry of an object*

As mentioned earlier, the object is represented by its circumferential edge points, so, the edge points of each object in Figure 4 must be extracted. The edges are extracted by applying the *Gradient Operator* on each layer. For an image function $f$, the gradient operator is defined by the equation:

$$G = \frac{\partial f}{\partial x}\boldsymbol{i} + \frac{\partial f}{\partial y}\boldsymbol{j} \quad (1)$$

If the value of $|G|$ (which is the length of the gradient vector of Eq. (1)) exceeds some threshold $(t)$ at a pixel $(x, y)$, then that pixel will be considered as an edge pixel. $|G|$ is determined by the equation:

$$|G| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad (2)$$

The edge image of Figure 4 is shown in Figure 5.

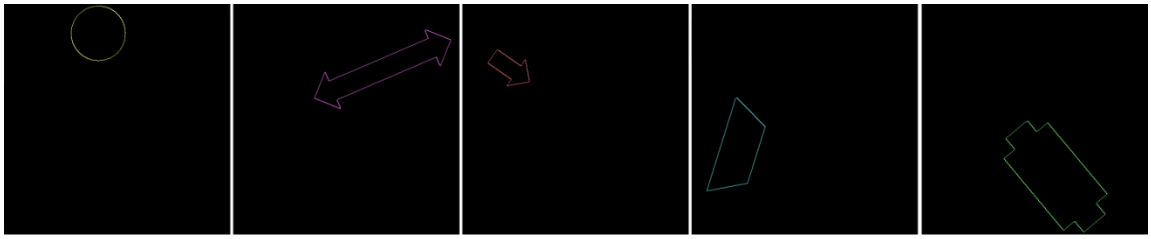

**Figure 5: Edge Image for all the objects**

The geometry of an object is determined by sorting the set of the object's edge points based on the angle of each point with respect to the center (mean) point of the set. As shown in Figure 6, an object edge point $(x_i, y_i)$ is located $\theta_i$ degrees with respect to the mean point $(x_c, y_c)$ at a distance $r_i$ away from it.

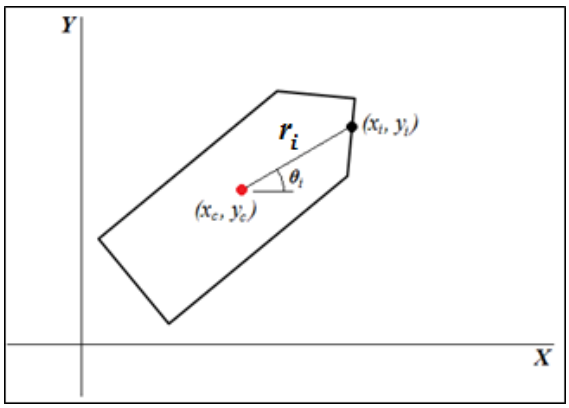


**Figure 6: Location of a general edge point $(x_i, y_i)$ w.r.t. $(x_c, y_c)$**

The value of $\theta_i$ is taken as an identity for its corresponding point $(x_i, y_i)$ and the $n$ points are sorted according to this identity starting from $\theta_i = 0^o$ to $\theta_i = 360^o$ around $(x_c, y_c)$. The values of $\theta_i$ and $r_i$ can be determined by the equations:

$$\theta_i = tan^{-1}\left(\frac{y_i - y_c}{x_i - x_c}\right) \quad (3)$$

$$r_i = \sqrt{(y_i - y_c)^2 + (x_i - x_c)^2} \quad (4)$$

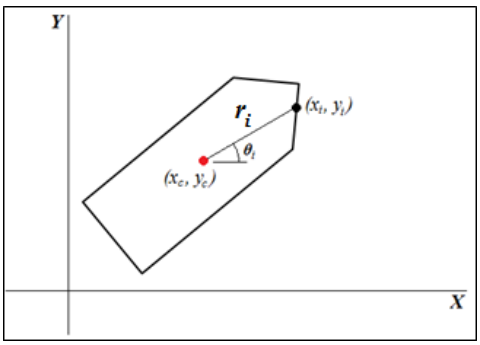


**Figure 6: Location of a general edge point $(x_i, y_i)$ w.r.t. $(x_c, y_c)$**

*III. Determination of object orientation (Pose) in the scene:*

In this paper the objects are classified into two categories based on the type of their symmetry:
1- Elongated object, for which the points that represent its edges have directional distribution, i.e. they are distributed (symmetrically or not symmetrically) about some axis (or a line) in the 2D plane. Examples of such objects include rectangles, ellipses, etc.
2- Non-elongated object (or circular symmetric object), for which the points are distributed around a center point within the group without a noticeable elongation in any specific direction in the 2D plane. Examples of such objects include circles, squares, circular polygons, etc. The type of object symmetry decides how the robotic arm manages to pick that object taking into account the size of the end effector gripper.
The distribution of the edge points of an object tells much about the object geometry and the best way to find how these points are distributed is to use a line fitting technique. The most commonly used methods for line fitting is the Minimum Square Error (MSE) method. The idea of MSE is to find a line that passes through the given set of points with coordinates $(x, y)$ which minimizes the sum of the squares of the distances in the $y$-axis direction from each point to that line. This is shown in Figure 7.

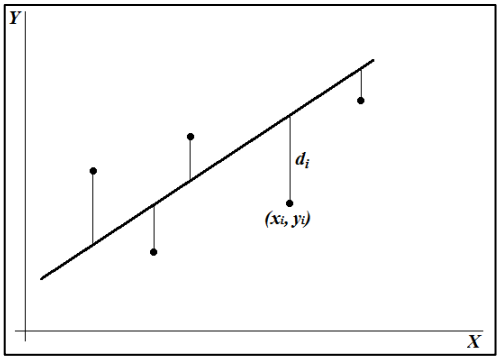


**Figure 7: Definition of the MSE line fitting.**

Unfortunately, this definition is not generally because in some cases MSE gives a wrong estimation of the direction of the point set distribution like the case shown in Figure 8. A better and more efficient method to find the best line fitting for a set of points is known as the Eigenvector Line Fitted [9]. It is based on characterizing the line passing through the mean point of the set in the 2D plane using the definition of $d_i$. This is clearly shown in Figure 9.

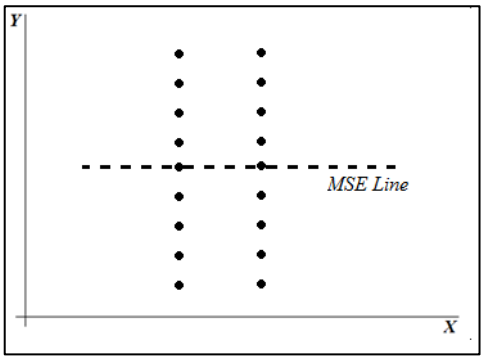


**Figure 8: Wrong geometry decision using MSE line fitting method.**

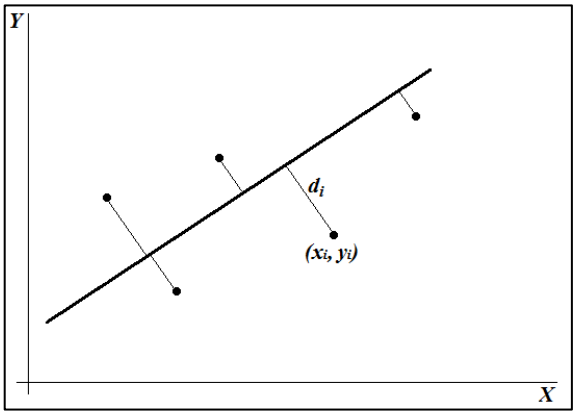


**Figure 9: Eigenvector fitting definition for the same set of points of Figure 1.**

The best-fitting line shown in Figure 9 is described by the standard form of a line in the $xy$-plane:
$ax + by + c = 0$ (5)
The perpendicular distance $d_i$ between the line and any point $(x_i, y_i)$ in the $xy$-plane is given by the relation:
$d_i = \frac{|ax_i+by_i+c|}{\sqrt{a^2+b^2}}$ (6)
As Eq. (6) shows, the expression for $d_i$ implicitly indicates features of the line itself, which means that the best line that fits a set of points is dependent more on the distribution of the points than the axes about which these points are distributed. The problem of determining the Eigenvector line fitting can be stated as follows:
Let's assume that we have a set of points$\{(x_i, y_i)\}, i = 1\ to\ n$, with $zero\ mean$ and we want to fit a straight line through them. Each point in the set can be described as a vector $(\boldsymbol{v_i})$. Figure 10 shows how each point is represented in the $xy$-plane.

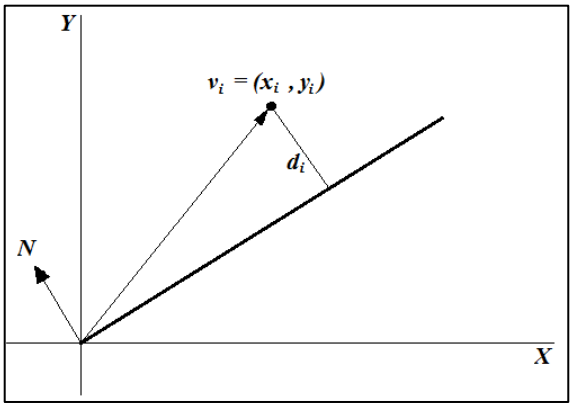


**Figure 10: Vector representation of a point in the xy-plane**

If the straight line is characterized by its normal $\boldsymbol{N}$ then the perpendicular distance $d_i$ will equal to the dot product between $\boldsymbol{N}$ and the vector $\boldsymbol{v}_i$ which can be expressed as:
$d_i^2(\boldsymbol{N}) = (\boldsymbol{N} \cdot \boldsymbol{v}_i)^2 = (\boldsymbol{N}^T\, \boldsymbol{v}_i)^2$ (7)
Where $\boldsymbol{v}_i = [x_i \quad y_i]^T$

For the whole set of points, Eq. (7) can be rewritten as

$$d^2(\boldsymbol{N}) = \sum_{i=1}^{n} d_i^2(\boldsymbol{N})$$
$$= \sum_{i=1}^{n} (\boldsymbol{N}^T\, \boldsymbol{v}_i)^2$$
$$= \sum_{i=1}^{n} (\boldsymbol{N}^T\, \boldsymbol{v}_i)(\, \boldsymbol{v}_i^T\, \boldsymbol{N}\,)$$
$$= \boldsymbol{N}^T \sum_{i=1}^{n} (\boldsymbol{v}_i\, \boldsymbol{v}_i^T)\, \boldsymbol{N}$$
$$= \boldsymbol{N}^T\, S\, \boldsymbol{N}$$

Where,

$$S = \sum_{i=1}^{n} (\boldsymbol{v}_i\, \boldsymbol{v}_i^T)$$
$$= \sum_{i=1}^{n} \begin{Bmatrix} x_i \\ y_i \end{Bmatrix} [x_i \quad y_i]$$
$$= \sum_{i=1}^{n} \begin{bmatrix} x_i^2 & x_i y_i \\ x_i y_i & y_i^2 \end{bmatrix} \quad (8)$$

The Scatter Matrix $S$ is a symmetric matrix and its Eigenvectors are orthogonal to each other. The straight line, that is chosen to be a best-fitting line for the given set of points, is that in the direction of the principal eigenvector which corresponds to the largest eigenvalue of $S$. The smallest eigenvalue corresponds to the eigenvector in the direction of the unit vector $\boldsymbol{N}$.

To summarize, the best Eigenvector line fitting for an $n$-set of points can be found by following these steps:

1- Determine the center point $(\bar{x}, \bar{y})$of the set, where

$$\bar{x} = (\sum_{i=1}^{n} x_i)/n$$
$$\bar{y} = (\sum_{i=1}^{n} y_i)/n$$

2- Normalize the set by subtracting $(\bar{x}, \bar{y})$ from each point in the set.

3- Find the principal Eigenvector of the scatter matrix $S$.

4- The required fitting line is the one that passes through $(\bar{x}, \bar{y})$ in the direction of the principal Eigenvector.

The above procedure helps the robotic system to know exactly how the object is directed in the scene.

*VI. Object identification*

For the robot to pick a specific object it must verify its existence in the scene first. If the object of interest doesn't exist then the robotic arm should retreat to its home position reporting an error indicating the nonexistence of the object. In the geometry extraction step, the edge points of an object were sorted according to the criterion mentioned. Let $r_i$ be the distance from an edge pixel $(x_i, y_i)$ to the center point of the set $(x_c, y_c)$. $r_i$ is given by Eq. (4) above.

If the sorted set of edge pixels is represented by an $r - \theta$ profile then this profile will be an identity profile for that object which is unique for each object. Each object in the scene can be placed at random orientation and for each orientation, there is a different $r - \theta$ profile even for the same object. So, to identify an object *at any orientation* among others, the whole set of its edge points must be rotated to a common reference orientation. Let this reference orientation be the horizontal line parallel to the $x$-axis. Comparing the identity profile of an object with a set of identity profiles for many objects stored in a database enables the robot to identify and pick the correct object and neglect the others in the same scene. To clarify the idea, let's take the rectangular shape in Figure 11 which is rotated by $50^o$ with the horizontal. The variation of its perimeter at this orientation gives the profile shown in Figure 12.

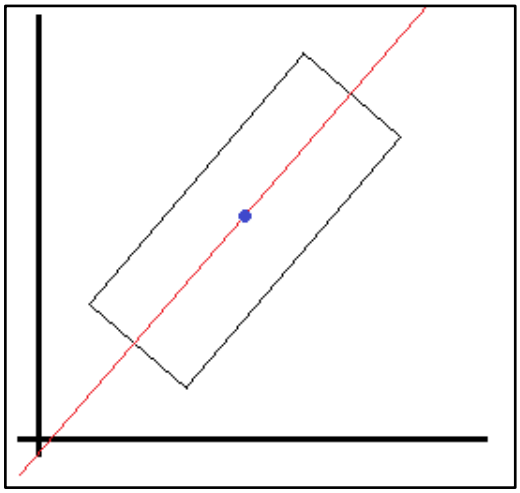

**Figure 11: Oriented rectangular object.**

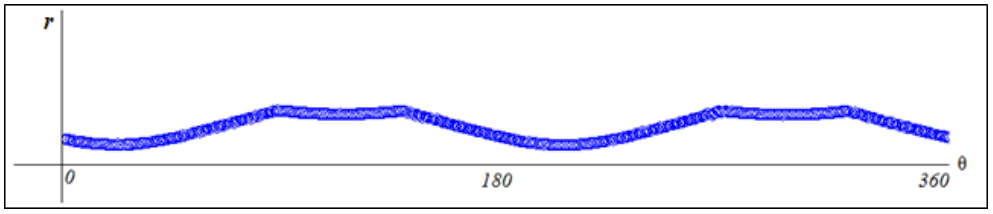


**Figure 12: Variation of the perimeter of the rectangular object of Figure 5 w.r.t. θ.**

For every orientation of an object in the scene, there is a different perimeter profile which makes the process of object identification a very hard (if not impossible). Here comes the advantage of rotating the object to a common reference orientation, the perimeter profile will be unique for every object which makes it easy for the robot to identify the target object among many others in the scene. Figure 13 shows the perimeter profile for the same object in Figure 5 after rotating it to become parallel to the $x$-axis.

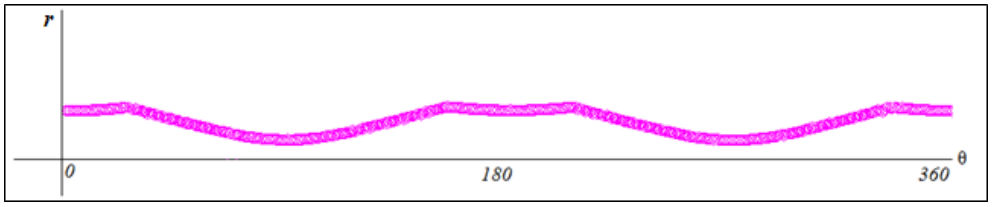


**Figure 13: Variation of the perimeter of the rectangular object of Figure 11 after been rotated to the reference angle.**

*V. Gripping the object*

Once the object of interest is identified, the robotic arm should start picking the object by adapting a selection criterion of the points on the object circumference that are most suitable for gripping. For a successful gripping, we seek a location on the object that needs minimum gripper opening to ensure that the distance determined for gripping the object is within the size of the gripper. The resulting sorted set will also include the distance $r_i$ defined in Eq. (4). The criterion is based on the fact that the gripper has two fingers $180^o$apart, hence, we will calculate the distance $D_i$ (given by Eq. (9) below) measured between the points lying on the opposite sides of the circumference and seek for the minimum value of $D_i$. The selected gripping points will be those corresponding to the minimum value of $D_i$.

$D_i = (r)_{\theta_i} + (r)_{\pi+\theta_i} : i = 1 \; to \; n/2$ (9)

## 4. Worked Example

Let's apply the above method for the set of points of an object extracted from an image after applying an edge detection filter. The resulting set of points represents the circumference (or the outer edge) of an object in front of the end effector of a robotic arm. See Figure 14.

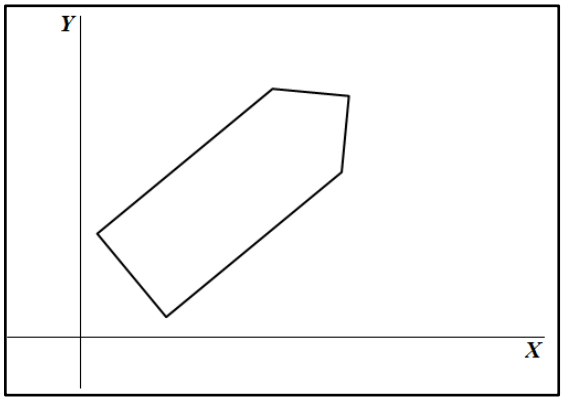


**Figure 14: Edge points of an object.**

- ***Determining the orientation of the object:***

Applying the Eigenvector Line Fitting for the given set of points, the following data were extracted:
1- The number of the points $(x_i, y_i)$ $(i = 1\ to\ n)$: $n = 918$ points.
2- The center point $(x_c, y_c)$ in pixels: $(157, 145)$
3- The calculated orientation: $40^o$
4- The equation of the axial line (in pixels): $y = 0.84x + 13.26$
The results are shown in Figure 15.

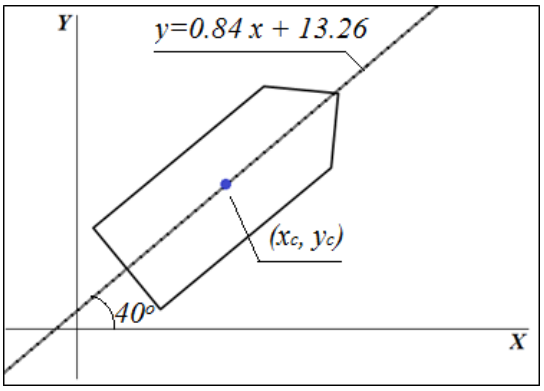


**Figure 15: Object orientation determination data.**

- ***Selection of the gripping points***

The object orientation estimation procedure helps the robotic system to know exactly how the object is directed in the scene. However, for the robot to pick the object, the object set of points need further processing. First, the set must be rotated around its center to the selected reference orientation, which is parallel to the $x$-axis; the object shown in Figure 15 must be rotated $40^o$ clockwise. This is shown in Figure 16.

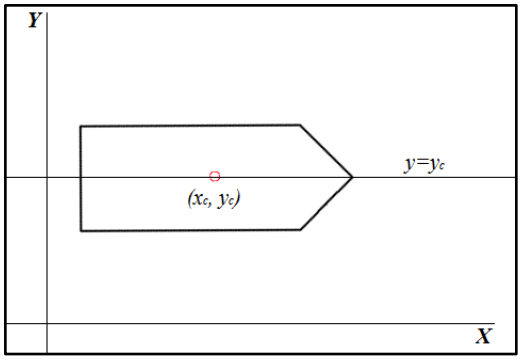


**Figure 16: Object set of edge points rotated to the reference orientation.**

Second, all the points in the referenced set will be sorted according to the angle that each point makes with respect to the center point starting from $\theta_i = 0^o$ and ending with $\theta_i = 360^o$. Figure 17 shows the variation of $r_i$ as a function of point number $n$.

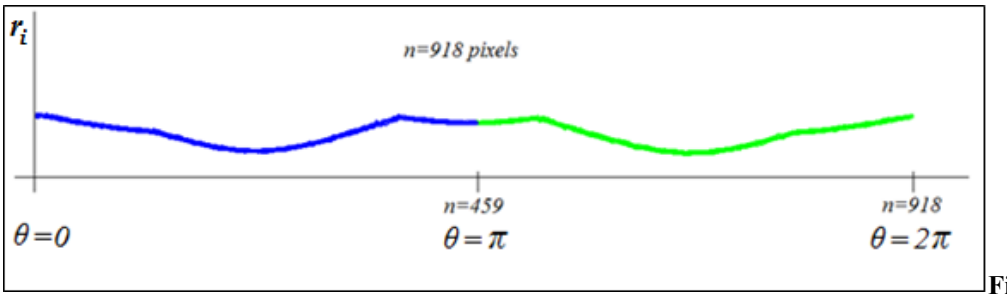


**Figure 17: Variation of $r_i$ w.r.t. point number n**

Using Eq. (11) to measure $D_i$ which is the distance between the points on the opposite sides of the circumference, we get the profile shown in Figure 18 which relates $\theta_i$ with $D_i$.
Figure 18 shows clearly that the minimum value for $D_i$ corresponds to $\theta = \pi/2$. This means that the best points for gripping this object are located $\pi/2$ with respect to the axis of symmetry. These two points are shown in Figure 19.

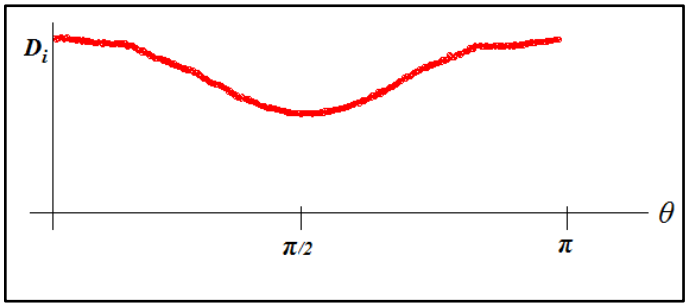


**Figure 18: Variation of $D_i$ w.r.t. $\theta_i$**

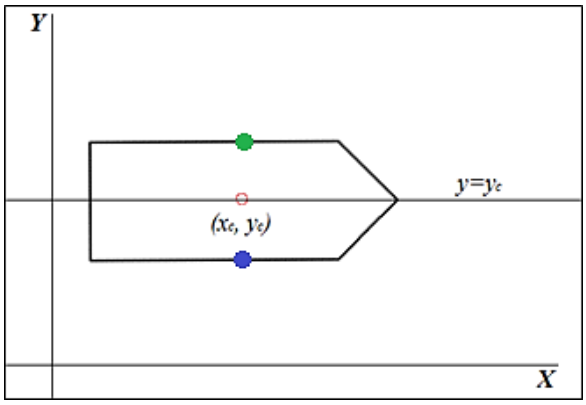


**Figure 19: The location of the best two points suitable for gripping this specific object.**

Applying the proposed algorithm on the test scene of Figure 2 enabled the robotic arm to calculate the orientation and locate the best gripping points for each object in the scene. The resulting image after applying the set of digital image processes described above will turn the original scene shown in Figure 2 into that shown in Figure 20. The number near each of the objects in Figure 20 represents the order of detection of that object and these numbers are not part of the image. The software will also prepare a detailed report about each object telling the location of its center of gravity, the best picking points, the gripper opening and the required orientation of the gripper to pick the object. The report format is shown in Figure 21.

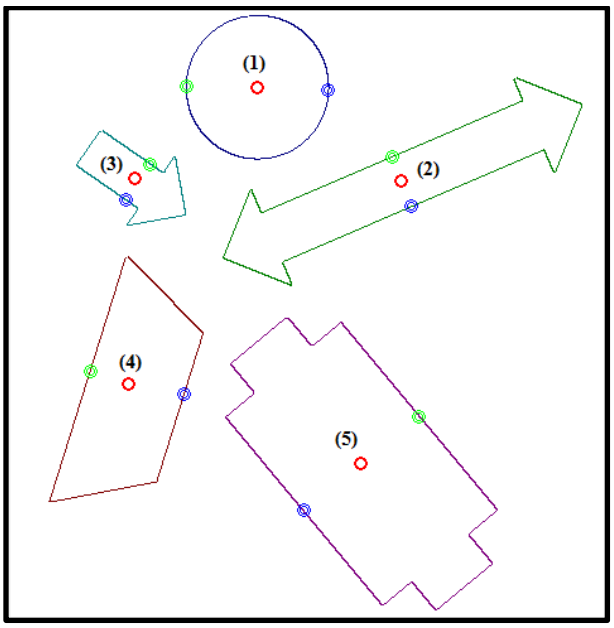


**Figure 20: The processed version of the original image of Figure 2.**

```
Data for object #1
Centre at 4.66 cm, 1.46 cm
Gripper opening = 2.68 cm
Gripper orientation: 1.53 degrees w.r.t. robotic arm horizontal reference axis.

Data for object #2
Centre at 7.38 cm, 3.22 cm
Gripper opening = 1 cm
Gripper orientation: 68.96 degrees w.r.t. robotic arm horizontal reference axis.

Data for object #3
Centre at 2.34 cm, 3.18 cm
Gripper opening = 0.81 cm
Gripper orientation: -55.84 degrees w.r.t. robotic arm horizontal reference axis.

Data for object #4
Centre at 2.22 cm, 7.07 cm
Gripper opening = 1.82 cm
Gripper orientation: 13.67 degrees w.r.t. robotic arm horizontal reference axis.

Data for object #5
Centre at 6.62 cm, 8.58 cm
Gripper opening = 2.8 cm
Gripper orientation: -39.12 degrees w.r.t. robotic arm horizontal reference axis.
```

**Figure 21: The Algorithm Output Report Format. Positive Orientation is in CW Direction**

## 5. Conclusions

Object handling is one of the most common applications in robotics. For a robot to perform the picking task successfully it needs to know, in advance, specific information about the object to be handled. First, the robot must have complete knowledge about the identity of the object so that it can recognize it; this information enables the robot to selectively pick the object among others that exist in the scene. Second, the object's dimensions and orientation in the scene are required by the robot for the complete fulfillment of the picking task. The robot acquires all the needed information by using digital image processing techniques which allow the robot to interact with its environment visually. In this paper, an algorithm for object identification and pose determination is discussed for object picking and placing an application. The algorithm showed good object identification and estimation of the orientation of the object as well as an acceptable selection of the best gripping points on the object's circumference.

The algorithm is general and can easily be modified for any gripper type. Also, a motor current sensor can be added to measure the gripper's motor current as an indication of the gripping force applied to the object.

## References

**[1]** F. Casado, Y. L. lapido, D. P. Losada, and A. Santana-Alonso, "Pose estimation and object tracking using 2D images," The 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, Modena, Italy, 27-30 June 2017.

**[2]** Y-K. Chen, G-J. Sun, H-Y. Lin, and S-L. Chen, "Random bin picking with multi-view image acquisition and cad-based pose estimation," IEEE International Conference on Systems, Man, and Cybernetics, 2018.

**[3]** A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, and others, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," IEEE International Conference on Robotics and Automation (ICRA), May 21-25, 2018, Brisbane, Australia.

**[4]** A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypothesis verification framework for 3D object recognition in clutter," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

**[5]** C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place," IEEE Robotics and automation letters, Pre-Print Version, 2016.

**[6]** M. Schwarz, A. Milan, C. Lenz, A. Mu˜noz, and others, "NimbRo picking: versatile part handling for warehouse automation," IEEE International Conference on Robotics and Automation (ICRA), May 29 - June 3, 2017, Singapore.

**[7]** K.-T. Song, C.-H. Wu, and S.-Y. Jiang, "CAD-based pose estimation design for random bin picking using a RGB-D camera," Springer Science+Business Media Dordrecht, 2017.

**[8]** Y. Xiang, T. Schmidt, V. Narayanan and D. Fox, "Pose CNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," Siemens and NSF STTR grant 63-5197, Lula Robotics, 2018.

**[9]** R. O. Duda and P. E. Hart, "Pattern classification and scene analysis," John Wiley & Sons Inc, 1973.

**[10]** Le Duc Hanh and Le Minh Duc, "Planar object recognition for bin picking application". The 5th NAFOSTED Conference on Information and Computer Science (NICS), 2018.

**[11]** P-C. Wu, H.-Y. Tseng, M.-H. Yang, and S.-Y. Chien, "Direct pose estimation for planar objects," Computer Vision and Image Understanding, Elsevier, 2018.

**[12]** K. M. Myint, Z. M. Min Htun, and H. M. Tun, "Position control method for pick and place robot arm for object sorting system," International Journal of Scientific & Technology Research, Vol. 5, Issue 06, June 2016.