POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Master of Science in Automation and Control Engineering
Dipartimento di Elettronica e Informazione

Flexible Model-Based Detection and Localization of A 3D Rotational Symmetric Object

Supervisor:
Prof. Luca Bascetta
Co-supervisor:
Sakcak Basak

Master Thesis by:
Sari Dewi Mutiara Matr. 858818

Academic Year 2016 – 2017

Abstract

Recently, the model-based detection and localization become a popular topic as their application for selecting an object has an important role in the industrial robotic vision field. However, it is a challenging task to make the robot can recognize the pose of the object and pick the best selection object because of its 3D rotational symmetric shape. Therefore, this thesis presents an Outer Line Mode algorithm for the template model generation. The proposed algorithm is used to produce an outer line (border) of a 3D flexible template model by scanning the evaluated line both in vertical and horizontal direction and select the minimum and the maximum point in each evaluated line. A Directional Chamfer Matching (DCM) algorithm computes an integral distance transform of the captured image to find the lowest cost of the distance and orientation mismatch between the chamfer and the captured image with the aim to recognize the object pose. In order to optimize the recognition of the object pose, the least squares projection error is minimized using a Levenberg-Marquardt algorithm together with creating a threshold based on the median value of DCM cost to encounter the missing edge case. Finally, a Gradient and Orientation Score algorithm will determine the best match selection object to be manipulated based on a computation of the local gradient and orientation of the image. The result shows that the average error between the real object pose and the pose found by the computational algorithms with respect to the world frame is 0.46 mm for the translation and 2.14° for the rotation, respectively. While, the highest score of the best match selection between the template model and the captured image reaches 0.67 out of 1. Overall, the proposed algorithm enables a flexible template model to detect and localize any pose of the rotational symmetric object.

Sommario

Recentemente, il rilevamento e la localizzazione basati su modelli sono diventati un argomento molto diffuso, in quanto la loro applicazione per la selezione di un oggetto ha assunto un ruolo importante nel campo della visione della robotica industriale. Tuttavia, rendere il robot in grado di riconoscere la posa dell'oggetto e di scegliere l'oggetto migliore da selezionare è un compito impegnativo a causa della forma simmetrica rotazionale 3D dell'oggetto. Pertanto, in questa tesi verrá presentato un algoritmo del tipo 'Modalitá Linea Esterna' per la generazione del modello. L'algoritmo proposto viene utilizzato per produrre il contorno di un modello flessibile 3D mediante la scansione verticale e orizzontale della linea considerata e permette di selezionare il punto minimo e massimo in ciascuna linea. Un algoritmo Directional Chamfer Matching (DCM) calcola una trasformata integral distance dell'immagine catturata per trovare il costo minimo della distanza e del disallineamento dell'orientamento tra il modello dell'immagine e l'immagine catturata con lo scopo di riconoscere la posa dell'oggetto. Per ottimizzare il riconoscimento della posa dell'oggetto, l'errore di proiezione dei minimi quadrati viene minimizzato utilizzando un algoritmo Levenberg-Marquardt insieme alla creazione di una soglia basata sul valore mediano del costo DCM per trovare il segmento mancante del modello. Infine, un algoritmo Gradient and Orientation Score, basato sul calcolo del gradiente locale e dell'orientamento dell'immagine, determinerà l'oggetto migliore da selezionare. Il risultato mostra che l'errore medio tra la posa reale dell'oggetto e la posa ottenuta dagli algoritmi computazionali rispetto alla tema mondo è, rispettivamente, di 0,46 mm per la traslazione e di 2,14° per la rotazione. Invece, il più alto punteggio relativo al miglior match tra il modello dell'immagine e l'immagine catturata é di 0.67 su 1. Nel complesso, l'algoritmo proposto consente a un modello flessibile di rilevare e localizzare qualsiasi posizione dell'oggetto simmetrico rotazionale.

Acknowladgement

Above all, I thank the Almighty God Who gives His blessings and grace towards me and for Muhammad pbuh who I missed.

I have no words to express my gratitude for my beloved family, Imam Hanafi Noor, Umiana, Vina Wahyuni Eka Putranti, and Rendi Kurniawan Tri Anggara, you are my strength when I am weak. Let's only God and me who know my best prayers for you.

With a great pleasure, I thank Lembaga Pengelola Dana Pendidikan (LPDP) who trust me to carry out this challenging mandate.

I sincerely thank my humble supervisor, Prof. Luca Bascetta, who leads me to a great science and always give remarkable advice. Also, I have been privileged to have a good hearth Basak Sakcak who guide and teach me patiently in every step of my progress. And for all my professors, lecturers, and teachers who are an intercessor of God for me to be free from ignorance. May God blessing you all.

My heartiest thanks to mas ilmi and his family for their sincere prayer and support for me. May God always devote His love for you.

And my warm thanks to Linda, Jun, Amel, Elena, Silvia, Furkan, Harini, and any other my friends and my colleagues in Milan, thanks for being my family here thanks for making Milan alive, I will miss all these fantastic moment, see you on top!. For Learning ROS team, Ima, Nurul, Fivit Ao, Ganesha, Hendra, Oppie, Gita, Tita, Risma and any other my friends and relatives in Indonesia for cheering me up and help me to survive this challenge, I will not forget all your kindness.

- Milan, Autumn 2017 -

Table of Contents

Abstract i

Sommario ii

Acknowladgement iii

Table of Contents iv

Figure List vi

Table List viii

Graph List ix

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Problems 1

1.3 Objective of The Thesis 2

1.4 Result 3

1.5 Organization of The Thesis 4

Chapter 2 3D Template Using Outer Line Mode 6

2.1 Transformation and Projection Matrix 6

2.2 Outer Line Mode 8

2.3 Implementation 9

Chapter 3 Object Image Edge Detection 14

3.1 Line Segment Detector (LSD) 14

3.2 Implementation 18

Chapter 4 Matching 20

4.1 Directional Chamfer Matching 20

4.2 Implementation 24

Chapter 5 Multi View Pose Refinement and Best Match Selection 32

5.1 Minimization Cost 32
5.2 Optimization 33
5.3 Best Match Selection 35
5.4 Implementation 36
Chapter 6 Experiment and Result 42
6.1 Outline Mode Result 42
6.2 Synthetic Camera 46
6.2.1 Single Object 46
6.2.2 Multiple Object 50
6.3 Real Camera 52
6.3.1 Set Up 52
6.3.2 Single Object 55
6.3.3 Multiple Object 57
Conclusion 63
Bibliography 64

Figure List

Figure 1. 3D Sprayer as A Rotational Symmetric Object 2
Figure 2. Block Diagram of The Whole Framework 4
Figure 3. Projection and Transformation Matrix Correlation 6
Figure 4. Scanning Line Coherence 8
Figure 5. The Rotated Original 3D Template in Camera View 9
Figure 6. The Projected Template in Image Frame 10
Figure 7. The Maximum and Minimum Points 11
Figure 8. The Outer Line Candidates 11
Figure 9. Outer Line Mode Result 12
Figure 10. Outline Mode Result in Out of Plane Case 13
Figure 11. 3D Outer Line Template Lines 13
Figure 12. Line Support Region 14
Figure 13. Rectangle of Line Support Regions 15
Figure 14. LSD Algorithm 17
Figure 15. LSD Representation with Different Scale 18
Figure 16. (a) Oriented Chamfer Matching (b) Directional Chamfer Matching 22
Figure 17. Step of Integral Distance Transform Computation 23
Figure 18. Illustration of Integral Distance Image 23
Figure 19. Initial Data Input of Matching Step 24
Figure 20. Angle Direction of Line Data 25
Figure 21. Sorted Data Based on The Length of Lines 25
Figure 22. The 1st Hypotheses Plot 26
Figure 23. Translated Template along The Longest Line 27
Figure 24. The Lowest Cost of Translated Template 27
Figure 25. The 2nd Hypotheses Plot 28
Figure 26. The 3rd Hypotheses Plot 29
Figure 27. Sorted Cost 30
Figure 28. In-Plane Parameters 30
Figure 29. Coarse Pose Hypotheses Plot 31

Figure 30. Plot of Hypotheses Parameters 37
Figure 31.Plot of Hypotheses Parameters in World Frame..... 38
Figure 32. The Cost Value of Each *i-th* Rasterized Template Point 39
Figure 33. Dimension Result..... 39
Figure 34. Pose Refinement Result 40
Figure 35. Plot of Refinement Pose..... 40
Figure 36. 3D Template Lines Data Base 45
Figure 37. Various Pose of In-Plane Case 47
Figure 38. Out of Plane Pose..... 47
Figure 39. Image Synthesis for Multiple Object Case..... 50
Figure 40. Lowest DCM Cost Object 51
Figure 41. The Top Most Object 51
Figure 42. World Frame Setup 53
Figure 43. The Whole System Setup 54
Figure 44. Out of Plane Rotation Setup 54
Figure 45. Translation on x-Axis and y-Axis in-Plane Parameter Test..... 55
Figure 46. Rotation in Plane Test 56
Figure 47. Out of Plane Rotation Test 56
Figure 48. The Topmost Object Selection Test In-Plane Case (1)..... 58
Figure 49. The Visible Edge Line of The Query Image 59
Figure 50. The Topmost Object Selection Test In-Plane Case (2)..... 60
Figure 51. The Topmost Object Selection Test In-Plane Case (3)..... 61
Figure 52. The Topmost Object Selection for Out-of-Plane Case 62

Table List

Table 1. Hypotheses Parameters in Image Frame 36
Table 2. Hypotheses Parameters in a World Frame 37
Table 3. Outer Line Mode Result in The Image Frame 45
Table 4. The Average of The Error Parameters Using Blender Images 48
Table 5. Scoring Result Table 52
Table 6. The Average of Error Parameters Using The Real Camera Images 57
Table 7. The Score of In-Plane Case (1) 58
Table 8. The Score of In-Plane Case (2) 60
Table 9. The Score of In-Plane Case (2) 61
Table 10. The Score for The Out-of-Plane Case 62

Graph List

Graph 1. The Different Between Reference and Experiment Parameters of Translation x-Axis 49
Graph 2. The Different Between Reference and Experiment Parameters of Translation y-Axis 49
Graph 3. The Different Between Reference and Experiment Parameters of Rotation z-Axis 50

Chapter 1

Introduction

1.1 Motivation

In the last decade, the machine vision and model-based detection and localization issues have been widely developed in the industrial field, especially for the robot vision with the task to select the object in the batch, then to pick the best object pose to be manipulated. In this case, the robot should pick the 3D sprayer as the object (illustrated in Figure 1) among the same kind objects.

The limitation cost and the complexity of the object shape to be detected employ the researcher to provide a set of algorithms which can satisfy the requirement using standard industrial mono camera design (in this case *uEye* camera). That is why the algorithm that enables to detect and localize the pose of the object and selecting the topmost object in the batch (the best selection object to be manipulated) should perform in a good result.

1.2 Problems

The 3D sprayer, as the object that will be detected, has some unique characteristics. Regarding the surface, it has a reflectance surface and regarding the shape, it doesn't have a simple planar shape, it has a 3D rotational symmetric shape (illustrated in Figure 1) that is the object has the same shape if it is rotated with respect to its y-axis. With this condition, it can have a high possibility that the object pose is either in-plane or out-of-plane. In this case, a condition in which the camera is located at the origin of the world coordinate frame while aligning the camera optical axis with the z-axis, then the rotation about this axis (θ_z) and the translation in the plane orthogonal to the optical axis (t_x, t_y) are defined as in-plane parameters. Meanwhile, the rotation about the x-axis (θ_x) defined as out-of-plane parameters, as with this kind of rotation in 2D image plane gives a result of a different object contour.

Then this thesis focus on several problems:

- How to generate a flexible 3D template model such that it only stores the outer line of the template in various poses both in-plane and out-of-plane condition?
- How to detect and estimate the 3D pose of the rotational symmetric object?
- How to select the topmost object (the best selection object to be manipulated) among the multiple objects of the same kind?

Therefore, a set of algorithms is proposed to solve the problems, which will be explained in the 'Organization of Thesis' section.

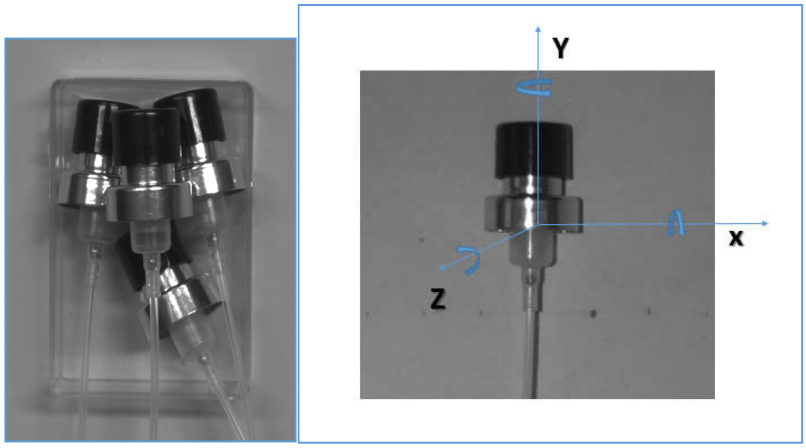


Figure 1. 3D Sprayer as A Rotational Symmetric Object

1.3 Objective of The Thesis

Given the technical drawing of the object (3D sprayer), then the purpose of the thesis were:

- Analyze the 3D template generation as a template model result in various poses

- Analyze the detection of 3D pose estimation result, both using the real camera image and using the synthetic image (generated by *Blender* software) as the acquired image (query image).
- Analyze the best selection object to be manipulated (the topmost object), both using the real camera image and using the synthetic image (generated by *Blender* software) as the acquired image (query image).

1.4 Result

Considering the main purpose defined in the previous section, the results can be expressed as follows,

i. The Outer Line Mode algorithm, adapted from Hidden Line Removal and Surface Removal algorithm. It can build an auto generate the 3D outer line template and easily change the pose of the template database. This condition is a benefit for the type of 3D rotational symmetric object.

ii. The algorithms, based on Directional Chamfer Matching algorithm, an optimization using Levenberg-Marquardt algorithm, and threshoding system based on the median of DCM cost, to detect and estimate the 3D pose of the object both single object and multiple objects cases, that also takes into consideration for missing edge and occlusion.

iii. A set of algorithm, based on Gradient and Orientation algorithm, to select the topmost object.

1.5 Organization of The Thesis

Based on the previous section, the whole framework is illustrated in Figure 2,

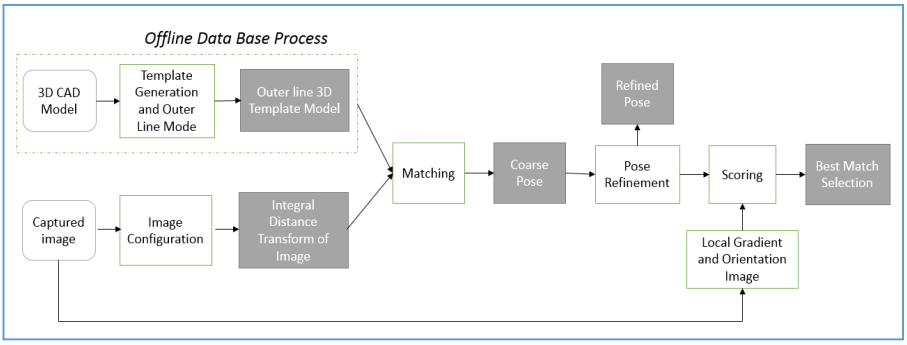


Figure 2. Block Diagram of The Whole Framework

We organize this thesis contents which refer to the framework in Figure 2, the thesis is outlined as follows,

Chapter 2 presents the template generation step which uses an Outer Line Mode method. Starting with the information of 3D sprayer technical drawing, we build a template generation step and using an Outer Line Mode method with the output is the outer line 3D template as a single database template. By using Outer Line Mode method, we can easily generate the various pose of the template database.

It consists of 3 sections. Firstly, regarding the related theoretical and formulation of a transformation and a projection matrix as a basic theory. Secondly, regarding the related theoretical and formulation of an Outer Line Mode algorithm. The last about how we implement the first and second section into the program.

Chapter 3 defines a captured image (query image) configuration that refers to the state of the art of the Line Segment Detector. The output of this step will be the input of matching step.

It consists of 2 sections. Firstly, regarding the related theoretical and formulation of the Line Segment Detector algorithm. Secondly, about how we implement the first section into the program.

Chapter 4 discuss the matching step based on the Directional Chamfer Matching algorithm. In this step, we use the input from the outer line of 3D template model (Chapter 2) and the configured of the captured image, that is the value of integral distance transform of the image (Chapter 3). The output of matching step is the pose of the object in-plane frame or we call it as coarse pose hypotheses.

It consists of 2 sections. Firstly, regarding the related theoretical and formulation of the Directional Chamfer Matching algorithm. Secondly, about how we implement the first section into the program.

Chapter 5 defines the pose refinement step using a Levenberg-Marquardt method to have more precise 3D object pose which also covers the out-of-plane parameter, and selects the topmost object, a Gradient and Orientation Score algorithm is applied.

It consists of 4 sections. First, about the related theoretical and formulation of cost minimization, that is how we define the cost that will be minimized. Second, regarding the related theoretical and formulation of cost minimization that is how we define the cost that will be minimized. Third, about the related theoretical and formulation of the optimization based on a Levenberg-Marquardt method. The last, about the theoretical and the formulation behind the Gradient and Orientation algorithm which is used in best selection object to be manipulated (the topmost object)

Chapter 6 defines the experiments and results for both the synthetic image (using *Blender* software) and the real camera image. Also the single object case to test the detection of object pose algorithms and the multiple objects case to test the algorithm of selecting the topmost object.

Chapter 2

3D Template Using Outer Line Mode

Given the 3D CAD model of the template, then the main challenge is how to have only the outer line of the object shape in any camera view, because in the next step, in order to have a precise matching object, it is safe for having the outer line of the object template.

2.1 Transformation and Projection Matrix

The basic thing when dealing with the camera is regarding the transformation and projection matrix[i]. It will help us to know the object in the world frame, the camera frame, and the image frame. The correlation is summarized in Figure 3.

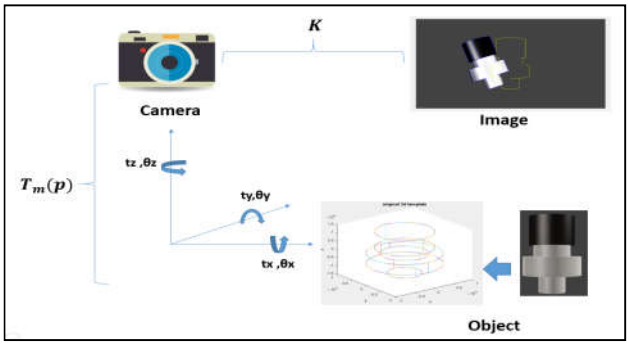


Figure 3. Projection and Transformation Matrix Correlation

Given n set 3D points of CAD model defined as $\widetilde{U} = \{\widetilde{u_i}\}_{i=1}^{n}$, then given the pose of the object with respect to camera frame $p = [t_x\ t_y\ t_z\ \theta_x\ \theta_y\ \theta_z]^T$, the transformation matrix T_m (4 x 4 matrix dimension) is expressed as following,

$$T_m(p) = \begin{bmatrix} R_m & t_m \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

With the rotation matrix of the Euler angle and the transformation matrix are,

$$R_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 \\ sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$t_m = [t_x \quad t_y \quad t_z]'$$

If it will be expressed in an image frame, we use the projection matrix P (3 x 4 dimension matrix) in which we need the intrinsic camera parameter K that we get from camera calibration step using the check board [ii],

$$P = [K \quad 0] * T_m(p) \tag{2.2}$$

Being,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

f_x, f_y, c_x, c_y are focal length and the center points of an image. So the 2D points of the template are expressed as,

$$u_i = P * \widetilde{u_i} \tag{2.3}$$

After we get the template points in an image frame, then the next step is applying an Outer line Mode method. Another basic thing that will be useful in this case is regarding back projection matrix, that is a conversion of 2D points set into 3D points set, it is explained in the following formulation,

$$\widetilde{u_i} = w[KR]^{-1} * u_i - {R_m}^{-1} t_m \tag{2.4}$$

With w is scaling value.

2.2 Outer Line Mode

The aim of outer line mode is storing the outer line of the template in any camera view. That is why the algorithm works in the image frame (2D points) then at the end it will be back-projected in 3D points.

By adopting from the Hidden Line Removal[iii] and Surface Removal algorithm[iv], some points that we will use are: first, about the visibility parts of an object, it depends on the location of the viewing eye, the viewing direction, the type of projection (orthogonal or perspective), and the depth (the distance between various object's faces in the scene and the viewing eye). Second, apply the priority algorithm[v] which can be the depth, the maximum and minimum value, and any other consideration. In our case, we consider the minimal and maximal points of the template by scanning every line segment which contains at least a point (Figure 4).

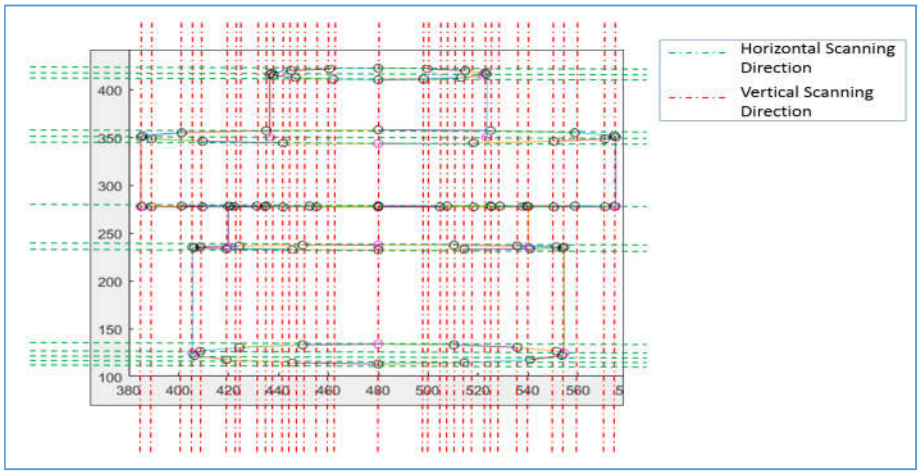


Figure 4. Scanning Line Coherence

To know the value of every point in a line, we use the straight line formulation

$$y = a * x + b \tag{2.5}$$

By given all data of the projected template points, we use the Polynomial method [vi] to get the value of the slope a and the coefficient b as the y-intercept. From there we can know any value of any points as long as along that analyzed line.

$$p(x) = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1} \tag{2.6}$$

With $p(x)$ is polynomial of degree n.

2.3 Implementation

To make it easier to understand how we implement in our system, we summarize into the following steps,

Firstly we consider camera viewing. So, we rotate the object with respect to camera frame (by assuming that the camera perpendicular to the object with a certain height distance).

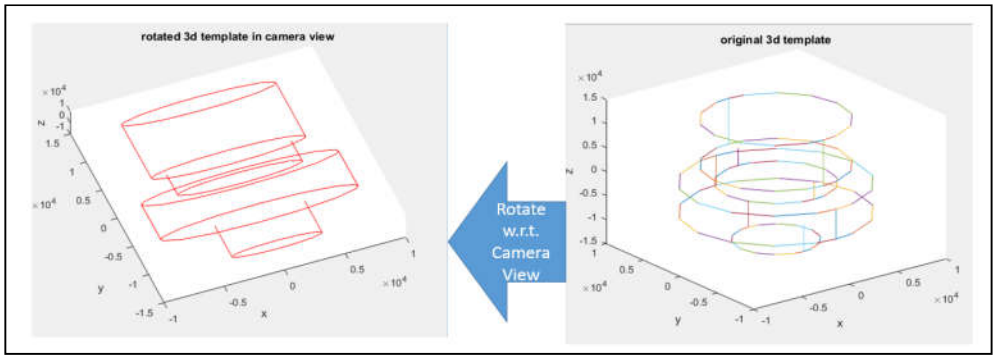


Figure 5. The Rotated Original 3D Template in Camera View

Then we project the rotated 3D template into image frame using formulation (2.3). The result is shown as following figure,

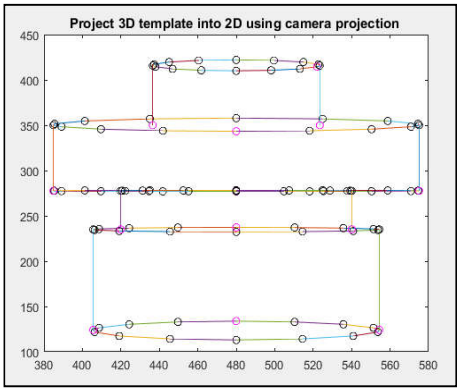


Figure 6. The Projected Template in Image Frame

In this image frame, we can apply the priority value (maximum and minimum points of the template) by scanning every line segment, as illustrated in Figure 4. But before doing so, we create a middle point in every line (to increase the precision when scanning the line by increasing the number of points), so currently, we have not only a start and an endpoint for each line but also a middle point. By applying line equation and *polyfit* function (2.5) and (2.6), we can get the exact coordinate of every point along the line and it will be used in the scanning process. In Figure 6, the total points indicated as a circle.

Regarding the scanning step (refer to Figure 4), we use 2 scanning directions, that is the horizontal scanning direction (with respect to the x axis of the image frame) and the vertical scanning direction (with respect to the y axis of the image frame). Now we can choose the minimum and maximum points in the vertical and the horizontal scanning direction for each line. As the result, shown in Figure 7, the green color indicates the maximum and minimum value in vertical direction meanwhile the blue color indicates the maximum and minimum value in horizontal direction

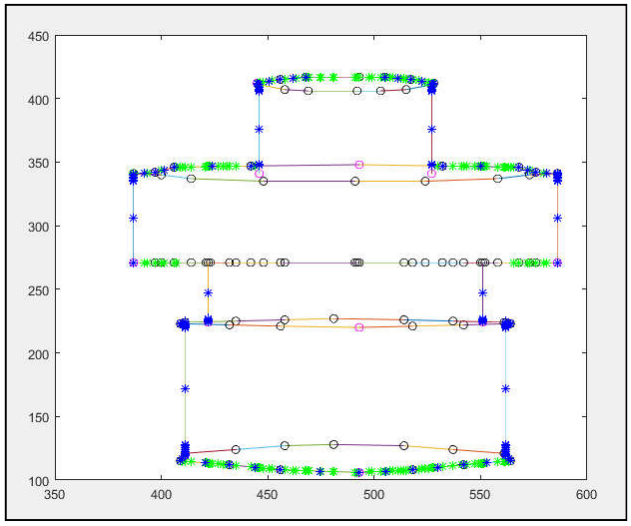


Figure 7. The Maximum and Minimum Points

Now we have the outer line candidates (Figure 8), by storing the initial and the endpoint of the projected template which contains only maximum and minimum points (indicated as green stars and blue stars symbol).

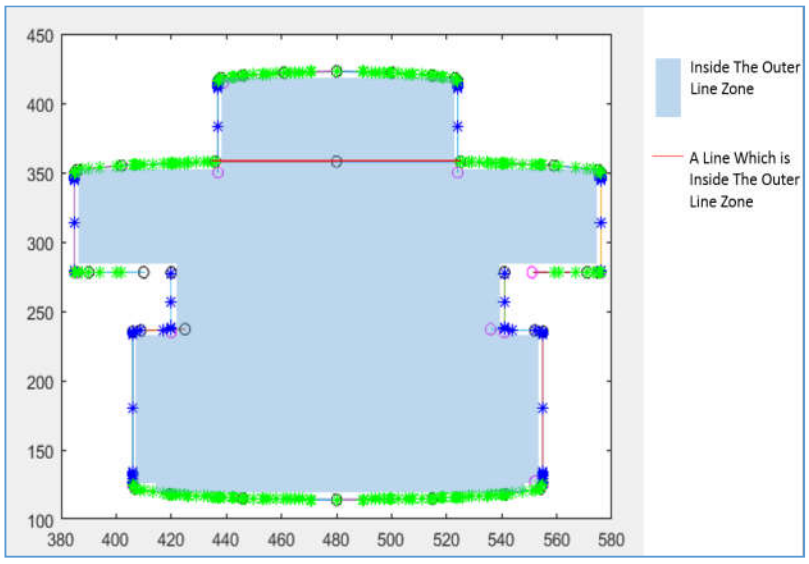


Figure 8. The Outer Line Candidates

The next step is to filter the outer line candidates such that we try to don't have any line inside the outer line zone. Because as shown in Figure 8, there are some lines inside the outer line zone, which are indicated by a red line. We try to decrease this kind of line by ordering the points in every outer line candidate, then eliminate and redefine a new point which belongs to the outer line candidate but inside the outer line zone such that the line is not inside the outer line zone or at least the length of the line which is inside the outer line zone is decreased. So, we will have a new line as the point is changed by using line equation. And finally, we get the final 2D template lines as shown in Figure 9,

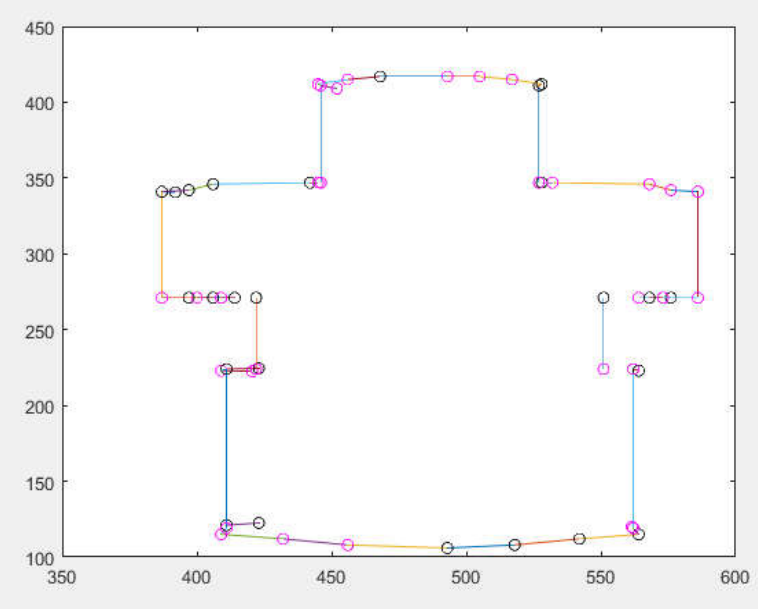


Figure 9. Outer Line Mode Result

Another template pose is shown in Figure 10, which is rotated in the out-of-plane case. The left side is the projected original template in the image frame then the right side is the result of Outer Line Mode method.

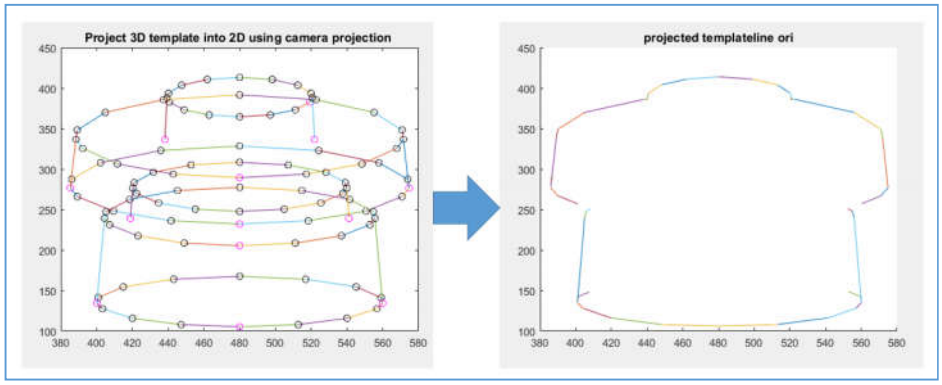


Figure 10. Outline Mode Result in Out of Plane Case

The outer line template mode in image frame should be back-projected in 3D point using formula (2.4). So the final result of template lines generation step is the outer line of 3D template lines as shown in Figure 11,

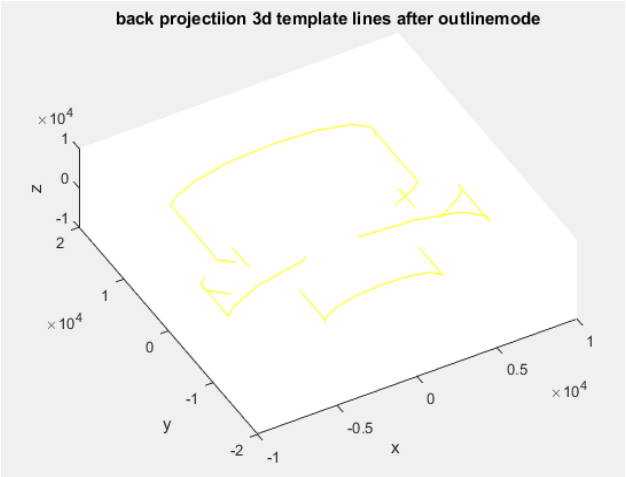


Figure 11. 3D Outer Line Template Lines

Chapter 3

Object Image Edge Detection

In this chapter, we focus on how to manage the captured image which is also called as a query image. We use a state of the art of the Edge Detection algorithm that is a Line Segment Detector that gives subpixel in accurate result. This is the reason why in the template generation step we only use the information of the outer line of the template lines.

3.1 Line Segment Detector (LSD)

LSD[vii] is aimed at detecting locally straight contours on images. The LSD algorithm takes a gray-level image as input and returns a list of detected *line segments*, that is where the gray level is changing dramatically from dark to light or the opposite. The algorithm starts by producing *level line field* that is computing the level line angle at each pixel (mentioned as line support region) which share the same level line. This step is illustrated in Figure 12,

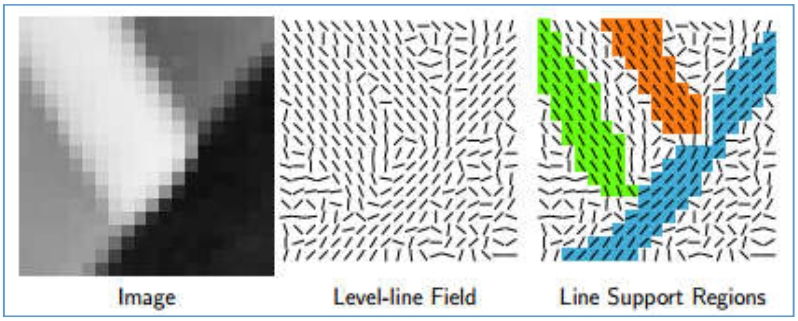


Figure 12. Line Support Region

A *Line segment* is every *line support region*. The corresponding geometrical object (a rectangle in this case) must be associated with it. The principal inertial

axis of the *line support region* is used as main rectangle direction; the size of the rectangle is chosen to cover the full region. It is depicted in the following figure,

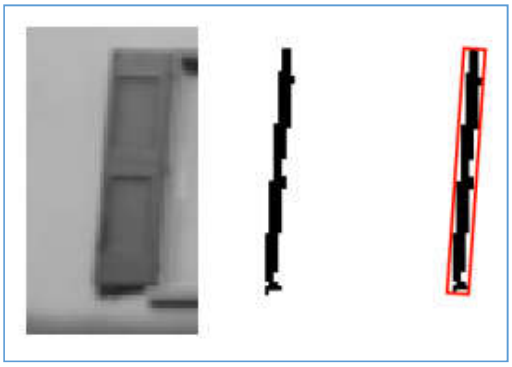


Figure 13. Rectangle of Line Support Regions

The approaches that are used are *a contrario* approach and the Helmholtz principle as the validation step [viii]. The Helmholtz principle is about the absence of noise on an image, meanwhile for *a contrario* approach is defining a noise or a *contrario* model H_o in which the desired structure is not present. So, when the structured events are defined as being rare in the *contrario* model, then the event is validated. We consider the event that a *line segment* in the *contrario* model has as many more aligned points, as in the observed *line segment*. The expected number of events which are as good as the observed one is formulated in (3.1),

$$N_{test} \,.\, P_{H_o}[k(r,I) \geq k(r,i)] = B(n(r), k(r,i), p) \tag{3.1}$$

$$B(n\ k, p) = \sum_{j=k}^{n} \binom{n}{j} p^j\,(1-p)^{n-j}$$

$$p = \text{tolerance}\ \tau/\pi$$

Where the *number of test* N_{test} is the total number of possible rectangles being considered, $k(r,i)$ is the number of *aligned points* of an image *i* and *n(r)* the total number of pixels in a rectangle *r*. P_{H_o} is the probability on the *contrario* model H_o and I is a random image following H_o, and H_o is a noise model for the image

gradient orientation. Meanwhile $B(n,k,p)$ is the tail of binomial distribution and p is the probability and precision that a pixel on the *contrario*. The *contrario* model of H_o used for LSD is defined as a stochastic model which also called as aligned point satisfying the following properties,

- $\{LLA(j)\}_{j\in Pixels}$ is composed of independent random variables and called as level line angle at pixel j
- $\{LLA(j)\}$ is uniformly distributed over span of [0,2π]

The number of tests N_{test} corresponds to the total number of rectangles that could show an alignment at a fixed precision. The rectangles are oriented, which means that the order of starting and ending points is not arbitrary (it encodes which side of the line segment is darker). Then the number of tests is considered in formulation (2.2) with γ is a number of different p values potentially tried

$$(NM)^{5/2}\gamma \tag{3.2}$$

Finally the Number of False Alarms (NFA) associated with a rectangle r on the image i defined as

$$NFA(r,i) = (NM)^{5/2}\gamma.B(n(r),k(r,i),p) \tag{3.3}$$

When *NFA* associated with an image rectangle is large, it means that it is not relevant one. Meanwhile, when NFA value is small, it means that the event is rare and probably the meaningful one. Hence it needs threshold ε such that when $NFA(r,i) < \varepsilon$ it produces detection. The algorithm of LSD is summarized in Figure 14,

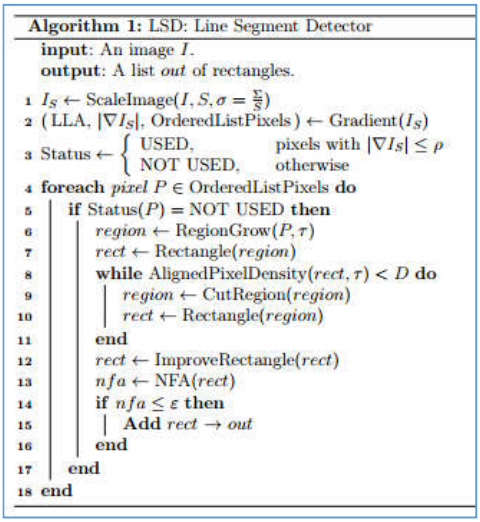


Figure 14. LSD Algorithm

Another information is that LSD was an automatic image analysis tool, it depends on several parameters, they are image scaling, gradient computation, gradient pseudo ordering, gradient threshold, region growing, rectangular approximation, NFA computation, and aligned points density.

- Image Scaling

The LSD algorithm produces a different result when it is applied to the different image because it has the different detail such as a number of tests. As the result of *a contrario* validation step, the detection threshold will automatically adapt to the image size. The scaling helps to cope with aliasing and quantization artifacts (especially the staircase effect). Meanwhile for blurring image, some structured would be detected on a blurred white noise. The scaling was performed by filtering the image using the Gaussian kernel to avoid aliasing and sub-sampled. The standard derivation of Gaussian kernel is computed as,

$$\sigma = \Sigma/S \tag{3.4}$$

In which S is scaling factor and Σ is set to 0.6 which will produce good balance between avoiding an aliasing and an image blurring

3.2 Implementation

As explained before, after having a query image in the image frame, then we applied LSD function to detect line segments on a query image.

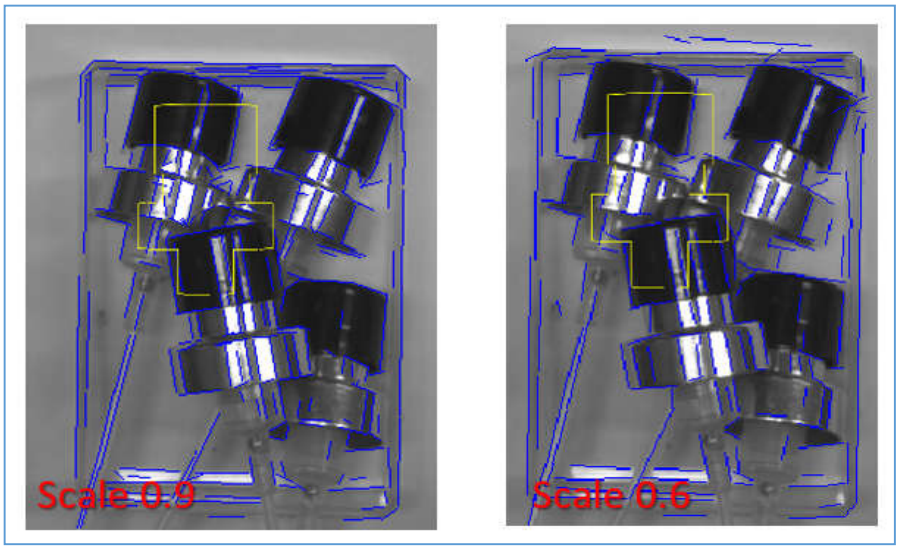


Figure 15. LSD Representation with Different Scale

As shown in Figure 15, the detected line segment is shown by a blue line meanwhile for the original template line is shown by a yellow line. Because the object is having high specularity, we tune the image scaling in order to have a good line representation. As the result, we get the data of line segments and their direction. For the direction, we determine to quantize the angle into 60 orientation channels of span [0,π).

Based on the 2 known points (x_1, y_1, x_2, y_2) from LSD result, to represent the angle orientation of line is formalized below,

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{3.5}$$

$$\tan(\theta) = m \tag{3.6}$$

$$\theta = \tan^{-1}(m)$$

Then we will use these data (start points, end points, and orientations of detected lines) to the next step that is Matching step.

Chapter 4

Matching

In this chapter, we calculate the cost in terms of distance and orientation between template and query image, then we store the one with the lowest cost and the in-plane parameters (translation t_x and t_y and rotation θ_z) will be the coarse pose parameters.

4.1 Directional Chamfer Matching

The common technique to find the best alignment parameter between template edge map and query edge map is using *Chamfer Matching* technique [ix] Let's define two point sets of a template edge map $U = \{u_i\}_{i=1}^{n}$ and a query image edge map $V = \{v_j\}_{j=1}^{|v|}$ with n and $|V|$ are the total number of the edge points for every set. Then the *chamfer distance* [x] between those two point sets which are defined in the average over all pixels $u_i \in U$ and their nearest pixel in V is,

$$d_{CM}(U,V) = \frac{1}{n}\sum_{u_i \in U} \min_{v_j \in V} ||u_i - v_j|| \tag{4.1}$$

The chamfer matching cost can be computed using the distance transform image of each pixel x to the nearest edge pixel in V by using,

$$DT_V(x) = \min_{v_j \in V} ||x - v_j|| \tag{4.2}$$

In order the distance transform can be computed using dynamic programing [xi], the cost function can be computed as,

$$d_{CM}(U,V) = \frac{1}{n}\sum_{u_i \in U} DT_V(u_i) \tag{4.3}$$

In order to match in term of orientation, every edge pixel is complemented with a direction term, then Directional Chamfer Matching (DCM)[xii] score is given by

$$d_{CM}(U,V) = \frac{1}{n}\sum_{u_i \in U} \overset{min}{v_j \in V}\left(\left|\left|u_i - v_j\right|\right| + \lambda \left|\left| \varphi(u_i) - \varphi(v_j)\right|\right|_{\pi}\right) \tag{4.4}$$

$\varphi(u_i)$ and $\varphi(v_j)$ are the orientation of template and edge point of query image meanwhile for λ is the weight for the direction term.

The three-dimensional distance transform ($DT3_v$) is a representation of three-dimensional image tensor. The first and second dimension are the location of an image plane, meanwhile for the third dimension is the channel orientation of a line, and by using this, it can minimize the matching cost in a linear time such that the complexity will be reduced. For every pixel and the direction channel, the cost of $DT3_v$ is expressed as

$$DT3_V(x, \varphi(x)) = \overset{min}{\widehat{\varphi}_i \in \Phi}||DT_{V\{\widehat{\varphi}_i\}} + \lambda\left|\left|\widehat{\varphi}(x) - \widehat{\varphi}_i\right|\right|_{\pi} \tag{4.5}$$

With $\widehat{\varphi}(x)$ and $\widehat{\varphi}_i$ are the quantized orientation assigned to a pixel and the orientation channel of the cost map the pixel belongs to. $DT_{V\{\widehat{\varphi}_i\}}$ represents the 2D distance transform of the edge points in V that has an edge orientation $\widehat{\varphi}_i$. The three-dimensional distance transform initially compute the 2D distance transform then updated with a forward recursion and a backward recursion for each pixel x. At most 1.5 cycles are needed for each of a forward and a backward recursion because the backward and forward recursion can not terminate after the full cycle, the values of three-dimensional distance transform entries continue to be updated until the value for a three-dimensional distance transform entry is not changed. The formulation of a forward recursion is following,

$$DT3_V(x, \widehat{\varphi_i}) = \min\{DT3_V(x, \widehat{\varphi}_i), DT3_V(x, \widehat{\varphi}_{i-1}) + \lambda\left|\left|\widehat{\varphi}_{i-1} - \widehat{\varphi}_i\right|\right|_{\pi}\} \tag{4.6}$$

Meanwhile for a backward recursion is formulated as following,

$$DT3_V(x, \widehat{\varphi_i}) = \min\{DT3_V(x, \widehat{\varphi}_i), DT3_V(x, \widehat{\varphi}_{i+1}) + \lambda\left|\left|\widehat{\varphi}_{i+1} - \widehat{\varphi}_i\right|\right|_\pi\} \quad (4.7)$$

So the directional chamfer matching score of the template *U* computed as

$$d_{CM}(U, V) = \frac{1}{n} \sum_{x_j \in |x_0, x|} DT3_V(x_j, \widehat{\varphi}_i) \quad (4.8)$$

In which it is equal to the total number of q orientation channels of distance transform images ($DT_{V(\widehat{\varphi}_i)}$) and by using dynamic programming, the difference in term of the orientation of the edge point that corresponds to a pixel and orientation channel reference q is added as a cost.

Figure 16 is adapted from Liu et. al.(2012), it illustrates the different between OCM and DCM.

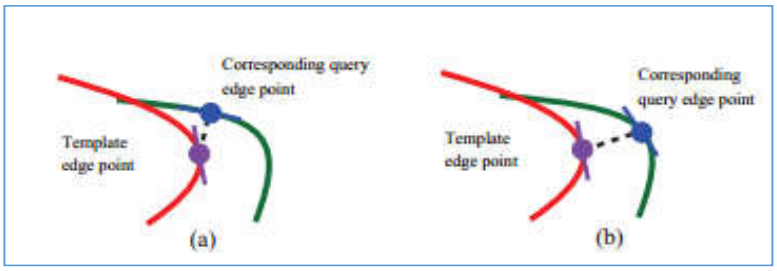


Figure 16. (a) Oriented Chamfer Matching (b) Directional Chamfer Matching

After having pixels which have the same orientation channel, we can compute the total cost as the integral distance transform which formulated as following,

$$IDT3_V(x, \widehat{\varphi}_i) = \sum_{u_i \in U} DT3_V(u_i, \widehat{\varphi} u_i) \quad (4.9)$$

Considering $L_U = l_{|s_i, e_i|}, i = 1, \ldots, m$ with s_i is the start point and e_i is the end points of i_{th} segment, then the chamfer matching score of each segment is,

$$d_{CM}(U,V) = \frac{1}{n}\sum_{l_i \in L_U} (IDT3_V(e_i, \hat{\varphi}(l_i)) - IDT3_V(s_i, \hat{\varphi}(l_i)) \quad (4.10)$$

If it is described in an illustration, it was described in Figure 17,

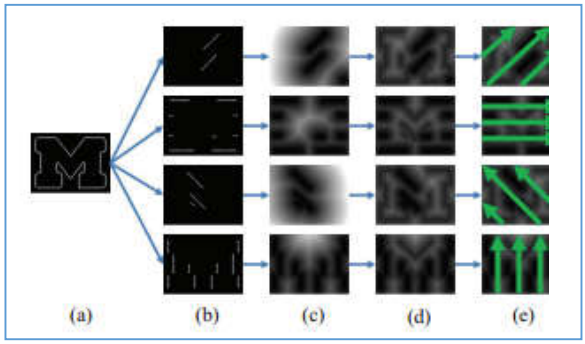


Figure 17. Step of Integral Distance Transform Computation

(a) Explains regarding the query edge map which it could use LSD. (b) quantize based on discrete orientation channel. (c) 2D distance transform of each orientation channel. (d) 3D distance transform which is updated based on orientation channel and (e) integrated 3D distance transform ($IDT3_V$) along orientation channel computation. Another illustration of integral image is shown in Figure 18,

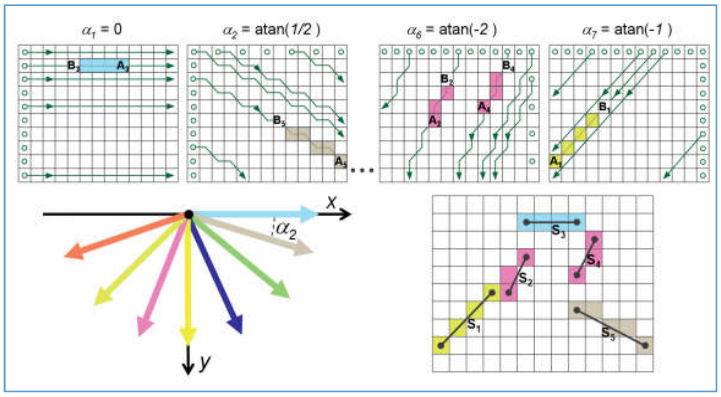


Figure 18. Illustration of Integral Distance Image

Summarized from (Beleznai et al.,2009) [xiii], Figure 18 shows the step of integral image with 8 channel orientations of span [0, π), the bottom right image tells an example of a contour template which consists of five line segments with the line integrals can be efficiently computed based on the integral images (summing up values at pixels color-coded according to the orientation).

4.2 Implementation

After having the projected 2D points data and edge line detected in a query image, then we manage the data such that we can find the minimum cost in term of distance and orientation between the template line segment and the edge line detected in a query image (the query line segment). Then from there, we will get the best in-plane parameters $\hat{s} = (\theta, t_x, t_y)$ with the lowest cost of DCM that will be a coarse pose.

Initially, the data input we get is 4 x number of lines, as an initial and an endpoint of every line in image frame either for the template line segment or the query line segment.

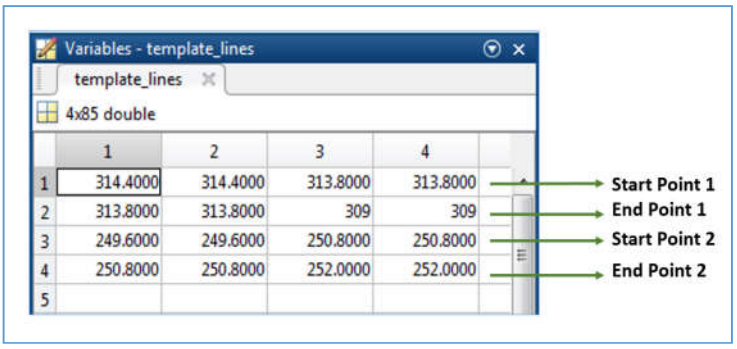


Figure 19. Initial Data Input of Matching Step

Then we applied the formulation (3.5) and (3.6), we get the angle direction of every line which is placed in the 5th row (in radian) of the data as shown in Figure 20,

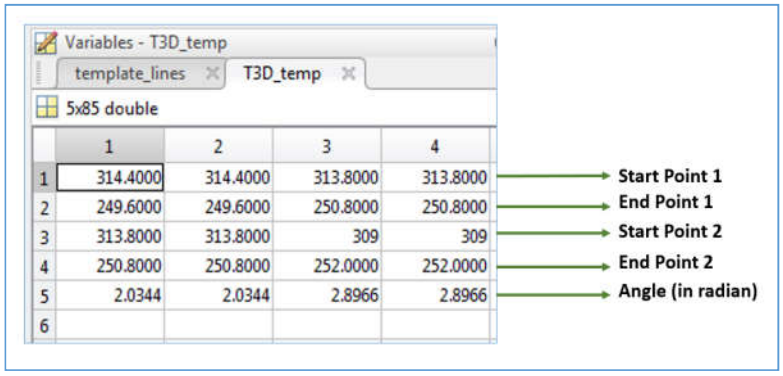


Figure 20. Angle Direction of Line Data

After that, we determine the orientation channel of the angle by using (4.10),

$$q_i = (\theta_i * n\_q)/\pi \tag{4.10}$$

With q_i is the orientation channel of *i-th* angle (in radian) and n_q is the total number of orientation channels q.

The next step is computing the length of every line and sort the data based on the length of the line (in Figure 21 is *7-th* row), that is starting from the longest one.

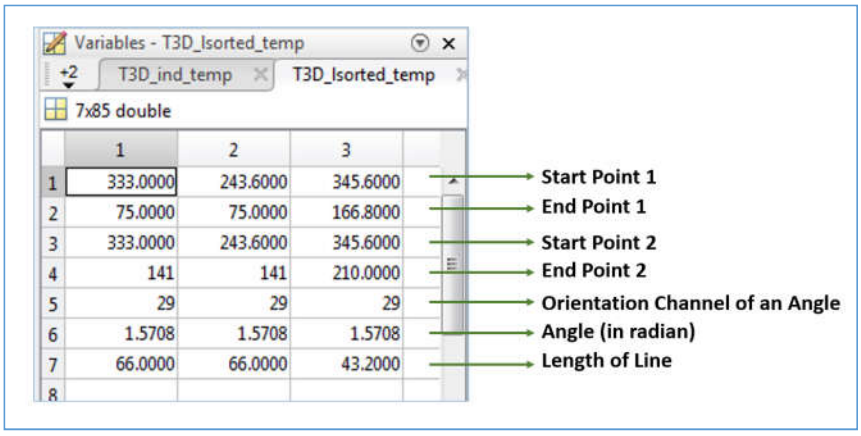


Figure 21. Sorted Data Based on The Length of Lines

Now, we apply the formulation of three-dimensional distance transform (image tensor), based on formulation (4.5), by choosing a pixel which has less different in terms of distance and orientation between the template line segment and the query line segment, such that it gives the lowest cost by scanning every orientation channel reference. Then the distance transform of an image will be checked in forward and backward recursion as formulated in (4.6) and (4.7), then we compute an integral distance transform based on (4.9) by computing the total costs based on the same orientation channel reference of an image (illustrated in Figure 18).

By using the data which is sorted by the longest line, we set the number of hypotheses by starting from the longest line one.

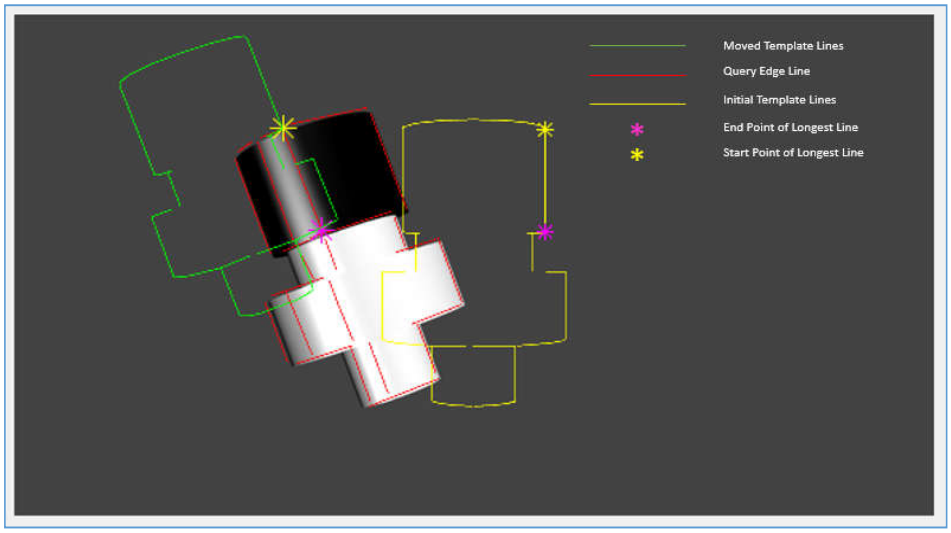


Figure 22. The 1st Hypotheses Plot

As shown in Figure 22, the end point of the longest line in the template line segment is coincided with the start point of the longest line in the query line segment. So, the initial template (yellow line) will move in the new position (green line) by rotating and translating the initial template lines (yellow line) to be aligned with the query line segment. Then it will translate along the longest line as illustrated in Figure 21,

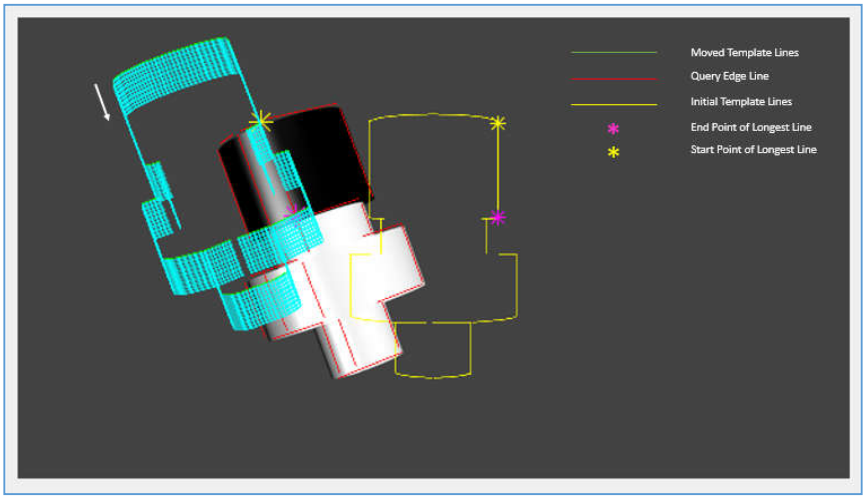


Figure 23. Translated Template along The Longest Line

As long as it coincides with the longest line in the query line segment, we compute the sum cost of integral distance transform which represented in formulation (4.10). We compute every movement and find out the lowest cost one. For example, in the case of Figure 23, the lowest cost is illustrated in the following figure,

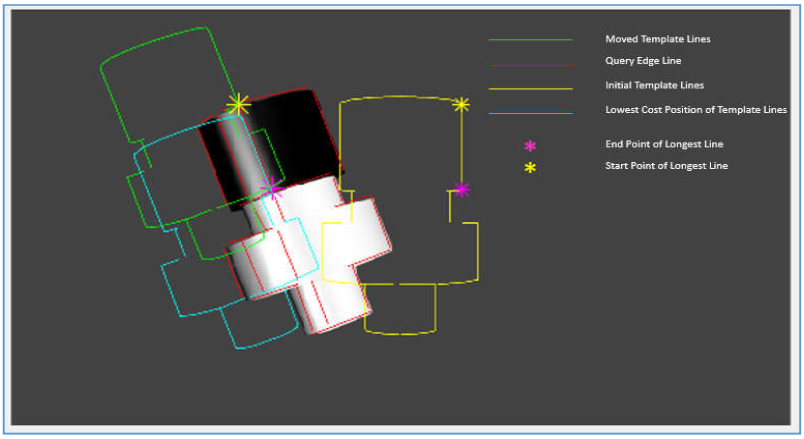


Figure 24. The Lowest Cost of Translated Template

Figure 24 shows that the lowest cost of the translated position is shown by a cyan line, that is in this case, the end point of the longest template line segment and the

end point of the longest query line segment coincide. But it is not always in the case, depends on the condition. Another hypotheses is plotted on Figure 25,

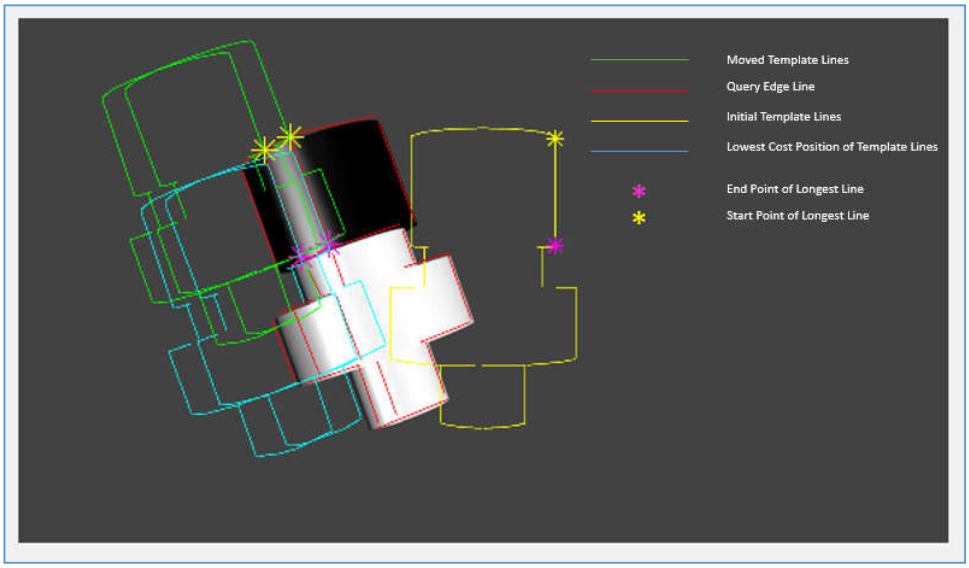


Figure 25. The 2nd Hypotheses Plot

Figure 25 is another hypotheses, that is the 2nd longest line in the query line segmnet. Then we treat the same procedure as explained before. Then the third hypotheses is shown in Figure 26,

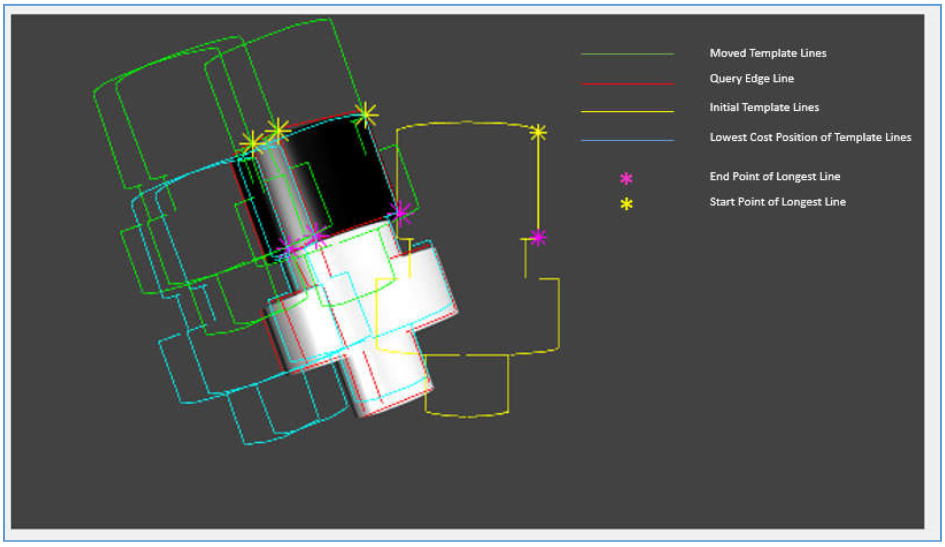


Figure 26. The 3rd Hypotheses Plot

The same as the previous case, the 3rd hypotheses is the 3rd longest line of the query line segment. And based on the example of 3 hypotheses, the 3rd one is having the lowest cost. As in image visualization (refer to Figure 26), it can be shown by noticing the cyan line of the 3rd hypotheses is the closest one with respect to the query line segment (red line), in which it means that the template point can 'find' the location (translation t_x and t_y and rotation θ_z) in the image frame. If we increase the number of hypotheses, it will have a result with the higher possibility to find the lowest DCM cost, but it has a drawback for the next step (Best Match Selection) that will be explained later and in term of time cycle, it takes more time to execute the program. Based on the above (three) hypotheses, we store the data as illustrated in Figure 27,

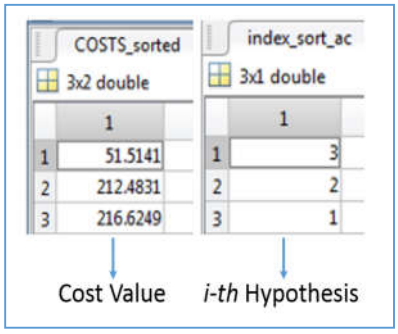


Figure 27. Sorted Cost

And the in-plane parameters (translation t_x and t_y and rotation θ_z) is summarized in Figure 28,

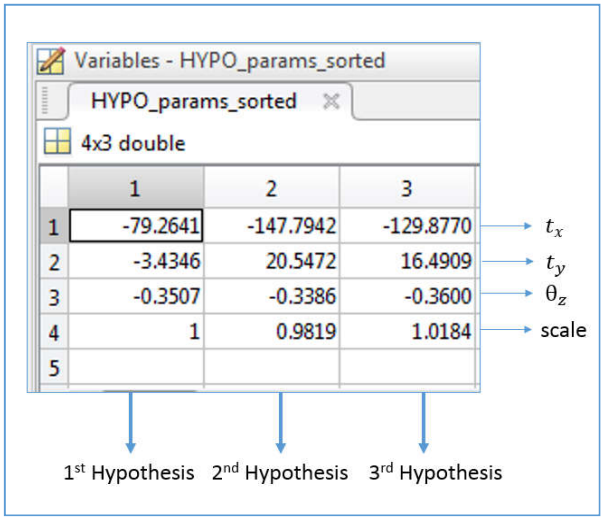


Figure 28. In-Plane Parameters

Beside the in-plane parameters, the last row is the info about scaling. Meanwhile for every column gives information about *i-th* hypotheses which also represents the *i-th* longest line. Note that all of these parameters are in the image frame. These parameters (called as the coarse pose parameters) will be used in the pose refinement step, which will be explained in Chapter 4. In this case, if we plot the best parameters (the coarse pose hypotheses), it can be illustrated in Figure 29,

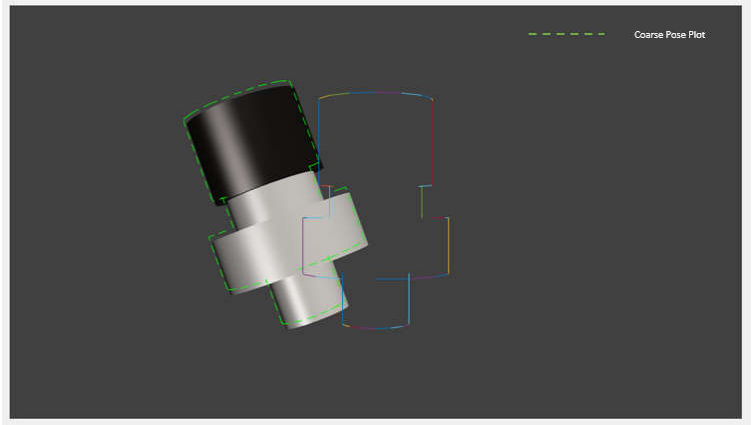


Figure 29. Coarse Pose Hypotheses Plot

Chapter 5

Multi View Pose Refinement and Best Match Selection

Since the coarse pose is done in the image frame, so the parameters we got is only in-plane parameters (translation t_x and t_y and rotation θ_z). In order to increase the accuracy of 3D pose of the object, we apply pose refinement step in which it also refine the out-of-plane parameters (θ_x and θ_y). So, it will be processed in world frame instead of the image frame with the input is from the coarse pose parameters data then it will be projected to the image frame to see the result of the plot.

5.1 Minimization Cost

In order to get the minimum cost, it is needed to define the function such that for every projected template point in image plane u_i find the nearest edge query image point v_i in V which has minimum the DCM cost [vii] as formulated in (5.1),

$$\arg\min_{v_i \in V} \left|\left|u_i - v_j\right|\right| + \lambda ||\varphi(u_i) - \varphi(v_j)|| \tag{5.1}$$

The established 3D to 2D point correspondences can be noted as $(\widetilde{u_i}, v_i)$. Then consider that there is missing edges due to low contrast or objects with partial occlusion, we use a thresholding system. A threshold σ which are based on the median of DCM cost of every template point $\delta_{median(d_{DCM})}$ and a certain value σ_{base} (*Sakcak,B. et al., 2016*), such that,

$$\sigma \begin{cases} \delta_{median(d_{DCM})}, & \text{if } \delta_{median(d_{DCM})} > \sigma_{base} \\ \sigma_{base}, & \text{if } \delta_{median(d_{DCM})} < \sigma_{base} \end{cases} \tag{5.2}$$

So, by using threshold σ, we can choose point pairs that having cost lower than the threshold. The result gives 3D-2D point correspondences $(\widetilde{u_k}, v_k)$ with $\widetilde{u_k}$ is subset of rasterized template points $\widetilde{u_i}$ that has correspondences v_k which below the cost bound which defined as threshold σ and it will be used in the refinement step.

5.2 Optimization

The concept for optimization is minimizing the error value between projected rasterized 3D template points $\widetilde{u_k}$ and the correspondences query edge points v_k. To do it, there are some algorithm such as Least Square Error [xiv]. If it is in the case, so the least square projection error given in (5.3),

$$\varepsilon(p) = \sum_{\widetilde{u_k}} \left|\left| PT_m(p)\widetilde{u_k} - v_k \right|\right|^2 \tag{5.3}$$

Error function is minimized for each hypotheses using Levenberg-Marquardt algorithm [xv] The concept of Levenberg-Marquardt method derivated as (5.4),

$$\begin{aligned} \gamma^2(p) &= \sum_{i=1}^{m} \left[\frac{y(t_i) - \hat{y}(t_i; p)}{\sigma_{y_i}} \right]^2 \\ &= (\mathrm{y} - \hat{y}(\mathrm{p}))^T \mathbf{W} (\mathrm{y} - \hat{y}(\mathrm{p})) \\ &= y^T \mathbf{W} \mathrm{y} - 2y^T \mathbf{W} \hat{y} + \hat{y}^T \mathbf{W} \hat{y} \end{aligned} \tag{5.4}$$

With σ_{y_i} is a measurement error for a measurement $y(t_i)$. $\mathbf{W}$ is the diagonal matrix with $\mathbf{W}_{ii} = 1/\sigma_{y_i}^2$ or can be set as the inverse of the measurement error covariance matrix. Because $\hat{y}$ is nonlinear in the model parameters **p**, so the minimization will be done iteratively with the goal to find a perturbation **h** to the parameters p that reduces γ^2.

Then apply the Gradient Descent Method which updates parameter values in the "downhill" direction: the direction opposite to the gradient of the objective

function. The gradient of γ^2 objective function with respect to the parameters is derivated in (5.5)

$$\begin{aligned}\frac{\partial}{\partial p}\gamma^2 &= 2(\mathrm{y} - \hat{y}(\mathrm{p}))^T \mathbf{W} \frac{\partial}{\partial p}(\mathrm{y} - \hat{y}(\mathrm{p})) \\ &= -2(\mathrm{y} - \hat{y}(\mathrm{p}))^T \mathbf{W} \frac{\partial \hat{y}(\mathrm{p})}{\partial p} \\ &= -2(\mathrm{y} - \hat{y}(\mathrm{p}))^T \mathbf{WJ}\end{aligned} \tag{5.5}$$

with the *m x n* Jacobian matrix $\frac{\partial \hat{y}}{\partial p}$ and it will be denoted as **J** for simplicity. The parameter update **h** that moves the parameters in the direction of steepest descent is given by

$$\mathbf{h}_{gd} = \alpha \mathbf{J}^{\mathrm{T}} \mathbf{W}(\mathrm{y} - \hat{y}) \tag{5.6}$$

With α is the length of the step in the steepest-descent direction. Then apply the Gauss Newton method to minimize sum-squares objective function. The function evaluated with perturbed model parameter approximated through first order Taylor series expansion stated as,

$$\hat{y}(\mathrm{p} + \mathbf{h}) \approx \hat{y}(\mathrm{p}) + \left[\frac{\partial \hat{y}(\mathrm{p})}{\partial p}\right] \mathbf{h} = \hat{y} + \mathbf{Jh} \tag{5.7}$$

Substituting the approximation for the perturbed function, $\hat{y} + \mathbf{Jh}$, with $\hat{y}$ in equation (4.4), so we get

$$\gamma^2(p + \mathbf{h}) \approx y^T \mathbf{W} \mathrm{y} + \hat{y}^T \mathbf{W} \hat{y} - 2y^T \mathbf{W} \hat{y} - 2(y - \hat{y})^T \mathbf{WJh} + \mathrm{h}^T \mathbf{J}^T \mathbf{WJh} \tag{5.8}$$

The parameter update **h** that minimize γ^2 is found from $\frac{\partial \gamma^2}{\partial \boldsymbol{h}} = 0$:

$$\frac{\partial}{\partial \boldsymbol{h}} \gamma^2(p + \mathbf{h}) \approx -2(y - \hat{y})^T \mathbf{WJ} + 2\mathrm{h}^T \mathbf{J}^T \mathbf{WJ} \tag{5.9}$$

And the resulting normal equation for the Gauss-Newton update is

$$[\mathbf{J}^T\mathbf{W}\mathbf{J}]\mathbf{h_{gn}} = \mathbf{J}^T\mathbf{W}(y - \hat{y}) \tag{5.10}$$

Finally the Levenberg-Marquardt algorithm adaptively varies the parameter updates between gradient descent update and the Gauss-Newton update,

$$[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\mathbf{I}]\mathbf{h_{lm}} = \mathbf{J}^T\mathbf{W}(y - \hat{y}) \tag{5.11}$$

In which small values of the algorithmic parameter λ result in a Gauss-Newton update and large values of λ result in a gradient descent update. If any iteration happens to result in a worse approximation ($\gamma^2(p + \mathbf{h_{lm}}) > \gamma^2(p)$), then λ is increased. Otherwise, as the solution improves, λ is decreased, the Levenberg-Marquardt method approaches the Gauss-Newton method, and the solution typically accelerates to the local minimum [xvi] [xvii]. Meanwhile the Marquardt's update relationship [xviii] stated as

$$[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})]\,\mathbf{h_{lm}} = \mathbf{J}^T\mathbf{W}(y - \hat{y}) \tag{5.12}$$

the values of λ are normalized to the values of $\mathbf{J}^T\mathbf{W}\mathbf{J}$

5.3 Best Match Selection

The low value of DCM doesn't guarantee that it is the best selection object to be manipulated. One of the cause is due to a lighting factor, because it can produce low contrast region. But it can be anticipated using scoring system value which is based on the gradient orientation of the image and the template lines [xix],

$$S(U, I_{go}) = \frac{1}{n}\sum_{i=1}^{n} |\cos(I_{go}(x_i) - \theta(u_i))| \tag{5.12}$$

With n is the total number of projected template points, $I_{go}(x_i)$ is the gradient orientation of the pixel corresponding to the projected template point on the image frame u_i, and $\theta(u_i)$ is the orientation of projected template point. The best matching is when the difference between the orientation of projected template point and the gradient orientation of the pixel corresponding to the projected template point on the image frame is low, which means that the scoring value result $S(U, I_{go})$ will give a high score because of cosine arithmetic operation. Based on Formulation (5.12), the scoring value is the average of the difference in the cosine over all the points and it is the normalized one, so the score will be [0,1] with 1 is in the ideal case of the topmost object. So, roughly said that the scoring value indicates the coincidence of the gradient and orientation between the template line segments and the visible query image line segments.

5.4 Implementation

The first step that we did in refinement step is changing the coarse pose parameters or we can say the hypotheses parameters, which are in an image frame becoming a world frame using formulation (2.4). Table 1 showing the example of hypotheses parameter data.

Parameters	Value	Unit
tx	-79.2641	pixel
ty	-3.4346	pixel
tz	-	pixel
theta_x	0	rad
theta_y	0	rad
theta_z	-0.3507	rad

Table 1. Hypotheses Parameters in Image Frame

The plot of applying hypotheses parameters is illustrated in Figure 30, in which the green dashed line with the red star symbol (as the middle point of the template) is the movement template after applying the hypotheses parameters

meanwhile, the multiple color line with the yellow star is the initial location of the template.

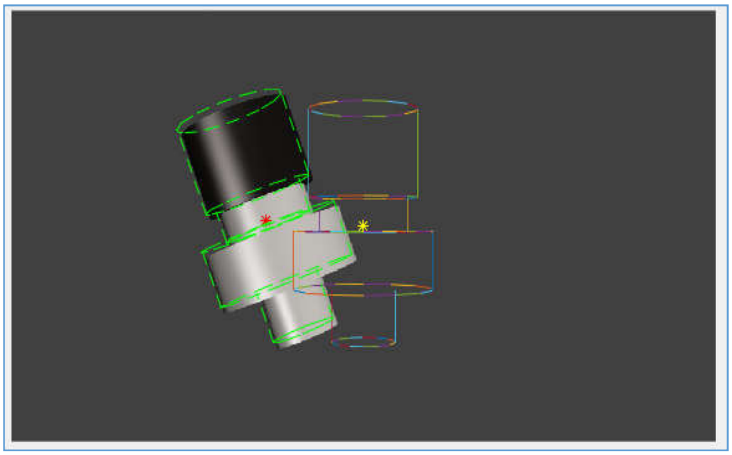


Figure 30. Plot of Hypotheses Parameters

The following table 2 represents the value of hypotheses parameters which are stated in the world frame.the value is obtained after the conversion formula (2.4).

Parameters	Value	Unit
tx	-11323.4	mm
ty	490.6507	mm
tz	90000	mm
theta_x	0	rad
theta_y	0	rad
theta_z	-0.3507	rad

Table 2. Hypotheses Parameters in a World Frame

Then apply the changed parameters, which stated in the Table 2, to the original 3D template. The result is illustrated in Figure 31, with the yellow dash line is the movement of the template after apply the parameters value in Table 2.

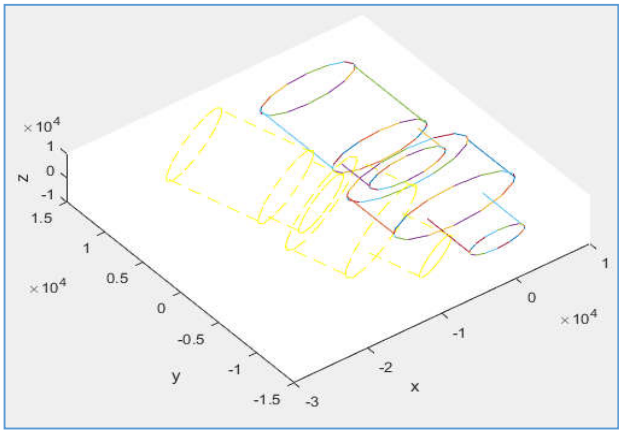


Figure 31.Plot of Hypotheses Parameters in World Frame

Now, we have a new 3D template lines with the different pose from the initial pose of the template. We rasterize the template with a certain length and project the rasterized template point on an image frame using Formulation (2.2), because we will do the next step which related with the query edge line detected which are in image frame.

The next step is finding the correspondences between rasterized template point and the detected point in query image by using Formulation (5.1). As explained previously, the idea is finding the correspondences point which can produce the lowest cost. By checking every point of the rasterized template and every point in a query image we apply Formulation (5.1). With using the same example as previously, Figure 32 shows the cost value of *i-th* rasterized template point,

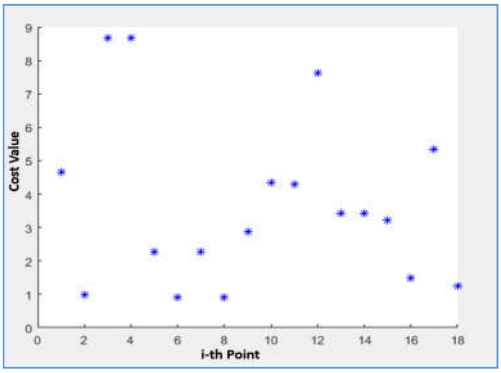


Figure 32. The Cost Value of Each *i-th* Rasterized Template Point

As a result, we store as the 'point_pairs' variable with the rasterized template point has a dimension (1:2 x number of points), the query image correspondences has a dimension (3:4 x number of points), and the cost value has a dimension (5x number of points).

The 'point_pairs' variable will be the input a of the thresholding system of the median value of DCM cost, by applying Formulation (5.2). with the same example as previous case, Figure 33 shows the dimension result after applying Formulation (5.1) and (5.2)

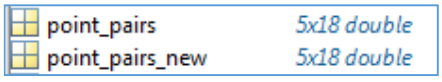


Figure 33. Dimension Result

In Figure 33, the 'point_pairs' variable is the result of Formulation (5.1) meanwhile 'point_pairs_new' variable is the result of thresholding system (Formulation (5.2)). In this case, before and after applying Formula (5.2) there is not any different dimension, it means that all the cost of every point is below the threshold value.

After that, we can find the error by using Formula (5.3), in which this error will be the input of the optimization step which uses the MATLAB function, the optimization we choose is using Least Square Error Levenberg-Marquardt

method. By setting the number of iteration and the tolerance, with the same case as before, it has the result as illustrated in Figure 34,

Iteration	Func-count	Residual	First-Order optimality	Lambda	Norm of step
0	7	349.074	219	0.005	
1	14	319.86	116	0.0005	28.6998
2	21	294.868	348	5e-05	112.435
3	30	286.871	2.48	0.005	24.2483
4	37	282.138	2.23	0.0005	20.9014
5	45	278.633	1.57	0.005	17.9768
6	52	276.036	1.29	0.0005	15.4772
7	61	275.8	0.458	0.05	1.52585
8	68	275.572	0.169	0.005	1.50252

```
x =

  1.0e+04 *

  -1.1133
   0.0413
   9.0000
   0.0000
  -0.0000
  -0.0000
```

Figure 34. Pose Refinement Result

In Figure 32, the left side is the examine process of Least Square Error Levenberg-Marquardt method using MATLAB function. The iteration is stopped because the final change in the sum of squares relative to its initial value is less than the default value of the function tolerance. It means it finds the local minimum of the error. Meanwhile the right side is the result of the parameters (in world frame) after being processed of optimization step. The plot of the refinement pose step shown in Figure 35,

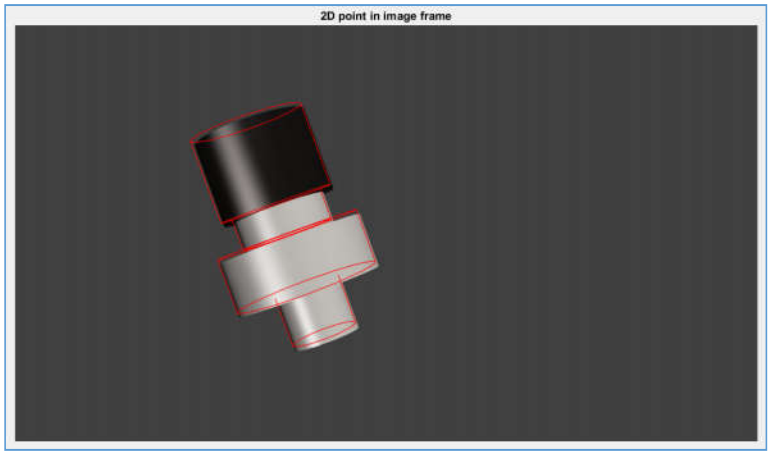


Figure 35. Plot of Refinement Pose

Figure 35 is after applying the parameter in the world frame that is we translate and rotate the 3D template (not the camera) based on refined parameters, then project the result into an image frame. It refines not only in-plane parameters but also out-of-plane parameters. That is why before projected on the image frame, the template is translated and rotated in the world frame.

After getting the refined parameters, the next step is the best matching selection. By using MATLAB function of '*imgradient*', it will return the gradient and orientation value of an image. In this case, the query image is in a grayscale. Meanwhile, the 3D template changes its pose based on the refined parameters result then project to image frame using (2.3) and also store the angle (orientation value) of the line by using (3.5) and (3.6), Then rasterize the template. The rasterized projected template points together with the orientation information are used to apply (5.13) to determine the score.

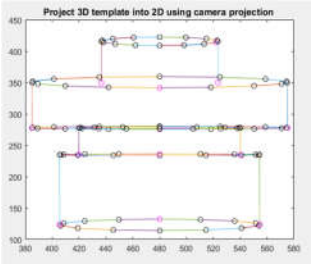
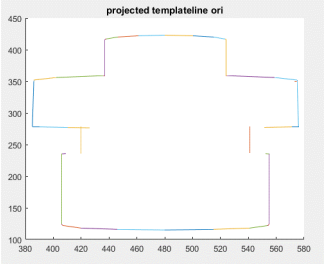
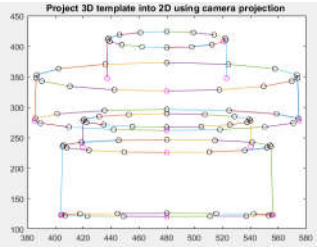
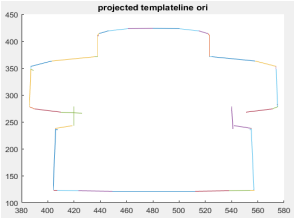
Chapter 6

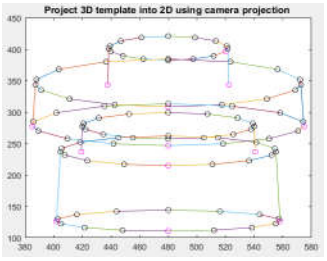
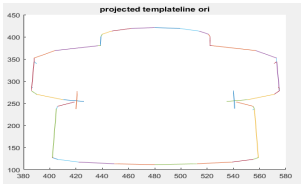
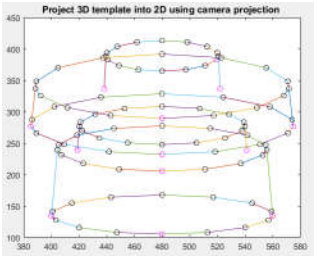
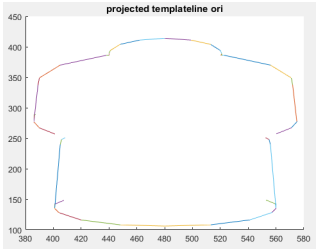
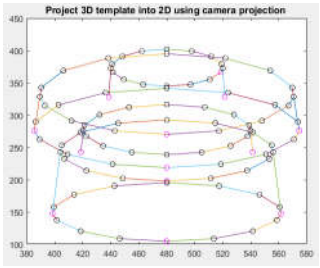
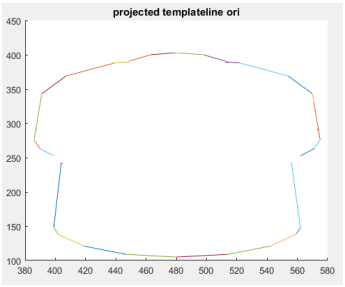
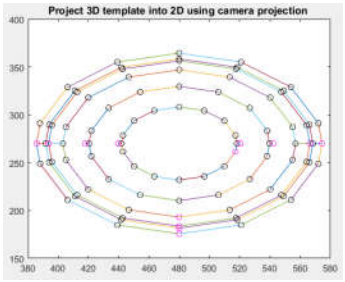
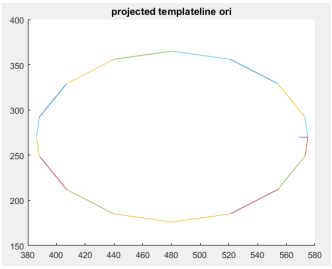
Experiment and Result

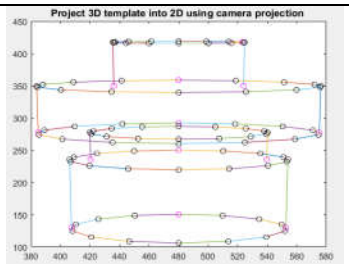
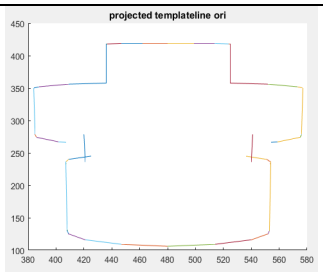
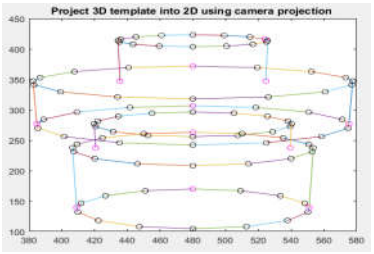
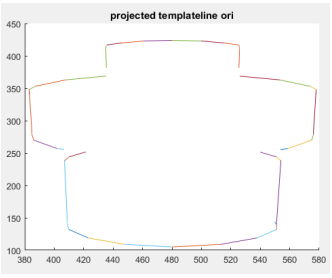
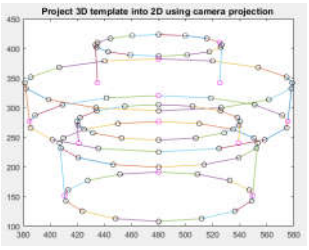
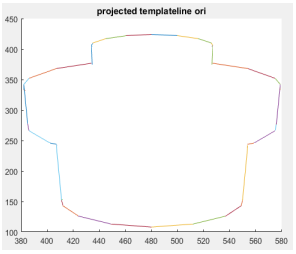
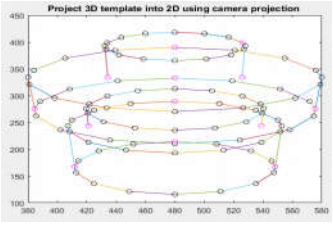
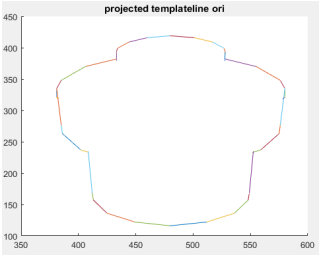
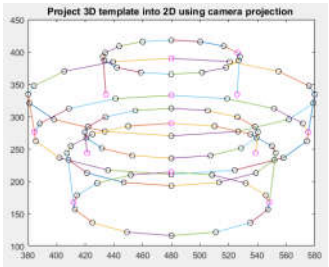
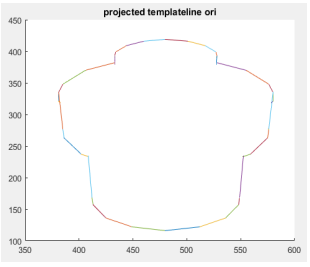
The experiments are done both in a synthetic and a real camera image as explained previously. As the object is 3D rotational symmetric (as illustrated in Figure 1), the setup also prepared both for in-plane and out-of-plane cases.

6.1 Outline Mode Result

Refer to the objective of this thesis is to analyze the flexibility of the Outer Line Mode method, then we analyze the results qualitatively. The expected result is we want to store only the outer line of the template in any object pose with respect to camera view, Here are the result,

No	Out of Plane θx (degree)	Original Template	Outline Mode Result
1	1		
2	10		

3	20		
4	30		
5	40		
6	90		

7	-10		
8	-20		
9	-30		
10	-40		
11	-50		

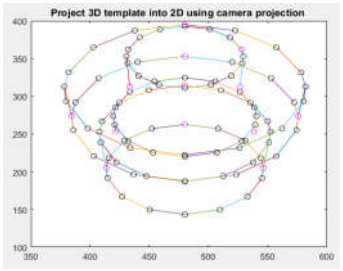
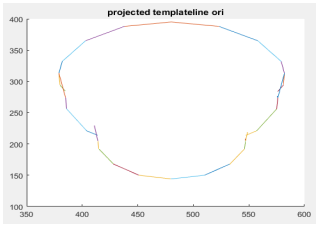
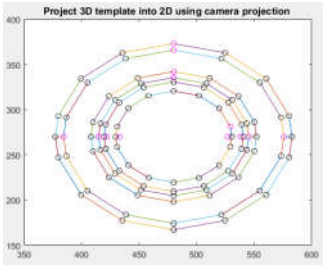
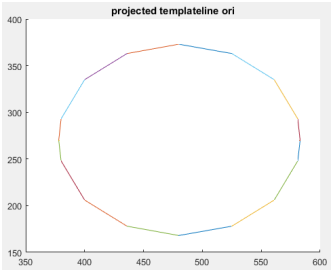
12	-60		
12	-90		

Table 3. Outer Line Mode Result in The Image Frame

As we can see, qualitatively, the result is as our expectation, only store the outer line. How we generate the template is fast. By only have a data of 3D template lines (Figure 37), then the user is allowed to change in any pose of the template, then it will generate automatically.

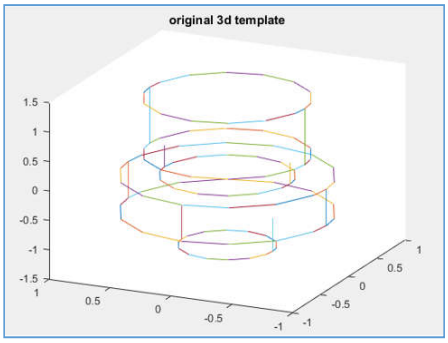


Figure 36. 3D Template Lines Data Base

Another information, we only allow the user changes the template in term of rotation with respect to x-axis (recall about the definition of an out-of-plane θ_x, in the introduction section), because for in-plane case (t_x, t_y, θ_z), it is enough using one and the same template when θ_x is around ± [0,10)°. Meanwhile, when the object position having $\theta_x \geq 10°$ we use another template (generate another template because we run the program using 1 template model) in which the user only change the value θ_x, then the Outer Line Mode will generate automatically to give the output, because it will be the template data base. Regarding θ_y, as explained before, we don't consider this parameter because the object's shape is rotational symmetric (recall in the introduction section).

6.2 Synthetic Camera

Refer to the purpose of this thesis is to test the accuracy of detecting the 3D pose estimation result, then we use a single object test set. Another goal of this thesis is analysing the best selection object to be manipulated or the top most object selection (in term of the score), then we use multiple objects test set. We tested using synthetic camera image (*Blender* software) as it is in the case of ideal camera condition (there is not distortion, etc).

6.2.1 Single Object

By employing the condition which represents many poses (different location and orientation both in plane and out of plane) of the object, we can express the accuracy of the object detection and localization in a quantitatively. Figure 38 shows some result of single object for detection and localization test in various poses of in-plane case.

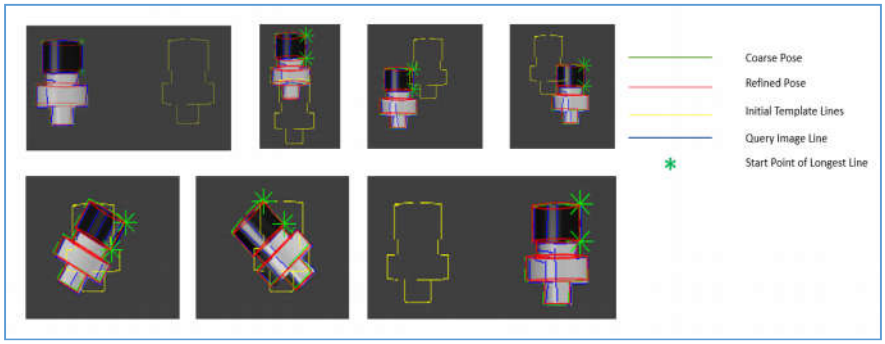


Figure 37. Various Pose of In-Plane Case

Another experiment is trying the out-of-plane parameters. By taking the object rotation with respect to its x-axis ±30°. The result for some poses when the object is out of plane ±30° is illustrated in Figure 39.

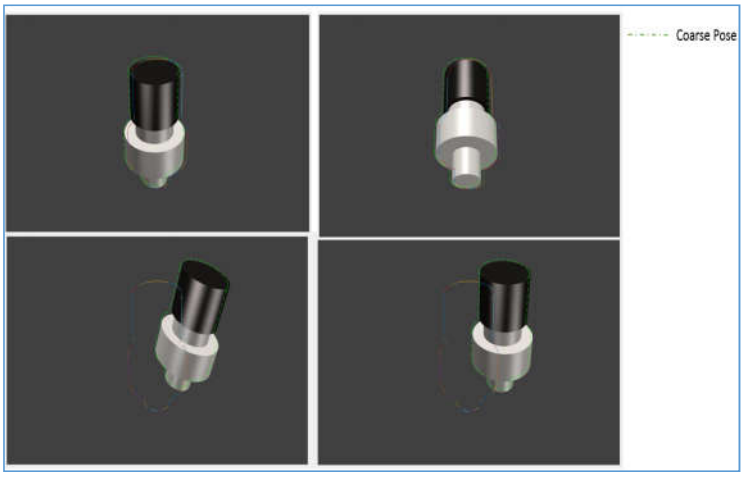


Figure 38. Out of Plane Pose

By using *Blender* software, we know exactly the position of the object (reference parameters), then we test our algorithm in 60 poses (we generate 60 Blender images) which represents any possibilities both in-plane poses and out-of-plane poses. The result of the average error is summarized in the following Table 4,

PARAMETERS	AVERAGE ERROR	UNIT
Translation x axis	0.16	mm
Translation y axis	0.3	mm
Translation z axis	-	mm
rotation x axis	1.7	degree
rotation y axis	-	degree
rotation z axis	0.7	degree

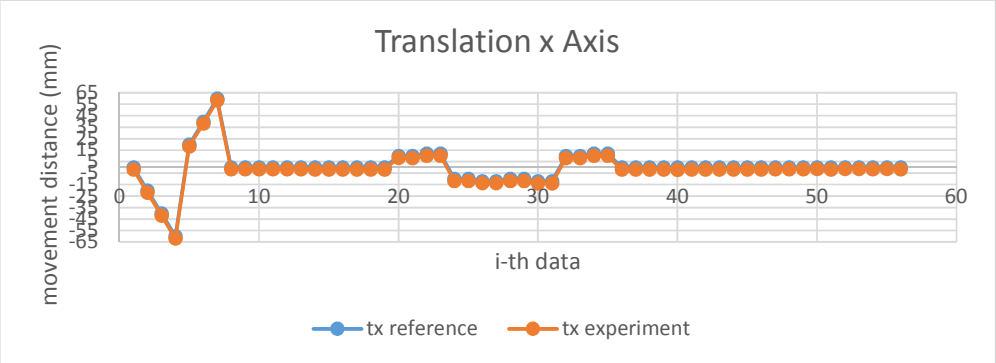
Table 4. The Average of The Error Parameters Using Blender Images

As shown in Table 4, the average of the error is calculated with respect to the world frame. In this case, the world frame is also the object frame. The error we get from the following formulation,

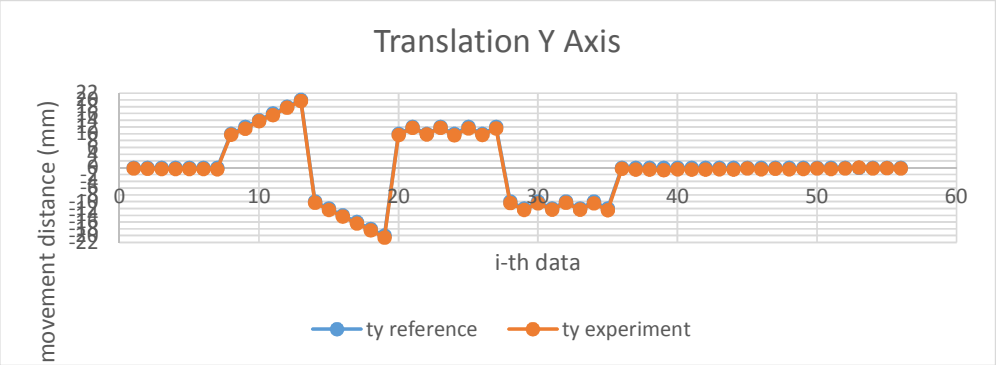
$$Error = \sum_{i=1}^{total\ pose} \left| \frac{refined\ parameters(i) - reference\ parameters(i)}{total\ pose} \right| \qquad (6.1)$$

The value of the refined parameters is stated in world frame, as explained in the previous chapter that the input of refinement pose parameters is converted in the world frame. Then for rotation y-axis (θ_y) we don't have an error because the object is rotational symmetric shape. Also for the translation with respect to z-axis (t_z) which is the height of camera with respect to the object, it doesn't have error because the camera position is fixed and our method that we use to refine the object pose is done by changing the pose of the object and make a fixed camera position instead of changing camera position.

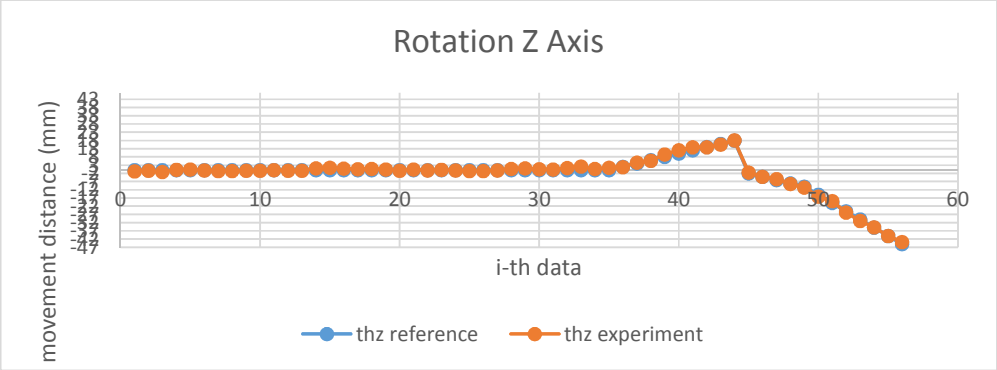
It is also convenient to present the differences between reference pose and the resulting pose of the algorithm (experiment) by plotting in the graph,



Graph 1. The Different Between Reference and Experiment Parameters of Translation x-Axis



Graph 2. The Different Between Reference and Experiment Parameters of Translation y-Axis



Graph 3. The Different Between Reference and Experiment Parameters of Rotation z-Axis

6.2.2 Multiple Object

The experiment of a single object is for checking the accuracy in term of detection and localization (object pose). Another purpose of this thesis is to select the topmost object. To do so, we set in the *Blender* image by creating multiple objects of the same kind which are overlapped each other as shown in Figure 39.

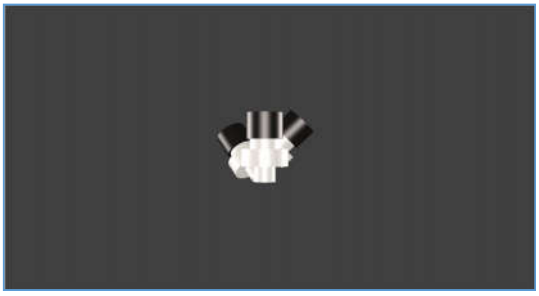


Figure 39. Image Synthesis for Multiple Object Case

The object which has the lowest DCM cost doesn't mean that it is the topmost object. The multiple objects condition which has missing edges object cannot be handled only using the DCM algorithm. For the case in Figure 40, the lowest DCM cost of the object is illustrated in Figure 41 with the red line. Another information that for the case in Figure 39 and Figure 40, the object with the lowest DCM cost is an object which has missing edge due to the partial occlusion. In

order to make precisely estimate the exact object pose, we cannot use all the correspondences because of the missing edge. That is why it is needed to use thresholding system based on the median of DCM cost (Formulation (5.2)).

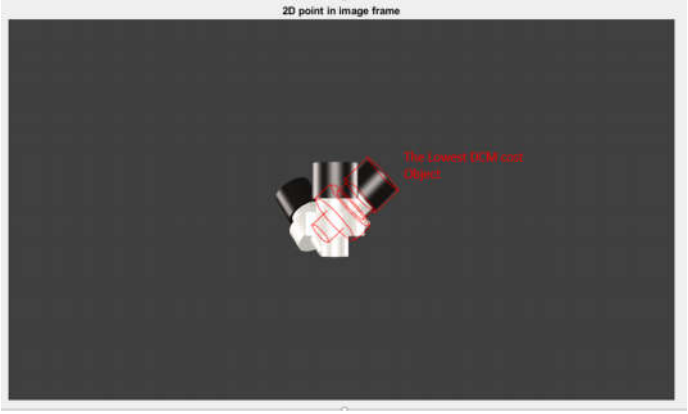


Figure 40. Lowest DCM Cost Object

But as we can see that it is not true if red line object in Figure 41 becomes the top most object. Then we apply the Gradient and Orientation Score algorithm. The result shown in Figure 42,

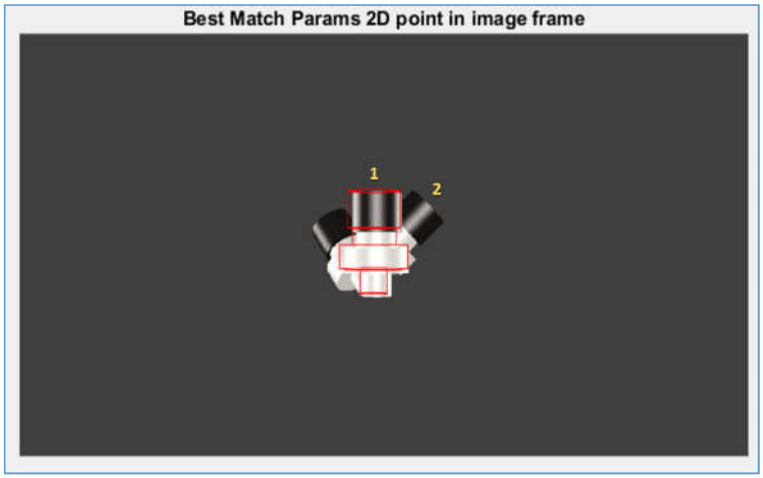


Figure 41. The Top Most Object

The scoring data is summarized in the Table 5,

INDEX	SCORE
1	0.6769
2	0.6319

Table 5. Scoring Result Table

The score value is the normalized one, so the highest score will be [0,1] with 1 is the ideal case of the topmost object. As explained in the Chapter 5, that the scoring value indicates the coincidence gradient and orientation of the template line segments and the visible edge line of captured image (query image).

The results in Table 5 and in Figure 42 show that the highest score indicated with index 1. The topmost object is the one which has the highest score because the topmost object should have more visible edges than the other objects and with using Formulation (5.12) that is the average of the difference in cosine over all the points, makes that the topmost object has the gradient and orientation coincide with respect to the projected template lines.

The best match object (highest score) will be the topmost object because, in a position point of view, it should be all edge lines is detected or at least it has more edge lines detected than the other objects and automatically it will have higher chance to have higher score than the other objects.

6.3 Real Camera

After completing the experiment with the synthetic image (*Blender* software), we also provide the experiment with the real camera. Some considerations are used in the real camera system since we don't use the robot that can be the world frame, so we set up the world frame and the whole system for real camera system.

6.3.1 Set Up

Refer to the objective of this paper, we need to compute an average error quantitatively, we need to set up the experiment. For the real camera images, we

use *uEye* camera which is placed perpendicular with respect to the object with the height 200 mm. Then we test some parameters by changing the pose of the object meanwhile for the camera position is fixed.

6.3.1.1 World Frame Setup

We decide the world frame origin will be the middle point where the template is placed for the first time on image frame (initial pose). So, we prepare a 'paper base' with the red point sign to match the corner point of the template line (red point in Figure 43 right) and the middle point (the green point in Figure 43 right) then we project the template on the captured image with the picture of that paper. We set the position of the paper with respect to the camera position such that the corner of the template line is touching the red point as shown in Figure 43. After successful touched the corner, then we can easily determine the middle point (as we know the dimension of the object) in which it will be the origin of the world frame and we make the position of the paper is fixed with respect to the camera then we do some tests.

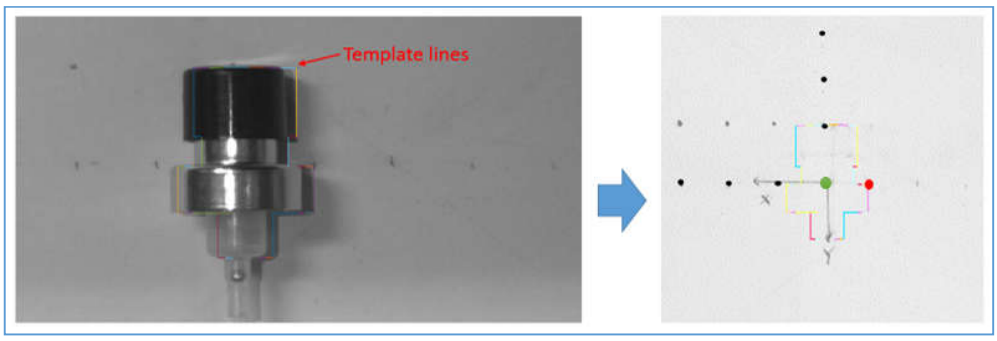


Figure 42. World Frame Setup

6.3.1.2 Translation and Rotation Movement Setup

Based on Figure 43 and previous explanation, after we set the paper position with respect to the camera position, and we got the coordinate of world frame's origin, then we make the sign of every movement step both for rotation and translation (Figure 41 right, it is signed with a black point). For the translation movement, we

use a ruler with the step variation is 10 mm. meanwhile for in-plane rotation, we use a protactor with the step variation is 10°. Figure 42 shown the whole setup.

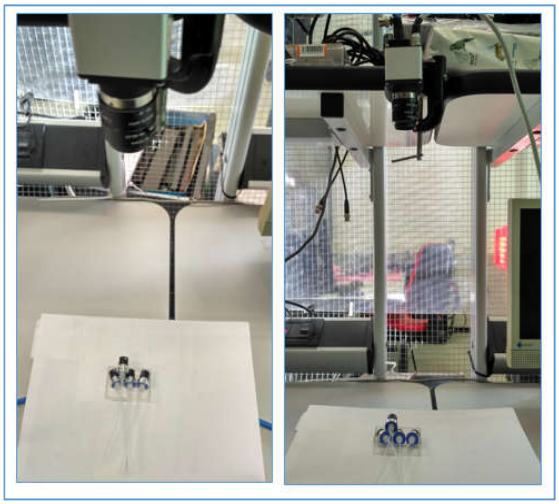


Figure 43. The Whole System Setup

Another setup is needed for the out-of-plane rotation that is shown in Figure 45,

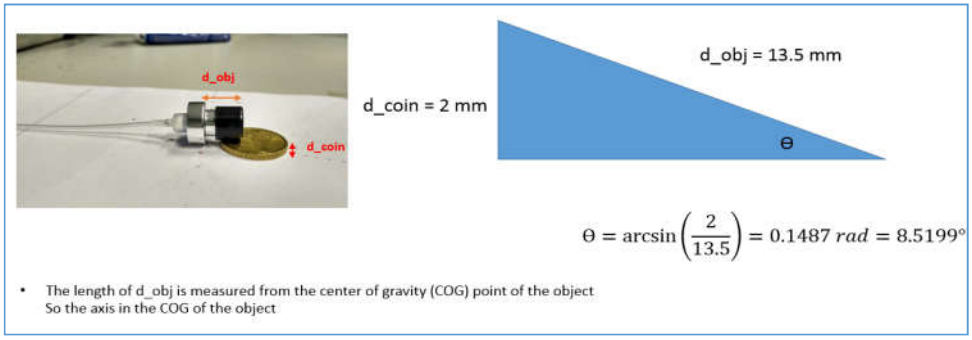


Figure 44. Out of Plane Rotation Setup

We use the coin with the height is 2 mm for 1 coin which means that we use 8.5199° step variation (the calculation shown in Figure 45) when we use 1 coin.

6.3.2 Single Object

With the same explanation as using a synthetic image, here are some results when using a single object with the real camera image in which the distance between the camera and the object is 200 mm. Figure 46 shows the translation with respect to x-axis and y-axis of the world frame test for in-plane parameter case.

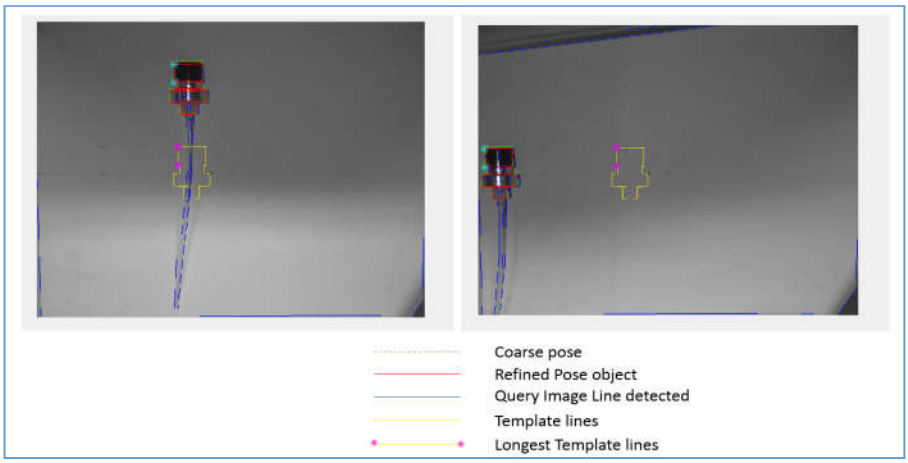


Figure 45. Translation on x-Axis and y-Axis in-Plane Parameter Test

Meanwhile Figure 47 shows the rotation in plane parameter (with respect to z-axis of the world frame) test.

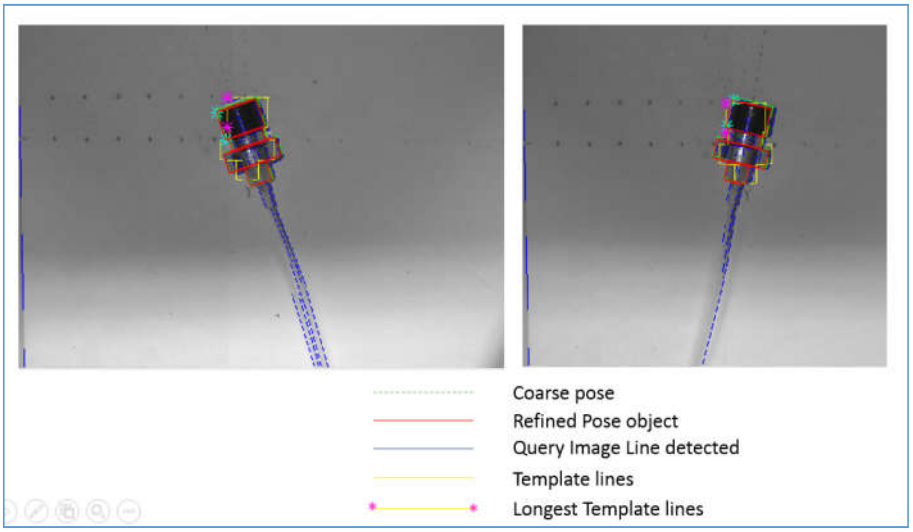


Figure 46. Rotation in Plane Test

Another test we did is for out-of-plane condition.it is shown in Figure 48,

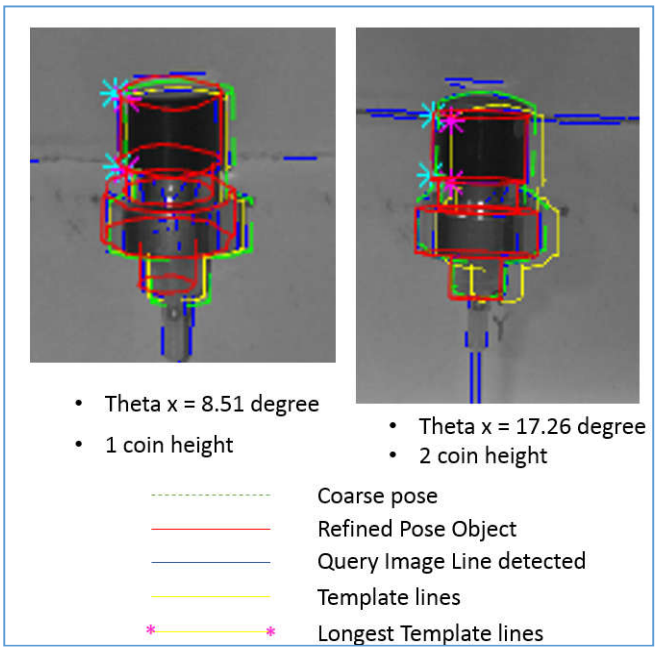


Figure 47. Out of Plane Rotation Test

To know the accuracy of detecting the object pose using the real camera, we summarized the average error in Table 6,

	VALUE	UNIT
tx	0.49	mm
ty	0.43	mm
tz	-	mm
theta_x	3.03	degree
theta_y	-	degree
theta_z	1.26	degree

Table 6. The Average of Error Parameters Using The Real Camera Images

The data in Table 6, we used 15 poses. Using the same formulation as (6.1) then we compute the error. As explained in Setup section, we know the middle point of the template and center of gravity (COG) of the object, we also have the translation sign, then we can calculate the error by comparing with the result of the algorithm. If we compare to the result when using the synthetic image, the result using the real camera in some parameters are less accurate because it can be the factor of internal camera parameters or when performing the calibration using check board.

6.3.3 Multiple Object

To test the accuracy of the algorithm for determining the topmost object, we set the condition by using the multiple objects of the same kind. For the multiple objects case, we change the distance between the object and the camera by setting the object distance closer to the camera, that is 100 mm. It is because with the multiple objects we have more lines detected, by setting the distance to be closer, the algorithm works well because it only focuses to the query image lines which correspond to the template lines.

There are 2 conditions that we test, that are using an in-plane template (the pose of the template is an in-plane case) and an out-of-plane template (the pose of the template is an out-of-plane, that is rotated with respect to its x-axis)). For the case when using an in-plane template, it is shown in Figure 48, Figure 49, and Figure 50.

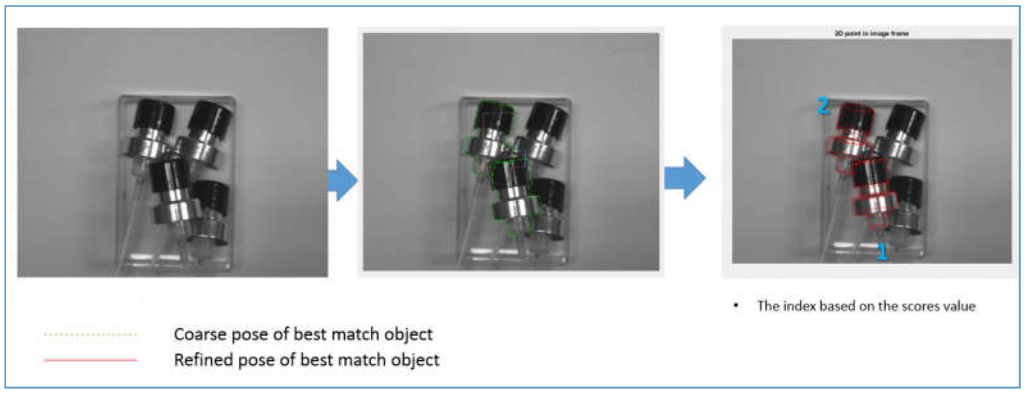


Figure 48. The Topmost Object Selection Test In-Plane Case (1)

Figure 48 shows that on the right side (final result) the index number represents the order of the score value. Meanwhile for the scoring value, it is summarized in Table 7.

INDEX	SCORE
1	0.6694
2	0.6657
3	0.6651

Table 7. The Score of In-Plane Case (1)

The highest scoring value reaches 0.6694 out of 1 because the visible edge of the query image is also captured the edge line which is not belongs to the object (illustrated in Figure 49).

Recall that the score value is the average of the difference in cosine over all the points (with respect to the template lines point). If the visible edge line in the query image is not only in the object's edge (Figure 49), it will make the score

value has lower average matched points because we have points which are not matched. Recall the Formulation (5.12) which also can be stated that the topmost object has the gradient and orientation coincide with respect to the projected template lines. So we can say that the scoring step also depending the quality of the captured image (the query image). Figure 49 shows the query image with the visible edge line is performed in blue line.

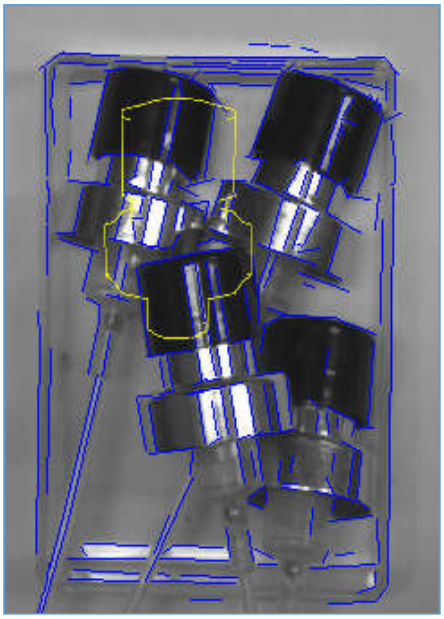


Figure 49. The Visible Edge Line of The Query Image

Another test is shown in Figure 50 with using 3 objects.

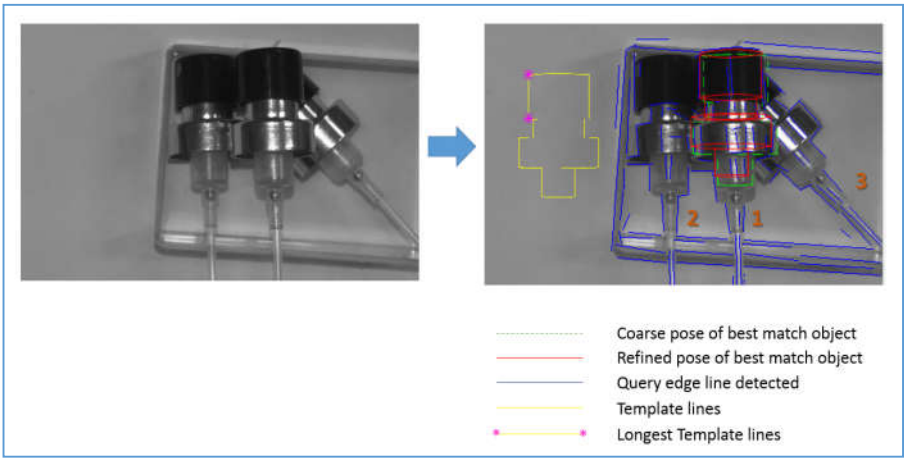


Figure 50. The Topmost Object Selection Test In-Plane Case (2)

When we test the algorithm using condition in Figure 50, it also works to select the topmost object which indicated with the red line. The index number of the right side in Figure 50 indicates the order of the scoring value. Index number 1 indicates the topmost object. Regarding the score, it is summarized in the Table 8.

INDEX	SCORE
1	0.6592
2	0.6438
3	0.6371

Table 8. The Score of In-Plane Case (2)

Another test with more objects is performed in Figure 51.

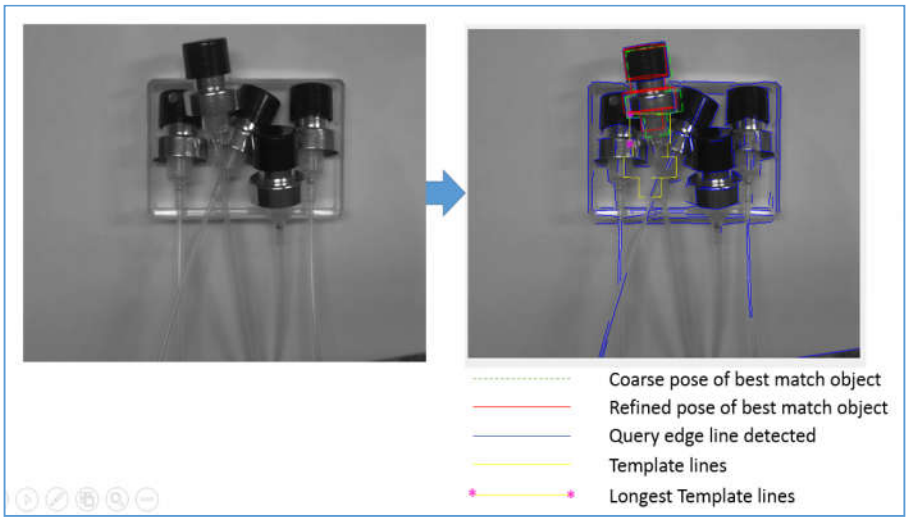


Figure 51. The Topmost Object Selection Test In-Plane Case (3)

In Figure 51, the object with the red line shows that it has the highest score among the others which means it has highest matching value compared with other objects, and it is the topmost object. The score for the in-plane case (3) is performed in Table 9.

INDEX	SCORE
1	0.6647
2	0.6377
3	0.6115

Table 9. The Score of In-Plane Case (2)

The next test is when the topmost object is out-of-plane. So, we use another template by only changing the degree of the rotation in the outline mode step. Figure 52 shows a test for out-of-plane case of multiple objects of the same kind.

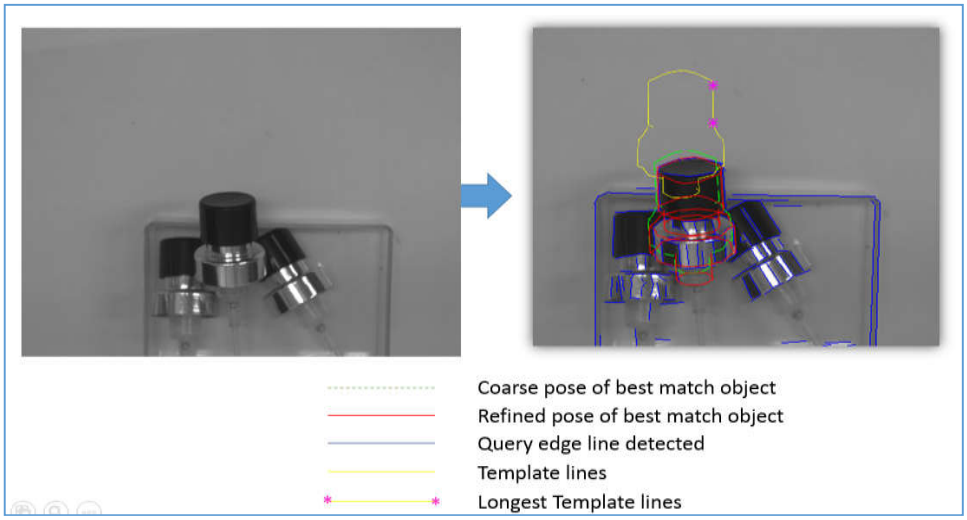


Figure 52. The Topmost Object Selection for Out-of-Plane Case

More or less the same as previous explanation. The red line shows the object which has the highest score ad becomes the topmost object. The degree of the out-of-plane rotation in this case is 30°. The score for the out-of-plane case is performed in Table 10.

INDEX	SCORE
1	0.6484
2	0.6243
3	0.588

Table 10. The Score for The Out-of-Plane Case

Conclusion

The main objective of this thesis are to have the algorithms for a 3D rotational symmetric object which creates a flexible 3D template generation in various pose both in-plane and out-of-plane then by using the generated template, it can detect the 3D pose of the object accurately and select the topmost object (best selection to be manipulated) among the multiple object of the same kind. Throughout the chapters it can be concluded that,

By using a 3D CAD template as an input then apply the Outer Line Mode method, qualitatively it gives the result to have an easy auto generated 3D template which provides only the outer line template with respect to camera view, with the experiment performed by changing the angle of rotation with respect to its x-axis with the out-of-plane rotation bounded [-90,90].

The Directional Chamfer Matching algorithm together with the Line Segment Detector method and an optimization using Levenberg-Marquardt method returns an accurate detection of a 3D pose object and the thresholding system based on the median of DCM cost to solve the missing edges problem. The experiment in the synthetic image using *Blender* software performed an average error for the translation reached 0.23 mm meanwhile for the rotation of an in-plane case reached 0.7° and the rotation of an out-of-plane case reached 1.7°. The experiment of the real camera image showed an average error for the translation achieved 0.46 mm then the average error of the rotation in-plane case reached 1.26° and the rotation of an out-of-plane case reached 3.03°.

The Gradient and Orientation Score algorithm for selecting the topmost object by selecting the highest score which indicate the coincidence of the gradient and orientation between the template line segments and the visible query image line segments by calculating the average of the difference in the cosine over all the points (with respect to template lines), so the quality of the query image also have a contribution to determine the score. As the experiment's result, quantitatively, the average of the topmost object score is 0.67 out of 1.

Bibliography

[i] Hartley, R. (2004). *Multiple View Geometry in Computer Vision Second Edition*. Cambridge University.

[ii] Zhang, Z. (2000). A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22, No. 11, pp. 1330–1334.

[iii] Zeng, Z., Yan, H. (2006). *Hidden Line Removal for 2D Cartoon Images*. School of Electrical and Information Engineering. University of Sydney.

[iv] Yeh-Liang Hsu. (2010). YZU Optimal Design Laboratory.

[v] Foley,J., et al,. (1990). Computer Graphics : Principle and Practice. USA.

[vi] Prasolov, Victor V. (2005). *Polynomials*. Springer.

[vii] von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. (2008). Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):722–732.

[viii] Agnes Desolneux, Lionel Moisan, Jean-Michel Morel. (2000). Meaningful Alignments. International Journal of Computer vision, vol. 40, no. 1, pp. 7-23

[ix] Barrow, H., Tenenbaum, J., Bolles, R., and Wolf, H. (1977).Parametric correspondence and chamfer matching:two new techniques for image matching. *5th international joint conference on Artificial intelligence*-Volume 2, pages 659–663. Morgan Kaufmann Publishers Inc.

[x] Sakcak,B., Bascetta,L., and Ferretti,G. (2016). Model based Detection and 3D Localization of Planar Objects for Industrial Setup. *ICINCO 13th International Conference on Informatics in Control, Automation and Robotics,* pages 360 - 367.

[xi] Felzenszwalb, P. and Huttenlocher, D. (2004). Distance transforms of sampled functions. Technical report, Cornell University

[xii] Liu, M.-Y., Tuzel, O., Veeraraghavan, A., Taguchi, Y., Marks, T. K., and Chellappa, R. (2012). Fast object localization and pos

[xiii] Beleznai,C., Bischof,H.(2009). Fast Human Detection in Crowded Scenes by Contour Integration and Local Shape Estimation, pages 2246-2253.*IEEE.*

[xiv] Simoncelli,E., Dew, N. (2017). Least Squares Optimization. *Center for Neural Science and Courant Institute of Mathematical Sciences*

[xv] Gavin, Henri P. (2017). *The Levenberg-Marquardt Method for Nonlinear Least Squares Curve Fitting Problems*. Department of Civil and Environmental Engineering, Duke University.

[xvi] M.I.A. Lourakis. (2005). A brief description of the Levenberg-Marquardt algorithm implemented by levmar*, Technical Report,* Institute of Computer Science, Foundation for Research and Technology – Hellas.

[xvii] K. Madsen, N.B. Nielsen, and O. Tingleff. (2004). *Methods for Nonlinear Least Squares Problems*. Technical Report. Informatics and Mathematical Modeling, Technical University of Denmark.

[xviii] D.W. Marquardt.(1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics,* 11(2):431-441,.

[xix] Pretto, A., Tonello, S., and Menegatti, E. (2013). Flexible 3d Localization of Planar Objects for Industrial Binpicking with Monocamera Vision System. *IEEE International Conference on*, pages 168–175. IEEE.

[XX] Piccinini,P. et al. (2012). Real-Time Object Detection and Localization with SIFT-Based Clustering. *ScienceDirect Journal*. *Image and Vision Computing*, page 573-587.

[XXI] Ihrke, I., eat al. (2008). *Transparant and SpecularObject Reconstruction*. The Eurographics Association.