Analyzing PID and Visual Servoing Control for 6DOF Mobile Robotics Platforms

Jasper Eng, Sam Scholnick-Hughes
1693023, 1662610

Abstract:

Autonomous Underwater Vehicles (AUVs) are an important part of the future of deep sea exploration. As part of this exploration it is important for these AUVs to have the ability to pick up and return samples to the surface for scientists to study. We investigate using PID controllers and Uncalibrated Visual Servoing as methods for picking up these samples to determine which shows more promise to be used on an AUV.

1. Introduction

The sea remains the least explored place on earth, one of the fundamental challenges with exploring the deep sea is that wireless remote control does not work underwater due to water quickly degrading wireless signals. As such Autonomous Underwater Vehicles (AUVs) are important to gain a better understanding of underwater environments. One specific problem for autonomous vehicles is picking up small samples from a video feed, solving this problem would allow AUVs to retrieve samples and bring them back to researchers on land to study. We investigate a more simplified domain where we know the true size of the object that we wish to retrieve and investigate if proportional integral derivative (PID) controllers or Uncalibrated Visual Servoing (UVS) perform better for this task. As we did not have access to an AUV for this project we constructed a robot that can move a platform with a camera in 6 degrees of freedom. This robot was based on a stewarts platform but used string and pulleys as opposed to actuated pistons.

2. Theory

The robot takes, as input, a single video feed from a camera that is mounted on the center of the platform facing directly down. We then place 4 trackers on the corners of the object. Then, given the true size of the object and the focal length of the camera we can compute 6 error terms, 1 for each degree of freedom, as follows.

Let p1, p2, p3, p4 be the true position of the tracked corners in 4 dimensional homogeneous space relative to the center of the object and the object's rotation frame (ie. $p_i[z]=0$ for each point).

Let f be the focal length of the camera

We can define the pixel coordinates such that the center of the image is the origin for convenience. We can then find the target positions of the 4 trackers in the camera view as

$$p_i' = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} p_i$$

Let t_i denote the location of the i^{th} tracker in the camera in homogeneous coordinates.

$$Let\ c\ =\ (t_1 \times t_3)\ \times\ (t_2\ \times\ t_4)$$

$$Let\ c'\ =\ (p_1' \times p_3')\ \times\ (p_2'\ \times\ p_4')$$

c and c' denote the centers of the tracked points and target points respectively.

$$Let\ d\ =\ \frac{1}{4}\sum_{i=1}^{4}(||c[0:1]\ -\ t_i[0:1]||)$$

$$Let\ d'\ =\ \frac{1}{4}\sum_{i=1}^{4}(||c'[0:1]\ -\ p_i'[0:1]||)$$

d and d' represent the average euclidean pixel distance from the tracked points to the center of the object and the average euclidean pixel distance from the target points to the center of the target points respectively.

The platform is then in the correct position to pick up the object when $p_i' = t_i \forall i \in [1, 4]$

We use this constraint to define 6 error functions, one for each degree of freedom as follows:

The error in the z axis is the difference between the average euclidean distance from the tracked corners to the center of the object, and the average euclidean distance between the corners and center of the object in its target position.

$$z_{error} = d' - d$$

The x and y error are the x and y components of the distance that the center of the tracked object is from the center of the target object, normalized by the average distance from the corners of the tracked object to the center of the tracked object.
This normalization keeps the error terms constant with respect to changes in the z error.

$$x_{error} = (c[0] - c'[0])/d$$

$$y_{error} = (c[1] - c'[1])/d$$

We know that in projective cameras points that are farther away from the camera appear farther away from each other than points closer to the camera. We can use this fact to compare the distance between the tracked corners and the center of the object to determine the roll and pitch errors. If the two trackers at the top of the object, t_1 and t_2 are farther away from the center than t_3 and t_4 we know that the platform must be pitched up. Similarly we can find if the platform is pitched down or rolled left or right. We then define the pitch and roll error terms as follows:

$$pitch_{error} = \frac{1}{d}(||c[0:1] - t_1[0:1]|| + ||c[0:1] - t_2[0:1]|| - ||c[0:1] - t_3[0:1]|| - ||c[0:1] - t_4[0:1]||)$$

$$roll_{error} = \frac{1}{d}(||c[0:1] - t_1[0:1]|| + ||c[0:1] - t_4[0:1]|| - ||c[0:1] - t_2[0:1]|| - ||c[0:1] - t_3[0:1]||)$$

To compute the yaw error we find the location of the intersection between the line that connects t_1 and t_4 and the line that connects t_2 and t_3. We know that when the object is in the correct location both these lines should be perfectly vertical therefore we can compute the yaw error as follows:

$V = (t_1 \times t_4) \times (t_2 \times t_3)$

$$yaw_{error} = \frac{V[0]}{||V||}$$

We use the same loss functions for both the PID control and the uncalibrated visual servoing.

2.1 PID Controller

The PID controls require knowledge of the dynamics of the robot. Pulling or releasing each of the 6 strings on the robot moves the robot in all 6 directions/rotations. For each string we compute whether pulling it creates a positive or negative change in each degree of freedom. Based on this we take the sum of each error term or the negative of each error term depending on if pulling the string moves the platform in the positive or negative direction for that DOF as the total error term for that string. We tune the PID to reduce oscillations and prevent any overshooting in the z axis as this could break the robot.

2.2 UVS controller

For Uncalibrated Visual Servoing we can use the same error functions and stack them in an error

vector e, and a vector that represent the power being applied to each motor q.

$$e = \begin{bmatrix} x_{error} \\ y_{error} \\ z_{error} \\ roll_{error} \\ pitch_{error} \\ yaw_{error} \end{bmatrix} \quad q = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}$$

We then initialize the Jacobian using the central difference method by moving each motor a small amount and computing the change in each error term.

$$J_{init} = \begin{bmatrix} \frac{\Delta x_{error}}{\Delta \theta_1} & \frac{\Delta x_{error}}{\Delta \theta_2} & \cdots & \frac{\Delta x_{error}}{\Delta \theta_n} \\ \frac{\Delta y_{error}}{\Delta \theta_1} & \frac{\Delta y_{error}}{\Delta \theta_2} & \cdots & \frac{\Delta y_{error}}{\Delta \theta_n} \\ \frac{\Delta z_{error}}{\Delta \theta_1} & \frac{\Delta z_{error}}{\Delta \theta_2} & \cdots & \frac{\Delta z_{error}}{\Delta \theta_n} \\ \frac{\Delta roll_{error}}{\Delta \theta_1} & \frac{\Delta roll_{error}}{\Delta \theta_2} & \cdots & \frac{\Delta roll_{error}}{\Delta \theta_n} \\ \frac{\Delta pitch_{error}}{\Delta \theta_1} & \frac{\Delta pitch_{error}}{\Delta \theta_2} & \cdots & \frac{\Delta pitch_{error}}{\Delta \theta_n} \\ \frac{\Delta yaw_{error}}{\Delta \theta_1} & \frac{\Delta yaw_{error}}{\Delta \theta_2} & \cdots & \frac{\Delta yaw_{error}}{\Delta \theta_n} \end{bmatrix}$$

We update the Jacobian with Broyden's method at every n time steps, for our purposes we choose n to be 50. Using all of the above we then updated each of the motor angles using the following formula

$$\Delta q = J^{-1} e$$

3. Experimental Results

We tested both the PID and Uncalibrated Visual Servoing control on the physical robot with a rectangular target object. During the experiment we had to disable the z error for both the PID and UVS as oscillations caused the platform to crash into the ground which put the camera at risk of damage. We then evaluated the performance of the algorithms based on which minimized the remaining error terms in the fewest iterations.

3.1 PID Results

The PID often did not converge to the target location and often moved in a way that resulted in some of the trackers leaving the camera frame. We theorize that this is due to the Spatially Variable Kinematics of the robot. Error terms and the PID do not take into account the position of the robot relative to the anchors the strings are attached to. This information is required to determine the kinematics of the robot. As a result there is not a combination of weights for the error terms that always guarantee that the change induced by each DOF's error term only affects the platform in that DOF.

3.2 UVS Results

Uncalibrated visual servoing does not need to know prior information about the kinematics of the system and instead models the kinematics directly on how changes in each motor position change the error terms. As a result the Visual Servoing is much better at moving the platform to the target location and is significantly more likely to achieve the task than the PID controller.

4. Conclusion

We found that when the Kinematics of the robot change depending on its location relative to the world frame it is better to use Visual Servoing over a PID controller. The robot that we created has very pronounced spatially dynamic kinematics as movements in any DOF change the kinematics of the robot. The Autonomous Underwater Vehicle also has spatially dynamic kinematics, however, only with respect to roll

and pitch due to the vehicle being positively buoyant and the center of buoyancy being above the center of mass. For this reason we believe that in practice using UVS would be significantly beneficial.

GitHub Repo: https://github.com/SamScholnickHughes/CMPUT428_Project

Sources

Chaumette, F., Hutchinson, S., & Corke, P. (2016). Visual servoing. Springer handbook of robotics, 841-866.

https://link.springer.com/chapter/10.1007/978-3-319-32552-1_34

Lab 2.1

https://ugweb.cs.ualberta.ca/~vis/courses/CompVis/assign/2DgeomAssign/lab4.html

CMPUT 312 Lab3

https://ugweb.cs.ualberta.ca/~vis/courses/robotics/assign/a3Control/images/uvs3.jpg