ME 650 - Advanced Manipulators
Puma 560 Visual Servoing

Nathaniel Burgdorfer
Stevens Institute of Technology
Hoboken, New Jersey, USA
nburgdor@stevens.edu

1. Introduction

Visual servoing is the process of utilizing the information captured by images from positioned cameras in the workspace of a robotic manipulator for the purpose of driving the motion of the robot. For a gentile introduction to visual servoing, we will begin by briefly discussing the choice of camera placement in relation to the manipulator, followed by the introduction of two different solution paradigms: position-based visual servoing and image-based visual servoing.

The choice of camera placement can be driven by the overall desired task of the system, or the specific paradigm that is being implemented. There are mainly two types of camera placements used in visual servoing. The first is labeled Eye-To-Hand visual servoing. This specifies a static camera at some vantage point in the workspace that has a view of the overall workspace, any objects of interest, and the manipulator. The second is labeled Eye-in-Hand visual servoing. In this configuration, the camera is mounted on the end-effector of the robot, allowing the camera to have a more direct vantage point of any objects of interest, reducing the number of possible object occlusions, while introducing a dynamic camera pose. In this work, we will be focusing on the latter camera configuration.

Heavily based on the desired end task, there are two choices of visual servoing algorithms. The first is position-based visual servoing and the second is image-based visual servoing.

Position-based Visual Servoing (PBVS) involves extracting information from images in order to recover or estimate object and camera pose. This method performs inference in a 3D space, generating pose errors in Cartesian space using a desired 3D pose. This method is more suited for tasks, such as grasping, that require precise 3D pose estimation in order to drive the robot motion.

Image-based Visual Servoing (IBVS) is a method that operates in a 2D space. Errors are computed directly in image space using the extracted features of the image. Robot movement in this method is driven by minimizing the current observed features with a set of desired features. There is some information loss in using an IBVS method, as orientation on some objects may be ambiguous, since the goal is to only obtain a desired image. With that in mind, it is quite simple to implement and works well for visual tasks, such as tracking.

2. Derivations

In this work, we will mainly be focusing on an IBVS method using an eye-in-hand system. The essential mathematical formulation for image-based visual servoing is the formulation of the image Jacobian. To construct the image Jacobian, the focal length $\hat{f}$, depth to object Z, and pixel coordinates (u, v) are needed.

$$J_I = \begin{pmatrix} \frac{-\hat{f}}{Z} & 0 & \frac{u}{Z} & \frac{uv}{\hat{f}} & -(\frac{\hat{f}+u^2}{\hat{f}}) & v \\ 0 & \frac{-\hat{f}}{Z} & \frac{v}{Z} & \frac{\hat{f}+v^2}{\hat{f}} & -\frac{uv}{\hat{f}} & -u \end{pmatrix} \quad (1)$$

This matrix relates the camera velocity (position and orientation) to the feature velocities, sometimes referred to as the pixel velocities, as follows:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = J_I \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \quad (2)$$

Since we measure the pixel velocities, we can use the pseudo-inverse of the image Jacobian to obtain the camera velocities. In order to drive the robot motion, we need to then relate the camera velocities to the joint velocities. The first step is to transform the camera velocities into end-effector velocities. This is done here with the transformation "Jacobian", J_c^{ee}.

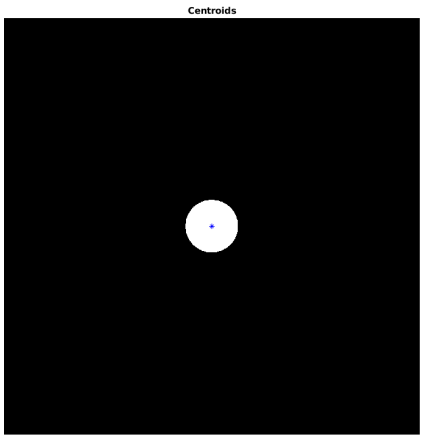


Figure 1. Centroid Feature Descriptor

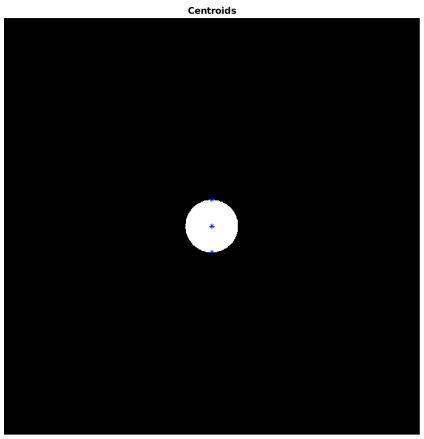


Figure 2. Center Line Feature Descriptor

$$R_{cam}^{ee} = R_{ee}^{T} R_{cam} \tag{3}$$

$$J_I = \begin{pmatrix} R_{cam}^{ee} & 0 \\ 0 & R_{cam}^{ee} \end{pmatrix} \tag{4}$$

where R_{ee} and R_{cam} are the rotation matrices in end-effector and camera frame, respectively. After transforming the velocities into the end-effector reference frame, we can then use the robot Jacobian to relate the end-effector velocities to joint velocities. The full equation is depicted as follows:

$$\dot{q} = kJ^{+} J_c^{ee} J_I^{+} \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} \tag{5}$$

Here, M^{+} represents the pseudo-inverse of a matrix M.

3. Implementation

The task that we will be focusing on with this work is the task of continually tracking an object in space. We will be using a red sphere in a simulated environment.

3.1. Basic Tracking

In the first iteration of performing this task, we will extract a single, simple feature from the image; the centroid of the sphere. Since this is a toy example, the segmentation process and centroid computations are relatively trivial and will not be expanded upon here. Once we have our single feature descriptor, we can use the pixel coordinates of the centroid to minimize the error between the measured features locations and the desired feature location. Essentially, we can drive the robot to move the end-effector in such a way to center the sphere in the camera view. The feature descriptor used in this iteration is shown in Figure 1.

3.2. Increasing Descriptors

The trajectory for the simple centroid tracking worked effectively to center the sphere in the image; however, the single feature descriptor is not enough to constrain the robot motion from moving too close or too far from the object. This formulation cannot overcome the scale ambiguity of just a singular projected feature in the image plane. To help better maintain a relative distance to the sphere, we introduce a three-descriptor formulation. Here, we use the centroid of the projection of the sphere in the image plane, as well as the extreme y-axis coordinates. The feature descriptors used in this iteration are shown in Figure 2.

3.3. Resolved Rates & Modified Descriptor Structure

Using the centroid and the bounding pixels along the y-axis end up causing non-optimal convergence, as well as continued (yet reduced) scale ambiguity due to all the features existing on the same line in the feature space. In order to produce non-col-linear features, we modify the feature extraction to take a triangular pattern around the circumference of the projection of the sphere into the image plane. This helps reduce the scale ambiguity, allowing the robot to not only converge to its desired position quicker, but also reduces some noise in maintaining the relative distance to the sphere. The feature descriptors used in this iteration are shown in Figure 3.

The second adjustment to the algorithm used in this iteration is the introduction of a resolved-rates approach to the end-effector velocity. Instead of using the direct velocity transformations, we compute the normal of the end-effector velocity to obtain the direction of change and produce the velocity magnitude based on the feature error. The modified formulation is the following:

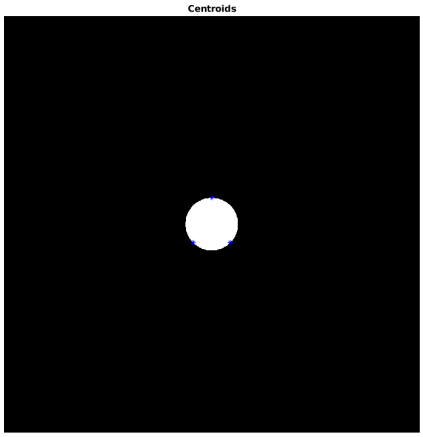


Figure 3. Triangular Feature Descriptor

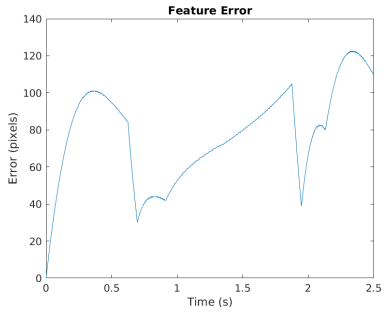
$$\dot{q} = kJ^{+}v_{rr}||J_c^{ee}J_I^{+}\begin{pmatrix}\dot{u}\\ \dot{v}\end{pmatrix}|| \tag{6}$$

4. Evaluation

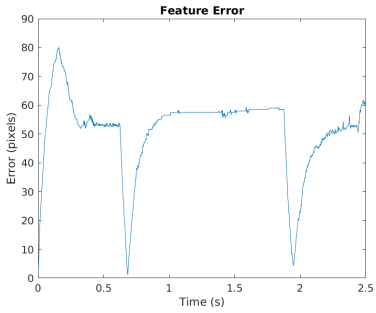
Evaluation of these iterations was done both qualitatively and quantitatively. The simulation videos where inspected to asses the overall robot movement and tracking speed. To form a quantitative evaluation, we compute the tracking error as a function of time. The errors are measured in pixels as the Mean Absolute Error (MAE) between the current measured feature pixel locations and the desired feature pixel locations. Plots for the three different algorithm versions are shown in Figure 4. As we can see, the tracking error for the first method was smooth, yet started to diverge as the simulation progressed. This is due to the unstable distance between the robot and the object. Increasing the number of feature descriptors allows us to have a more consistent tracking error, yet it is still relatively large error magnitudes and is slowly increasing. The best results are observed after we modify the feature structure, as well as the Cartesian velocity magnitude through a resolved rates algorithm. We can see our convergence is relatively quick and stable.

5. Conclusion

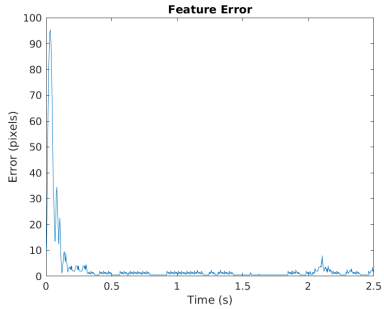
The comparison of the three different visual servoing methods for tracking an object is relatively straight forward. The single feature used in the basic tracking method lead to slow convergence and decreasing stability. Adding more features helped in both regards; however, similar convergence issues were still present. The robot failed to converge quickly and stably on the object as it moved through the space. Using a resolved rates approach increased the speed



(a) Basic Tracking



(b) Increased Descriptors



(c) Resolved Rates & Modified Descriptor Structure

Figure 4. Feature Tracking Error

of convergence and modifying the feature layout helped to properly maintain the distance to the object without the issues of the previous scale ambiguous formulations. There is still room for improvement, as the robot movement is still too jittery. We can see that in order to converge quickly, there exists small, high-frequency changes in robot pose. In future work, we look to alleviate some of these issues and produce a smooth, yet accurate tracking trajectory free of these high-frequency pose changes.