AUTOMATIC TRACKING OF MOVING OBJECT WITH EYE-IN-HAND ROBOT MANIPULATOR

[1]JIN-SIANG SHAW, [2]WEN-LIN CHI

Institute of Mechatronic Engineering, Taipei Tech, Taipei, Taiwan
Email: [1]jshaw@ntut.edu.tw, [2]tony455448@gmail.com

Abstract: In this paper, tracking strategies of a moving object using visual servoing are presented with an eye-in-hand robot arm. First of all, CAMshift (Continuously Adaptive Meanshift) algorithm is employed to track continuously the moving object in the image plane. Then by comparing the tracking window from CAMshift and the rectangle generated by the minimum rectangle method after back projection of the object model, a new rectangle is formed and can be seen as the object itself. Finally, object features for tracking purpose can be extracted from the new rectangle. Furthermore, through application of image Jacobian matrix, the tracking error in the image plane can be transformed to be the displacements of robot's end effector. Accordingly, robot arm can be moved for tracking this moving object. In addition, trajectory planning of robot arm with blending technique is used for smoothly and continuously tracking the moving object. The developed object tracking strategies using eye-in-hand robot arm can be applied to an assembly task in a moving conveyor without knowing the conveyor speed or having to stop the conveyor for assembly.

Keywords: Automatic Tracking, Visual Servoing, Eye-in-Hand

I. INTRODUCTION

The focus of our work is to achieve translation and rotation information with moving object in real time via eye-in-hand robot arm. In this paper, we attached a camera to a gripper module and fix it on the end effector of a robot arm. Eye-in-hand architecture is used primarily to guide end effector and ensure that the tool properly engages the intended target. Like the work in [1], the application is peg-and-hole alignment. It needed to keep detecting whether the target is on the desired position and to react quickly for doing the job successfully.

Visual servoing plays an important role in the robot applications. It mainly applies with moving objects in unknown environment. Image based visual servoing (IBVS) is an important control technique used for solving the complex problems of a 6-DOF robot control for object tracking. Using image Jacobian matrix, we can transfer the image space control errors into errors in Cartesian space, and then with a suitable control algorithm, it is possible to obtain the local convergence to a desired set point. Furthermore, using prediction approach can make the result more robust [2]. The performance of tracking static object is good in IBVS, so we try to use this method to track moving object. In this paper, we set the desired point to be the position between the two finger tips of a robot arm so that we can grasp the target in the future. The image Jacobian matrix depends on the camera parameter, visual feature coordinates and depth of target with respect to camera frame. Using transform coordinate frame, details in computing image Jacobian matrix can be found in [3]. In this paper, we tend to track the target in 3-dimensional space, hence two feature points are needed. The reason we will explain section 2.

In order to acquire 2 feature points of the object, we need to use robust method of machine vision to keep the feature points detected when the target object is moving. CAMshift is an object tracking algorithm proposed by Bradski [4] that used color histogram as its target model. This algorithm may not be easily influenced by the changes in the shape of the target. It usd color probability so that it can efficiently solve the problem that the target is moving or partly sheltered. Although it can achieve the purpose of the target tracking fast, CAMshift only relies on back projection. Therefore, it would fail in some cases. For example, the object's color is changed by environment or similar color objects are detected and hence influenced the result. Moreover, it would lose the target. Therefore, [5] and [6] also used other control methods or combined other algorithms to improve this problem and make it more robust.

THE PROPOSED APPROACH

A 6-axis robot arm TX60L from Staubli, with a gripper module having a camera on it, is attached to its end effector, forming an eye-in-hand robot manipulator [7]. For the Staubli robot manipulator, only the point-to-point control command is available to users and will be used here to track moving objects on a conveyor. Therefore, we first need to detect the object and extract features by using digital image processing techniques [8].

There are two major methods we used in digital image processing. The first one is CAMshift algorithm and the other is minimum area rectangle method.

2.1 CAMshift Algorithm

CAMshift is an object tracking algorithm. When the target is chosen, the track window would keep following it. CAMshift uses color information to track the moving target. It is mainly based on back projection calculation and mean shift algorithm. CAMshift uses color histogram as its target model.

Extracted image contains many number of pixels and each pixel has associates with a set of values of Hue(H), Saturation(S) and Value(V). Through back projection, we can generate color probability of this image from the Hue histogram distribution. Then, it would come out a gray scale image which means the lighter pixel is the most possible with the target model. CAMshift uses an elliptical search window and the calculation of window's location is a converging process. Each time the new window is computed according to the window of the last frame. The location of new window is determined by the difference of the two windows and it needs to be smaller than the pre-set threshold. Fig. 1 shows the block diagram of CAMshift algorithm. There are some formulas which show how to calculate the centre's coordinate of the search window as follows

0-order moment: $M_{00} = \sum_x \sum_y I(x,y)$

1-order moment: $M_{01} = \sum_x \sum_y yI(x,y)$

$$M_{10} = \sum_x \sum_y xI(x,y)$$

2-order moment:

$$M_{20} = \sum_x \sum_y x^2 I(x,y)$$

$$M_{02} = \sum_x \sum_y y^2 I(x,y)$$

$$M_{11} = \sum_x \sum_y xyI(x,y)$$

where $I(x,y)$ is each pixel's value in back projection. Then the centre's coordinates are given as

$$x_c = \frac{M_{10}}{M_{00}}, \quad y_c = \frac{M_{01}}{M_{00}} \tag{1}$$

Using the coordinates of center point to build an ellipse as the search window with length L, width W and angle θ with respect to x axis, we obtain

$$L = \sqrt{\frac{a+c+\sqrt{b^2+(a-c)^2}}{2}} \tag{2}$$

$$W = \sqrt{\frac{a+c-\sqrt{b^2+(a-c)^2}}{2}} \tag{3}$$

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{b}{a-c}\right) \tag{4}$$

where

$$a = \frac{M_{20}}{M_{00}} - x_c^2, \quad b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2$$

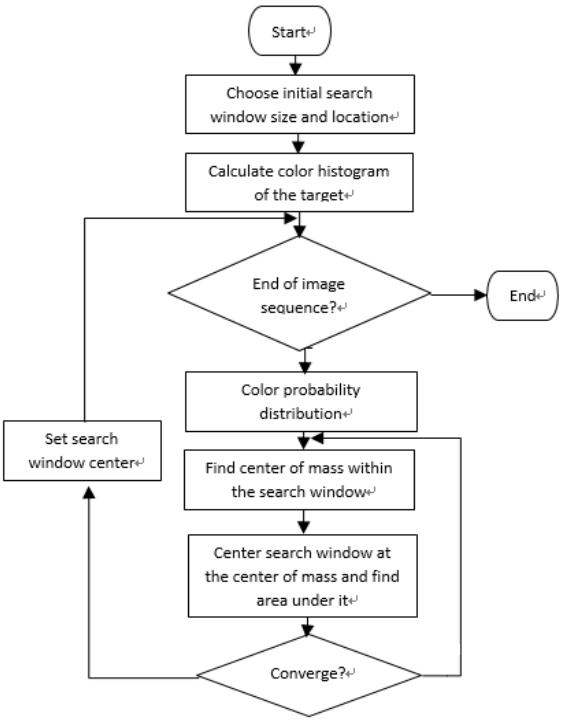


Fig.1. Block Diagram of CAMshift algorithm

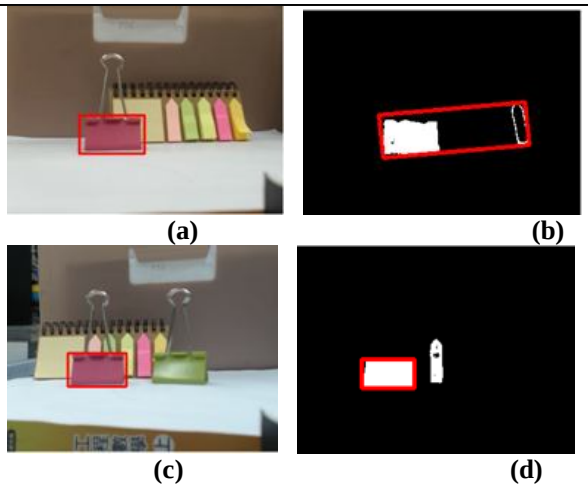
2.2 Minimum Area Rectangle

After using CAMshift, we only acquire track window which is a rectangle due to using Emgu, which is a cross platform .NET wrapper to the OpenCV image processing library. This track window can't represent the target, but we do know where the target is. The main goal is to find 2 robust feature points in the object so that visual servoing has enough parameters to work with. In this paper, we use minimum area rectangle method which is one type of bounding rectangle and it has a function to call in Emgu. It can generate the minimum rectangle by a set of points so the rectangle may rotate. Therefore, we use minimum area rectangle from back projection image and generate a rectangle to represent target object because the rectangle has enough information about the rotation and the size of the object if the image is clear. However, back projection may have some noises which is the problem of color probability.

We combine track window which is generated by CAMshift algorithm and minimum area rectangle to generate new rectangle which is more robust and have useful information of the object. We define the way to determine a point (x,y) is true point of the object or noise in the back projection by Euclidean distance to track window:

$$(x-x_g)^2 + (y-y_g)^2 \le \left(\frac{width}{2}\right)^2 + \left(\frac{height}{2}\right)^2$$

where x, y are coordinates in pixels in image plane; (x_g, y_g) are center point of the track window and *width*, *height* are the width and height of the track window.



(a) **(b)** **(c)** **(d)**

Fig.2. generate rectangle in Back Projection

(a) original image with track window
(b) use only Minimum Area Rectangle from Back Projection of Fig.1(a)
(c) original image with track window
(d) use track window and Minimum Area Rectangle from Back Projection of Fig.1 (c)

Fig. 2 illustrates results of applying CAMshift and minimum area rectangle algorithms. It is clearly seen that if we only use CAMshift algorithm, the track window is not good enough to represent the object. If we only use minimum area rectangle, the rectangle would be easily influenced by noises. But if we combine both, the result is good. Finally, we set the two feature points as the middle points in the long side of the rectangle.

2.3 Visual Servoing

In this paper, image based visual servoing is employed, which is based on the error between current and desired feature points on the image plane. The goal of the visual servoing process is to minimize the image control error

$$e(k) = f_d - f(k) \tag{5}$$

so that the image feature f can reach the desired target f_d .

In the IBVS structure, there is a feature space control law which uses the error as input and generates relatively robot command. In the following computation, we would find out the relationship between the changing rate of feature point in the image space and end effector velocity in the task space. First, we assume the camera is in the static environment. When the camera moves in the task space with a linear velocity $T = [T_x, T_y, T_z]$ and angular velocity $\Omega = [\omega_x, \omega_y, \omega_z]$. The velocity of a point P relative to the camera frame can be expressed as

$$\frac{dP}{dt} = -\Omega \times P - T \tag{6}$$

Using classical perspective projection model,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{z} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7}$$

where λ is the focus distance, (u,v) is the feature point coordinate in the image plane, and (x,y) is coordinate in space. In equation (7), the unit of (u,v) is length, but we acquire (u,v) in the unit of pixel. Using transformation to pixel units, equation (7) becomes:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{z} \begin{bmatrix} x / s_x \\ y / s_y \end{bmatrix} \tag{8}$$

where s_x, s_y are transformation coefficient of x, y axis in the image plane respectively. s_x, s_y and λ are associated camera parameters. We can measure them through experiments. To simplify equation (8), we define

$$\lambda_x = \lambda / s_x, \quad \lambda_y = \lambda / s_y$$

Equation (8) now becomes

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x\lambda_x \\ y\lambda_y \end{bmatrix} \tag{9}$$

The relationship between an image point and the velocity of P can be calculated as follows. Let $P = [x, y, z]$ be coordinate relative to camera frame. According to equations (6) and (9). The derivative of coordinate P in terms of image feature coordinate u,v is,

$$\begin{aligned} \dot{x} &= -z\omega_y + \frac{zv}{\lambda_y}\omega_z - T_x \\ \dot{y} &= -\frac{zu}{\lambda_x}\omega_z + z\omega_x - T_y \\ \dot{z} &= -\frac{zv}{\lambda_y}\omega_x + \frac{zu}{\lambda_x}\omega_y - T_z \end{aligned} \tag{10}$$

Derivative of coordinates of feature parameters in terms of P are

$$\begin{aligned} \dot{u} &= \frac{-\lambda_x}{z}T_x + \frac{u}{z}T_z + \frac{uv}{\lambda_y}\omega_x - \frac{\lambda_x^2 + u^2}{\lambda_x}\omega_y + \frac{\lambda_x}{\lambda_y}v\omega_z \\ \dot{v} &= \frac{-\lambda_y}{z}T_y + \frac{v}{z}T_z + \frac{\lambda_y^2 + v^2}{\lambda_y}\omega_y - \frac{uv}{\lambda_x}\omega_y - \frac{\lambda_y}{\lambda_x}u\omega_z \end{aligned} \tag{11}$$

Rewrite the above equation (11) in vector-matrix form as

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix}=\begin{bmatrix}\frac{-\lambda_x}{z} & 0 & \frac{u}{z} & \frac{uv}{\lambda_y} & \frac{-\lambda_x^2-u^2}{\lambda_x} & \frac{\lambda_x}{\lambda_y}v\\ 0 & \frac{-\lambda_y}{z} & \frac{v}{z} & \frac{\lambda_y^2+v^2}{\lambda_y} & -\frac{uv}{\lambda_x} & -\frac{\lambda_y}{\lambda_x}u\end{bmatrix}\begin{bmatrix}T_x\\ T_y\\ T_z\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \quad (12)$$

Equation (12) can be written as

$$\dot{f} = J_{img} V_p^c \quad (13)$$

where J_{img} is the image Jacobian matrix, V_p^c is velocity of p point with respect to camera frame. In this paper, we need to control robot manipulator in 3 dimensional space, hence at least 2 feature points are required. In addition, we further assume that $T_z = \omega_x = \omega_y = 0$ for in-plane object tracking. Consequently, the final feature points in image plan can be obtained

$$\begin{bmatrix}\dot{u}_1\\ \dot{v}_1\\ \dot{u}_2\\ \dot{v}_2\end{bmatrix}=\begin{bmatrix}\frac{-\lambda_x}{z} & 0 & \frac{\lambda_x}{\lambda_y}v_1\\ 0 & \frac{-\lambda_y}{z} & -\frac{\lambda_y}{\lambda_x}u_1\\ \frac{-\lambda_x}{z} & 0 & \frac{\lambda_x}{\lambda_y}v_2\\ 0 & \frac{-\lambda_y}{z} & -\frac{\lambda_y}{\lambda_x}u_2\end{bmatrix}\begin{bmatrix}T_x\\ T_y\\ \omega_z\end{bmatrix} \quad (14)$$

Because this image Jacobian matrix is not a square matrix, we can get the relationship below through pseudo inverse matrix

$$V_p^c = J_{img}^+ \dot{f} \quad (15)$$

where $J_{img}^+ = (J_{img}^T J_{img})^{-1} J_{img}^T$. Because velocity of P is relative to camera frame, so we can set camera frame on the tool frame due to tool frame and camera frame moving together. Then, we can see the velocity of P as the displacement command of the robot end effector. In this paper, the displacement command of end effector also needs to transfer the tool frame to the global frame of robot manipulator. Using forward kinematics and the Denavit-Hartenberg convention, we can know each of coordinate transformation matrix from the previous coordinate system to the next coordinate system on the each joint of robot. Multiply all transformation matrix, we can know the coordinate transformation from inertial coordinate to end effector coordinate

$$T_6^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5$$

Transfer the position of P with respect to the tool frame P^6 to the inertial frame P^0

$$P^0 = T_6^0 P^6$$

we can know the relationship between tool frame and inertial frame of robot manipulator.

Finally, send this displacement command to robot arm controller for tracking the target object. It is noted that the *x*,*y* axis of image plane needs to check whether they are in the same directions as *x*,*y* axis of tool frame in the perspective projection or not. Fig. 3 shows a block diagram for image based visual servoing control for.

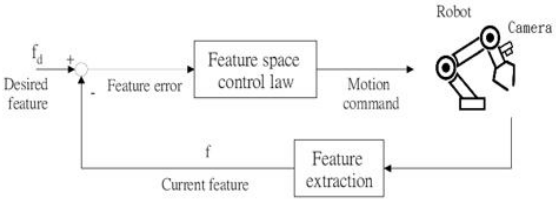


Fig.3. Image Based Visual Servo

After introduction of machine vision based digital image processing and visual servoing approaches, the overall flow chart for object tracking using an eye-in-hand robot manipulator is shown in Fig. 4.

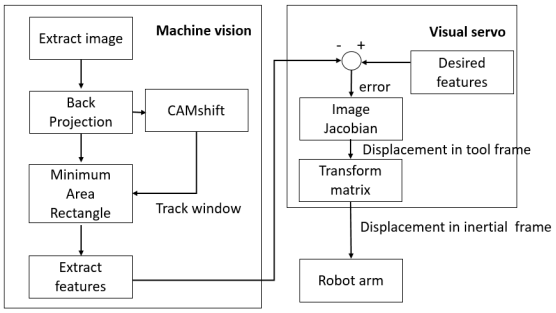


Fig.4. Block diagram of the tracking system

2.4 Blending Move

In this paper, we use a robot arm from Staubli company. There is a movement control command called blending move in the controller. Next, we will explain how this function is defined. Assume there is a typical pick-and-place task: picking up at pPick point and putting down at the pPlace point, as shown in Fig. 5. The robot is initially at the pPick point. Straight line movements are used for disengagement and approach. However, the main movement is point-to-point movement, as the geometry of this part of the trajectory does not need to be accurately controlled, because it is desirable to move robot arm as quickly as possible.

In the absence of any specific movement requirement, the robot will stops at the pDepart and pAppro points, as the trajectory is sharply turning at these points. This unnecessarily increases the duration of the operation and there is no need to pass through these

points precisely. The duration of the movement can be significantly reduced by "blending" the trajectory in the vicinity of the pDepart and pAppro points. In the blending move, there are two parameters which are called leave and reach respectively. Leave is the distance from arrival point to start blending point. Reach is the distance from arrival point to end blending point.

Once we know these two parameters for blending, we can add blending move to straight line movements. In this way, the robot manipulator no longer stops at the pDepart and pAppro points. This movement is therefore faster and smoother than the point-to-point movement.

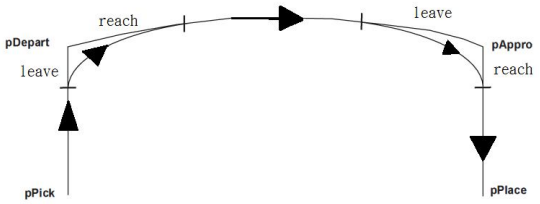


Fig.5. Pick and place trajectory (with blending)

EXPERIMENTAL RESULTS

3.1 Experimental Setting

We place target objects on a continuously operating conveyor and make sure the object movement is in the workspace of the robot arm. Then, we attached a gripper module with a camera to the end effector. We assume the tracking movement has same height at 390 mm from camera to object. We also need to set the desired points at coordinate (191,158) and (145,158) which are the positions of the two fingure tips. In this study, we track object moving in *x*-*y* plane and the robot can also rotate about the *z* axis if the object is randomly orientated.

In addition, we need to measure λ_x, λ_y mentioned earlier. In this paper, we use background subtraction method and edge detect method to easily know the pixels of the object. Then, measure the length, width of object and the distance from camera to object. Through equation (9), we can acquire the values for λ_x, λ_y.

3.2 Results

Fig. 6 shows time response of the error of feature points. Before the tracking started, we can see the error of *u* which is *x* axis of image plane was monotonically decreasing and the error of *v* which is *y* axis of image plane didn't have obviously change. When the tracking was started, the error of *v* suddenly decreased and was close to 0 pixel. However, the error of *u*, trying to converge, can't go to 0 pixel because the object was moving. It maintained the error around 40 pixels that means robot arm kept tracking the object and didn't lose it.

Fig. 7 shows the processes of robot arm tracking a moving object: (a) we put the object on the conveyor; (b) we selected the target and the robot began to track; (c),(d),(e) robot kept following the object;(f) track mode was closed at the end of conveyor.

In Fig.8 we can see that after track mode started, robot tried to move toward the desired position. At the end, the error of *v* was close to 0 pixel and only had error of *u* in the *x* axis because the object is moving with a velocity. Moreover, we can find the value of *u* was changing in a bounded small area, meaning the robot was able to track the moving object without divergence. Trajectory of the robot arm tracking a moving object is shown in Fig. 9.

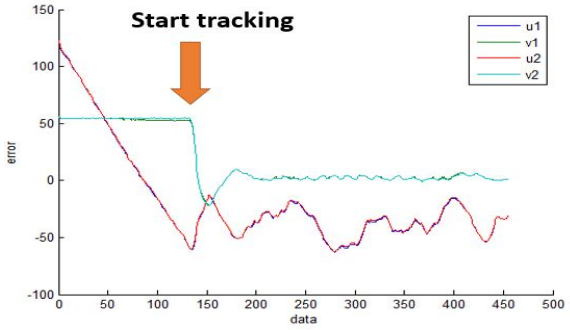


Fig.6. Error of feature points in the image plane

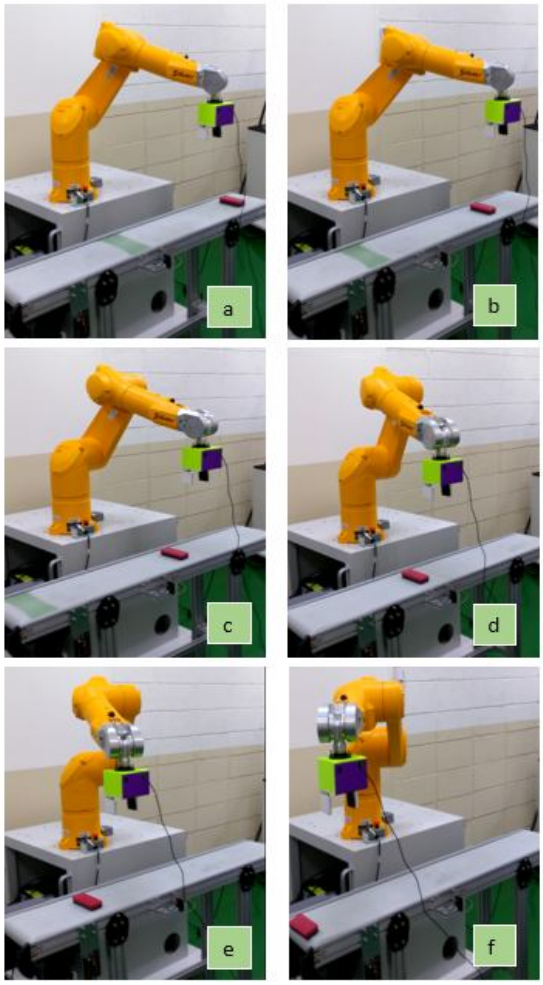


Fig.7. Snapshots of robot arm tracking a moving object

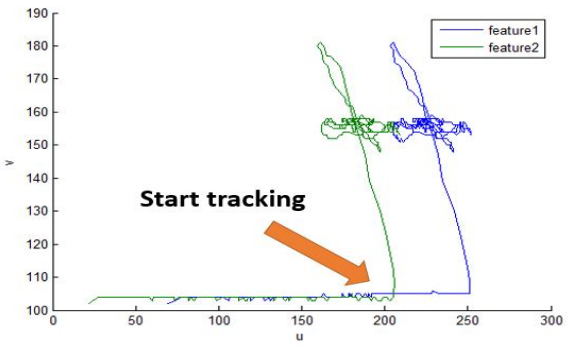


Fig.8. Path of feature points in the image plane

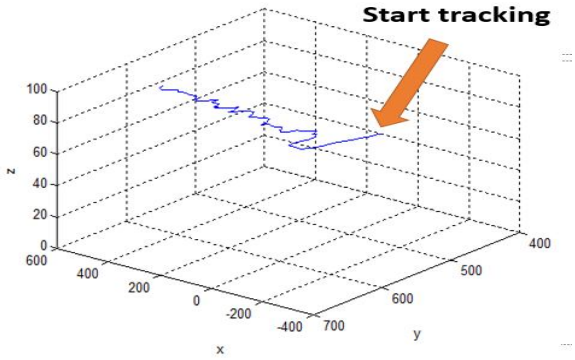


Fig.9. Trajectory of robot arm

3.3. Discussion

During the experiment, we found that if we didn't use blending movement, the robot arm might not catch up the moving object in high speed movement and the error of *u*, *v* would be bigger than with blending. Because the camera extracts the image much faster than robot movement, we can't keep sending all feature errors to robot controller. It would have the problem of stacking commands because we use point to point movement no matter we use the movement with blending or not. Therefore we use an asynchronous method to handle the two things: one is to keep extracting image. The other is to send robot command only if the last movement is finished over 50%.

CONCLUSIONS

In this paper, we combined CAMshift algorithm, minimum area rectangle method and visual servoing for tracking moving object with an eye-in-hand robot manipulator. In order to make feature points more robust, we generated new rectangle to represent the target object itself from track window by CAMshift algorithm and minimum area rectangle in back projection image. Although the robot arm control only use point-to-point movement, robot arm still move smoothly with blending.

In future works, the robot gripper is to grasp the moving object by using some predictive control methods, like Kalman filter or adaptive linear predictor. By employing these predictive control methods, a moving object can be successfully grabbed as soon as possible by the robot arm.

REFERENCES

1. Shouren Huang, Yuji Yamakawa, Taku Senoo and Masatoshi Ishikawa, "Realizing Peg-and-Hole Alignment with One Eye-in-Hand High-Speed Camera", 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp.1127-1132 July 2013.
2. C. Lazar and A. Burlacu, "Visual Servoing of Robot Manipulators Using Model-based Predictive Control", 7th IEEE International Conference on Industrial Informatics June 2009.
3. Chung-Hsien Yeh, "Grasping Control of a Mobile Manipulator", Department of Electrical and Control Engineering in National Chiao Tung University, pp1-57 July 2006.
4. G.R. Bradski, "Computer Vision Face Tracking for Use in a Perceptual User Interface", Intel Technology Journal, 2nd Quarter, 1998.
5. Saket Joshi, Shounak Gujarathi, and Abhishek Mirge "Moving object tracking method using improved CAMshift with SURF algorithm", Inernational Journal of Advances in Science Engineering and Technology, ISNN:2321-9009, pp.14–18 April 2004.
6. Sheldon Xu and Anthony Chang "Robust Object Tracking Using Kalman Filters with Dynamic Covariance" Cornell University 2014
7. J. Shaw and Vipul Dubey, "Design of Servo Actuated Robotic Gripper Using Force Control for Range of Objects," International Conference on Advanced Robotics and Intelligent Systems, Taipei, Taiwan, Aug. 31~Sept. 2, 2016.
8. R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison Wesley, 2009.

★★★