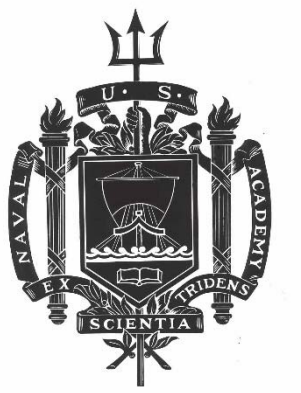
A TRIDENT SCHOLAR PROJECT REPORT

NO. 489

Development of Hybrid Robotic Controller for Autonomous, On-Orbit Spacecraft Assembly Applications

by

Midshipman 1/C Dakota L. Wenberg, USN



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
5-20-19		

4. TITLE AND SUBTITLE

Development of Hybrid Robotic Controller for Autonomous, On-Orbit Spacecraft Assembly Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Wenberg, Dakota L.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Naval Academy
Annapolis, MD 21402

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Trident Scholar Report no. 489 (2019)

12. DISTRIBUTION / AVAILABILITY STATEMENT

This document has been approved for public release; its distribution is UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The use of robotics in the space environment has been common throughout the last half century of space exploration. However, robotic arms are rarely entrusted to perform large assembly tasks without a human in the loop because of the high cost of the hardware. As our presence in space grows, developing advanced autonomous robotic systems may be the way forward as time lags associated with teleoperation are expected to cripple future space missions beyond earth orbit.

This project focuses on the derivation of an autonomous control system for spacecraft assembly applications that blends Jacobian path following and visual servoing. Jacobian path following assumes the environment is well known and plans end effector trajectories whose performance for assembly may suffer in dynamic or uncertain environments. Visual servoing approaches use feedback from an effector attached camera and can avoid dynamic obstacles, but provides no guarantee of success if the sensor cannot keep the goal location in its field of view and often traverses inefficient manipulator trajectories. This work proposes a hybrid approach that combines both approaches to improve the performance of a robotic manipulator in both known and unknown environments.

The robotic systems are simulated using the proposed hybrid controller. The hybrid controller is developed in MATLAB using a kinematic simulation of a two degree of freedom robotic arm operating in a single plane with a simplified camera model. Following successful implementation of the simulation, a more complex robotic arm is simulated in 3D space. The controller is integrated into an existing robotic arm platform (UR5 Industrial Manipulator) for a proof of concept. The results of the testing highlights the path and final position of each controller to demonstrate the advantages of each controller individually and the advantages of the hybrid approach.

15. SUBJECT TERMS
robotics, autonomous control, visual servoing, Jacobian path following

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		34	**19b. TELEPHONE NUMBER** *(include area code)*

U.S.N.A. --- Trident Scholar project report; no. 489 (2019)

DEVELOPMENT OF HYBRID ROBOTIC CONTROLLER FOR AUTONOMOUS, ON-ORBIT SPACECRAFT ASSEMBLY APPLICATIONS

by

Midshipman 1/C Dakota L. Wenberg
United States Naval Academy
Annapolis, Maryland

Certification of Adviser(s) Approval

Assistant Professor Michael D.M. Kutzer
Weapons, Robotics and Controls Engineering Department

Associate Professor Jin S. Kang
Aerospace Engineering Department

Assistant Professor Levi D. DeVries
Weapons, Robotics and Controls Engineering Department

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

USNA-1531-2

Abstract
The use of robotics in the space environment has been common throughout the last half century of space exploration. However, robotic arms are rarely entrusted to perform large assembly tasks without a human in the loop because of the high cost of the hardware. As our presence in space grows, developing advanced autonomous robotic systems may be the way forward as time lags associated with teleoperation are expected to cripple future space missions beyond earth orbit.

This project focuses on the derivation of an autonomous control system for spacecraft assembly applications that blends Jacobian path following and visual servoing. Jacobian path following assumes the environment is well known and plans end effector trajectories whose performance for assembly may suffer in dynamic or uncertain environments. Visual servoing approaches use feedback from an effector attached camera and can avoid dynamic obstacles, but provides no guarantee of success if the sensor cannot keep the goal location in its field of view and often traverses inefficient manipulator trajectories. This work proposes a hybrid approach that combines both approaches to improve the performance of a robotic manipulator in both known and unknown environments.

The robotic systems are simulated using the proposed hybrid controller. The hybrid controller is developed in MATLAB using a kinematic simulation of a two degree of freedom robotic arm operating in a single plane with a simplified camera model. Following successful implementation of the simulation, a more complex robotic arm is simulated in 3D space. The controller is integrated into an existing robotic arm platform (UR5 Industrial Manipulator) for a proof of concept. The results of the testing highlights the path and final position of each controller to demonstrate the advantages of each controller individually and the advantages of the hybrid approach.

Keywords: robotics, autonomous control, visual servoing, Jacobian path following

Acknowledgements

Thank you to the advisors of this project. To my professional mentors Dr. Radice, Dr. Piepmier, LCDR Sewell and LCDR Shey, thank you for guiding me throughout this challenging time. Many thanks to the Trident committee for supporting this project and Dr. Schroeder for shepherding my class through the program. Thank you to my parents and sister who now know way more about robotics than any economists should.

Table of Contents

Introduction

The advent and proliferation of autonomous robotic systems has revolutionized terrestrial manufacturing. Automated systems may improve production rates, lower costs, and eliminate danger to human operators. This same revolution has yet to make the jump to space. Current robotic arms in space are primarily human controlled and are not entrusted to independent operation [1]. While construction of larger objects in orbit has been augmented by teleoperated robotic arms, their operation is slowed by communications delays from the ground as well as the cost and time required to launch a human operator into space [2]. These impediments to robotic operations in space impact the ability to assemble larger, complex spacecraft structures. The on orbit assembly of spacecraft has been demonstrated to decrease deployment risk, and enable the assembly of systems too large to be flown in a single launch vehicle. On orbit assembly also enables satellites and spacecraft to be repaired and upgraded in the future [2].

Increased and improved autonomy may increase the efficiency and reliability of robotic operations in space. A move away from an overreliance on human-in-the-loop robotics may extend the usable reach of robots for space applications and decrease the time required to perform robotic operations in and beyond earth orbit. There are several accepted methods for providing closed-loop feedback to a robotic system to enable autonomous robotic operations. Many approaches use visual feedback to a system to verify the relative position of the robotic arm and to inform future movements. Two control approaches, eye-in-hand visual servoing and Jacobian path following, have been extensively researched and applied in terrestrial robotics [3][4][5]. While current research has extensively explored autonomous control using each of these approaches separately, there does not appear to be research in the field of a sliding hybrid controller informed by the two approaches. By combining the two approaches an autonomous robotic system will be able to assemble spacecraft using feedback to safely operate in the dynamic environment.

The work done focuses on developing the control algorithms for the robotic arm. The algorithms are tested in a simulated 3D environment using a 6 Degree of Freedom (DOF) robotic arm which is kinematically comparable to the robotic arm that will be used during the International Space Station (ISS) testing. The simulation is also developed to be able to inject errors into the system. The errors are a result of sensor measurement error and are injected when taking 2D and 3D images and while moving the joints of the arm. Next a stability proof is done for the weighted system to validate the approach. The proof results in an upper bound on the error at the end of the trajectory. The goal for the weighting function is to minimize that error.

The report is organized as follows. The Background section motives the need for a hybrid approach to robotic manipulator control and provides an overview of related work. The Theoretical Results section walks through the fundamental algorithms that govern the control of the robot, the proposed weighting function, the error models used in simulation, and the simulation results. The Experimental Results section presents the results of the UR5 demonstration. The conclusion section concludes the report and summarizes future work.

Background

The biggest initial question for the project is: why is a hybrid controller necessary? The two discussed approaches to autonomous control, eye-in-hand visual servoing and model-based trajectory planning, are each viable methods of determining how to move a robotic arm on their own. So why combine them? Problems arise when the requirements of the space environment, and the limitations of the available sensors are taken into consideration. Jacobian path following works incredibly well in the terrestrial environment. Excellent cameras with great resolution and large high powered computers can be used to get incredibly accurate depth data to determine the position of a robotic arm. This means that the path that it follows is accurate. However because of size, weight and power requirements, in the space environment less accurate sensors are the only options available. This causes end position error in the Jacobian path following approach. This error is dangerous when looking at spacecraft assembly because of the sensitivity and the cost of the parts being assembled. Another inherent problem with Jacobian path following is the difficulty of the system to respond to a dynamic environment. For example if an object obstructs the field of view of the 3D camera then it is difficult to acquire the necessary depth data.

There are also problems with eye-in-hand visual servoing. Most significantly, if the object which is being used as a reference leaves the field of view of the camera, the robot can no longer be controlled using this method. There are also errors in the 2D image produced by the resolution of the camera. These errors are worse when the camera is a distance away from the object. However one of the advantages of eye-in-hand visual servoing is that as the distance to the object decreases, the error decreases. This enables highly accurate end effector position at close proximity. This advantage of visual servoing suggests that this method should be used for fine movements close to an object at the end of the robotic trajectory. Following this conclusion, it is clear that a hybrid controller is necessary to be able to use both of these approaches.

By definition, a hybrid controller combines these two approaches. Jacobian path following is used to form a generalized understanding of the trajectory through free-space, and eye-in-hand visual servoing reduces the positional error that exists in Jacobian path following while allowing the robotic arm to react to a dynamic field of obstacles.

1. Past Research in this Field

One of the more recent advances in robotic control came with the advent of digital camera technology and subsequent study of machine vision. With this technology, researchers began investigating the use of measured parameters from digital images to inform robotic movements. One such method, known as visual servoing, leverages image parameters to create a closed-loop velocity controller for robots. Early work in visual servoing began in the early 1970s and has continued ever since. Extensive reviews of visual servoing methods have been published including those by Peter Corke who has authored numerous papers, books, and code packages on visual servoing and more broadly visual robotic control [6]. His work details visual servoing methods starting with the characterizing electromechanical systems, summarizing methods and models for image capture, and detailing various algorithms associated with creating a closed-loop control system using feedback from a digital camera.

Substantial research also exists in the field of motion planning and manipulation. In the introduction to Algorithmic Foundations of Robotics Jean- Claude Latombe notes, “As our understanding of algorithms grows, machines will improve on their ability to appear as machines executing algorithms, independent of any ephemeral technology” [7]. The book outlines a number of established approaches to the mathematical modeling of and planning for robotic systems. Issues such as point sampling, image segmentation, obstacle avoidance algorithms and other methods are discussed in the text. These are robustly studied areas of robotic planning and have many examples of real world implementation.

Current projects such as DARPA’s Robotic Servicing of Geosynchronous Satellites Program use closed-loop visual control methods to autonomously rendezvous with large satellites to repair and refuel them without needing to return them to the ground [8]. This project and others like it build on over fifty years of knowledge and experience in the field of robotic operations and control. Because the DARPA project is aimed at large expensive geosynchronous satellites the control method for the robotic arm is a combination of autonomous and teleoperation methods [9]. The autonomous functions are performed using visual servoing and a compliance control system. However due to the complex required manipulation tasks, the fine movement tasks will be carried out using teleoperation. Autonomy will only be used in cases where the spacecraft does not have reliable communications with the ground. This means that although many of the functions on RSGS are autonomous, full autonomy is not achieved in this platform despite the visual servoing control.

Another autonomous space robotic arm was demonstrated on the Orbital Express Satellite Servicing project in 2007 [10]. The demonstration included autonomous capture and refueling of a satellite. The capture was performed using pre-planned movements along with visual servoing. While visual servoing was implemented in this approach the heavy reliance on pre-scripted commands means that the system was not fully autonomous, nor did it combine model-based planning with visual servoing.

There are robotic projects that have developed hybrid robotic control techniques leveraging multiple inputs to a system. The first example is the linear combination of Position Based Visual Servoing (PVBS) and Image Based Visual Servoing (IVBS) [3]. The two inputs are added and scaled based on an error term. This approach is more complex than the elegant sliding hybrid which is proposed in this paper. The second example is a project that looks at supplementing human motion with an attached robotic arm for patients with neurological injuries undergoing rehabilitation [11]. The objective is to provide robotic assistance on an as-needed basis when a patient cannot perform a function in rehabilitation training. The discrete event nature of this approach (either the robot arm is on or it is off) means that either a preprogrammed rule or input from an observing therapist is needed to discretely switch back and forth between the human motion and the motion of the robotic arm. The goal of this research is to optimize the interaction between the robot and human. The described hybrid controller proposes an adaptive rule to change modes using sliding surfaces and a reference trajectory in a Lyapunov-based algorithm. The controller also adjusts assistance levels based on the patient’s ability, it does not need new input parameters for each new patient and each new session as information collected during

sessions adjusts the system parameters. The stability of the system is shown through Lyapunov-based analysis. Overall this research indicates that multiple inputs to a system from different control approaches can be combined to produce a stable robotic system.

Theoretical Results

1. Algorithm Development

The algorithms developed as part of this work are broken into two sections. The Jacobian path following section describes the derivation of a Jacobian matrix which is used to relate joint velocities to task space velocities to execute Jacobian path following. The visual servoing section details a previously derived method of executing visual servoing that accounts for the translation and the orientation change in the robotic arm using a camera.

The main purpose of all of these approaches is to relate a change in coordinates in 3D space to a change in the joint angles of the robotic arm. Then from there the understanding of how the joint position needs to change can be used to move the robotic arm in 3D space to produce that desired change.

Jacobian Path Following

Jacobian path following executes linear motion in task space, so the trajectory looks like a straight line from the start position to the end position. However this approach is limited by the accuracy of the sensor (a 3D camera). Jacobian path following is executed by relating the joint variables ($\dot{\vec{q}}$) to the task variables ($\dot{\vec{X}}$). Task variables are a 6x1 vector that represents the desired change in translation and orientation of the end-effector. The equation

$$\dot{\vec{X}}^0 = J(\vec{q})\dot{\vec{q}}, \tag{1}$$

defines the relationship between joint and end-effector velocities using a Jacobian matrix, $J(\vec{q})$. Here, $\dot{\vec{X}}^0$ is defined as the task velocities of the end-effector referenced to Frame 0 (the base frame of the robotic manipulator) and $\dot{\vec{q}}$ represents a vector containing the joint velocities of the manipulator. Because this relationship is linear, it provides a simple way of converting desired end-effector velocities into desired joint velocities

$$\dot{\vec{q}} = sJ(\vec{q})^{-1}\dot{\vec{X}}^0. \tag{2}$$

The robotic Jacobian is configuration dependent meaning that the velocity relationship between the end-effector and joint velocities changes as the arm moves. The Jacobian relating end-effector velocities and joint velocities for a 6 degree of freedom (DOF) arm is calculated in two parts. The upper half of the matrix that relates the joints to the translation of the arm ($x(\vec{q}), y(\vec{q}), z(\vec{q})$) is calculated by taking the partial derivatives with respect to each joint

$$J_t(\vec{q}) = \begin{bmatrix} \frac{\partial x(\vec{q})}{\partial q_1} & \dots & \frac{\partial x(\vec{q})}{\partial q_6} \\ \frac{\partial y(\vec{q})}{\partial q_1} & \dots & \frac{\partial y(\vec{q})}{\partial q_6} \\ \frac{\partial z(\vec{q})}{\partial q_1} & \dots & \frac{\partial z(\vec{q})}{\partial q_6} \end{bmatrix} \tag{3}$$

The equations for $x(\vec{q})$, $y(\vec{q})$, and $z(\vec{q})$ are found by calculating the forward kinematics of the arm, that relate the end effector to the base, symbolically

$$H_e^0(\vec{q}) = \begin{bmatrix} R_e^0(\vec{q}) & \vec{d}_e^0(\vec{q}) \\ \vec{0}^T & 1 \end{bmatrix} \tag{4}$$

and then extracting the translation matrix,

$$\vec{d}_e^0(\vec{q}) = \begin{bmatrix} x(\vec{q}) \\ y(\vec{q}) \\ z(\vec{q}) \end{bmatrix} \tag{5}$$

The second half of the matrix describes the rotation component of the Jacobian and relates the orientation to the joint angles. For simplicity not all variables that are a function of q are going to be explicitly called out. This method is described in [12]. The method uses the rotation matrix R_e^0 that is extracted from the forward kinematics to produce the Jacobian

$$J_r(\vec{q}) = \begin{bmatrix} \left(\frac{\partial R_e^0}{\partial q_1}(R_e^0)^T\right)^\vee & \dots & \left(\frac{\partial R_e^0}{\partial q_6}(R_e^0)^T\right)^\vee \end{bmatrix} \tag{6}$$

where $\vee$ is notation for the vee function. The method for finding the desired change in translation $\frac{\Delta\vec{x}^0}{\Delta t}$ and orientation $\frac{\Delta\vec{k}^0}{\Delta t}$ relies upon the homogeneous transform between the different poses. The transform between the current end effector position and the goal position is found by manipulating the transforms of each individual from with respect to the base,

$$H_g^e = (H_e^0)^{-1} H_g^0 \tag{7}$$

$$H_g^e = \begin{bmatrix} R_g^e & \vec{d}_g^e \\ \vec{0}^T & 1 \end{bmatrix}. \tag{8}$$

Where the relationship for the translation change can be extracted from the resultant 4X4 matrix,

$$\Delta\vec{x}^e = \vec{d}_g^e. \tag{9}$$

And the relationship for the rotation ($\Delta\vec{k}^e$) is given by taking the rotation matrix from the homogeneous transform and taking the vee of the matrix natural log,

$$\Delta\vec{k}^e = [logm(R_g^e)]^\vee \tag{10}$$

Then each calculated change is modified such that it is given in terms of the base frame instead of the end effector frame,

$$\frac{\Delta\vec{x}^0}{\Delta t} = R_e^0 \Delta\vec{x}^e \tag{11}$$

$$\frac{\Delta\vec{k}^0}{\Delta t} = R_e^0 \Delta\vec{k}^e \tag{12}$$

When combined, we achieve a term in 3D task space relative to the base frame that describes the translation and orientation difference between the current end effector pose and the desired end effector pose.

$$\dot{\vec{X}}^0 \approx \frac{\Delta\vec{X}^0}{\Delta t} = \begin{bmatrix} \frac{\Delta\vec{x}^0}{\Delta t} \\ \frac{\Delta\vec{k}^0}{\Delta t} \end{bmatrix}. \tag{13}$$

The result is the task variable matrix ($\frac{\Delta\vec{X}^0}{\Delta t}$) which is used in the fundamental equation for Jacobian path following

$$\frac{\Delta\vec{q}}{\Delta t} = J(\vec{q})^{-1} \frac{\Delta\vec{X}^0}{\Delta t}, \tag{14}$$

which by the fundamental theorem of calculus results in

$$\Delta\vec{q} = sJ(\vec{q})^{-1} \Delta\vec{X}^0, \tag{15}$$

where s is a scaling term used to decrease the step size for each increment preserving the fundamental theorem of calculus assumption. The result is an equation that converts the desired change in orientation and position into the required joint velocities needed to accomplish that change. The assumption in this approach is that the inverse of the Jacobian matrix is never singular. When taking an inverse of a matrix every value in the new matrix is, by definition, divided by the determinant of the original matrix. If the determinant is equal to zero then the resultant inverse becomes undefined. This can occur at many different configurations of the robotic arm called singularities. Avoidance of singularities is well documented in literature [13]. A temporary solution to avoid this situation is to avoid these configurations and to monitor the Jacobian throughout the trajectory and if it nears a singularity, the calculated velocity is regulated.

Visual Servoing

The basic principle of visual servoing is that the projected image of an object changes if the object moves relative to the camera or if the camera moves relative to the object. Visual servoing commands a robotic arm to move such that a particular object within a camera's field of view changes position and orientation from a starting configuration to match a desired ending configuration as shown in Figure 1.

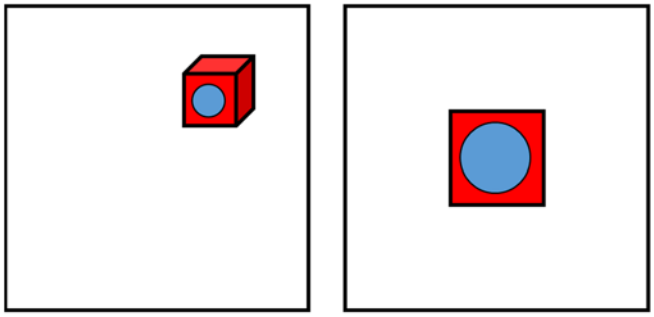


Figure 1: Initial view of cube (left) and desired view of cube (right).

Eye-in-hand visual servoing is where the camera is positioned on the end-effector of the arm and moves with the robotic arm as shown in Figure 2. This allows the system to see more and to view objects from different angles. Image based visual servoing is implemented using what's called an image Jacobian. The image Jacobian is the product of the manipulator Jacobian, the camera's intrinsic matrix, and the camera's extrinsic matrix relating the camera frame to a frame on the robotic end-effector.

The method for executing the visual servo is derived from [14]. This approach combines both an understanding of the object view in the camera, while estimating its orientation in space. The fundamentals of this approach are shown in Figure 2 where not only the position of the reference point in the camera frame is used in the calculations but also the distances from the reference point to the current and desired camera frame and the normalized distance to the plane the reference point sits on.

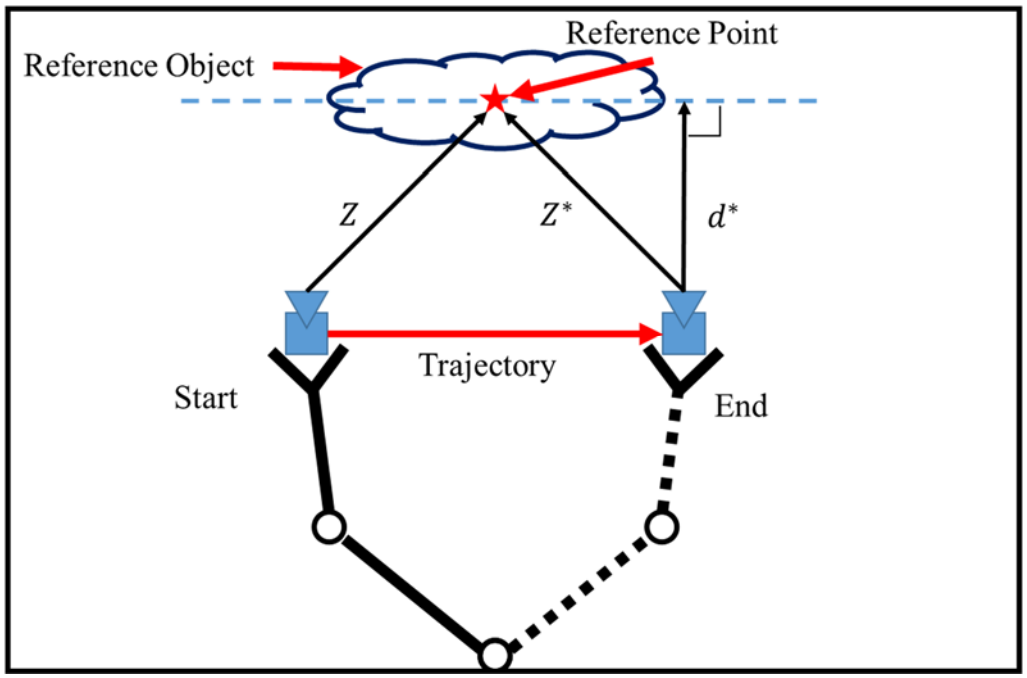


Figure 2: Basic principles of 2 ½ D visual servoing.

Z is the current distance between the camera frame and the reference target point and normalized goal distance is d^*. Z^* is the distance between the camera frame and the reference target point at the desired end position and orientation. These properties are all shown in Figure 2.

Some of the basic parameters are,

$$\vec{P}^m = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \vec{P}^{m_g} = \begin{bmatrix} u_g \\ v_g \\ 1 \end{bmatrix}, \rho_1 = \frac{Z}{d^*}, \tag{16}$$

Are extracted from the image. $\vec{P}^m$ which is the current image point as expressed as a 2X1 matrix that contains the 2D coordinate points where the reference point shows up on the image frame and $\vec{P}^{m_g}$ which is the desired image point are both points referenced relative to the matrix of the camera. ρ_1 is a method of scaling the trajectory as the camera reaches the end position.

The image points are related to the desired change in camera orientation and translation through a series of matrices in the 2 ½ D Visual Servoing method [14]:

$$L_v = \begin{bmatrix} -1 & 0 & u \\ 0 & -1 & v \\ 0 & 0 & -1 \end{bmatrix} \tag{17}$$

$$L_{vw} = \begin{bmatrix} uv & -(1+u^2) & v \\ (1+v^2) & -uv & -u \\ -v & u & 0 \end{bmatrix} \tag{18}$$

$$L = \begin{bmatrix} Z^*(L_v)^{-1} & -Z^*(L_v)^{-1}L_{vw} \\ \overrightarrow{0_3} & I_3 \end{bmatrix} \tag{19}$$

where the L matrix is the matrix used to manipulate the error term ($\vec{e}$), which is a compact method of expressing translational and rotational error ($\Delta\vec{k}^0$ is calculated using the same methods of calculating the term as described in Equations 10 and 12),

$$\vec{e} = \begin{bmatrix} u - u_g \\ v - v_g \\ \log\left(\frac{Z}{Z^*}\right) \\ \Delta\vec{k}^0 \end{bmatrix} \tag{20}$$

$$\vec{V} = -sL\vec{e} \text{ [14].} \tag{21}$$

The error term is related to the camera velocity screw ($\vec{V}$), by a scalar term (s).

The method for converting from camera velocity to the joint velocity is by using the intrinsic matrix (A_c^m), which holds camera parameters and the rotation matrices extracted from the forward kinematics. The intrinsic matrix describes the internal parameters of the camera and can be used to convert points in 3D space to a projection onto a camera. The homogeneous transform between the camera and the base frame is found using forward kinematics,

$$H_C^0 = \begin{bmatrix} R_C^0 & \vec{d}_C^0 \\ \vec{0}^T & 1 \end{bmatrix} \tag{22}$$

then the rotation matrix is extracted from the transform

$$R_0^c = (R_c^0)^{-1} = (R_c^0)^T. \tag{23}$$

Then the image Jacobian, used in calculating the translation, is calculated by combining the intrinsic matrix, the rotation matrix extracted in Equations 16 and 17, and the first three rows of the Jacobian matrix ($J_t(\vec{q})$) which relate to the translational velocity of the end effector. The result

$$J_i(\vec{q}) = A_c^m R_0^c J_t(\vec{q}) \tag{24}$$

Is then concatenated with the last three rows of the Jacobian matrix which relate the rotational velocity to the end effector to create the combined Jacobian matrix.

$$J(\vec{q}) = \begin{bmatrix} J_i(\vec{q}) \\ J_r(\vec{q}) \end{bmatrix} \tag{25}$$

This matrix is then used to related the calculate camera velocity screw to the change in joint angles,

$$\Delta\vec{q} = J(\vec{q})^{-1}\vec{V}. \tag{26}$$

One of the key assumptions in 2 ½ D visual servoing is that the identity matrix can be used in the lower right hand 3x3 matrix of the L matrix which is shown in Equation 19. This assumption comes from a small angle approximation for this approach where the method assumes that only small changes in orientation are occurring. There is a term that is based on the change in orientation in the equations that is assumed to be small. The term which results in this approximation is plotted in Figure 3.

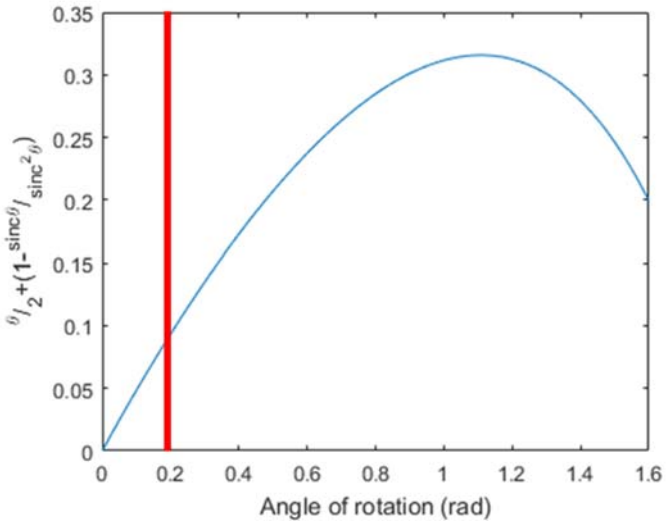


Figure 3: Small angle approximation with maximum change in orientation plotted with red line.

This graph shows how the term $(\frac{\theta}{2} + (1 - \frac{\sin(\theta)/\theta}{\sin^2(\theta)/\theta^2}))$ changes as the angle of rotation, θ, is increased. The plotted term is subtracted from 1 in the equations for this visual servoing approach. The result is that if the term is small then for this approach it can be assumed that the value is approximately equal to 1 and the identity matrix can be used in Equation 19. It is estimated that this assumption will hold as long as the term is within 10% of 1 so it would be equal to 0.1. The 10% bound is decided because it is a reasonable bound around the value and it is a common bounding value for an approximation. This result occurs for all orientation values between 0 and 0.2 rad. The conclusion is that the small angle approximation is valid for small orientation changes (-0.2 rad to 0.2 rad).

2. Error Models

There are two main error models used to add noise to the system. The goal is to quantify the performance of the system under realistic error. The 2D image section outlines the method of simulating error in a regular camera. The 3D image section walks through an approach to

modeling error when a 3D image of a point is space is taken using a stereoscopic vision system with two cameras.

2D Camera Error

The 2D Image error model is based on an assumption that the camera calibration will be the source of the error. Camera calibration is used to calculate the intrinsic matrix (A_N^m). When calibrating a camera the functions used to execute this calibration return both calculated parameters and error terms (δ_N^M) [14]

$$\delta_N^M = \begin{bmatrix} \sigma_\alpha & \sigma_\gamma & \sigma_{u_0} \\ 0 & \sigma_\beta & \sigma_{v_0} \\ 0 & 0 & 1 \end{bmatrix}. \tag{27}$$

An example camera calibration is used in the simulation. Taking an image in simulation is done by converting the coordinates of a point relative to the camera into image plane coordinates by using the intrinsic matrix A_N^m

$$\vec{P}^m = A_N^m \vec{P}^N. \tag{28}$$

To simulate error a term is added to account for the increase in error in the camera as the camera gets further away from the point

$$a = s\left(e^{\frac{z - z_{min}}{z_{max}}} - 1\right). \tag{29}$$

Where there is a minimum (z_{min}) and maximum (z_{max}) distance for which the camera can resolve two points. The error is maximum when the point being imaged is at the furthest distance and minimum at the closest distance as shown in Figure 4.

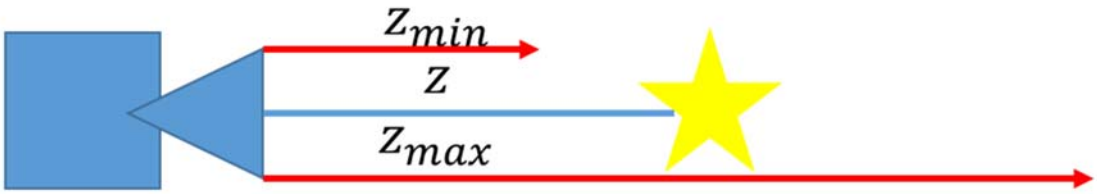


Figure 4: A point (star) that lies closer to the z_{min} has a smaller error than a point closer to the z_{max} because the camera can better resolve points at closer distances.

The error is then added to the intrinsic matrix by multiplying the term that correlates to distance by the matrix of error,

$$\tilde{A}_N^m = A_N^m + a\delta_N^M. \tag{30}$$

The result is a matrix whose error increases as the distance to the reference point increases. This noisy intrinsic matrix is then used to take a noisy image of the target by substituting $A\,{}_N^m$ for $\tilde{A}_N^m$ in Equation 28.

3D Camera Error

3D imager data is used in the Jacobian path following approach. The goal position is taken from the output of the camera with is a depth image that represents the 3D environment around the camera. The imager most commonly used is a stereoscopic 3D camera which uses two separate cameras to take images and the offset difference between the two images is used to calculate depth. The error in the position reading of a stereoscopic camera is the result of the overlap that is used to determine distance. An image of the generated error is given in Figure 5.

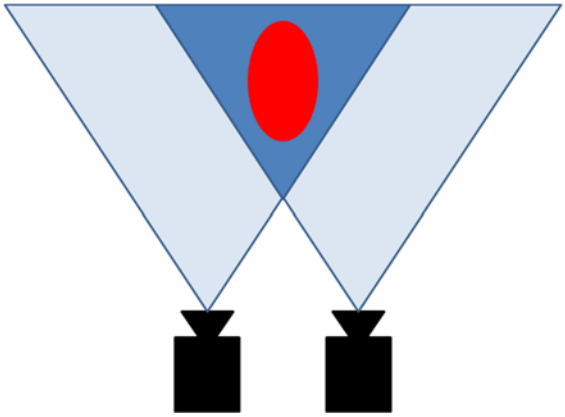


Figure 5: The overlap between two images resulting in an error (red) that is approximately circular.

The equation for determining the position of a point using stereo vision is [15]

$$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{B(u_l+u_r)}{u_l-u_r} \\ \frac{B(v_l+v_r)}{u_l-u_r} \\ \frac{2B}{u_l-u_r} \end{bmatrix}. \tag{31}$$

Where B is the baseline distance between the centers of each camera, the u and v terms are the position of the point in the view of either the right (r) or left (l) camera. A covariance (error) matrix for the system is calculated by combining the covariance matrix of each individual camera (V_l, V_r) and a Jacobian matrix:

$$J_f = \begin{bmatrix} \frac{\partial X}{\partial u_l} & \frac{\partial X}{\partial v_l} & \frac{\partial X}{\partial u_r} & \frac{\partial X}{\partial v_r} \\ \frac{\partial Y}{\partial u_l} & \frac{\partial Y}{\partial v_l} & \frac{\partial Y}{\partial u_r} & \frac{\partial Y}{\partial v_r} \\ \frac{\partial Z}{\partial u_l} & \frac{\partial Z}{\partial v_l} & \frac{\partial Z}{\partial u_r} & \frac{\partial Z}{\partial v_r} \end{bmatrix} \tag{32}$$

Which gives,

$$V_p = J_f \begin{bmatrix} V_l & 0 \\ 0 & V_r \end{bmatrix} (J_f)^T. \tag{33}$$

As a way to visualize the error produced around a point by this covariance matrix a Monte Carlo simulation is run on a series of three points in the field of view of the camera. The goal is to create a sphere of points around the point that symbolizes the range of the error that could exist for a single point. Figure 6 is the result of that simulation.

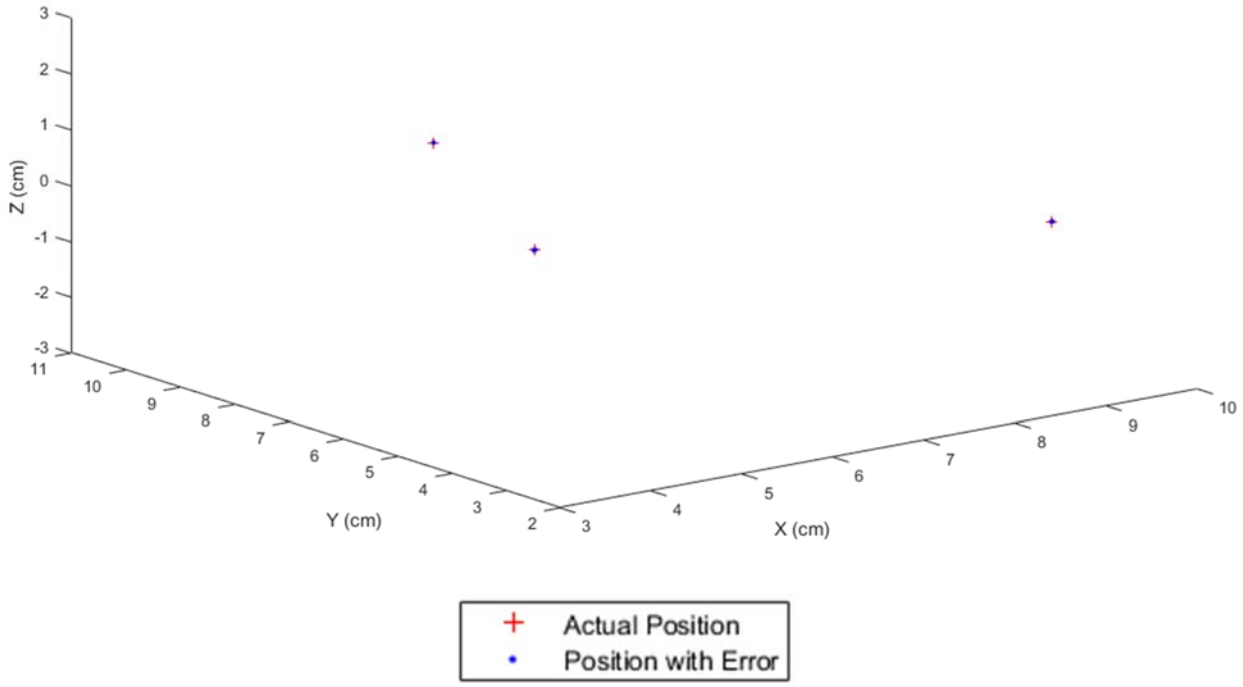


Figure 6: Visualization of error surrounding a point as produced by a 3D stereoscopic vision camera.

A close up example image of 10,000 sample points of a single point using this error method is given in Figure 7.

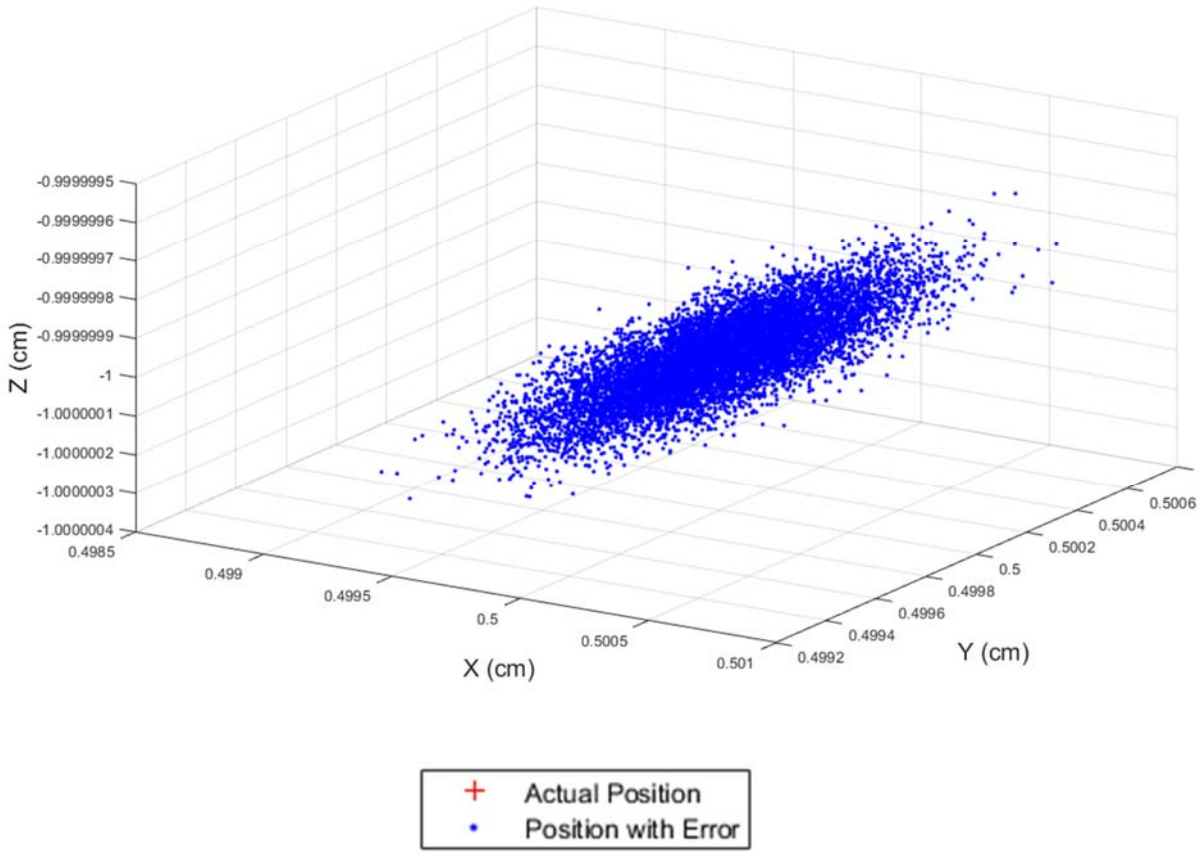


Figure 7: A random sample of 10,000 points of one location using the 3D camera errors.

The simulation of the 3D error gives an oval shape which matches the expected shape resented in the paper where the overlap between the cameras results in an error profile that is longer in the normal direction to the camera than laterally within the field of view.

3. Weighting Function

The weighting section describes a derived method of weighting these two approaches using the current and desired position and orientation as well as the position of the reference pixel in the field of view of the camera.

The proposed hybrid controller is a weighted sum of the two joint velocities: one from the visual servoing and the other from the robotic Jacobian following waypoints in a pre-planned path. The weight of each controller is decided based on the distance to the target position and orientation. As an analogy, this approach can be applied to a cruise missile. The missile, when initially launched, is guided by long range guidance systems such as satellite guidance and tracking. However as the missile approaches its target, it switches to automatic target recognition to refine the path to the target. In that same fashion the hybrid controller seamlessly moves from the initial control using a pre-planned path to a controller that refines movement using visual feedback until it reaches the desired position.

The weight of the visual servoing control coefficient is zero at the starting position where the fraction of total distance to the target is 1. This means that, initially, the visual servoing controller has no effect on the movement of the robotic arm. The weight of the visual servoing control coefficient grows as the fraction of the total distance to the target falls to zero, ending at the final position equal to 1. The result should be a movement that initially matches the pre-planned path and diverges as from the planned path to the actual target position as the end-effector gets closer to the target. Since visual servoing is less accurate when it is far away from an object, following a pre-planned path is used until the robotic arm sufficiently close to the object of interest. Once the object is sufficiently close, visual servoing begins to drive the arm to an accurate end-effector position. In the proposal the weighting function took only the translation of the arm into account while sliding between each approach. However, in the 3D approach this term does not encompass the entire story. It is necessary to take into account both translation and orientation. Therefore each weighting term is averaged into a combined term that describes the error is orientation (W_r) and translation (W_t)

$$W_t = \frac{|X_g - X|}{|X_g - X_{init}|} \in [0,1] \tag{34}$$

$$W_r = \frac{\left|logm\left(R_{e_g}^0\right)^\vee - logm(R_e^0)^\vee\right|}{\left|logm\left(R_{e_g}^0\right)^\vee - logm\left(R_{e_{init}}^0\right)^\vee\right|} \in [0,1] \tag{35}$$

$$W_e = \frac{W_t + W_r}{2}. \tag{36}$$

This combined term (W_e) weights the error in translation and rotation equally while still returning a result that is between 0 and 1.

Stability Proof

The stability proof is conducted using the principle of Lyapunov functions that if the derivative of the potential function can be shown to be negative, within a bound, from any starting configuration, then the system is stable in the sense of Lyapunov. Imagine a bowl. The goal of a Lyapunov approach is to prove the shape of the bowl so that it can be concluded that for any initial position on the surface of the bowl that the final position will be the bottom of the bowl. This approach is completed in two steps. First a generic controller is used to demonstrate that a weighted function can be proved stable. Then specific matrices for each controller are substituted to complete the proof.

Consider two separate approaches to robotic control which are each individually stable. The goal of the proof is to show that weighting two stable systems does not render the system unstable [16]. There are simplified stable systems are used in the first proof with error terms included.

The goal of the simplified proof is to model the structure of the problem and it sets a template for the full proof.

Assume there is a system:

$$\dot{x} = ax + u. \tag{37}$$

The term u could be calculated one of two ways (similar to how the desired change in joint velocity can be calculated in two ways either Jacobian path following or visual servoing),

$$u = -k_1 x + g_1(t, x) \tag{38}$$
$$u = -k_2 x + g_2(t, x) \tag{39}$$

where $g_1(t, x)$ and $g_2(t, x)$ are error terms that are time and state dependent. The system that results from each is,

System 1	System 2	
$\dot{x} = (a - k_1)x + g_1(t, x)$	$\dot{x} = (a - k_2)x + g_2(t, x).$	(40)

The Lyapunov function for this example is

$$\dot{V} = x\dot{x} \tag{41}$$

which results in a system that has the following potential functions,

$$\dot{V} = (a - k_1)x^2 + g_1(t, x)x \qquad \dot{V} = (a - k_2)x^2 + g_2(t, x)x. \tag{42}$$

If the error terms are bounded,

$$-\delta_1|x| \le g_1(t, x)x \le \delta_1|x| \qquad -\delta_2|x| \le g_2(t, x)x \le \delta_2|x| \tag{43}$$

and the placeholder term

$$0 \le \theta \le 1 \tag{44}$$

is used to separate the terms.

$\dot{V} = -(1 - \theta)\lvert a - k_1\rvert\lvert x\rvert^2$	$\dot{V} = -(1 - \theta)\lvert a - k_2\rvert\lvert x\rvert^2$	(45)
$-\theta\lvert a - k_1\rvert\lvert x\rvert^2 + \delta_1\lvert x\rvert$	$-\theta\lvert a - k_2\rvert\lvert x\rvert^2 + \delta_2\lvert x\rvert.$	

The $-(1 - \theta)|a - k_1||x|^2$ portion of the function will always be negative because of the bounds on θ. So the next step is to determine for what values of x will result in negative values therefore proving the Lyapunov stability of the system. So how to make the rest of the equation always be negative. When the last two terms are compared to each other the following bound is calculated,

$$|x| \ge \frac{\delta_1}{\theta|a - k_1|} \qquad |x| \ge \frac{\delta_2}{\theta|a - k_2|}. \tag{46}$$

This means that each example system individually have a bound on the error seen above. The system will converge to within that bound which is around the end state of the system.

The goal of the next step of the proof is to show that there is still a bound on the error and that the system is stable even when weighting the two stable approaches. The system equation for the weighting function is

$$\dot{x} = W\big((a - k_1)x + g_1(t, x)\big) + (1 - W)\big((a - k_2)x + g_2(t, x)\big) \tag{47}$$

With the Lyapunov equation bounded by

$$\dot{V} \leq W|a-k_1||x|^2 + (1-W)|a-k_2||x|^2 + \delta_1|x| + \delta_2|x| \tag{48}$$

with the bound on the disturbances due to error being added the same as the previous proof. The result being a bounded region around the origin to which the system will converge

$$|x| \geq \frac{\delta_1 + \delta_2}{\theta(W|a-k_1| + (1-W)|a-k_2|).} \tag{49}$$

The full proof is derived from two separate proofs which show that Jacobian Path Following and 2 ½ D Visual Servoing are stable when operating independently [17] [14]. The goal of the following proof is to show that by weighting the functions that the stability of the system is still maintained. The equations for the error term which is the control input to the system is

$$\dot{\bar{e}} = w(-Q_1\bar{e} + g_1) + (1-w)(-Q_2\bar{e} + g_2) \tag{50}$$

where Q_1 and Q_2 are matrices that represent the negative semi-definite matrices that are used in the equations for Jacobian path following and visual servoing. $g_1$and g_2 are errors due to the camera errors which inject error into the system. The Lyapunov function is similar to the previous proof

$$V = \frac{1}{2}\bar{e}^T\bar{e} \tag{51}$$

and the derivative results in

$$\dot{V} = \bar{e}^T\dot{\bar{e}}. \tag{52}$$

Inserting Equation 50 into Equation 52 results in

$$\dot{V} = \bar{e}^T\big(w(-Q_1\bar{e} + g_1) + (1-w)(-Q_2\bar{e} + g_2)\big). \tag{53}$$

Rearranging the equation gives

$$\dot{V} = -w\bar{e}^TQ_1\bar{e} - (1-w)\bar{e}^TQ_2\bar{e} + w\bar{e}^Tg_1 + (1-w)\bar{e}^Tg_2. \tag{54}$$

The error terms are then bounded

$$-\delta_1|\bar{e}^T| \leq w\bar{e}^Tg_1 \leq \delta_1|\bar{e}^T| \tag{55}$$

$$-\delta_2|\bar{e}^T| \leq (1-w)\bar{e}^Tg_2 \leq \delta_2|\bar{e}^T| \tag{56}$$

And the θ term is added to the system

$$0 \leq \theta \leq 1 \tag{57}$$

$$\dot{V} = (1-\theta)(-w|Q_1||\bar{e}|^2 - (1-w)|Q_2||\bar{e}|^2) + \theta(-w|Q_1||\bar{e}|^2 - (1-w)|Q_2||\bar{e}|^2) + \delta_1|\bar{e}^T| + \delta_2|\bar{e}^T|. \tag{58}$$

Then isolating the error terms and the other portion of the equation,

$$(\delta_1 + \delta_2)|\bar{e}^T| \leq \theta\big(-w\big|Q_1\big||\bar{e}|^2 - (1-w)\big|Q_2\big||\bar{e}|^2\big) \tag{59}$$

$$(\delta_1 + \delta_2) \leq \theta\big(-w\big|Q_1\big||\bar{e}| - (1-w)\big|Q_2\big||\bar{e}|\big) \tag{60}$$

results in an error bound which is

$$|\bar{e}| \geq \frac{\delta_1 + \delta_2}{\theta}(-w|Q_1| - (1-w)|Q_2|)^{-1}. \tag{61}$$

This all shows that for any particular configuration that the system will converge to a bounded region around where the potential function is zero. However the one simplifying assumption in this proof is that w is not state dependent which is not the case because it actually depends on the joint configuration and the distance to the target end state. This assumption allows this proof to be conducted however it could mean that for certain states the system may not be stable.

Resulting Weighting Function

The paper which lays out the proof for the stability of the 2 ½ D visual servoing approach dictates that in order for the system to be stable, the target reference point must stay within the field of view of the camera. Since this is a major condition of stability, the rotation and translation error terms are not sufficient for determining the weighting function. Jacobian path following only moves linearly in task space which opens up the possibility that the reference point leaves the field of view. As a result the weighting function must also take into account the position of the target in the field of view of the camera such that when the point is about to leave the field of view the function shifts quickly to visual servoing in order to keep the point in the field of view. To accomplish this the distance between the center of the image and the point is taken

$$\mu = \sqrt{(u-u_0)^2 + (v-v_0)^2}. \tag{62}$$

This implementation assumes that the image is wider than it is tall. A saddle function is built,

$$W = W_e\left(1 - \left(\frac{|\mu - v_0|}{v_0}\right)^n\right), \tag{63}$$

and the total weight W is based on the error term and the position of the pixel, where $n \in \{2,4,6 \ldots\}$.

A visualization of this function is given in Figure 8.

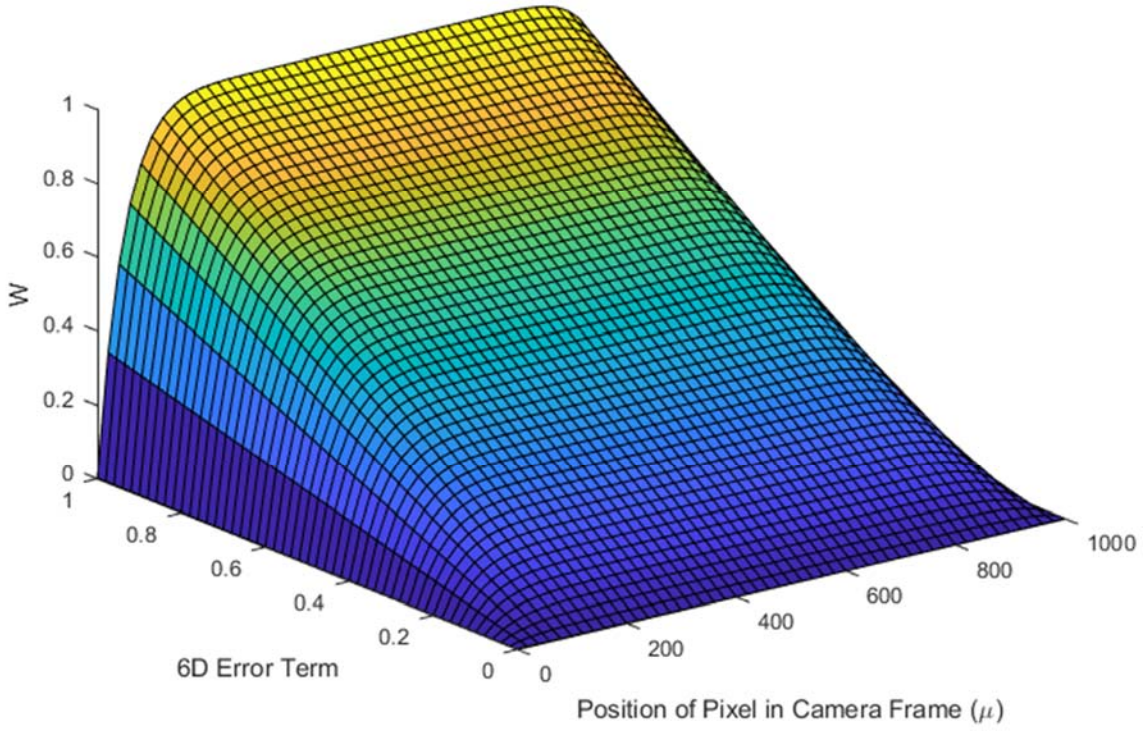


Figure 8: Visualization of weighting function with respect to error and pixel location.

Then the function is weighted such that when W=1 Jacobian path following is implemented and when W=0 visual servoing is implemented.

$$k_1 = 1 - W \tag{64}$$

$$k_2 = W \tag{65}$$

The final weighting equation is

$$\dot{q}_h = k_2 \dot{q}_t + k_1 \dot{q}_{vs} \tag{66}$$

Simulation and Experimental Results

The results to demonstrate this system are in simulation and in hardware testing. There are three arms that are built in simulation. They are a theoretical 2DOF and 6DOF arm, a ScorBot, and a UR5. The system is also simulated on a real UR5 arm.

The initial testing is done using a simple 2DOF robotic arm with several simplifying assumptions which means that this simulation only moves the robotic arm in a single plane. The robotic arm used is shown in Figure 9.

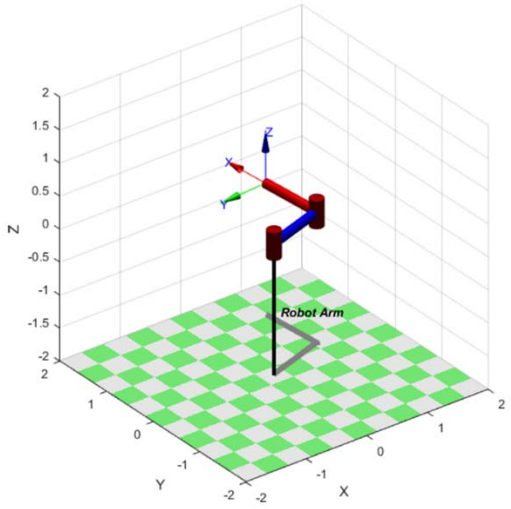


Figure 9: 2DOF robotic arm in single plane simulation.

The initial results of the simulations show that the system is initially feasible. Figure 10 indicates that the trajectory path planning is the most direct method of moving from the starting configuration to the ending position, where the path is essentially a straight line from start to end. This is because when executing Jacobian path following the arm moves both of its degrees of freedom to achieve a mostly straight line where visual servoing will primarily operate fewer joints which is why the path is curved. What this simulation fails to take into account is the errors that are inherent in this approach. These errors are due to errors inherent in the sensor, for this application a 3D camera, where at certain distances from the sensor the Jacobian path following is expected to be less accurate. This means that in practice the robotic arm will not always follow such a straight line from start to finish and will probably arrive at an inaccurate ending position.

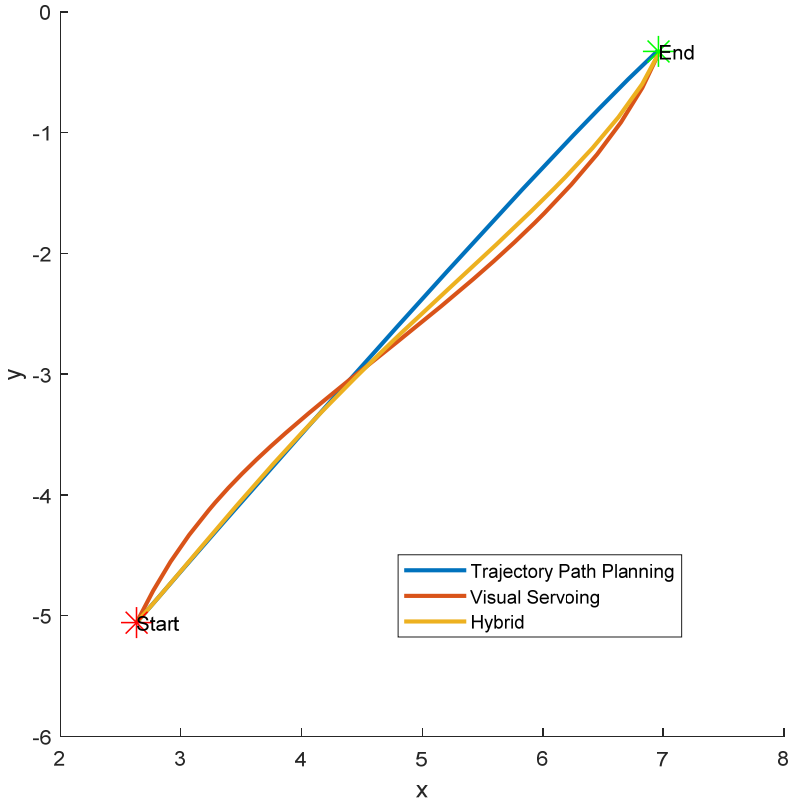


Figure 10: Initial Results of 2D representation of camera and point

The path which the hybrid controller followed was initially similar to the model-based trajectory planned path and then as weight was driven to zero as it approached the end position the hybrid controller was informed increasingly by the visual servoing control.

The simulation of a 6 DOF robotic arm has a simulated 2D camera at the end-effector and a 3D camera mounted close to the origin of the arm. The simulation tracks the position and orientation of the arm, the commanded joint angles as is shown in Figure 11.

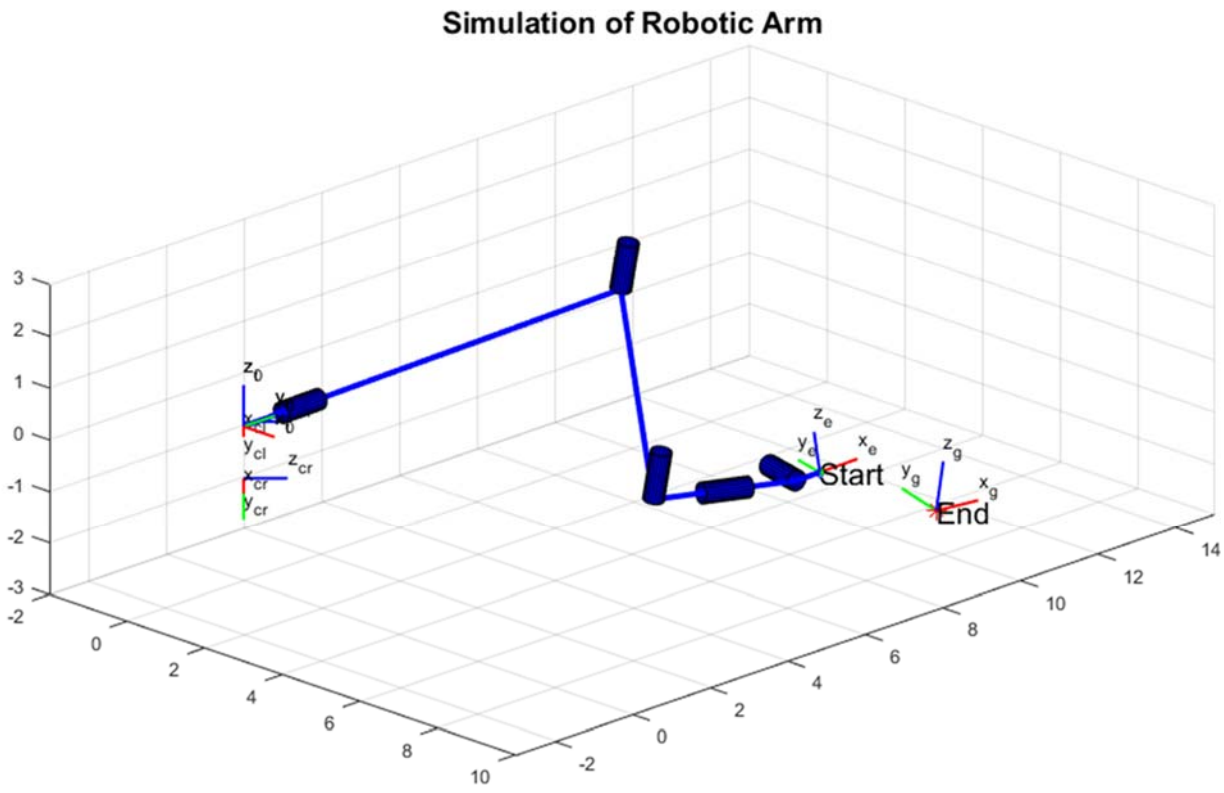


Figure 11: Simulation setup for 6DOF arm.

The kinematics of the arm are described in the Denavit-Hartenberg table given in Table 1.

Table 1: D-H Table for 6DOF Simulated Arm

Joint i	α_i	a_i	d_i	θ_i
1	0	0	0	θ_1^*
2	0	0	l_1	0
3	θ_2^*	l_2	0	θ_3^*
4	0	l_3	0	θ_3^*
5	0	l_4	0	0
6	θ_5^*	l_5	0	0

It also tracks the position of the target point in the field of view of the 2D camera. The view from the camera can be seen in Figure 12.

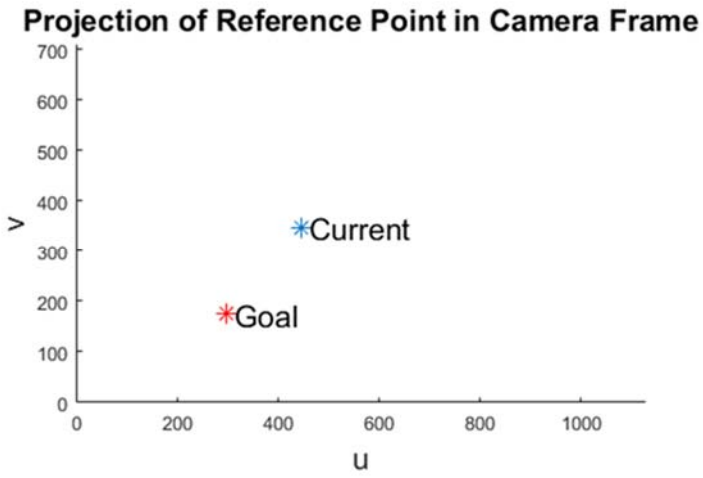


Figure 12: Simulated view of target point in the frame of a camera mounted onto the end effector of the robotic arm.

The simulation runs each method (Jacobian path following, visual servoing, and the hybrid) and tracks the required data over iterations. Then each method is compared in the graphs below. Figure 13 shows a close up of each trajectory.

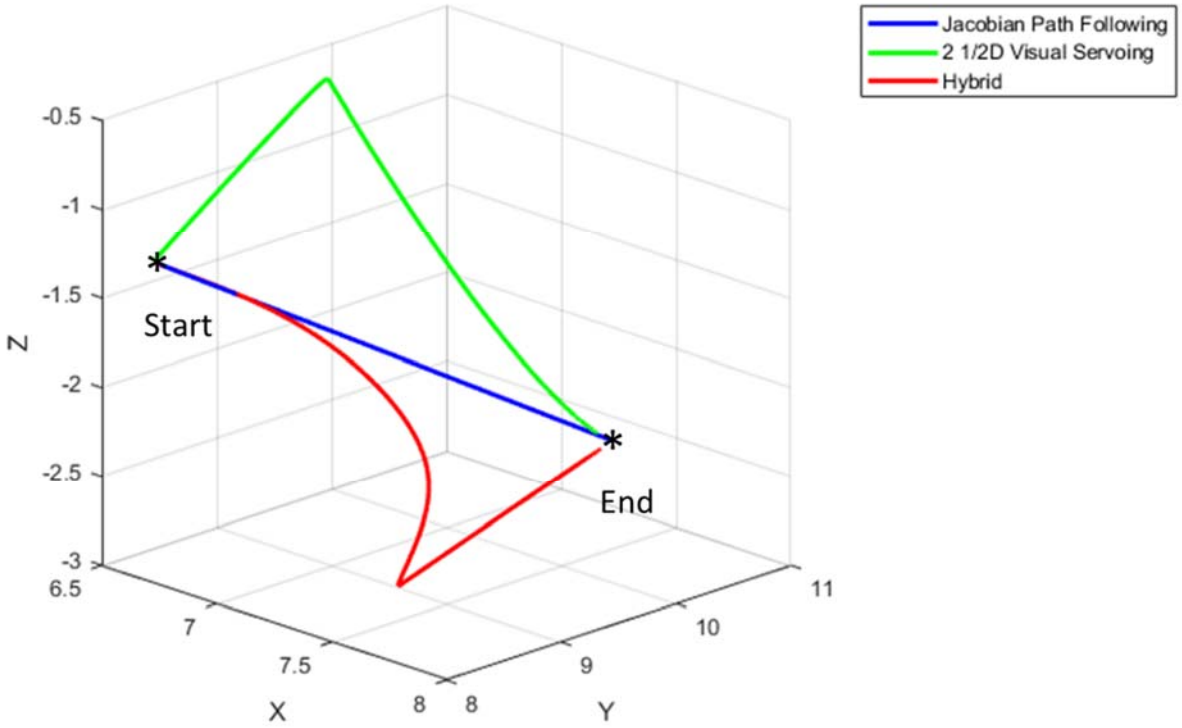


Figure 13: The trajectory of the Jacobian path following, visual servoing and hybrid.

As expected the Jacobian path following traces an almost straight line from the start position to the ending position. The visual servoing firsts orients itself in the correct direction and deviates from the path before servoing in on the final position. The hybrid first traverses the path planning

trajectory before lining up the orientation as it switches to visual servoing. Figure 14 shows the translational and rotation error over each iteration of each controller.

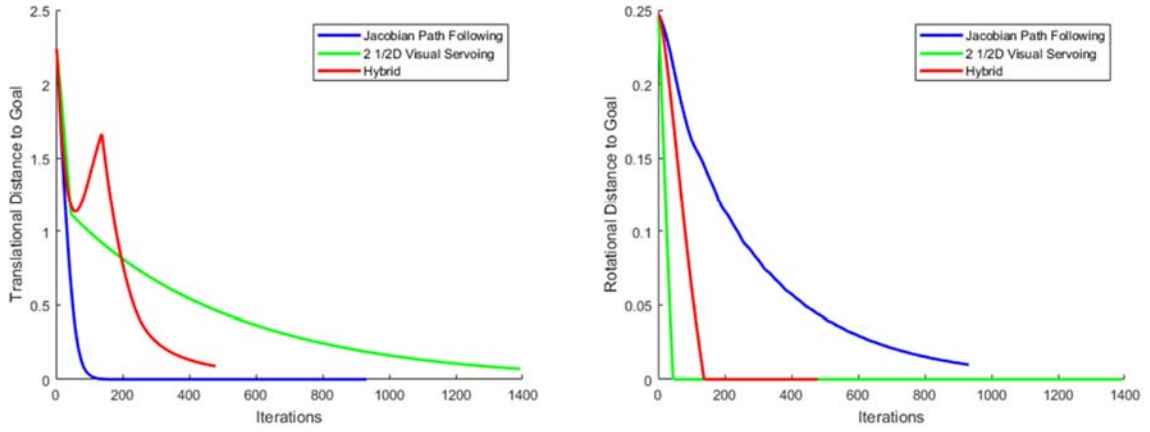


Figure 14: Translational (left) and rotation error (right) for each controller over iterations

The combined error terms are critical to the weighting function and as the graph shows, the hybrid approach balances both by decreasing the translational steady state error as compared to visual servoing and decreasing the rotational error of the system as compared to Jacobian path following. The inclusion of the error models for the cameras in the simulation does not lessen these effects as shown in Figure 15.

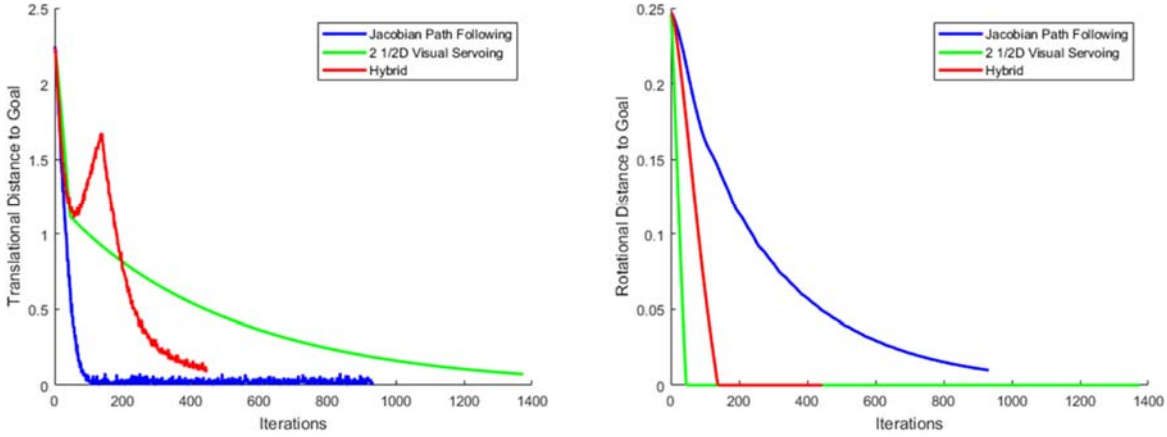


Figure 15: Translational (left) and rotational error (right) for each controller over iterations with camera error

There are significant changes in joint angle right around the halfway point in the trajectory which can be seen in Figure 16. Those changes are mitigated by setting a maximum absolute change in the commanded joint movements.

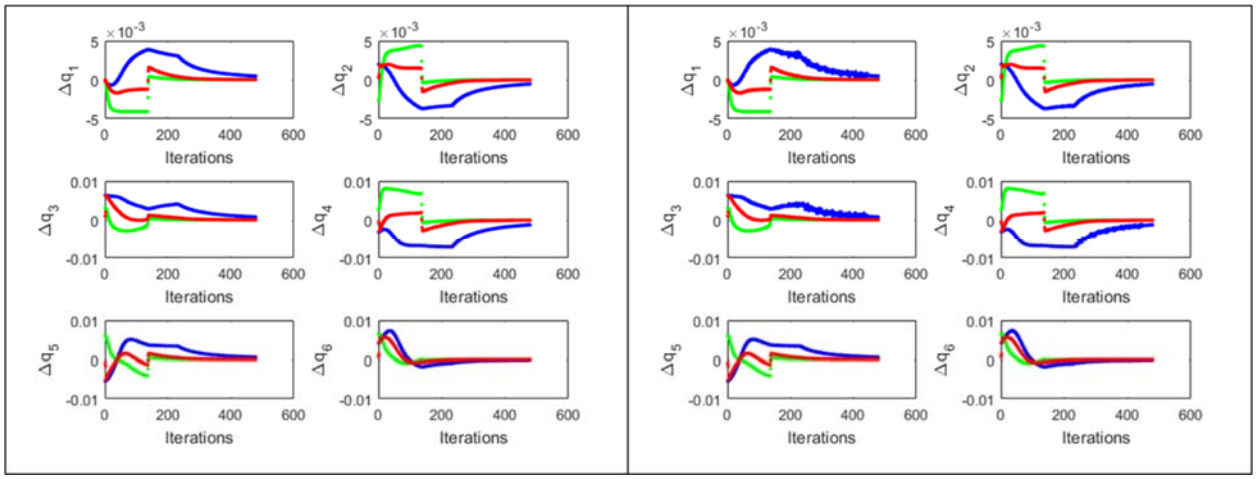


Figure 16: Joint angles over iterations without error (left) and with error (right).

The second simulation is the ScorBot ER-4U which is a 5DOF robotic arm that is run in simulation. In order to be able to invert the Jacobian matrix which is now 5X6 the Moore Penrose Pseudo Inverse is used in place of the inverse in the algorithms. Otherwise the robot executes the exact same controller as demonstrated in the previous simulation.

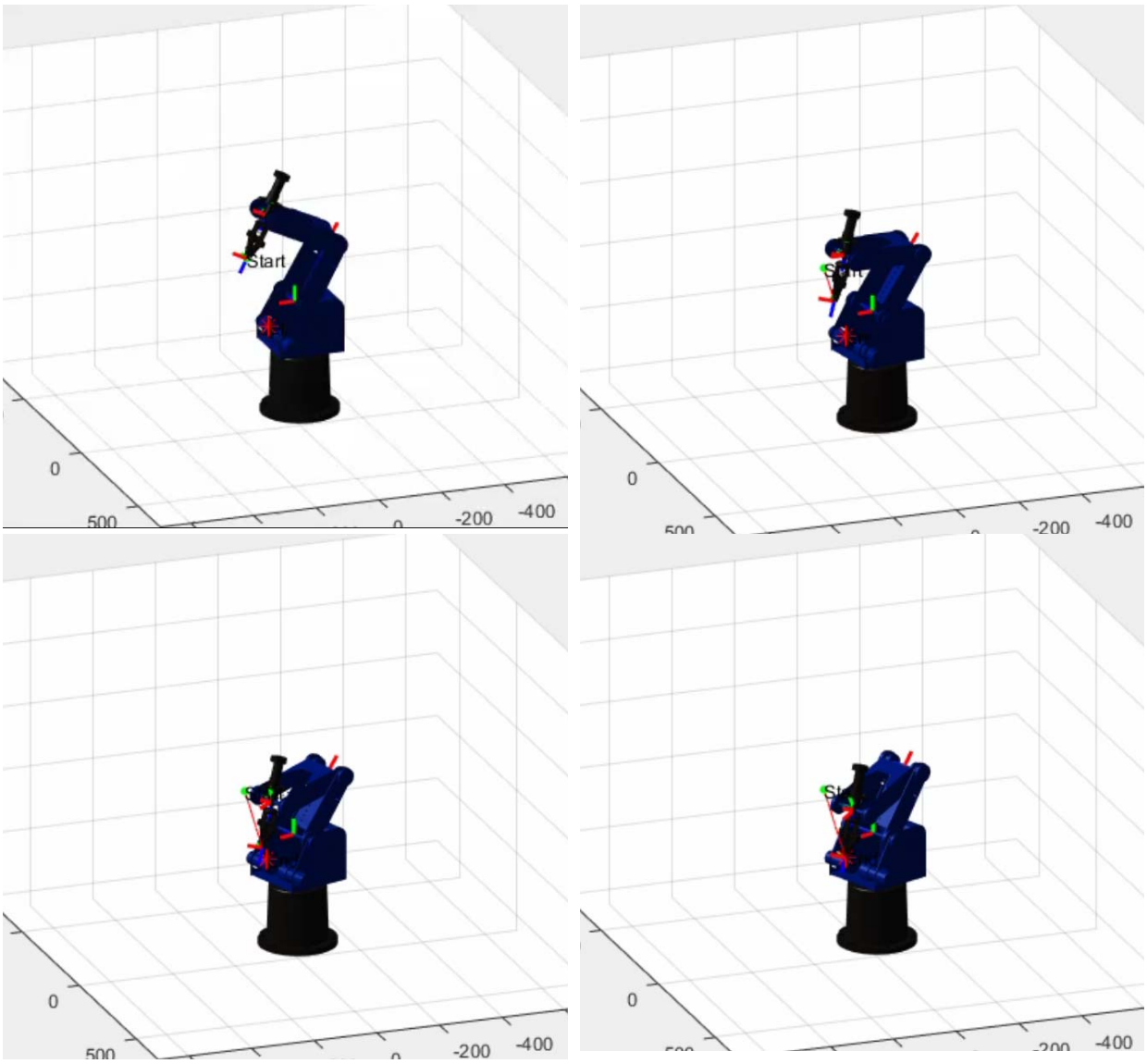


Figure 17: Trajectory of ScoreBot executing the hybrid controller.

This implementation suggests that this hybrid controller can work for industrial manipulators. The trajectory the arm follows is mostly a linear trajectory in the workspace which suggests that the visual servoing and the Jacobian path following methods resulted in similar calculations for the trajectory. This results in the seamless transition between the two controllers.

The next arm tested is the UR5 industrial manipulator. The trajectory of that manipulator is given in Figure 18.

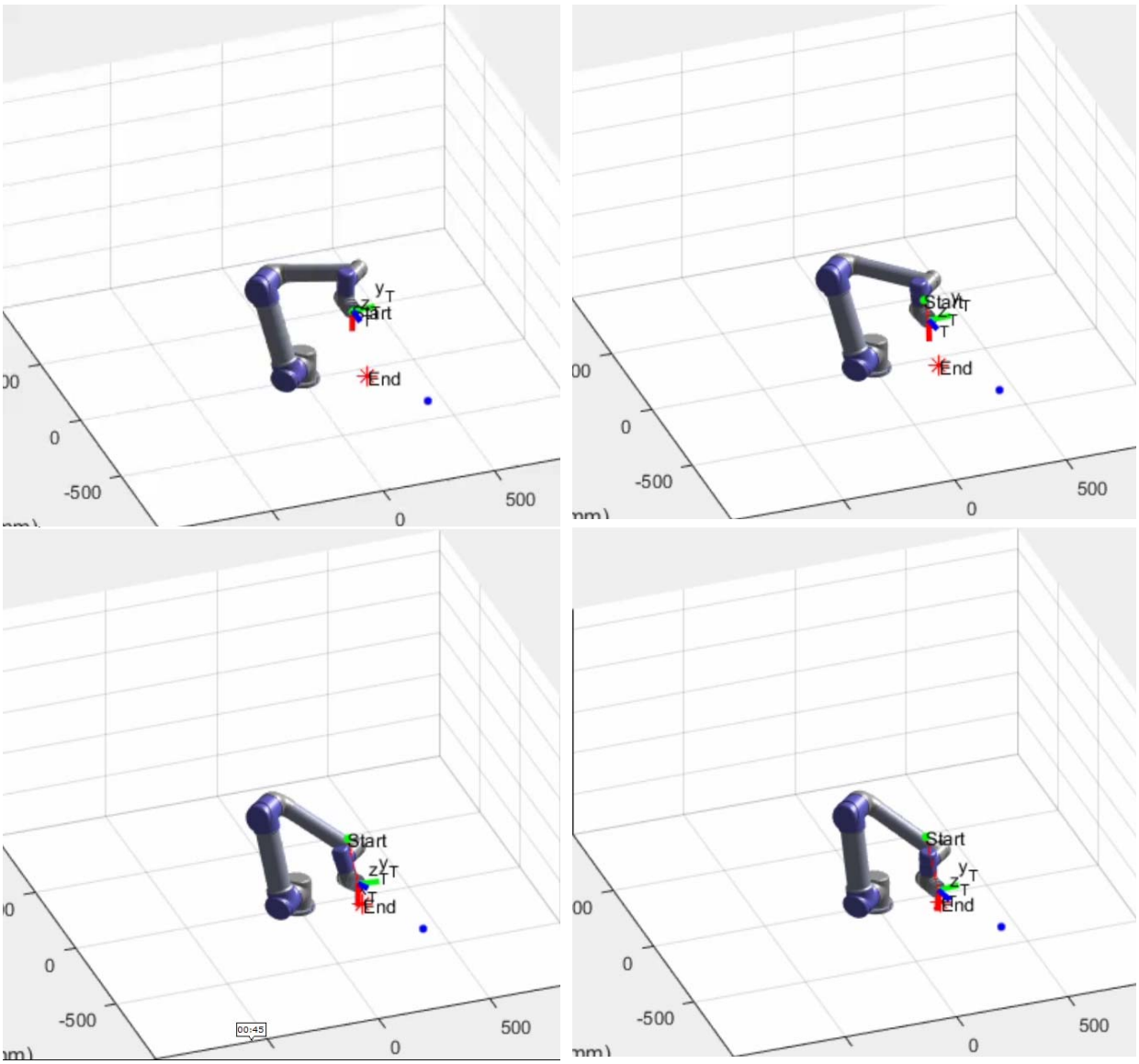


Figure 18: Trajectory of UR 5 executing the hybrid controller in simulation.

The trajectory of the arm is as expected with the arm moving from star to end configuration. The weight shifts from 1 to 0 over time in a predictable manner which suggests the arm is executing the hybrid controller as the equations predict. Then that same controller is tested to see if it can traverse the same trajectory when running on the actual UR5 pictured in Figure 19.

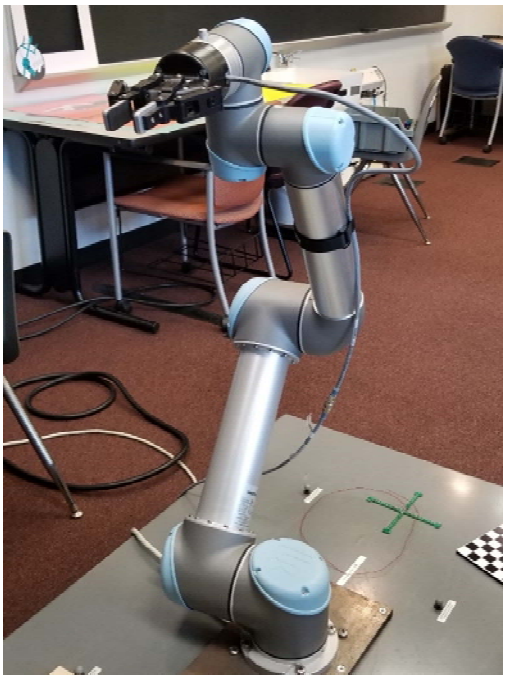


Figure 19: The U5 manipulator.

A frame breakdown of the motion is shown in Figure 20.

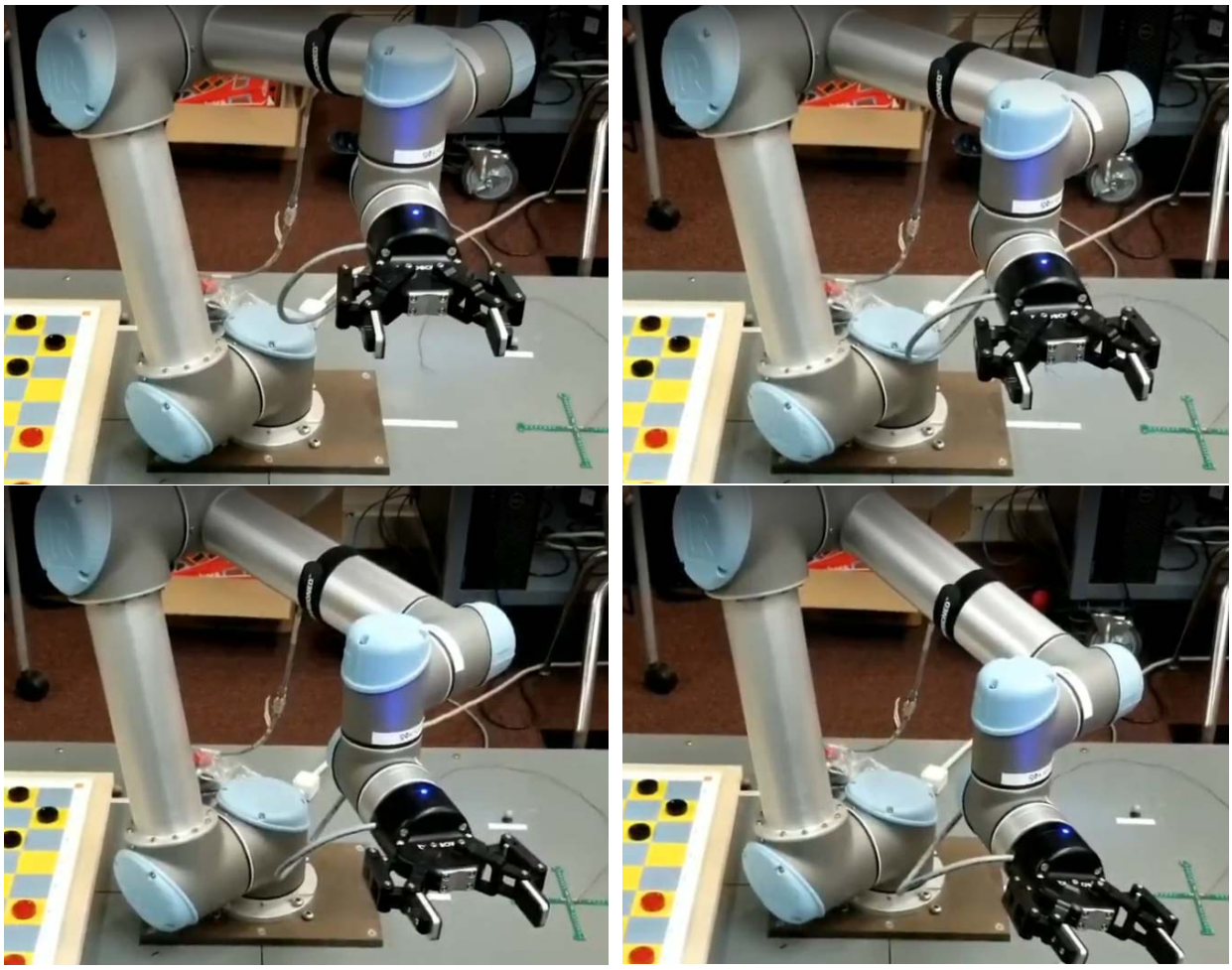


Figure 20: Trajectory of UR 5 executing the hybrid controller.

The UR5 performs the trajectory just as it is executed in the simulation with the weight changing over iterations and the arm performing visual servoing at the end of its path. This test positively suggests that this controller can be used safely on industrial, commercially available robotic arms.

All of these tests demonstrate the viability of this method to be used in the general field of robotics. Specific to the application of this project, space, these results demonstrate the possibility of integrating the system into common robotic systems. Though the method was tested in hardware and simulation with robots used on the ground, their general construction and standard operation is comparable to robotic arms in space. In addition the initial simulation of the system used the kinematics of a robotic arm that will be used in spacecraft applications.

Conclusions

As expected the Jacobian path following traces an almost straight line, visual servoing firsts orients itself in the correct direction and deviates from the path before servoing in on the final position. The combined error terms are critical to the weighting function and as the graph shows,

the hybrid approach balances both by decreasing the translational steady state error as compared to visual servoing and decreasing the rotational error of the system as compared to Jacobian path following. The inclusion of the error models for the cameras in the simulation does not lessen these effects. The stability proof shows that for these two methods that are independently stable that they can be combined using a weighting function and as long as that weight is between 0 and 1 the system is stable. The simulations and experiments support that this is a viable controller for robotic arms.

1. Discussion

The hardware testing for this system was limited due to problems with integrating all of the hardware components into ROS. The hardware required for a full proof of concept would take a considerable amount of effort. But a combined hardware system would better demonstrate the feasibility of using this controller with actual robotic arms.

2. Future Work

This system could be tested in a number of robotic arms to show that it is a robust approach to robotics. This could also be implemented into a robotic arm that is going to space in 2020 as part of the Intelligent Space Assembly Robot project.

References

[1] D. Sternberg et al., "Reconfigurable ground and flight testing facility for robotic servicing, capture, and assembly," 2016 IEEE Aerospace Conference, Big Sky, MT, 2016, pp. 1-13.

[2] K. Belvin, B. Doggett, and J. Watson, "In-Space Structural Assembly: Applications and Technology," AIAA SciTech 2016. pp. 1-5, 2016.

[3] F. Chaumette and S. Hutchinson, "Visual servo control. II. Advanced approaches [Tutorial]," IEEE Robotics & Automation Magazine, vol. 14, no. 1, pp. 109-118, March 2007.

[4] M. Aull, "Visual Servoing for an Autonomous Rendezvous and Capture System," Intelligent Service Robotics. vol 2. pp. 131-137, 2009.

[5] R. Paul, "Robot Manipulators: Mathematics, Programming and Control," The MIT Press, Cambridge, Massachusetts. pp. 128-130, 1981.

[6] P. Corke, "Visual Control of Robot Manipulators-a Review," pp. 1-31, 1993.

[7] K. Goldberg, D. Halperin, J. Latombe, and R. Wilson, "Algorithmic Foundations of Robotics," A.K. Peters., 1995.

[8] G. Roesler, "Robotic Servicing of Geosynchronous Satellites (RSGS)," DARPA., 2016.

[9] C. Henshaw, "The DARPA Phoenix Spacecraft Servicing Program: Overview and Plans for

Risk Reduction," International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2014.

[10] A. Ogilvie, J. Allport, "Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System," MDA Corporation, 2008

[11] E. Wolbrecht, V. Chan, D. Reinkensmeyer, and J. Bobrow, "Optimizing Compliant, Model-Based Robotic Assistance to Promote Neurorehabilitation," IEEE Transactions on Neural Systems and Rehabilitation Engineering. vol. 16. pp. 1-12, 2008.

[12] G. Chirikjian, A. Kyatkin, "Engineering Applications of Noncommutative Harmonic Analysis," CRC Press, 2000.

[13] O. Nadal, "Numerical Computation and Avoidance of Manipulator Singularities," Universitat Poletecnica de Catalunya, 2013.

[14] E. Malis, F. Chaumette and S. Boudet, "2 1/2 D visual servoing," in *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238-250, April 1999.

[15] L. Matthies and S. Shafer, "Error modeling in stereo navigation," in IEEE Journal on Robotics and Automation vol. 3 no. 3 pp. 239-248 June 1987.

[16] H. Khalil, "Nonlinear Systems," Prentice-Hall, New Jersey, vol. 2, no. 5, 1996.

[17] C. Cheah and H. Liaw, "Inverse Jacobian regulator with gravity compensation: stability and experiment," in IEEE Transactions on Robotics vol. 21 no. 4 pp. 741-747 August 2005.