Division of Engineering Science
UNIVERSITY OF TORONTO

ROB501: Computer Vision for Robotics

Assignment #4: Image-Based Visual Servoing

Fall 2023

Overview

Image-based visual servoing (IBVS) is a popular form of closed-loop feedback control based on errors measured in image space, between the observed and desired positions of image feature points. In this assignment, you will write a simple IBVS controller. The goals are to:

- provide some additional practice deriving and using Jacobian matrices,
- introduce the matrix pseudo-inverse to solve overdetermined systems in a least squares sense,
- assist in understanding how simple proportional controllers are built and used for vision tasks, and
- determine the sensitivity of IBVS to 3D point depth uncertainty/errors.

Prior to starting the assignment, you should install the Lie groups Python library that is available from the following GitHub URL: `https://github.com/utiasSTARS/liegroups`.

The due date for assignment submission is **Friday, November 24, 2023, by 11:59 p.m. EDT**. All submissions will be in Python 3 via Autolab; you may submit as many times as you wish until the deadline. To complete the assignment, you will need to review some material that goes beyond that discussed in the lectures—more details are provided below. The project has four parts, worth a total of **50 points**.

Please clearly comment your code and ensure that you only make use of the Python modules and functions listed at the top of the code templates. We will view and run your code.

Part 1: Image-Based Jacobian

IBVS effectively operates entirely in image space (i.e., on the image plane, $\mathbb{R}^2$), that is, no explicit references to 3D are made (beyond having depth estimates for feature points). The first step is to derive the Jacobian matrix that relates motions of the camera in 3D space to motions of points on the image plane. Conveniently, most of this work has already been done for you and the result is given by Equation 15.6 in the Corke text (on pg. 544). For this portion of the assignment, you should submit:

- A function in `ibvs_jacobian.py` that computes and returns the x and y velocities of a point on the image plane, given the translational and rotational velocities of the camera and the (estimated) depth of the point.

Later, we will invert several stacked Jacobian matrices to calculate the desired camera motion. Note that, in the Jacobian computation, the image point coordinates (in pixels) must first convert to *normalized image plane coordinates* (see early lectures in the course).

Part 2: Image-Based Servo Controller

The next step is to design and code up a simple proportional controller that will moving the camera (i.e., by providing velocity commands) in an effort to reduce the distance (error) between the observed positions of several feature points in the current camera image and their desired positions (which we assume to be known in advance).

We seek to determine the translational and rotational velocities of the camera (so six numbers), and the motion of each image plane point gives us two pieces of information—hence, we require at least three points on the image plane. However, it is often desirable to use more than three points, in which case we have an *overdetermined* system; recalling the normal equations from linear least squares, we can solve our problem by computing the *Moore-Penrose pseudo-inverse* of the (stacked) Jacobian matrix:

$$\mathbf{J}^{+} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T$$

Wikipedia has a good discussion of the matrix pseudo-inverse. Note that the **J** in the equation above is formed from the stacked 2×6 sub-Jacobians for each image plane point. The complete controller is specified by Equation 15.11 in the Corke text (on pg. 548). For this portion of the assignment, you should submit:

- A function in `ibvs_controller.py` that implements a simple proportional controller for IVBS. The single gain value will be passed as an argument to the controller. The function should compute the output velocities given three or more image plane point correspondences.

Part 3: Depth Estimation

As noted in the lecture session and in the Corke book, IBVS is relatively tolerant to errors in the estimated depths of feature points. Given an initial (incorrect) set of estimated feature depths, it is possible to refine the depth values by making use of (known) camera motion and observed image plane motion (of the feature points). The Corke text provides a description of this approach on pg. 553, summarized by Equations 15.13 and 15.14. For this portion of the assignment, you should submit:

- A function in `ibvs_depth_finder.py` that produces a new set of depth estimates for feature points, given the commanded camera velocity and changes in image feature positions (i.e., a first order estimate of the image feature velocities).

Part 4: Performance Evaluation

With your IBVS system now up and running, the last step is to run a few experiments to determine its performance and overall sensitivity to errors in the estimated feature depths. Fist, you should choose a challenging initial camera pose (i.e., with a large difference in feature alignment). You may use the same test harness provided in the learner examples and just change the initial pose. Then, you should attempt to answer the following questions:

- What is the optimal gain value (experimentally) for the case where feature depths are known exactly? That is, what gain value leads to the fastest convergence?
- What is the optimal gain value (experimentally) for the case where feature depths are estimated? That is, what gain value leads to the fastest convergence?
- How much worse is the performance with estimated depths compared to with known depths? Is the difference significant.

For this portion of the assignment, you should submit:

- A single PDF document called `report.pdf` that briefly answers the questions above. The document should be a couple pages (three maximum) and include a handful of plots that show your results.

Grading

Points for each portion of the assignment will be determined as follows:

- IBVS Jacobian function – **10 points** (5 tests $\times$ 2 points per test)

  Each test uses a different image plane point and a different camera intrinsic matrix. The 12 entries in the Jacobian matrix must be exact (up to a small tolerance) to pass.

- IBVS controller function – **20 points** (5 tests $\times$ 4 points per test)

  Each test uses a different set of image plane points and a different camera intrinsic matrix. The six output values (velocities) must be exact (up to a small tolerance) to pass.

- Depth estimation function – **10 points** (5 tests $\times$ 2 points per test)

  Each test uses a different set of image plane points. The n output values (depths) must be exact (up to a small tolerance) to pass.

- PDF performance report document – **10 points** (Up to 10 points assigned for convincing results)

  The document should contain a few plots and perhaps a table or two. It **must** be a valid PDF and have the filename `report.pdf`, *otherwise it will not be graded*.

Total: **50 points**

Grading criteria include: correctness and succinctness of the implementation of support functions, and proper overall program operation and code commenting and details. Please note that we will test your code *and it must run successfully*. Code that is not properly commented or that looks like 'spaghetti' may result in an overall deduction of up to **10%**.