AUTONOMOUS RENDEZVOUS CONTROLLER DESIGN USING 3D DEPTH DATA

APPROVED BY SUPERVISING COMMITTEE:

________________________________________
Mo Jamshidi, Ph.D., Chair

________________________________________
Chunjieang Qian, Ph.D.

________________________________________
Daniel Pack, Ph.D.

Accepted:

________________________________________
Dean, Graduate School
DEDICATION
This thesis is dedicated to my family and all my friends for continuing to support me through my time here at UTSA and for the constant motivation to keep working.
AUTONOMOUS RENDEZVOUS CONTROLLER DESIGN USING 3D DEPTH DATA

by

JOAQUIN DANIEL LABRADO B.S.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Engineering
Department of Electrical and Computer Engineering
December 2013
ACKNOWLEDGEMENTS

I would like to extend many thanks to Dr. Mo Jamshidi for continuing to inspire me to follow my passions in robotics and beyond. If it wasn’t for his ACE Lab I never would have realized my passion, and desire in engineering. I would also like to thank Dr. Chunjiang Qian for guiding me in understanding how different controllers work. Also I would like to thank Dr. Daniel Pack who helped me to understand more complex systems and for always being helpful and understanding. I would also like to thank of Yasher for helping me understand how a fuzzy logic system should work in MATLAB and for always giving advice when I needed. Danna and Josh for letting me bounce ideas off them on how to fix problems, and for always making sure I never worked myself to death.

I would like to give special thanks to Patrick Benavidez for everything he has done to help me to learn about robotics and the Kinect. He has always been extremely helpful since the day I met him and has become a close friend of mine over the years. It is safe to say that most of this thesis would not be possible without the guidance of Patrick though out all the years. I also need to thank Brandy Alger for everything she has done to help me reach this point. We have worked together since we have been in undergrads and she helped me many times over. Being my closest friend all these years, I can say without her I would not have pushed myself to the point I am at. I would like to say thank you for continuing to give me inspiration and motivation to be the very best in the field.
Space is starting to become very crowded mostly in low earth orbit (LEO). Current projections are showing that without active debris removal (ADR) missions manned and unmanned operations in LEO will become in danger of collisions. While research is being done on removal techniques most are piloted or sent after only one target. This thesis compares three controllers to be used on a nonlinear active satellite system to rendezvous with a certain target that can be identified from a depth map acquired from the Microsoft Xbox Kinect. The two types of controllers that are compared are a backstepping controller and a Fuzzy logic controller, these controllers are ran independently of each other first then the Lyapunov and fuzzy logic controllers are tested in unison. Initial conditions are provided to the controller
at first before finally using the Kinect, though the use of image processing techniques, are given from the 3D depth data.
2.2: Previous and Proposed Techniques ...........................................8
2.3: Objective of Thesis ......................................................................9

Chapter Three: Modeling of the Spacecraft .....................................11
3.1: The Spacecraft Frame .................................................................11
3.2: The Chaser Model ....................................................................14
3.3: Control Methods .....................................................................15
   3.3.1 Backstepping State Feedback Controller ..............................16
   3.3.2: Fuzzy Controller ..............................................................19

Chapter Four: Target Identification ................................................24
4.1: Targets .................................................................................24
4.2: The Xbox Kinect .................................................................26
   4.2.1: OpenCV and ROS ..........................................................28
4.3: Identification .......................................................................29
   4.3.1: Image Difference ............................................................30
   4.3.2: Grab Cut and Centroid Algorithm .................................30

Chapter Five: Simulation Set-Up and Results .................................33
5.1: Mission Objectives and Assumptions ....................................33
5.2: Simulation Set-Up ...............................................................35
   5.2.1: Controller Simulations ....................................................35
   5.2.2: Kinect Setup ...............................................................36
5.3: Chaser Simulations ................................................................. 37
  5.3.1: Backstepping Controller ................................................... 37
  5.3.2: Fuzzy Logic Controller .................................................... 57
5.4: Kinect Images ........................................................................ 76

Chapter Six: Conclusions and Future Work ..................................... 95
References ..................................................................................... 97
Vita

LIST OF TABLES

Table 1 The “Box Score” of debris by international agency ................. 4
Table 2 Rule set chosen for the Fuzzy Logic Controller ..................... 21
Table 3  Comparison of the Kinect and two flight tested LIDAR cameras ............................ 28
Table 4  Mass of the Chaser Craft .................................................................................. 34
Table 5  Constants for simulations .................................................................................. 35
Table 6  Initial conditions for the controller .................................................................... 36
Table 7  Distance reading of the centroid for the target .................................................. 87

LIST OF FIGURES

Figure 1  Number of objects in Low earth Orbit - LEO ................................................ 3
Figure 2  All fragments from the Iridium, Cosmos and FY-1C collisions........5
Figure 3  Worst case scenario..............................................................................7
Figure 4  A magnetic tether to capture an object..................................................8
Figure 5  Coordinate system for Earth central inertia ..........................................12
Figure 6  The orbital reference frame ....................................................................13
Figure 7  The membership functions for the Fuzzy Controller ..............................20
Figure 8  Rule surface for all three outputs of the Fuzzy Controller...................23
Figure 9  The analogs of the Satellites used ........................................................25
Figure 10 Noisy shadow created from the offset cameras......................................26
Figure 11 Shows the IR projection from the Kinect..................................................27
Figure 12 LDRI range information of the ISS from testing during STS-91 ..........27
Figure 13 Kinect taking two pictures to use for the difference .............................30
Figure 14 The process used to find the target and eliminate the background .........31
Figure 15 The centroid is label as the red X............................................................32
Figure 16 The Chaser Craft design........................................................................34
Figure 17 Setup of the Kinect looking at the targets ............................................36
Figure 18 Output of the ode45 simulation with no controller ..............................37
Figure 19 Output of the ode45 simulation with the backstepping controller ......38
Figure 20 The position of the chaser ....................................................................39
Figure 21 The velocity of the chaser ......................................................................40
Figure 22  Forces of the chaser.................................................................41
Figure 23  3D Plots of the Chaser’s path.........................................................41
Figure 24  The position of the chaser ...............................................................42
Figure 25  The velocity of the chaser ...............................................................43
Figure 26  Forces of the chaser ..................................................................44
Figure 27  3D Plots of the Chaser’s path.........................................................45
Figure 28  The position of the chaser ...............................................................46
Figure 29  The velocity of the chaser ...............................................................47
Figure 30  Forces of the chaser ..................................................................48
Figure 31  3D Plots of the Chaser’s path.........................................................49
Figure 32  The position of the chaser ...............................................................50
Figure 33  The velocity of the chaser ...............................................................51
Figure 34  Forces of the chaser ..................................................................52
Figure 35  3D Plots of the Chaser’s path.........................................................52
Figure 36  The position of the chaser ...............................................................53
Figure 37  The velocity of the chaser ...............................................................54
Figure 38  Forces of the chaser ..................................................................55
Figure 39  3D Plots of the Chaser’s path.........................................................56
Figure 40  3D plots of the Chaser’s path.........................................................57
Figure 41  The position of the chaser ...............................................................58
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>The velocity of the chaser</td>
<td>59</td>
</tr>
<tr>
<td>43</td>
<td>Forces of the chaser</td>
<td>60</td>
</tr>
<tr>
<td>44</td>
<td>3D plots of the Chaser’s path</td>
<td>61</td>
</tr>
<tr>
<td>45</td>
<td>The position of the chaser</td>
<td>62</td>
</tr>
<tr>
<td>46</td>
<td>The velocity of the chaser</td>
<td>63</td>
</tr>
<tr>
<td>47</td>
<td>Forces of the chaser</td>
<td>64</td>
</tr>
<tr>
<td>48</td>
<td>3D plots of the Chaser’s path</td>
<td>65</td>
</tr>
<tr>
<td>49</td>
<td>The position of the chaser</td>
<td>66</td>
</tr>
<tr>
<td>50</td>
<td>The velocity of the chaser</td>
<td>67</td>
</tr>
<tr>
<td>51</td>
<td>Forces of the chaser</td>
<td>68</td>
</tr>
<tr>
<td>52</td>
<td>3D plots of the Chaser’s path</td>
<td>69</td>
</tr>
<tr>
<td>53</td>
<td>The position of the chaser</td>
<td>69</td>
</tr>
<tr>
<td>54</td>
<td>The velocity of the chaser</td>
<td>70</td>
</tr>
<tr>
<td>55</td>
<td>Forces of the chaser</td>
<td>71</td>
</tr>
<tr>
<td>56</td>
<td>3D plots of the Chaser’s path</td>
<td>72</td>
</tr>
<tr>
<td>57</td>
<td>The position of the chaser</td>
<td>73</td>
</tr>
<tr>
<td>58</td>
<td>The velocity of the chaser</td>
<td>74</td>
</tr>
<tr>
<td>59</td>
<td>Forces of the chaser</td>
<td>75</td>
</tr>
<tr>
<td>60</td>
<td>The depth background of the simulation area</td>
<td>76</td>
</tr>
<tr>
<td>61</td>
<td>Depth images of the targets</td>
<td>77</td>
</tr>
</tbody>
</table>
Figure 62  Results of the image difference for the targets..........................78
Figure 63  Rectangle around each of the targets .....................................79
Figure 64  Background removed for each of the targets .............................80
Figure 65  Location of the centroid for the FASTSAT-HSV ......................81
Figure 66  Location of the centroid for the Iridium .................................81
Figure 67  Location of the centroid for the NAVSTAR-2F .......................82
Figure 68  Location of the centroid for the FASTSAT-HSV ......................83
Figure 69  Location of the centroid for the Iridium .................................83
Figure 70  Location of the centroid for the NAVSTAR-2F .......................84
Figure 71  Location of the centroid for the FASTSAT-HSV ......................85
Figure 72  Location of the centroid for the Iridium .................................85
Figure 73  Location of the centroid for the NAVSTAR-2F .......................86
Figure 74  Position graphs for the chaser approaching the FASTSAT-HSV ....87
Figure 75  Velocity graphs for the chaser approaching the FASTSAT-HSV .....88
Figure 76  Position graphs for the chaser approaching the NAVSTAR-2F .......88
Figure 77  Velocity graphs for the chaser approaching the NAVSTAR-2F .......89
Figure 78  Position graphs for the chaser approaching the Iridium ...............89
Figure 79  Velocity graphs for the chaser approaching the Iridium ...............90
Figure 80  Position graphs for the chaser approaching the FASTSAT-HSV ......91
Figure 81  Velocity graphs for the chaser approaching the FASTSAT-HSV ......91
Figure 82  Position graphs for the chaser approaching the NAVSTAR-2F........92
Figure 83  Velocity graphs for the chaser approaching the NAVSTAR-2F........92
Figure 84  Position graphs for the chaser approaching the Iridium ..................93
Figure 85  Velocity graphs for the chaser approaching the Iridium.................93
CHAPTER ONE: INTRODUCTION

The expansion into space has brought many new challenges to overcome. One challenge is the growing threat of orbital debris. According to NASA, orbital debris is classified as any object in Earth orbit that no longer serves a useful function[1]. These objects create numerous problems for manned and unmanned operations in orbit, because of the damage that can be done to these expensive systems. A study put out by NASA calls for more active debris removal (ADR) missions to help alleviate the growing threat of debris. The study showed that with no debris mitigation, the number of debris would skyrocket by 2030[1, 2]. More autonomous systems are being considered for the removal of debris or even for on-orbit servicing to repair object before they can become debris. With autonomous ADR systems there are many different stages to be able to rendezvous and capture the debris. The most important part of the whole mission is the rendezvous phases of the operation. The whole rendezvous stage of the operations is comprised of five phases[3].

1. Separate orbit: This phase started after the chaser craft has achieved orbit and is in the process of calculating how to start the next phase

2. Drift Orbit A: When this phase starts the target is still out of sight, out of contact and the chaser is in the process of orbital transfers thrusts to get into the same orbit as the target
3. Drift Orbit B: The target is now in sight and relative navigation begins

4. Proximity Operations A: This phase starts when the target is about 1km away from the chaser. This allows the chaser to get into a parking orbit to begin docking procedures.

5. Proximity Operations B: This phase starts when the chaser is about 100m away from the target and has begun to move into dock or capture the target.

This thesis explores two types of controllers to allow a chaser craft to rendezvous with a target as well as targeting techniques using 3D imaging inside of the Proximity Operations B phase. The types of controllers proposed are a nonlinear backstepping controller and a fuzzy logic controller that control the movement of the chaser craft. These types of controllers have been used in the past for control of mobile robots, unmanned aerial vehicles, and on some spacecraft[4-9].

Though the use of a Microsoft Xbox Kinects point cloud data, different techniques are tested to identify the target and obstacles that are in front of the chaser craft so it will not crash and be able to successfully navigate to the target. The Kinect not only uses a RGB camera but it comes equipped with an IR emitter and detector to produce a depth map of the scene in front of the camera. This depth image is
comparable to the types of LIDAR that are used in most spacecraft operations already[10].

This thesis is starts off in Chapter 2 with the background of the problem of orbital debris and different methods of ADR missions. Chapter 3 discusses the design of the backstepping controller and the fuzzy logic controller, then chapter 4 discusses the 3D depth data and how it has been processed in flights and how we are processing it in the thesis. Chapters 5 and 6 give the conditions for the simulations and the results of the controller and Kinect simulations. Lastly all the conclusions and future work planned are discussed in Chapter 7.

CHAPTER TWO: ORBITAL DEBRIS REMOVAL

2.1 Motivation for Removal

Orbital Debris has become a major problem for both manned and unmanned operations in orbit. NASA estimates that 94% of all orbital objects are debris, where 20,000 of these objects are softball size or larger. While the thesis focuses on these larger objects it is important to note that smaller debris also pose a serious threat to extra-vehicular activities (EVA). These objects come from many different sources
such as non-operational spacecraft, mission-related debris, derelict launch vehicle stages, and fragmentation debris.

Figure 1 Number of objects in Low earth Orbit - LEO[1]

There have been numerous events in the past, such as collisions and explosions, which have created more and more debris in the area. In three cases to date, the International Space Station (ISS) had to be evacuated or had to preform maneuvers to avoid debris[11-13]. These types of debris usually come from other satellites colliding. Collisions are very costly because it not only creates more debris but it can also destroy functioning satellites or put man operations in danger from the debris cloud. One such case of this is the accidental collision of the Iridium 33
and the Kosmos 2251 satellites[1, 13]. This event destroyed both satellites and created more than 2000 objects in low earth orbit (LEO). According to the NASA Orbital Debris Program office about 1800 are still in orbit. It is classified as the worst collisions in history because of the number of fragments that were created as a result of the collision.

Table 1 : The “Box Score” of debris by international agency[14]

<table>
<thead>
<tr>
<th>Country/Organization</th>
<th>Payloads</th>
<th>Rocket Bodies &amp; Debris</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHINA</td>
<td>140</td>
<td>3612</td>
<td>3752</td>
</tr>
<tr>
<td>CIS</td>
<td>1427</td>
<td>4830</td>
<td>6257</td>
</tr>
<tr>
<td>ESA</td>
<td>42</td>
<td>46</td>
<td>88</td>
</tr>
<tr>
<td>FRANCE</td>
<td>56</td>
<td>442</td>
<td>498</td>
</tr>
<tr>
<td>INDIA</td>
<td>49</td>
<td>125</td>
<td>174</td>
</tr>
<tr>
<td>JAPAN</td>
<td>125</td>
<td>83</td>
<td>208</td>
</tr>
<tr>
<td>USA</td>
<td>1134</td>
<td>3804</td>
<td>4938</td>
</tr>
<tr>
<td>OTHER</td>
<td>615</td>
<td>119</td>
<td>734</td>
</tr>
<tr>
<td>TOTAL</td>
<td>3588</td>
<td>13061</td>
<td>16649</td>
</tr>
</tbody>
</table>

As table 1 shows many countries have contributed to orbital debris. Most of the debris that the Chinese have generated came from their antisatellite tests. The
worst test was the collision of the FY-1C satellite. This generated more debris than the collision of the Iridium and Cosmos.

Figure 2 All fragments from the Iridium, Cosmos and FY-1C collisions[13]

Figure 2 shows the density of the debris that was created after the events of the collisions and antisatellite tests. This figure shows that most of the debris was created in LEO where most of human operations take place.

These fragments that are created can go off and collide with other satellites or manned spacecraft. Such was the case for a small Russian satellite called BLITS (Ball Lens In The Space) was struck by a small fragment from a Chinese satellite. Due to the small size of both objects the only way this collision was noticed was due
to BLITS having reported a slower spin rate[14]. While collisions are relatively rare they are very significant when they do occur.

Collisions are not the only danger that comes from orbital debris; explosions can create just as many if not more fragments. Explosions mainly originate from the upper stages of launch vehicles which were due to left over fuel that ignited. One such event occurred in 2001 when a Russian satellite, Cosmos 2367, experienced a major fragmentation event in an orbit just 30 km above the International Space Station (ISS). Approximately 200 objects were detected by the Space Surveillance Network and about 40% crossed the orbit of the ISS [15]. Luckily, these events were minimized in 2007 with the passivation of the upper stages to help discharge any stored up energy by discharging batteries and venting any remaining fuel. Since the adopting of this technique orbital debris that originated from explosions has been reduced. Without passivation of the upper stages one runs the risk of a time bomb sitting in orbit that could explode next to the recently launched object or another one passing by.

Orbital debris projections from NASA’s Johnson Space Center for the worst case scenario show debris in LEO to be increasing over the next 200 years with just regular satellite launches. Figure 3 shows the worst case scenario of no mitigation of orbital debris. As it can be seen LEO will become very crowded if something is not done[16].
One major thing learned from many studies is that current policies of mitigation will not stem the influx of orbital debris. Kessler Syndrome was an idea pushed by Donald Kessler in 1978 which states that as the number of debris increases the rate of collisions will increase as well. Each one of these collisions would create more debris and thus increase the rate of collisions. Because of Kessler Syndrome many spacefaring nations would create policies that would help mitigate the population of space debris. One of the main policies that have been adopted is the 25 year decay rule which states that anything that is put into orbit must be able to deorbit in 25 years. The Inter-Agency Space Debris Coordination Committee (IDAC), which includes all the major space agencies of the world, has a set of strict guidelines that each agency must follow when putting something in orbit.
However, this alone is not good enough to help mitigate the debris, which is the reason why active debris removal missions (ADR) are being considered. While there are many challenges that face ADR operations many spacefaring nations have begun to look at ways to remove debris with autonomous and non-autonomous ADR missions[17].

2.2 Previous and Proposed Techniques

The concept of ADR has been bouncing around the space community for some time now. When attempting to rendezvous with the debris the first consideration is the size of debris that the chaser craft will be targeting as well as the location of the debris. The majority of the objects that are targeted for being ADR missions are in LEO. This is due to the fact that LEO is a high traffic area for both manned and unmanned missions. A number of different methods have been proposed to remove debris such as the use of tethers or nets to be able to actively capture the target.
Figure 4: a magnetic tether to capture an object

These techniques would involve a chaser craft that could rendezvous with the target in a higher orbit and “drop” these objects down to capture the target such as shown in figure 4. While this method could be used the logistics of the entire mission plan make it a very complicated mission.

Another method described is the most widely proposed method to remove debris and to manipulate satellites in orbit. This involves having the chaser craft reach the same orbit as the target then slowly maneuver, before using the end effectors, to capture the target in order to maneuver the target to a lower orbit need to de-orbit or return it to a human operated spacecraft. This method is planned to remove the FASTSAT satellite and could be used on a number of other debris. Grappling the debris is a challenge unto itself and there are numerous examples of different methods such as the use of robotic arms with different end effectors, the use of a net to capture the debris, harpoon systems, etc. Despite the popularity of this method one of the draw backs would be the chaser would need to be refueled in order to make many of the orbital maneuvers it would need.

DARPA has also begun to look into the issue of orbital debris with a project called Catcher’s Mitt. This is a study to cover all the issues and challenges in
removing orbital debris. DARPA hopes the study will help create better models for debris; then determine what the best approach, if there is, to remove the debris.

### 2.3 Objectives of thesis

A full scale space debris mission would entail the following steps for the chaser craft:

1.) Reach the orbit of the target craft

2.) Identify the target craft

3.) Align itself for the proper approach with the target craft

4.) Begin approach to the target craft

5.) During approach phase check for obstacles

6.) Inspect target craft

7.) Capture

This thesis focuses on steps 2 – 5 where chaser will use depth data from the Kinect in order to identify the target before the approach begins. Using the depth data from the Kinect we will use a grab cut method to filter out the background noise before finding the centroid of the target. This data will then be used as the initial conditions for both of the controllers. Once the chaser is in the right location proximity operations can begin which includes the rendezvous and inspection of the target. Two controllers will be compared to a nonlinear controller derived through
backstepping and a Fuzzy Logic Controller. Both controllers are tested along the same parameters.

CHAPTER THREE: MODELING OF THE SPACECRAFT

This chapter covers the chaser model and the controllers that were designed. First, we cover the frame of reference that the chaser is using. Next, we define the model
and state the assumptions before arriving at the final states of the system. Lastly, we designed a nonlinear backstepping controller, and fuzzy logic controller.

3.1 The Spacecraft Frame

Before the equations of motion of the chaser can be described, the coordinate system needs to be established. The reference frame is very important for all orbital operations. Since there are many different types of operations that can be done in orbit, there are also many different reference frames that can be used to describe orbital motion. Due to the large emptiness of space and lack of identifiable landmarks makes creating reference frames rather difficult. One of the first considerations for a reference frame would be the Earth centered inertial (ECI), often referred to as the celestial coordinate system, which places the Earth at the origin of the coordinate system and is a nonrotating frame. Figure 5 shows this coordinate system.
Another called Earth centered and Earth fixed (ECEF) which is very much like ECI except that objects in orbit stay relative to the surface of the Earth due to the fact that ECEF frames are rotating frames. The problem with ECI frames is that they do not take into consideration the acceleration of the Earth itself. ECI and ECEF frames, while useful, are not ideal for rendezvous operations due to the vast distances and orbital angles at which targets and chaser craft alike will have [Reference??]. This is why a reference frame that is orbit based would be ideal for any orbital debris maneuvers or rendezvous procedures.

Another frame of reference that could be used is the Body-Fixed Frame which is more commonly used for aircraft but can be used for spacecraft. The origin of this frame is centered on the center of mass for the craft, yet one major problem is that it is fixed and not rotating. Since modeling spacecraft’s requires a rotating reference
frame, the orbital frame is chosen which is popular for rendezvous and formation flights in orbits. The frame places the origin at the center of mass of the main spacecraft, in this case the target, and creates a coordinate system around the spacecraft using the orbital elements of motion[18]. For the rendezvous operation the system is described in relative to the target. This is done because as the chaser approaches the target most of the relative orbital errors between the target and the chaser have minimal impact on the system. This coordinate system has its origin at the center of mass of the target craft. This type of rotating frame is called the Local-Vertical-Local-Horizontal (LVLH) frame which allows for two separate spacecraft to either fly in formation or any rendezvous procedures. Figure 6 shows the reference frame that is used in this thesis.

![Figure 6: The orbital reference frame](image)

Using the orbital reference frame, we can use the rotating Hill frame to better describe the motion of the chaser craft[19]. The orbital reference frame is earth centered, orbit based, and rotating which allows for the concept of row, pitch, and
yaw to be used to described the motion in any direction. From Figure 6 we can see that the positive x direction is an extension of the $r_c$ vector going from the center of the earth to the center of mass of the target. The positive y is the vector direction of the orbital velocity vector, and the positive z direction is the orbital normal. This reference frame is used in textbooks for formation flying of spacecraft and rendezvous operations in orbit because it places the target or the main craft as the origin.

### 3.2 The Chaser Model

Because of the LVLH reference fame the full nonlinear equations of motion are as follows\cite{19}:

\begin{align}
\ddot{x} - 2\dot{\theta} \dot{y} - \dot{\theta} y - \dot{\theta}^2 x - 2\left(\frac{\mu}{r_c^3}\right)x &= -\frac{\mu(r_c+x)}{\rho^3} + \frac{\mu}{r_c^2} - 2\left(\frac{\mu}{r_c^3}\right)x + F_x/m \quad (1) \\
\ddot{y} + 2\dot{\theta} x + \dot{\theta} x - \dot{\theta}^2 y - \left(\frac{\mu}{r_c^3}\right)y &= -\frac{\mu y}{\rho^3} + \left(\frac{\mu}{r_c^3}\right)y + F_y/m \quad (2) \\
\ddot{z} + \left(\frac{\mu}{r_c^3}\right) z &= -\frac{\mu z}{\rho^3} + \left(\frac{\mu}{r_c^3}\right)z + F_z/m \quad (3) \\
\ddot{r_c} &= r_c \dot{\theta}^2 - \frac{\mu}{r_c^2} \quad (4) \\
\ddot{\theta} &= -2 \left(\frac{\dot{r}_c \dot{\theta}}{r_c}\right) \quad (5) \\
\rho &= \sqrt{(r_c + x)^2 + y^2 + z^2} \quad (6)
\end{align}
where $x$, $y$, and $z$ represent the distances the chaser is with respect to the target relative to each axis, $r_c$ represents the scalar distance from the center of the Earth to the target, $\rho$ represents the scalar distance from the center of the Earth to the chaser, $\theta$ is the true anomaly at which the target debris is orbiting at, the values $F_x$, $F_y$ and $F_z$ represent the thrust force applied to the system and $m$ represents the mass of the spacecraft. Some assumptions must be made in order to model the chaser system as it must move in a very complex environment. The following assumptions are made in this thesis:

1) All orbital maneuvers have been completed upon the start of the simulations and the chaser is at the start of Proximity Operations A (~1000m apart).

2) The target and the chaser are in a circular orbit in LEO therefore the true anomaly has a constant velocity.

3) There are no outside disturbances on the chaser or the target

Assumption one is chosen due to the nature of the thesis is for the approach phase of the rendezvous not the orbital mechanics of catching up to the target. LEO is where most of the human operations take place in orbital operations and where most debris is located. The circular orbit is picked to help reduce the amount of nonlinearities in the state equations.
By using equations 4 and 5 the Clohessy-Wlitshire model (Hill’s equations) can be formed which can be formed into nonlinear states, 
\[ [x_1, x_2, x_3, x_4, x_5, x_6] = [x, \dot{x}, y, \dot{y}, z, \dot{z}] \].

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 \\
\dot{x}_4 \\
\dot{x}_5 \\
\dot{x}_6 \\
\end{bmatrix}
= 
\begin{bmatrix}
2\dot{\theta}x_4 + \dot{\theta}^2x_1 - \frac{\mu(r_c + x_1)}{\rho^3} + \frac{\mu}{r_c^2} + F_x/m \\
x_4 \\
-2\dot{\theta}x_2 + \dot{\theta}^2x_3 - \frac{\mu x_3}{\rho^3} + F_y/m \\
x_6 \\
-\frac{\mu x_5}{\rho^3} + F_z/m
\end{bmatrix}
\]

(7)

3.3 Control Methods

Autonomous motion of spacecraft for rendezvous procedures is widely used for resupply missions with the ISS and is now beginning to gain ground for debris removal and refueling operations. Recently the Dragon capsule, developed by SpaceX, uses LiDAR and thermal cameras as “eyes” as it approaches the ISS. These types of maneuvers are meant to be done very slowly so there is a lot of time to processes the data from the LiDAR and cameras. Preferred controllers of systems in orbit such as docking with the ISS or any object with humans inside will be piloted by humans for safety purposes.

For autonomous flight the more popular controllers tend to be Bang-Bang and PID controllers although research has been increasing in more intelligent controllers.
The two controllers that are derived in this thesis are a nonlinear backstepping controller and a fuzzy logic controller. These types of controllers have been proposed before but have never been used in real flight missions.

3.3.1 Backstepping State Feedback Controller

Backstepping is a method to create a nonlinear stabilizer for systems in cascade. This method is based on the use of Lyapunov theory which ensures that the system is stable for each state. The idea behind the design of the controller is create a virtual controller that will stabilize each state until the final “true” controller is reached. These types of controllers are used in a variety of systems such as control of motors, path tracking, and flight controls. One of the advantages of using the backstepping method is to help create a controller that stabilizes each state. Backstepping controllers can be the basis for more advanced controllers such as state feedback and output feedback controllers.

The controller, that is to be built for the chaser system, has to be able to successfully maneuver the chaser craft to the target, which is at the origin of the system. The controller is designed for each direction therefore three separate controllers are built for the thrust force, \( F = [F_x \ F_y \ F_z] \). The first is for the \( F_x \) component which use the states \( x_1 \) and \( x_2 \). The first step to the force is to create a virtual controller for only the \( x_1 \) state.
\[ V_1(x_1) = \frac{x_1^2}{2} \]

From equation (8):

\[ \dot{V}_1(x_1) = x_1 \dot{x}_1 \]

From equation (9):

\[ \dot{V}_1(x_1) = x_1 x_2 \]

The objective is to make \( \dot{V}_1 \) stable for all \( x_1 \) while \( x_2 \) becomes the virtual controller needed to make \( \dot{V}_1 \) stable.

\[ \dot{V}_1(x_1) = x_1 x_2^* + x_1 (x_2 - x_2^*) \]

Where:

\[ x_2^* = -x_1 \]

\[ \dot{V}_1(x_1) = -x_1^2 + x_1 \xi_2 \]

Where:

\[ \xi_2 = (x_2 - x_2^*) \]

The next step is to create \( V_2 \) for all \( x_1 \) and \( x_2 \).

\[ V_2(x_1, x_2) = V_1 + \frac{\xi_2^2}{2} \]
\[ \dot{V}_2(x_1, x_2) = \dot{V}_1 + \xi_2 \dot{\xi}_2 \]

Where:

\[ \dot{\xi}_2 = (\dot{x}_2 - \dot{x}_2^*) \]

\[ \dot{V}_2(x_1, x_2) = \dot{V}_1 + \xi_2 (\dot{x}_2 - \dot{x}_2^*) \]

Next we plug in the states to arrive at the following equation.

\[ \dot{V}_2(x_1, x_2) = -x_1^2 + x_1 \xi_2 + \xi_2 \left( 2 \dot{\theta} x_4 + \dot{\theta}^2 x_1 - \frac{\mu (r_c + x_1)}{\rho^3} + \frac{\mu}{r_c^2} + \frac{F_x}{m} + x_2 \right) \]

The point of this next step is to create a \( F_x \) that will stabilize \( \dot{V}_2 \) for all \( x_1 \) and \( x_2 \).

\[ F_x = m \left( (-x_1 - x_2) - 2 \dot{\theta} x_4 - \dot{\theta}^2 x_1 + \frac{\mu (r_c + x_1)}{\rho^3} - \frac{\mu}{r_c^2} - \xi_2 \right) \]

Plugging \( F_x \) into \( \dot{V}_2 \) shows that this Lyapunov equation is stable for \( x_1 \) and \( x_2 \). Using the same procedure that was used to derive \( F_x \), the control forces \( F_y \) and \( F_z \) can be derived.

\[
F = \begin{bmatrix}
m \left( (-x_1 - x_2) - 2 \dot{\theta} x_4 - \dot{\theta}^2 x_1 + \frac{\mu (r_c + x_1)}{\rho^3} - \frac{\mu}{r_c^2} - \xi_2 \right) \\
m \left( (-x_3 - x_4) + 2 \dot{\theta} x_2 - \dot{\theta}^2 x_3 + \frac{\mu x_3}{\rho^3} - \xi_4 \right) \\
m \left( (-x_5 - x_6) + \frac{\mu x_5}{\rho^3} - \xi_6 \right)
\end{bmatrix}
\]
Where:

\[ \xi_4 = (x_4 - x_4^*) \]

\[ \xi_6 = (x_6 - x_6^*) \]

\[ x_4^* = -x_3 \]

\[ x_6^* = -x_5 \]

This controller ideally should be more responsive than the linear stabilizing controller.

3.3.3 Fuzzy Controller

Fuzzy logic is widely used in control of mobile systems to assist in obstacle avoidance while navigating along paths. Fuzzy logic controllers have a crucial advantage because one can transfer experience to the controller. This means that the fuzzy controller can be trained with human knowledge, such as from a pilot. Due to the usefulness of the controller, many papers have been written on 2D simulations of applications of fuzzy controllers. 3D controllers have been used for spacecraft and aerial robotics. Most of the work done covers stabilization of spacecraft and UAVs [4, 20, 21]. Of the work that has been done for spacecrafts, most work has been done with attitude stabilization of small satellites and obstacle avoidance for rendezvous
Most of the fuzzy controllers that are designed tend to be Takagi-Sugeno type instead of the Mamdani type. The Takagi-Sugeno type uses linguistic rules and linear functions and is used in many nonlinear systems[24]. These types of fuzzy systems are very useful for spacecraft operations.

The controller designed in this thesis is a Mamdani model, which is the most common type of fuzzy controller to use. This type of model is preferred in most cases because it’s easier to translate human experience into the controller. The fuzzy controller that is being designed will take in the error of the chaser’s position in relation to the target and the error of the velocities of the chaser to create rules in the form of IF…AND…THEN… to ensure that the chaser applies the proper thrust to move in the intended direction. Because each of the thrusts that are produced from the controller controls one of the three directions the rule based was designed so that the error in the position and velocity for x only changes the thrust in the x direction. The same was done for the y and z states in order to create a fuzzy controller that is able to control the chaser states to get close enough to the target point.
Figure 7 The membership functions for the Fuzzy Controller

Figure 7 describes the membership functions for the error in the inputs and the resulting output force. The membership functions that were chosen were
Gaussian, Z, and S functions in order to create smoother transitions between the functions. The membership functions for the position and velocity errors where designed to be the same across all three states due to the fact that all three thrust values must all accomplish the same goal.

The fuzzy controller has 25 rules per state, which means that there are 75 rules total as table 2 shows. Then figure 8 (a)

<table>
<thead>
<tr>
<th>Position error (X, Y, Z)</th>
<th>Velocity error (X, Y, Z)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Negative</td>
<td>Big Negative</td>
<td>Big Negative</td>
</tr>
<tr>
<td>Big Negative</td>
<td>Small Negative</td>
<td>Small Negative</td>
</tr>
<tr>
<td>Big Negative</td>
<td>Zero</td>
<td>Small Negative</td>
</tr>
<tr>
<td>Big Negative</td>
<td>Small Positive</td>
<td>Zero</td>
</tr>
<tr>
<td>Big Negative</td>
<td>Big Positive</td>
<td>Zero</td>
</tr>
<tr>
<td>Small Negative</td>
<td>Big Negative</td>
<td>Small Negative</td>
</tr>
<tr>
<td>Small Negative</td>
<td>Small Negative</td>
<td>Small Negative</td>
</tr>
<tr>
<td>Small Negative</td>
<td>Zero</td>
<td>Small Negative</td>
</tr>
<tr>
<td>Small Negative</td>
<td>Small Positive</td>
<td>Zero</td>
</tr>
<tr>
<td>Small Negative</td>
<td>Big Positive</td>
<td>Zero</td>
</tr>
<tr>
<td>Zero</td>
<td>Big Negative</td>
<td>Small Negative</td>
</tr>
<tr>
<td>Zero</td>
<td>Small Negative</td>
<td>Zero</td>
</tr>
<tr>
<td>Zero</td>
<td>Zero</td>
<td>Zero</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Zero</td>
<td>Small Positive</td>
<td>Zero</td>
</tr>
<tr>
<td>Zero</td>
<td>Big Positive</td>
<td>Small Positive</td>
</tr>
<tr>
<td>Small Positive</td>
<td>Big Negative</td>
<td>Zero</td>
</tr>
<tr>
<td>Small Positive</td>
<td>Small Negative</td>
<td>Zero</td>
</tr>
<tr>
<td>Small Positive</td>
<td>Zero</td>
<td>Small Positive</td>
</tr>
<tr>
<td>Small Positive</td>
<td>Small Positive</td>
<td>Small Positive</td>
</tr>
<tr>
<td>Small Positive</td>
<td>Big Positive</td>
<td>Small Positive</td>
</tr>
<tr>
<td>Big Positive</td>
<td>Big Negative</td>
<td>Zero</td>
</tr>
<tr>
<td>Big Positive</td>
<td>Small Negative</td>
<td>Zero</td>
</tr>
<tr>
<td>Big Positive</td>
<td>Zero</td>
<td>Small Positive</td>
</tr>
<tr>
<td>Big Positive</td>
<td>Small Positive</td>
<td>Small Positive</td>
</tr>
<tr>
<td>Big Positive</td>
<td>Big Positive</td>
<td>Big Positive</td>
</tr>
</tbody>
</table>

Table 2 Rule set chosen for the Fuzzy Logic Controller
Figure 8: Rule surface for all three outputs of the Fuzzy Controller
As it can be seen the fuzzy controller designed is basically three separate controllers built for each state. The output of the controller is the amount of thrust needed to control the chaser.

CHAPTER FOUR: TARGET IDENTIFICATION

While a majority of the work in this thesis was done on controller design, some image processing was done in order to identify a target and use the depth information from the Microsoft Xbox Kinect as the initial conditions for the
controllers. The first section covers the targets that are being used to identify, then we cover the identification techniques used to locate the centroid of the image.

4.1 Targets

Before talking about the methods of detections it is important to know what targets that the chaser will be identifying and rendezvousing with. For this thesis there were three different targets where chosen, the FASTSAT-HSV (Fast Affordable Science and Technology Satellite – Huntsville), NAVSTAR-2F (Navigation System using Timing And Ranging) references ???, and an Iridium satellite. These satellites were chosen due to their differences in shapes and orbital properties.
While there are many different satellites in LEO, these three were chosen due to their more general geometric body configurations. The Iridium is part of a constellation of satellites commonly used for communication. NAVSTAR-2F is a GPS satellite that is still in use today and shares many features with other GPS satellites. FASTSAT-HSV is a less common satellite that is designed to house numerous small experiments to be done in orbit and has the simplest hull shape. Figure 9 shows the analogs of the satellites used for identification in the laboratory setting.
Since these satellites will be in space, the chaser cannot use normal RGB cameras for identification. Therefore, the depth image is used to locate the target which is formed from infrared (IR) light.

4.2 The Xbox Kinect

The Microsoft Xbox Kinect is a device that was developed by PrimeSense in collaboration with Microsoft. This device comes equipped with an RGB camera and an IR camera in order to obtain color and depth for each pixel in the image[10]. One problem that the Kinect has to deal with is that the IR projector and the detector are off by about 7.5 cm creating a “shadow” in the depth images. This problem is accounted for in the pixel information when the camera is calibrated, but the shadow will stay present in the image. Figure 10 shows a depth image capture by the Kinect with the shadow present from the offset cameras. The output of the IR projector is shown in figure 11 aimed at the floor. The light in the image is in IR with the lights in the lab off.
Figure 10: Noisy shadow created from the offset cameras

Figure 11 Shows the IR projection from the Kinect

While the RGB camera is not capable for spacecraft operations, the IR camera is comparable to most 3D scanners that are currently used in for rendezvous operations. Most spacecraft operations use 3D laser scanners such as LIDAR (Light
Detection and Radar) to determine range information. Two examples of the types of laser scanners that have been tested and flown are the DragonEye flash LIDAR camera and the Laser Dynamic Ranger Imager, as shown in figure 12, (LDRI) that flew on STS-97 in 2007 took test images of the ISS[25, 26].

Figure 12 LDRI range information of the ISS from testing during STS-91 [25]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>≤ 1.5 Km</td>
<td>.8 – 6 m</td>
<td>≤~45.72 m</td>
</tr>
<tr>
<td>Resolution</td>
<td>128x128 pixels</td>
<td>640x480 pixels</td>
<td>640x480 pixels</td>
</tr>
<tr>
<td>Frequency</td>
<td>5 Hz capture</td>
<td>30Hz</td>
<td>7.8 Hz</td>
</tr>
<tr>
<td></td>
<td>can get up to 30Hz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 Comparison of the Kinect and two flight tested LIDAR cameras
From table 3 it can be seen that the Kinect can be used as stand in for a LIDAR camera to obtain a depth map of the scene. While it is not as accurate at long distances the size of the images and the frequency is comparable to the other two flight tested cameras.

4.2.1 OpenCV and ROS

In order to be able to pull and process the depth images from the Kinect a combination of the robot operating system (ROS) and OpenCV are used. ROS operates inside of Ubuntu Linux and provides simplified access method to the Kinect through the Open Natural Interface. This approach creates topics that contain the depth information that can be shared between processes that are run on the computer[27].

Along with using ROS to access the depth information we also use OpenCV to process the images. OpenCV allows us to use a variety of filters and processing techniques for identifying the targets, also it allows access to the depth information for each pixel[28]. OpenCV not only can handle each single image but it allows for real time streaming and processing of the depth images as well.

4.3 Identification

Once we have the gray scaled depth information the next step is to identify what the spacecraft is looking at. In space the background of the depth image is
black simple image processing techniques such as edge detection or image subtraction can be used to find the target or a specific location on the target craft[29]. For the most part, image subtraction is used for the VGS (Video Guidance System) which looks for a set pattern of reflectors that are placed on the target craft[3]. Once the reflectors are found the chaser craft then has distance and orientation data about the target craft. Image difference seems to be the most widely used approach because of the static background of space. Other research has been done by the European Space Agency with 3D model tracking, which creates a model of what the satellite should look like and is able to obtain position and velocity in nominal conditions[30]. It can be seen that most of the approaches taken for autonomous rendezvous is done with knowing what the target craft looks like before the approach. This is done mostly for the safety of these expensive satellites so there are no mistakes in identification.

In this thesis, the detailed shape of the target isn’t known by the main computer but the target must be in the center of the camera for optimal processing. This is due to the noise in the background. In these tests the noisy background could not be subtracted perfectly nor could a uniform background be placed in the scene the satellite has to be in the center and the length of the satellite must be known. The next step is to create a rectangle around the target and use a grab cut algorithm to eliminate the background pixels. Once the target is the only object left in the image,
the next step is to find the centroid of the image which is what is used as the origin of the coordinate system.

4.3.1 Image Difference

Since image difference techniques are primarily used for targeting object in orbits we did try this approach at first. Figure 13 shows the process of subtracting two images. The idea of this procedure is to take two images, one image of the background and one image with the target in front of the camera. The two images are subtracted from each other which should show the target only.

![Figure 13 Kinect taking two pictures to use for the difference.](image)

This procedure will have some difficulties with the noisy background, but should still be able to pull the target out with the difference. Once the image had been taken out we then use the centroid algorithm described in the next section to find the center of the target.
4.3.2 Grab Cut and Centroid Algorithm

Once the image is first captured from the Kinect the first step is to use the grab cut algorithm to help eliminate the background noise generated from the uniformed background in the lab. The main idea is to select an area where the foreground image is, any pixel outside of this area is assumed to be background, it then searches for anything similar inside the selected area in order to have only the background remaining. This method was chosen to best simulate a uniform background where the difference of images would be a much better choice to use. This grab cut is a function provided in the OpenCV library[31].

(a)                                                                 (b)

(c)                                                                 (d)
Figure 14 The process used to find the target and eliminate the background

Figure 14 shows the process of the grab cut algorithm. A.) show the scene before the Kinect acquires an image. B.) shows the Kinect illuminating the scene with the IR beam. C.) shows the Kinect acquiring the depth map from the IR beam. D.) shows a rectangle being drawn around the target for the grab cut process. E.) is the final result where all the background has been cut of the image.

After that function is applied the next step is to find the centroid of what is left in the image. In order to find what pixel is the centroid we chose to take an average of the pixels.

The process of the centroid algorithm is as follows:

1. Look at the first pixel in the image and determine if the value is less than a threshold
2. If not move on to next pixel
3. If it is less than the threshold then:
   a. Increase total pixel counter and increase x and y counter.

4. After the entire image has been scanned the average is taken in the x and y directions to calculate and place the centroid on the image.

After this procedure has been completed the centroid pixel can then be accessed to have the target position information. Figure 15 shows the final output of the algorithm with the centroid

Figure 15 The centroid is label as the red X

CHAPTER FIVE: SIMULATION SET-UP AND RESULTS

5.1 Mission Objectives and Assumptions
Before the simulations can be started, several assumptions must be made to interpret the results. While most of the assumptions have been stated earlier in the thesis, they will be reiterated here since they are also important to the simulation of the controllers.

1) All orbital maneuvers have been completed upon the start of the simulations and the chaser is at the start of Proximity Operations A (~1000m apart).

2) The target and the chaser are in a circular orbit in LEO therefore the true anomaly has a constant velocity.

3) There are no outside disturbances on the chaser or the target

4) The Chaser cannot run out of fuel

5) Force applied is in the form of $F = ma$

6) There is no roll applied to the chaser

These assumptions are chosen to help simplify a very complex system in order to accomplish the task of controlling the movements of the chaser craft.

The main mission objective is to have the chaser craft maneuver to the target craft starting from a point no less than 100m away. Since the center of mass of the target is the origin of the coordinate system the position states must converge to zero, as well as the velocities. In the simulations the target point is fixed, but in the real
world system the chaser would end up following the target after it successfully rendezvoused.

The chaser craft that will be modeled in these simulations is designed to a small satellite much like a Microsatellite or Nanosatellite which are flown for inspection or for various other tasks on orbit. These are a type of small satellites have a wet mass of up to 10kg for Nanosatellites, and 100kg for Microsatellites. These types are designed for station keeping operations, constellations for communications, or inspections of larger satellites.

<table>
<thead>
<tr>
<th>Parameter (units)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dry Mass (kg)</td>
<td>30kg</td>
</tr>
<tr>
<td>Wet Mass (kg)</td>
<td>50kg</td>
</tr>
</tbody>
</table>

Table 4 mass of the Chaser Craft

After the mass of the craft is chosen, we choose to set the shape of the chaser up as a cube. Figure 16 shows the chaser shape and thrust output. The control thrusts of the chaser are represented as the red arrows. This allows for a more ideal application of thrusts for the simulation.
5.2 Simulation Setup

5.2.1 Controller simulations

The simulations for each of the controllers are done in the MATLAB and Simulink environment. This environment is chosen because of the ode functions that are used in solving the system. The membership functions and rules were created with the aide of the Fuzzy Logic toolbox GUI provided by MATLAB.

Before the simulations can be ran we have to set up the constants that will be used in each trial. These are constants that are important to orbital maneuvers and are listed in table 5.
<table>
<thead>
<tr>
<th>Parameter (units)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gravitational Parameter ( \frac{km^3}{s^2} )</td>
<td>3.986004418e14</td>
</tr>
<tr>
<td>Distance from the center of the earth (Km)</td>
<td>6371</td>
</tr>
<tr>
<td>LEO altitude (above the Earth’s surface) (Km)</td>
<td>500</td>
</tr>
<tr>
<td>Orbital period (seconds)</td>
<td>5668</td>
</tr>
</tbody>
</table>

**Table 5 Constants for simulations**

The simulations are run with different parameters for the different approaches such as starting far away, close, high initial velocities.

<table>
<thead>
<tr>
<th>Trial #</th>
<th># number</th>
<th>X initial (m)</th>
<th>Y initial (m)</th>
<th>Z initial (m)</th>
<th>X velocity initial (m/s)</th>
<th>Y velocity initial (m/s)</th>
<th>Z velocity initial (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>-500</td>
<td>-100</td>
<td>-50</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>-500</td>
<td>100</td>
<td>50</td>
<td>50</td>
<td>100</td>
<td>5</td>
</tr>
</tbody>
</table>
Table 6: initial conditions for the controller

Table 6 shows the tests values that were tested in each controller; the only difference between the Fuzzy logic controller and the other two controllers is that the fuzzy logic controller requires more iterations to complete a simulation.

5.2.1 Kinect Setup

The Kinect is setup in the lab to look out at the target on an even plane. Figure 5.2 shows the depth image of the Kinect with no target in the frame, only the background. This is done so the target is in the center of the depth image that is taken. The distance from each target to the Kinect is about 2 meters.

Figure 17 Setup of the Kinect looking at the targets

All the algorithms for image processing are written in C++ using the OpenCV libraries and handled in ROS.
5.3 Chaser Simulations

Using the values described in the previous section the chaser is tested with each controller. The goal of the simulation was to approach the target position which is the origin, and rendezvous from very different initial conditions.

5.3.1 Backstepping controller

Before simulations were ran, we first used MATLAB’s differential equations solver ode45 function to ensure that the controller designed indeed stabilized the system. The first run used no control input, and used the initial conditions described in Table 5.3 for trial #1 at t = 0, .., 1000. This time was chosen to fully see the output of the uncontrolled system.
As Figure 18 shows the system with no control is indeed unstable. This simulation was a baseline test to be able to see the difference between the uncontrolled and the controller chaser system. Next using the same initial conditions as before and at time $t = 0, \ldots, 10$ we again used ode45 to evaluate the system with the backstepping controller this time.
From the output of the ode45 function showed that the controller did stabilize the system at its equilibrium point. The velocity and position states are equal to each other because the system and controller are linear. After proving that the controller was indeed working we were then able to change the properties of the simulation to test the different initial conditions.

The first simulation ran using the backstepping controller that had the same initial conditions that were used in the ode45 simulation. These conditions mean that at the start of the simulation the chaser is behind and below the target position. We then plot their position, velocity and thrust vs. time and plot the trajectory in 3D.
Figure 20: The position of the chaser
Figure 21: The velocities of the chaser
Figure 22 Forces of the chaser

Figure 23: 3D Plots of the Chaser’s path
The plots above are all for trial #1 and show that the controller worked just as expected from the output of the ode45 function. Next trial #2 is run using an initial position that is further away from the target, but still has no initial velocity. Just as in the simulation before this means that the chaser is behind and below the target at the start of the rendezvous phase.

Figure 24: The position of the chaser
Figure 25: The velocities of the Chaser
Figure 26: The forces of the chaser
From the plots of the two simulations it can be seen that the controller worked as expected in controlling the movements of the chaser. The problem with assuming that the controller works well for all cases, is that here the chaser has no initial velocity. This is a more ideal scenario where the chaser is sitting in the orbit of the target “waiting” for the target to pass. In most situations in orbits it’s hard to achieve this waiting scenario because of all the orbital maneuvers used to achieve this rendezvous phase. The third simulation that is ran uses the initial conditions of trial #3 which contains small positive initial velocities. In the real system these conditions mean that the chaser starts off behind the target and is slowly moving towards the target.
Figure 28: The position of the chaser
Figure 29: The velocities of the chaser
Figure 30: The forces of the chaser during the simulation
Even with negative initial velocities the controller continued to rendezvous with the target point. For trial #4 we then use the same conditions as the previous trial but use negative velocities. This means that at the start of the rendezvous the chaser is below and behind the target, as in the previous trial, but also moving away initially.

Figure 31: 3D Plots of the Chaser’s path
Figure 32: The position of the chaser
Figure 33: The velocities of the chaser
Figure 34: The forces of the chaser
With negative initial velocities the chaser is initially moving away from the target point so it must slow down to turn around. Even as the chaser is moving away the controller still works as expected. Trial #5 is the last simulation that has large positive initial velocities and a mix of positive and negative initial positions. This represents that the chaser is above and in front the target at the start of the rendezvous phase.

Figure 36: The position of the chaser
Figure 37: The velocities of the chaser
Figure 38: The forces of the chaser
Figure 39: 3D Plots of the Chaser’s path

Again the controller works just as expected showing that the rendezvous phase encountered no problems. It does however have a very different approach to the target. This is probably due to the initial conditions for the velocities in the simulation. Throughout each of these simulations the chaser was able to rendezvous with the target point each time using the backstepping controller. This controller works ideally from great distances and shorter approaches during the rendezvous phase of the mission. On a full chaser system with sensors this controller would be able to approach to the designated point with ease.
5.1.2 Fuzzy Logic Controller

The next couple of simulations were run on the fuzzy logic controller. The controller uses the same initial conditions as the backstepping controller. The two differences between the controllers are the environment that they are run in and the run time for the fuzzy system is 2000 steps compared to the 500 that the backstepping uses. For the first simulation we ran the same conditions described in table 6. We create the same plots as before the 3D position, the position and velocities states and the forces.
Figure 40: 3D Plots of the Chaser’s path
Figure 41: The position plots for the Chaser

Figure 42: The velocities plot for the Chaser
When compared to the results of the backstepping controller it can be seen that this controller reacts very differently, as it does not initially converge to zero. Instead this controller orbits very closely around zero. This is due to the membership function for zero, as it could have no overlap of the small positive and small negative membership functions. In the real world this would not be cause for concern as the chaser would have collision avoidance procedures in the control algorithm. Next we run the simulation using the initial conditions of trial #2.

Figure 43: The Force plots for the Chaser
Figure 44 3D Plots of the Chaser’s path
Figure 45: The position plots for the Chaser
Figure 46: The velocities plot for the Chaser
Figure 47: The Force plots for the Chaser

Again we notice the same behavior as before except this time the chaser orbited at a greater distance. After testing the controller with no initial velocity we then test the controller with different initial velocities. The next simulation used the conditions of trial #3.
Figure 48 3D Plots of the Chaser’s path
Figure 49: The position plots for the Chaser
Figure 50: The velocity plots for the Chaser
Figure 51: The force plots for the Chaser

We again see a tighter orbit around the target point much like the response in the first simulation. Next we run the initial conditions for trial #4 to simulate negative initial conditions for the velocity of the chaser.
Figure 52 3D Plots of the Chaser’s path
Figure 53: The position plots for the Chaser

Figure 54: The velocity plots for the Chaser
Figure 55: The force plots for the Chaser

Again we see the tight orbit as before. We noticed that when the x and y initial positions were the same this behavior would surface. Next we try much larger initial velocity for trial #5.
Figure 56 3D Plots of the Chaser’s path
Figure 57: The position plots for the Chaser
Figure 58: The velocity plots for the Chaser
This time we see an orbital that is not as tight as the previous simulations. This behavior is very much like that of the second simulation. This is caused when the initial conditions of the x and y positions are different, which is caused from the model of the chaser. While z is independent of the x and y velocities, both x and y depend on each other. When the positions are different they have different velocities as the FLC tries to control the movements. As it can be seen from the results of the simulations the FLC would be ideal for rendezvous operations from a great distance or to set up a safety orbit to inspect a target. More work would need to be done so the controller could converge to the target position.
5.2 *Kinect Images*

The first step to processing the depth images of the Kinect is to first take and save the images of the background and of all three targets. All the images were acquired through the use of Ubuntu Linux, ROS, and OpenCV.

Figure 60: the depth background of the simulation area
Once these depth images were taken we used OpenCV to process the images. The first technique was image difference. Using the depth images of the background (Figure 60) and of each target (Figure 61) we produce three separate images of the image differences.
While it the targets seem to dominate the images there is still background noise that appears. All the noise in the background can create problems in targeting that could be avoided with a more uniform background. To avoid this issue we decided to use the grabcut algorithm provided in the OpenCV library. Using the depth images the first step was to draw a rectangle around the target for the grab cut algorithm.
The rectangle width and length is set in program. Since there are three different targets to identify the easiest draw the rectangle was to set it manually. The grabcut process then marks all the pixels outside of the rectangle as background and sets their color to white (a value of 255).
It then compares the known background pixels to those inside the rectangle. This is done to fully remove all the background pixels from the image. The grabcut algorithm can be run through multiple iterations to clean up the image. Once the background has been all removed then the algorithm would find the centroid of the remaining image, which should be the target. This process is done by moving though the entire image and counting only the pixels that are not white. After it has averaged the number of non-white pixels it then circles the centroid pixel and labels it with green text.

The first trial used only one pass of the grabcut process to identify the target.
Figure 65: Location of the centroid for the FASTSAT-HSV

Figure 66: Location of the centroid for the Iridium
The resulting images show that just one pass is enough to clean up the background for the FASTSAT but not for the other two. Next we tried two passes for each image.
Figure 68: Location of the centroid for the FASTSAT-HSV

Figure 69: Location of the centroid for the Iridium
Figure 70: Location of the centroid for the NAVSTAR-2F

After two passes the figures show that most of the noise has been eliminated from the images and that the centroid has moved closer to the center of the target. For the FASTSAT and the Iridium the grabcut has eliminated all the noise from the image but the NAVSTAR still has noise in the image. Lastly we run the grabcut process for 5 passes to see if it successfully removes all the background noise from the image.
Figure 71: Location of the centroid for the FASTSAT-HSV

Figure 72: Location of the centroid for the Iridium
After the algorithm was run for five iterations it can be seen that the background has been all successfully removed from the images. Once the centroid pixel had been found we next needed to access the information stored in the pixel. While normally the depth data is stored in the pixel during the grabcut process OpenCV changes the pixel information to a value between 0 and 255. Inorder to get a distance reading we took the values from each centroid pixel and used a mapping function that ranged from the minimum and maximum ranges of the Kinect. From here we then used the distance away from the target as the x distance to the target and assume that the y and z distance are zero for these images.
<table>
<thead>
<tr>
<th>Target name</th>
<th>Distance reading (meters)</th>
<th>Orbital altitude (km)</th>
<th>Orbital period (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTSAT-HSV</td>
<td>1.58</td>
<td>500</td>
<td>5668</td>
</tr>
<tr>
<td>NAVSTAR-2F</td>
<td>1.54</td>
<td>20,000</td>
<td>43200</td>
</tr>
<tr>
<td>Iridium</td>
<td>1.53</td>
<td>781</td>
<td>6000</td>
</tr>
</tbody>
</table>

Table 7: Distance reading of the centroid for the target

The depth data as well as the orbital information was then put into the both controllers as the initial conditions to see how the controllers were able to react.
Figure 74: Position graphs for the chaser approaching the FASTSAT-HSV

Figure 75: Velocity graphs for the chaser approaching the FASTSAT-HSV
Figure 76: Position graphs for the chaser approaching the NAVSTAR-2F

Figure 77: Velocity graphs for the chaser approaching the NAVSTAR-2F
Figure 78: Position graphs for the chaser approaching the Iridium

Figure 79: Velocity graphs for the chaser approaching the Iridium
The backstepping graphs show that this controller works optimally for a short approach. Not only does this shows that controller works ideally for these shorter approaches but also shows that the controller can work for different orbits. This is a very important fact since this makes the controller work for general use and not just for one specific mission.

Figure 80: Position graphs for the chaser approaching the FASTSAT-HSV
Figure 81: Velocity graphs for the chaser approaching the FASTSAT-HSV

Figure 82: Position graphs for the chaser approaching the NAVSTAR-2F
Figure 83: Velocity graphs for the chaser approaching the NAVSTAR-2F

Figure 84: Position graphs for the chaser approaching the Iridium
Figure 85: Velocity graphs for the chaser approaching the Iridium

The output of the FLC shows that the controller has difficulties on shorter approaches but works well for approaches from further away. Although because of the orbit that the FLC outputs as it moves closer to the target point it allows for a useful mission procedure for inspection of the target debris.
CHAPTER SIX: CONCLUSIONS AND FUTURE WORK

The main goal of this thesis was to design and simulate these controllers during proximity operations B of a rendezvous phase for a spacecraft with given initial conditions and depth data from the Microsoft Xbox Kinect. In order to create controllers for the chaser craft, we had to use assumptions that aid in the design of the controller. After both controllers were designed they were tested individually to see their behavior during a simulated rendezvous phase. Then the controllers were tested using the depth data collected from the Kinect in order to simulate a more realistic approach.
During simulations of each controller it could be seen each controller worked well and got to the target point. There were differences between the two controllers the fuzzy controller would orbit around the target point. Depending on the initial conditions, the orbit would be either tighter or would have more space. This orbit can also be adjusted by making the zero membership function of the velocity wider. What this effectively does is set the parking velocity for the chaser. The skinner the zero membership function for position and velocity are the close the chaser gets to the target position. Therefore, a delta function would be the most ideal to use as the membership function. As MATLAB does not allow for this, we plan to move the fuzzy system to fuzzylite a program that is run on a Linux system. The backstepping controller performed very well in every simulation. Even though the controller had some overshoot it was already very close to the target point which in a real system with a constant update to the position would be stop sooner than the FLC. The next step for this controller is to rewrite the model and controller into C++ to be used in ROS with the Kinect running simultaneously. This would represent the chaser docking to the target exactly. Though the use of the 3D depth data we were able to simulate the target being found by the chaser before the rendezvous phase can began. Using this we were able to identify the target then began a simulated approach.

The most significant future goal is to add the rotation of the chaser craft into the system. This will allow us to simulate true satellite system to get a better idea of
how the chaser craft will react during the approach. Also we wish to model the orbit as an ellipse rather than a circle since most orbits are more ellipses in shape thus resulting in a nonlinear system. The controllers will be modified to better control the movements of the chaser. One planned modification is to design a Takagi-Suegno FLC to determine if it can control the nonlinear system better than the current FLC in use. With that we will be able to simulate a fully functional chaser system that can identify any target and being rendezvous procedures.

REFERENCES


[31] OpenCV. *Miscellaneous Image Transformations.*
Joaquin Daniel Labrado was born in March of 1988 and is originally from Austin Texas. He received his B.S. in Electrical Engineering in May of 2011 from The University of Texas at San Antonio and is currently pursuing a M.S. in Electrical Engineering. He has worked in the UTSA ACE (Autonomous Control Engineering) lab since 2009 and has worked on a variety of projects in robotics. After an internship at the Marshall Space Flight Center he began to focus on autonomous rendezvous projects and had an interest in orbital debris removal. His interests include control systems, mobile robotics, 3D depth imaging, and autonomous rendezvous missions.