Virtual Simulation for Testing Robotic Control Architectures for Autonomous Rendezvous Docking

John Karasinski[1], Christopher Lorenzen[2], and Melanie Stich[3]
[1] *Department of Mechanical & Aerospace Engineering, University of California, Davis, USA, karasinski@ucdavis.edu*
[2] *Department of Mechanical & Aerospace Engineering, University of California, Davis, USA, calorenzen@ucdavis.edu*
[3] *Department of Mechanical & Aerospace Engineering, University of California, Davis, USA, mkstich@ucdavis.edu*

***Abstract*—CURRENTLY 163 WORDS...NEED TO CUT OUT 13 WORDS**
With small failures leading to the loss of an entire spacecraft, innovations in inspection methods are required. The viability of a small inspection satellite with a deliberative or a reactive robotic architecture is discussed and methods for testing the robustness of these systems are developed. Some past docking and machine vision examples in the field are presented as well as their applicable contributions. A navigation system based primarily on machine vision position determination is used to compare the success of two different control system architectures. Each of the two architectures are applied to the same environment, and varying initial states and errors are used to test their ability to adapt to different applications or situations. The fuel burned and time to target are used to quantify the relative performance of the two architectures. The proposed test environment and simulation methods are introduced to serve as a means for acquiring data to develop empirical cause and effect relationships between the different control design parameters.

***Index Terms*—robots, satellite, machine vision, docking, navigation, deliberate, adaptive**

I. Introduction

A. Background

In recent years the small satellite platform has increased in popularity. With the ability to launch multiple small satellites as a secondary payload and the relatively low development cost involved, they are one of the most viable testbeds for new technologies. One great example of this is the class of satellites that adhere to the CubeSat standard [1]. CubeSat initiatives have made many projects possible due to the minimal costs required. Although a small size limits their capabilities, higher fidelity projects usually require a larger volume to contain all desired systems. There are several projects that use slightly larger free flyers that have produced unique navigation demonstrations such as AERCam Sprint, Mini-AERCam, and SPHERES [2–4]. An obvious limitation to small satellites is the ability to fit all required architecture within the size restrictions set by the design. One way to reduce the space required is to choose your subsystems such that some may fill multiple roles. An example of this is to use a science instrument to collect navigation data. More specifically, an optical sensor can be used to take stereoscopic visual recordings for inspection purposes, and to determine the position and attitude in relation to a target. To do this, the image captured must be used in conjunction with a machine vision algorithm in order to estimate navigational parameters.

The decision to use stereoscopic visual sensors to inspect was chosen to increase ease of inspection for a human operator. Using stereo vision and the full visual light spectrum, a more familiar virtual image is produced. Operators are able to view the subject as if it was in front of them, allowing for quick and informed decision making. This, as among other reasons, has been realized by many research teams and can be seen in many projects such as SPHERES VERTIGO and the aforementioned AERCam projects [2], [3], [5].

The ability to dock to the target craft could greatly increase the effective mission operation time, allowing the inspector to refuel, recharge, and download data. Instead of a small disposable craft that has a single use, it could become a long term asset and an integral part of the target craft's diagnostic routines. Most current docking procedures utilize a laser range finder for determining distance. Laser range finders have flight heritage and have moderate accuracy [6]. Simple fiducial markers can be used to increase the performance of rangefinders, and multiple markers can allow rangefinders to triangulate a relative position. Markers for visual navigation systems, however, can vary in complexity. Marker distribution and marker shape can allow a machine vision system to gather a large amount of data with a single image, whereas a laser rangefinder must take multiple measurements. An example of visual cues for docking can be seen in Figure **??**. The alignment markers clearly display position error. This system was originally developed for manual docking operations, but it also provides visual information to allow more accurate machine vision position measurements.

B. Motivation

The advantages presented by the addition of an inspector satellite to a larger craft show the necessity for the development of this technology. Whether the target craft is inhabited by astronauts or not, it can still increase the probability of completing a mission successfully.

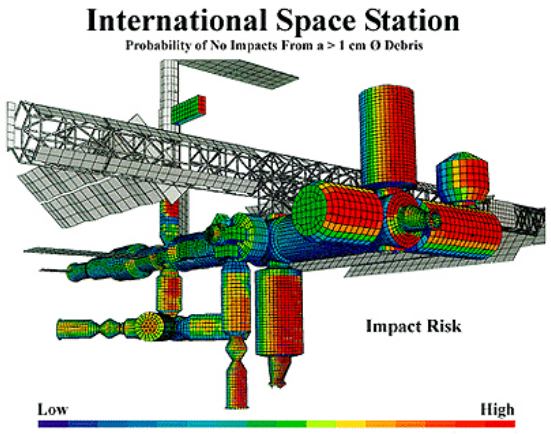

Fig. 1. Debris impact risk by region of the International Space Station

There are two main types of damage that can lead to concern. The first cause for alarm is physical damage due to collision with orbital debris or micrometeorites. The second is component failure. Either of these events can be disastrous to a mission. The best case scenario is that the craft is not inhabited and the only loss is a multimillion dollar piece of equipment. With inspection capabilities, damage and failures can be identified in a timely manner. The possibility of salvaging the mission increases greatly with more information about the problem area and additional time. Additional opportunities would arise if inspection capabilities were to be integrated with other projects like DARPA's Phoenix Satellites [7]. The Phoenix project's goal is to have a cluster of modular "satlets" rendezvous and integrate with an inoperative target craft. An inspector could be sent beforehand, or with the satlets, to increase the success rate of the Phoenix's mission. If damage or component failure could be accurately identified, then the Phoenix system could dock and efficiently repair or repurpose a damaged satellite.

Using the International Space Station as an example, NASA has shown concern for both damage types that could be observed with a visual inspector. Numerous studies have been conducted, attempting to quantify the danger involved with debris collision [8]. As seen in Figure 1, the impact risk for different parts of the station have already been determined and an inspection satellite could be made to center its efforts in patrolling high risk areas. There have also been close calls due to component failures. One such incident requiring emergency extra-vehicular activity was when an ammonia pump developed a leak in May 2013 [9].

Beyond the overall need for an inspection resource, there is also a driving factor to attain efficiency. With increased navigational efficiency, it extends the life of the inspector as well as increasing the amount of data that can be gathered with each excursion. As the inspection capability of the free flyer increases, the overall feasibility of the inspection mission also increases.

C. Objective

The purpose of this exercise is to create and test two navigation control schemes that are built on a vision based pose determination system. The first scheme will be a deliberative control architecture that will utilize a dynamic model to calculate corrections, and the second will be a reactive control architecture that will incorporate several weighted parallel sub-architectures. Several parameters will be modified, and the two systems' results will be compared. Once each control scheme has been evaluated, the method with the best performance could be implemented in a low cost inspection satellite.

The two control architectures should be able to perform the same task while using significantly different decision making procedures. Each will have a means to approach the target, maintain its orbit relative to the target, hold its final position, and, if necessary, avoid a collision with the target craft. Since both methods will be capable of achieving a successful autonomous rendezvous docking, several metrics can be compared to rate the quality of the two controllers.

Under normal conditions, the only drift from the intended rendezvous trajectory is contributed to error or disturbances, but in this situation there is also drift due to the inspector and the target occupying different orbits. Even with a perfect initial trajectory and planning, corrections are required. The navigation scheme must be as efficient as possible as course corrections are unavoidable. To measure each navigation scheme's efficiency, the most relevant metrics to investigate are the fuel used and time to target. By examining the fuel use and travel time, direct performance comparisons can be made. Multiple simulations will be run to test how each simulation reacts to varying initial states and errors. This will allow the robustness of the two control schemes to also be quantified and compared.

The primary goal of the study will be to answer the following question:

> "What is the relative performance between deliberative and reactive control architectures in an autonomous rendezvous docking simulation? Additionally, which architecture executes with the highest capture rate while maintaining the lowest fuel use and time to capture?"

For many space applications, deliberative controllers are used because the dynamics involved are very predictable. Deliberative controllers based on system dynamics are sometimes obvious solutions, however, they may not be the best solution. The authors would like to determine if a simple reactive control scheme, with no knowledge of the system dynamics, can rival a deliberative controller that is programmed to make calculated reactions within the test environment.

II. Literature Review

A. Overview of Small Inspection Satellites

Over the past 20 years, there have been incredible advancements within the realm of semi-autonomous satellites. Beginning in 1997, the Autonomous Extravehicular Activity

Robotic Camera Sprint (AERCam Sprint) was the first semi-autonomous satellite to demonstrate the use of a free-flying prototype camera aboard the International Space Station (ISS). While operating alongside STS-87 Mission Specialist Winston Scott, the AERCam Sprint flew under the remote-control guidance of Steve Lindsey for approximately 75 minutes, and relayed live television images to Columbia's Mission Control [2], [3]. After successfully completing this experiment, researchers and analysts decided to incorporate a higher level of autonomy, and produced a second prototype known as the Mini AERCam in 2000. While this satellite never made it to space, the Mini AERCam underwent multiple tests on an air-bearing table and in an orbital test simulation facility at Johnson Space Center. This newly designed satellite was given automatic position hold, point-to-point maneuvering, and an additional camera to provide an orthogonal view, allowing astronauts to navigate the Mini AERCam with respect to the ISS. Through these multiple additions, researchers expanded the satellite's capability to encompass supervised autonomous and/or remotely piloted operations [3], [10].

In 2006, the first Synchronized Position Hold Engage Reorient Experiment Satellites (SPHERES), a self-contained nanosatellite made by MIT's Space Systems Laboratory, was launched to the ISS and taken to the US Laboratory. Since that time, this semi-autonomous satellite has been joined by two additional SPHERES, making this system the first consistent experimental nanosatellite testbed aboard the ISS. Unlike the AERCam Sprint and the Mini AERCam, SPHERES is a modular satellite where each system is self-contained in individual capsules. This configuration allows SPHERES to easily incorporate system expansions onto specific platforms, such as navigation, without needing to reconfigure the entire craft. Furthermore, its modularity helps researchers efficiently address system failures, making it easier for astronauts to perform on-site repairs. To navigate SPHERES within the ISS, the system utilizes wall-mounted ultrasonic beacons and corresponding ultrasonic receivers attached to the nanosatellite [4]. SPHERES emits an infrared flash to determine its location. Once emitted, the satellite waits for the wall-mounted beacons to emit corresponding ultrasonic pulses. After receiving these ultrasonic pulses, the satellite measures its range based on the pulse's time of flight, and can then calculate its relative position, attitude, and angular velocity [4], [5]. This unique navigation system allows SPHERES to emulate a "pseudo-GPS" time-of-flight sensing system, and ultimately estimate its position, angular velocity, and attitude without the potential for signal interference and noise – a challenge that has been previously encountered with GPS systems [5]. Through this autonomous navigation and modular design, the SPHERES testbed has become a versatile platform for developing vision-based navigation, anti-collision, and formation flying algorithms. By allowing research teams to create algorithms that can then be uplinked to the SPHERES test system aboard the ISS, researchers can receive live feedback, and ultimately find the exact areas within their algorithms that need improvement.

B. SPHERES VERTIGO

In 2008 the MIT Space Systems Laboratory began building an upgrade to the SPHERES system, known as the Low Impact Inspection Vehicle (LIIVe), as part of the Visual Estimation and Relative Tracking for Inspection of Generic Objects (VERTIGO) program. Once completed, this upgrade would later be attached to the existing SPHERES system and act as VERTIGO "goggles," allowing SPHERES to perform vision-based navigation experiments in the microgravity environment aboard the ISS. After adjusting these VERTIGO Goggles to suit the space station's environment, the final system was upgraded to include two monochrome stereo cameras, two illuminating LEDs, a 1.2 GHz Via Nano processor, an 802.11n network card, and optics that included a larger aperture lens in a synchronized stereo configuration [4], [5], [11], [12].

When the modified SPHERES VERTIGO was ready for experimentation, numerous flight algorithms were tested to demonstrate the spacecraft's complete autonomy. After ISS Expedition 34, it was confirmed that SPHERES VERTIGO was capable of autonomously conducting a circular orbit about an uncooperative object, while simultaneously maintaining a constant relative position between SPHERES VERTIGO and the target. This objective was achieved through the primary use of inertial sensors and cameras, and was considered an unprecedented success [11], [12].

While the SPHERES VERTIGO made considerable progress over the past few years, this advanced system continues to exhibit an ongoing navigational limitation through the use of ultrasonic beacons and receptors. The navigational hardware requires a total of five wall-mounted ultrasound beacons, confining the SPHERES VERTIGO system to one room within the ISS. This ultimately limits its overall use, and essentially forces SPHERES VERTIGO to remain as an experimental testbed [4], [5], [11], [12]. Today, researchers are working to expand the system's navigational range by incorporating Google's Project Tango – a mobile device that can track 3D motion allowing autonomous navigation within a building. Researchers hope that once this hybrid Project Tango and SPHERES system, also known as "Smart SPHERES," has been successfully implemented, the SPHERES nanosatellite will ultimately be able to traverse the entire ISS, performing interior maintenance and inspections [13].

C. Collision Avoidance and Docking Algorithms

Since the SPHERES nanosatellite's first microgravity guidance, navigation, and control experiment, there have been three classes of algorithms pertaining to collision avoidance and docking that have emerged: metrology, control, and autonomy [14].

1) Metrology Algorithms: The metrology algorithms were implemented using a SPHERES-specific interface, and utilized a series of Extended Kalman Filters to obtain the system's state vector from the sensor outputs. This approach has been typically utilized in position, attitude, and determination systems. Despite many successful implementations, recent literature suggests that the second and third classes have

demonstrated greater accuracy in the areas pertaining to collision avoidance and docking [14].

2) Control Algorithms: The control algorithm class involves both closed-loop controls and path-planning algorithms. One prominent control algorithm that has been frequently tested is the glideslope algorithm. This algorithm is a hybrid between a path-planning and a velocity-control algorithm, where the incoming spacecraft is given commands to slow its velocity as it approaches its target [14–17]. The glideslope algorithm was the first autonomous docking algorithm to successfully attach an incoming spacecraft to its tumbling target, and its simple, yet robust, controller makes this algorithm easy to store aboard SPHERES [15]. Another promising algorithm is the "safe" trajectory algorithm. This innovative algorithm computes a pre-planned trajectory using the solution from a Mixed-Integer Linear Program, and, using this pre-computed trajectory, is able to optimize fuel and avoid incoming obstacles [15]. However, while this algorithm is guaranteed to produce a safe trajectory, its overall complexity requires it to be computed on an external computer. Once the computations have been completed, the final trajectory is transferred to SPHERES. This entire computational process creates an approximate nine second delay, and can potentially create a catastrophic outcome if the spacecraft requires an immediate trajectory path to avoid an incoming collision. To remedy this solution, researchers have begun to trade trajectory and fuel optimality for computational time, and can reduce the total computation time to about 0.17 seconds [15]. Lastly, the "close point of approach" algorithm has been demonstrated to be both compact and computationally efficient, and has served as a background safety routine for the high school SPHERES Zero Robotics program [17]. While all three aforementioned algorithms have accurately performed numerous tests pertaining to collision avoidance and docking, each algorithm is associated with its own specific set of pros and cons. As it currently stands, researchers have yet to find a way to optimize fuel usage, pre-planned trajectories, and computational power, and thus must decide which factors are most important for any given mission [14–17].

3) Autonomous Algorithms: Finally, the autonomous algorithm class is used to execute the control class algorithms and determine the current mode of operation [14]. As a result, the glideslope, "safe" trajectory, and "close point of approach" algorithms all utilize autonomy to properly perform their respective procedures.

D. Computer Vision Based Navigation

While there are numerous computer vision based navigation algorithms, the MIT Space Systems Laboratory (SSL) developed a unique algorithm using fiducial markers to validate the performance of the SPHERES VERTIGO Goggles. This algorithm utilized methods that were originally seen in visual navigation algorithms for unknown environments, and through this study, an upper bound for its precision and accuracy was established.

When implementing this computer vision algorithm, researchers had to determine two features regarding the system's target:

1) The number of fiducial markers needed to solve the relative pose estimation problem with minimal ambiguity
2) The design of each fiducial marker

According to previous studies, the researchers determined that four coplanar fiducial markers were the minimum number of points needed to obtain a unique solution to the exterior orientation problem [18]. Since image processing algorithms are considered to be computationally expensive, the researchers were also looking to minimize the complexity of their fiducial markers, to then increase performance for this algorithm. Thus to maintain simplicity, and to also comply with the system's size constraints, the researchers decided to use four fiducial markers to maximize their individual size, as seen in Figure **??** [18], [19]. In designing the fiducial markers, researchers were looking to maintain the target's simplicity, while simultaneously attaining detection in space's lighting environment. After conducting individual literature reviews, the MIT SSL research team determined that concentric contrasting circles would achieve optimal results. These concentric contrasting circles proved advantageous because the centroids of the contrasting circles remained constant under both rotation and translation. Additionally, the relative ratio between the concentric circles also remained constant under these same operations. Through these two features, the image processing algorithm was able to see this target as a large point, which led to easier and more accurate detection [18], [19].

Next, the MIT SSL research team implemented a seven-step computer vision algorithm to detect their corresponding target from a processed image. First, the detection algorithm utilized an adaptive threshold algorithm to compensate for the variation in lighting, dividing the image into segmented black/white components. Next, the algorithm grouped and labeled similar segments based on location and pixel color. Once these segments were labeled, the algorithm searched each segment for collocated centroids, and ignored segments that did not comply with this condition. The algorithm then filtered through the remaining collocated centroid regions, and removed pairs that did not have an area and color ratio of larger black regions to smaller white regions. Next, the algorithm determined whether there were exactly four contrasting collocated circles in the remaining segments. If yes, the algorithm concluded that a target was properly found, and if no, the algorithm determined no target was found and exited. As the final step, the algorithm conducted an area ratio sorting and correspondence to extract any false positives, or rather outliers, from the final solution [18], [19].

This complicated image processing algorithm was just the first step in the overall computer vision navigation algorithm that was tested on the SPHERES VERTIGO platform. The navigational algorithm then proceeded to estimate the relative pose of the system, and then smoothed out the results using a Multiplicative Extended Kalman Filter. As this study concluded, this navigational approach, while originally

designed for small spacecrafts, could theoretically be extended to apply to larger spacecrafts and continue to attain accurate results. Due to the larger vehicle size, however, the upper bounds regarding precision and accuracy would differ from those attained in this original experiment [18], [19].

While this image processing algorithm is similar to the algorithm the authors will ultimately try to simulate, this detection algorithm's innate complexity will only serve as a future model. For this project, the authors will focus on creating a simpler image processing algorithm that will extract circles, as opposed to contrasting collocated centroids, from a processed image.

E. Virtual Simulations

When testing a computer-vision based navigational algorithm, the first step is to simulate the algorithm in a comparable environment to test the system's design. For example, the research team at the West Virginia Robotic Technology Center (WVRTC) facility created a virtual environment to attempt to test their vision based pose estimation system, consisting of a monocular camera mounted to the tip of a robotic manipulator [17]. In doing so, the designers were able to exhibit how their system parameters effected their system's performance, and make dynamic modifications to their computer vision algorithms. Similarly, when the NASA Jet Propulsion Laboratory scientists simulated NASA's Mars Sample Return Mission, particularly the high-risk operation of capturing the Orbiting Sample, this research team performed numerous virtual simulations to test as many situations as possible, before implementing the final algorithm [20]. As a result, advanced virtual simulations have played an integral part in emulating the system's environment, using tools such as EDGE and OpenCV, to aid in the detection of algorithmic deficiencies. The authors recognize that this is an optimal way for testing a vision-based autonomous navigation algorithm, and will thus utilize a virtual simulation to test and demonstrate the navigational and docking algorithm's capability.

F. Literature Review Conclusions

Since the AERCam Sprint, there have been numerous advancements pertaining to small semi-autonomous and autonomous satellites, particularly in the areas regarding guidance, navigation, and control. Through the glideslope, "safe" trajectory, and "close point approach" algorithms, researchers have attained many viable options for docking and collision avoidance, but there still remains room for improvement. With the need for both computational efficiency and vehicle safety, there remains a wide area of computer science that researchers are continuing to explore, to ultimately find a solution to this problem. The most intriguing aspect of these programs is the fact that all satellites, excluding the AERCam Sprint, have been solely implemented in interior environments, when the AERCam Sprint was originally designed to assist astronauts in Extra Vehicular Activities (EVA) [2], [3]. While the MIT Space Systems Laboratory is the closest team to attaining a functioning exterior inspection satellite through their SPHERES-X proposal, there still remains a relatively open field that has yet to be explored [4]. In this paper, the authors hope to take preliminary steps towards creating an exterior inspection satellite by developing two navigation control architectures, and ultimately determine which architecture could later be applied to a completed satellite system.

III. Design Specifications

Assume that there is a preexisting small satellite with machine vision navigation capabilities, and the following features are attached to implement the autonomous machine vision algorithms:

- Two color stereo cameras attached to the side of the satellite.
- A wide angle 2.8mm f/1.3 CCTV lens with manual iris and focus, capable of a 96° horizontal angle and a 71° vertical field of view.
- Two illuminating light-emitting diodes.
- A computer capable of 1.2 GHz x86 Via Nano Processor, 4GB RAM, 1 MB L2 Cache, SSE3.
- An 802.11n network card.

[11], [12], [17], [21]

For this simulation, assume that the physical dynamics of the satellite have been previously implemented in secondary algorithms, and thus have no impact on the implemented algorithms. Furthermore, assume that this small satellite has reaction control wheels for attitude control and a sufficient power supply to properly conduct all virtual test simulations. Although the inspector craft will have sufficient fuel, it will be considered finite to determine accuracy of the different control schemes.

IV. The Environment

A. The Spacecraft Environment

Space is an extremely dangerous environment with many challenges of different natures and magnitudes. Some effects on a spacecraft can arise from radiation, space debris and micrometeorite impact, upper atmospheric drag, spacecraft electrostatic charging, and many other factors. While the resulting dynamics of these phenomena have various degrees of effect on a small spacecraft, these are outside the scope of this work. For the purposes of the study, all of these factors are ignored. It is assumed here that the spacecraft is operating under nominal conditions in an otherwise empty environment.

B. Dynamical Equations of Motion

The Clohessy-Wiltshire equations were used to model the dynamics of the spacecraft while it was in close proximity to the target vehicle. The Clohessy-Wiltshire equations describe a simplified model of orbital relative motion, in which the target is in a circular orbit, and the inspection spacecraft is in an elliptical or circular orbit. This model gives a first-order approximation of the chaser's motion in a target-centered coordinate system. It is used here in planning a rendezvous between the inspector and the target [23].

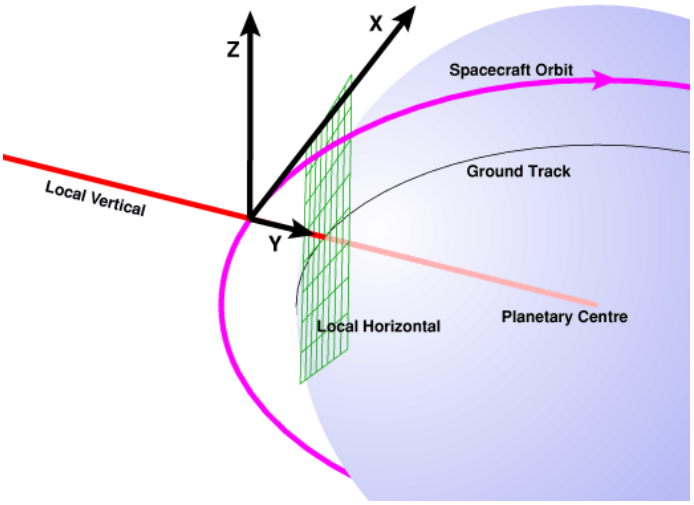

Fig. 2. LVLH (Local vertical/local horizontal) is a frame of reference that is relative to a line drawn from the spacecraft to the centre of the body it is orbiting [22].

These equations of motion can be expressed as

$$\begin{aligned} \ddot{x} &= 3n^2 + 2n\dot{y} \\ \ddot{y} &= -2n\dot{x} \\ \ddot{z} &= -n^2 z \end{aligned} \tag{1}$$

and have the closed form solution given by

$$\begin{aligned} x(t) &= (4 - 3\cos(nt))x_0 + \frac{\sin(nt)}{n}\dot{x}_0 + \frac{2}{n}(1 - \cos(nt))\dot{y}_0 \\ y(t) &= 6(\sin(nt) - nt)x_0 + y_0 \frac{-2}{n}(1 - \cos(nt))\dot{x}_0 \\ &\quad + \frac{4\sin(nt) - 3nt}{n}\dot{y}_0 \\ z(t) &= z_0 \cos(nt) + \frac{\dot{z}_0}{n}\sin(nt) \end{aligned} \tag{2}$$

where

$$n = \sqrt{\frac{\mu}{a_t^3}} \tag{3}$$

and a_t is the semi-major axis of the target vehicle's orbit, and μ is the standard gravitational parameter.

V. Simulation

The virtual simulation was driven with a variety of software packages. The dynamics and control systems were modeled with several Python modules, while the visualization will be rendered in EDGE. EDGE is a graphics display tool developed at NASA's Johnson Space Center that combines key elements from graphics software developed for the space shuttle and the International Space Station programs, and adapts them for integration with other engineering simulations and facilities [24].

EDGE makes use of a node tree to structure data, objects, and models. Each node has many properties, the most notable of which are the node's position, orientation, and parent. Each node's position and orientation are defined relative to its parent's position and attitude. A node's parent can be changed to a different node, at which time the node's position and attitude are automatically updated to the correct values.

Three Python 2.7.9 scripts were developed for use in the virtual simulation:

cv
: A computer vision image processing module which takes advantage of OpenCV-Python version 2.4.9.

dynamics
: A dynamics and controls module.

dcomm
: A wrapper library developed to interface with EDGE's C++ DCOMM library.

A. Python Modules

1) cv: The *cv* module takes advantage of Python bindings for OpenCV. OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision [25]. This module takes the live video feed from EDGE and attempts to identify features visible on the target spacecraft. The features of the target are supplied to the module a priori, and these descriptions were used to estimate the 3D pose of the target vehicle relative to the spacecraft's camera. This pose information is then passed on to the *dynamics* module.

2) dynamics: The *dynamics* module drives the dynamics and control of the simulation. The movement of the spacecraft around the target craft is modeled by the Clohessy-Wiltshire equations, see section IV-B. This module controls how the spacecraft operates in its different modes.

3) DCOMM: The *DCOMM* module was previously developed to network between Python scripts and EDGE. The Python DCOMM interface allows a user to call various C++ functions from EDGE's DCOMM module and communicate with an EDGE server. Users can move and rotate nodes, and can also change a node's parents, units, and principal axis definitions. Commands to set the spacecraft's attitude and position were sent from the *dynamics* module through DCOMM and passed on to EDGE.

B. Sensors

A system's sensors play an integral role when performing autonomous docking operations with a noncooperative target. As a result, a sensor's ability to accurately determine the system's range to target is imperative for mission success. In this simulation, an autonomous satellite conducts a docking operation with *a priori* knowledge of the target, and uses

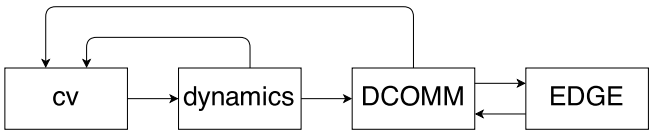

Fig. 3. Block diagram of software package interactions

sensors to calculate a real-time orbital trajectory. Additionally, this simulation utilizes two different sensors, computer vision and laser range finder, for each robotic architecture from start to target capture. These results will be compared and discussed in Section VII, and the superior sensor will be identified for each set of initial conditions.

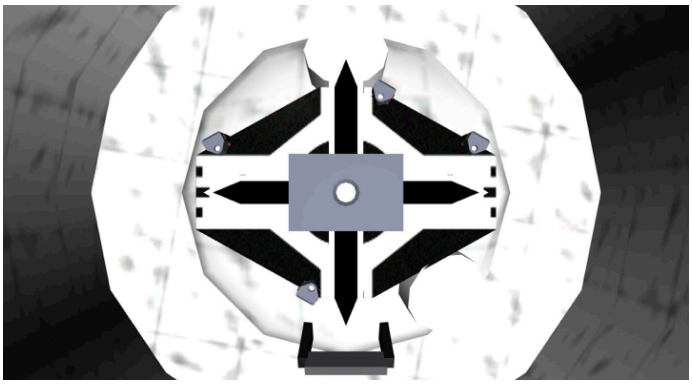

Fig. 4. Single picture taken from satellite

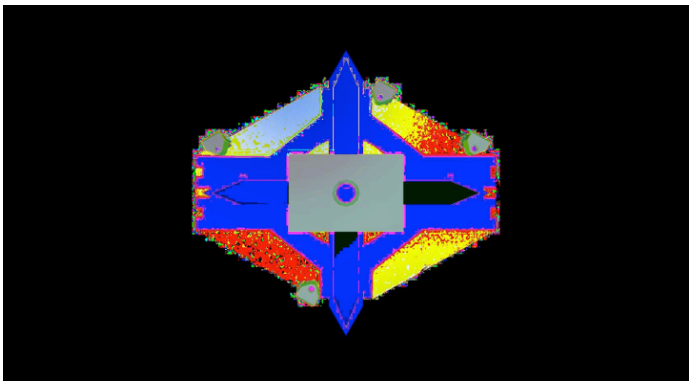

Fig. 5. An identified docking port

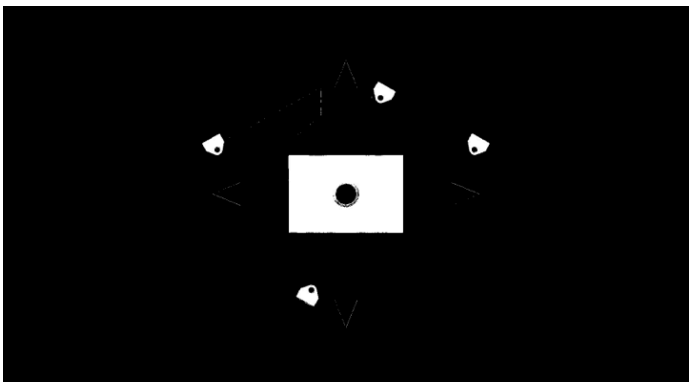

Fig. 6. Mask used to identify features on docking port

1) Computer Vision: The cameras on the spacecraft can serve the dual purpose of visual inspection and relative location estimation. The camera used in the simulation takes images of width 1600 pixels and height 900 pixels, and performs several algorithms to identify markers and estimate the relative location of the target spacecraft (an example image taken by the inspector can be seen in Figure 4).

To identify the markers on the docking port, the inspector takes two pictures of the docking port, one with the inspector's lights on, and the other with its lights off. The difference between these two images is used to identify the docking port (an example of the resultant image can be seen in Figure 5). The highly reflective nature of the port makes it standout compared to the rest of the target. Certain predetermined color ranges are selected from the resultant image to create a mask identifying the 5 known markers (an example of the mask can be seen in Figure 6).

Contours are then extracted from the masked image using the algorithm described by Suzuki et al. [26]. The size of these contours is then sorted from largest to smallest, and the largest is assumed to be the largest of the markers. A clustering algorithm is then used to locate the next four largest markers of roughly the same size. The area of these four markers is averaged, and this average is used as a scaling factor to estimate the relative distance in the x direction. The offset of the center of the largest marker from the center of the image and the estimated distance in the x direction, is used to estimate the relative distances in the y and z directions.

While this method proved effective a majority of the time, it did occasionally lead to failures when the largest marker was misidentified. The error between the estimated state and the true state as a function of distance can be seen in Figure 7. This error was comparable to the error for the laser sensor at large distances, but was quickly reduced to less than a few degrees for shorter distances.

2) Laser Range Finder: The second sensor that estimates the satellite's relative location is the laser range finder. Whenever a sensor tries to estimate the position of its respective system, there is an innate measurement error, or "noise," as a result of environmental disturbances and drift. To accurately simulate how noise effects the system, the simulation trials were first conducted using an exact laser range finder, and then again using a "noisy" laser sensor.

For the noisy laser range finder sensor, the current (x, y, z) position of the system is passed into a function, and noise is propagated throughout the system using a two-step method. First, error is added to the position by randomly generating a number between the positive and negative value of the predetermined laser error. Next, a secondary error is generated between the bounds of one plus/minus the laser error value. This secondary error is multiplied by the modified position, and these new x, y, and z values become the system's estimated position. For this simulation, it is important to note that the predetermined laser error is manually set by the user, and thus varying the laser error will subsequently adjust the estimated position's magnitude. Refer to the code below for an example of a noise propagation function for the x position.

```
x += np.random.uniform(-error, error)
x *= np.random.uniform(1 - error, 1 + error)
```

3) Thrusters:

VI. Robotic Architectures

Two different robotic architectures were utilized in this simulation, deliberative and reactive, and both architectures were tested in unique trials using either the computer vision or the laser range finder sensors. Therefore for a given set of initial conditions, a minimum number of six trials were conducted: Deliberative – Computer Vision, Deliberative – Laser Range Finder (noisy), Deliberative – Laser Range Finder (exact), Reactive – Computer Vision, Reactive – Laser Range Finder (noisy), and Reactive – Laser Range Finder (exact). As previously mentioned, the results comparing both architectures, as well as both sensors, will be discussed in detail in Section VII.

A. Deliberative Architecture

1) Overview: A deliberative robotic architecture utilizes a hierarchical paradigm such that data is gathered to create a world model, and, after each iteration, this world model is updated before planning and calculating an action to take place. This architecture conducts steps in series, as opposed to parallel, and places a large emphasis on planning [27].

To mimic this paradigm, the deliberative architecture algorithm will consist of three states. The architecture will first Sense, and use sensors to determine the current state and to build the world model. The information gathered in Sense is passed on to Plan, which uses this information to determine parameters for an ideal orbital maneuver. The Plan state will then pick one of three different trajectory types: Homing, Closing, Final Approach. Once this trajectory is planned, the trajectory is passed on to the Act state. The Act state performs open-loop thruster inputs. Once the burn is complete, the state switches back to Sense and the cycle continues until the target is reached.

2) Algorithm: The deliberative robotic architecture utilizes the Sense-Plan-Act paradigm, and its algorithm was intuitively organized in a similar manner. Below is a pseudo-code description of the three main functions that are used to build the world model, compute the desired trajectory, and conduct a corresponding burn.

Sense

From the main simulation, an argument is passed into the sense function specifying which sensor, "CV", "laser-noise", or "laser-exact", will be utilized for this trial run. Using this specified argument, the Sense function links the deliberative architecture simulation to its respective sensor, and the sensor updates the satellite's estimated state vector accordingly. Refer to Sections V-B1 and V-B2 for the respective details pertaining to the computer vision and laser range finder sensors.

Plan

Using the Sense estimated position values, Plan calculates the distance to target and burn time, or rather the time duration of a particular burn. If the distance is less than 127 cm, the satellite enters the Final Approach trajectory, with a burn time equal to 0.2s. Else if the distance to target is less than 1270 cm, the satellite enters the Closing trajectory and updates the burn time to equal 1.0s. Else, the satellite enters the Homing trajectory and utilizes a 5.0s burn time. Then after the appropriate trajectory and burn time have been selected, Plan computes the necessary burn needed to reach the target and perform the appropriate orbital maneuver.

Act

Act receives the calculated burn values from Plan, and decides if the specified thrust can be performed. If the desired burn rate is less than the minimum thrust, no burn is conducted. If the desired burn rate is within a specified value, allow the system to burn for a fraction of the desired burn rate. Else, if the desired burn rate is too large, conduct a burn equal to the previous burn rate. This particular series of checks and balances prevents the satellite from performing large impractical burns, limiting the craft's relative velocity and increases the fuel burn efficiency. As Act is conducting this calculated burn, the satellite's position is simultaneously updated using the Clohessy-Wiltshire equations.

This algorithm will continue to perform in a Sense-Plan-Act cycle until the satellite has reached the target.

B. Reactive Architecture

1) Overview: Unlike the deliberative architecture, a reactive architecture eliminates planning, and thus operates without a global world model. Instead, this method utilizes a parallel approach, and uses a coupled Sense-Act relationship to quickly implement independent behaviors [27].

This reactive architecture algorithm makes use of a fuzzy approach where each behavior has a weighted contribution. In this paradigm, the robot takes sensing data and computes the best action to take independently of what the other behaviors decide to do. The robot will then do a combination of the possible behaviors. Five behaviors are used: "Move closer", which uses no path planning, but attempts to get closer in local vertical/local horizon frame (LVLH); "Don't hit", which fires away from the target based off distance and relative velocity of the target; "Station keeping", which holds its position when the target region is reached; "Stay on orbit", which pushes closer to the horizontal axis if the system is too far off the orbital path; and "Stay in Plane", which returns the inspector to the orbital plane of the host. The resulting thrust, T, is calculated by

$$T = \frac{\sum_{i=0}^{n} T_i w_i}{\sum_{i=0}^{n} w_i}, \tag{4}$$

where T_i is the thrust calculated by behavior i, w_i is the weight of the behavior, and n is the total number of behaviors.

2) Algorithm: The reactive architecture uses the same algorithms for the Sense procedure as the deliberative architecture. As previously mentioned, Act is split into five independent sub-behaviors, and each behavior calculates a unique burn necessary to perform its desired function. Below is a further description of each sub-behavior along with an explanation of it's requested burn and weighting value.

Move Closer

Calculates a burn weight, w_{mc}, for a maneuver that is intended to move the inspector closer to the host in the LVLH frame. Using the estimated position, an estimate can be made for the distance from the target along the orbital path (x-axis). The weight for this burn is proportional to the distance to target, meaning the weight decreases linearly as the inspector approaches the host. The burn is applied along the orbital velocity vector towards the target.

$$w = |x_{target} - x| + 1 \tag{5}$$

Don't Hit

Calculates a burn weight, for a maneuver that is intended to prevent the inspector from colliding with the host craft. As before, an estimate is made for the distance from the target along the x-axis. The weight for this burn increases exponentially as the estimated distance to the host craft approaches zero. The burn is applied along the orbital velocity vector away from the host.

$$w = A^{x - x_{target}} \tag{6}$$

Station Keeping

Calculates a weight for a zero burn condition that is intended to keep the inspector within an acceptable distance from the target location. The previous estimate is for the distance from the target along the x-axis is used to determine the weight. The weight for this reaction is gaussian as it approaches the target position. This sub-behavior calls for zero burn, so it simply nullifies other commands when the target position has been reached.

$$w = Be^{\frac{-(x - x_{target})^2}{2 * \sigma^2}} \tag{7}$$

Stay on Orbit

Calculates a burn weight for a maneuver that is intended to keep the inspector on the same orbit as the host craft. This sub-behavior is only concerned with changes in altitude, and calculates the weight by using the difference between the desired altitude and current estimated altitude. It uses both a linear and parabolic relationship to obtain a weight that increases as the altitude of the craft becomes further from that of the host craft. The burn is applied along the y-axis (radial axis from the inspector through center of the Earth) in the direction of the host's orbital path.

$$w = (y_{target} - y)^2 + |y_{target} - y| \tag{8}$$

Stay in Plane

Calculates a burn weight for a maneuver that is intended keep the inspector on the same orbital plane as the host craft. This algorithm is similar to "Stay on Orbit," but is instead concerned with the position changes along the z-axis (axis perpendicular to both orbital velocity vector and radial vector). It calculates the weight by using the distance that the craft has traveled out of plane. The algorithm uses both a linear and parabolic relationship to obtain a weight that increases as the craft becomes further out of the plane occupied by the host craft. The burn is applied along the z-axis in the direction of the host's orbital plane.

$$w = (z_{target} - z)^2 + |z_{target} - z| \tag{9}$$

As with the deliberative architecture, the algorithm will continue until the target is reached.

VII. Simulation Results and Discussion

A complete simulation was conducted using an initial condition state vector such that the initial x displacement was equal to 254, 762, or 1270 cm; the initial y and z displacements were equal to 0, +/- 127, or +/- 93.98 cm; and the initial velocities were all equal to zero. All initial condition test combinations were simulated, creating a total of 27 different initial condition state vectors. The deliberative and reactive architecture algorithms were independently simulated for both the laser range finder and the computer vision sensors, and 10 trials were simulated for each initial condition state vector. A total of 1620 trial runs were conducted to test which sensor and architecture method achieved an optimal efficiency in fuel usage, time to completion, or both. Refer to Tables I and II for the mean value and standard deviations for the satellite's time to completion, the total ΔV used, the radial distance from the docking port, and the "rate," or total velocity, of the satellite when it reached the docking port. It is important to note that an acceptable docking rate is 3.00 cm/s – a rate that is approximately 10 times larger than the docking rates attained through this simulation. Additionally, refer to Table III for the mean and standard deviation for fuel usage given an initial state vector with zero y and z offsets.

A. Deliberative Architecture: Computer Vision vs. Laser Range Finder

The deliberative algorithm was simulated for a total of 810 trials, and the means and standard deviations of its results were organized into Tables I and II, respectively. As seen in these tables, the computer vision (CV) and both laser range finder sensors (exact and noisy) had an average run time of approximately 353 seconds, with a standard deviation of about 166 seconds. For both time statistics, the CV was faster by approximately 0.02 seconds – a slight difference that could essentially be attributed to calculation errors, and is too small to accurately distinguish which sensor performed with better time efficiency using the deliberative method.

Next, to determine which sensor was able to successfully dock the satellite, the final distance between the satellite and the docking port was measured. This distance should essentially go to zero, and as seen in Table I, all sensors successfully attained a small docking distance of approximately 0.0018 cm at a rate of 0.35 cm/s. Additionally, it is important to note that in 1987, NASA JSC deemed an acceptable approach rate to be between 3.05 cm/s and 6.10 cm/s– a value that is almost 10 to 20 times larger than the

TABLE I
SIMULATION MEAN VALUE RESULTS FOR ALL TRIAL RUNS

Architecture	Sensor	Time (s)	Fuel (cm/s)	Distance (cm)	Rate (cm/s)
Deliberative	CV	353.307037	6.318384	0.001898	0.350590
	laser (exact)	353.503704	10.124424	0.001811	0.356138
	laser (noisy)	353.557407	9.219628	0.001824	0.351206
Reactive	CV	371.858889	161.473580	0.303015	0.132178
	laser (exact)	364.675926	49.945871	0.141140	0.127627
	laser (noisy)	363.357407	277.725357	0.032213	0.410119

TABLE II
SIMULATION STANDARD DEVIATION RESULTS FOR ALL TRIAL RUNS

Architecture	Sensor	Time (s)	Fuel (cm/s)	Distance (cm)	Rate (cm/s)
Deliberative	CV	166.281398	1.455229	0.001860	0.429374
	laser (exact)	166.424436	4.650075	0.001864	0.412567
	laser (noisy)	166.462108	3.635545	0.002010	0.415513
Reactive	CV	167.096079	72.930148	0.125167	0.079502
	laser (exact)	167.232888	38.174648	0.043787	0.011663
	laser (noisy)	167.138923	130.190847	0.016381	0.209043

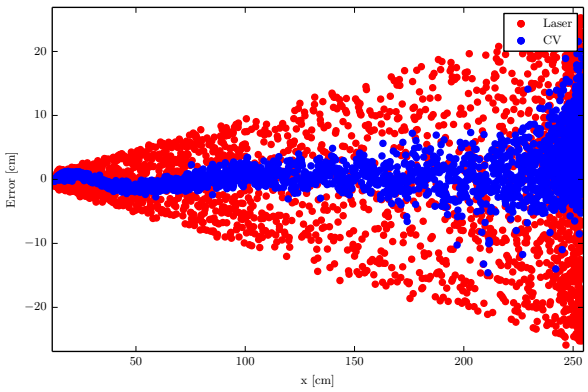

Fig. 7. Sensor Errors

approach rate attained for either simulated architecture [29]. However, while the distribution of these three satellite-to-port distances is roughly equal to $8.7e-5$cm, the exact and noisy lasers had a greater mean fuel usage by about 4 cm/s and 3 cm/s, respectively, in comparison to the CV sensor. Furthermore, both lasers' total fuel usage had a standard deviation equal to about 4.65 cm/s and 3.63 cm/s, whereas the computer vision had a standard deviation equal to about 1.46 cm/s, indicating the laser's fuel usage had a greater variance. These significant differences suggest that the computer vision sensor conducted more efficient ΔV burns in comparison to the laser range finder sensor. Additionally, it is important to note that the noisy laser had a better fuel efficiency than the exact laser sensor – an unexpected result seeing as the exact laser was simulated without added noise error, and thus mimics a "perfect" sensor.

To examine the mean fuel usage as a function of distance, Table III shows the mean fuel usage for both sensors when the only non-zero initial condition was the displacement along the x-direction. As the satellite is initially displaced farther from the target, the average total fuel consumption subsequently increases, and the difference between the CV's and the laser's fuel efficiency begins to increase as a function of distance. For example, when the satellite has a small initial displacement equal to 254 cm, the noisy laser sensor has the best performance with an average total burn rate of about 5.78 cm/s, followed by the exact laser sensor at 5.81 cm/s, and then finally the CV sensor with a burn rate of 5.87 cm/s. However, as the initial displacement between the satellite and the target increases to 762 cm, the "most efficient" sensor switches from the noisy laser range finder to the computer vision sensor, with an increased performance by about 0.074 cm/s. Then for the farthest offset of 1270 cm, the difference in fuel efficiency between the CV and noisy laser sensor increases by almost a factor of ten to 0.496 cm/s. Despite this distribution increase, the overall magnitude of this distribution was small, and essentially all three sensors had a comparable performance for this given set of initial conditions.

The main reason why the efficient sensor switches from the noisy laser range finder to the computer vision sensor is directly related to noise. When utilizing a computer vision sensor, an innate amount of natural error exists from computing image gradients and conducting Canny edge detection algorithms. Since computer vision has a substantial amount of preexisting noise, no additional noise was added to these measurements. However, since noise was only systematically added to the laser's position estimations, there exists a noticeable difference between the amount of error in the noisy laser's calculations as compared to those attained through computer vision. As seen in Figure 7, the laser's

TABLE III
FUEL USAGE FOR TRIALS CONTAINING ZERO INITIAL Y AND Z OFFSETS

Architecture	Sensor	x(cm)	Mean Fuel(cm/s)	Std. Dev Fuel (cm/s)
Deliberative	CV	254	5.873499	0.375036
		762	6.272073	0.434431
		1270	6.617597	0.400670
	laser (exact)	254	5.810814	0.399440
		762	6.379098	0.373520
		1270	7.058843	0.484380
	laser (noisy)	254	5.780540	0.383680
		762	6.346101	0.336987
		1270	7.113471	0.416979
Reactive	CV	254	66.394292	5.097201
		762	76.760390	4.843838
		1270	85.431635	12.143697
	laser (exact)	254	8.475532	0.060457
		762	15.361138	0.104953
		1270	22.956919	0.235527
	laser (noisy)	254	49.102620	2.463416
		762	58.001444	2.857584
		1270	65.3067299	2.863291

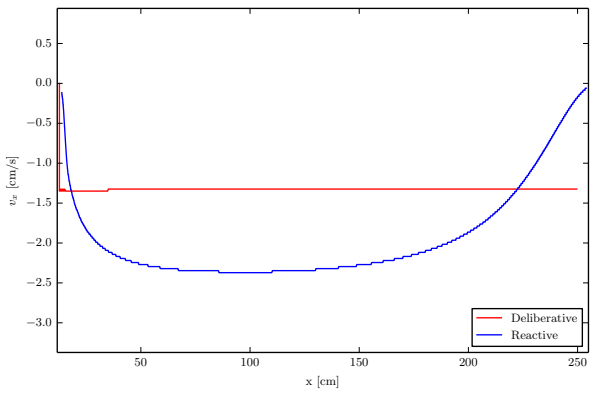

Fig. 8. X-Velocity for a Deliberative and Reactive Time Exact Laser Sensor Trial

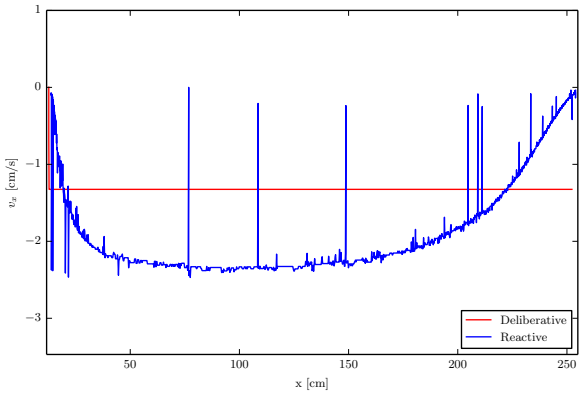

Fig. 9. X-Velocity for a Deliberative and Reactive Time Noisy Laser Sensor Trial

error has a canonical shape, where the cone has a larger error distribution the farther the satellite is displaced from the docking port. In comparison, the computer vision has a more refined error boundary region, exhibiting a slight sinusoidal behavior as the satellite is displaced closer to the target. Since the laser sensor has a larger error boundary, it is logical to conclude that this larger error was propagated, and created a higher fuel consumption from conducting a greater number of correction burns. To determine whether a reduction in added laser error would lead to a more optimal performance, an exact laser sensor was simulated. As discussed above, this exact laser surprisingly had the least efficient mean fuel usage for all time trials, and only performed the second best for a subset of trials in Table III.

B. Reactive Architecture: Computer Vision vs. Laser Range Finder

The reactive architecture's lack of a planning stage makes it much more sensitive to noisy measurements. When the exact state is used, the fuel usage is relatively low, at 8.47 cm/s for an initial state of x = 254 cm, and increases linearly with distance to a total fuel usage of 22.96 cm/s at x = 1270 cm. Once some noise is added to the laser sensor, the fuel usage goes up by about 40 cm/s, to 49.10 cm/s at x = 254 cm, and again increases at the same linear rate as before to 65.41 cm/s at 1270 cm. The CV sensor results in the largest total fuel usage for the reactive method as the CV error has the largest variance. The CV fuel usage goes up by another 15 cm/s, to 66.39 cm/s at x = 254 cm, and again linearly increasing, to 85.43 cm/s at 1270 cm. The standard deviations of all of these

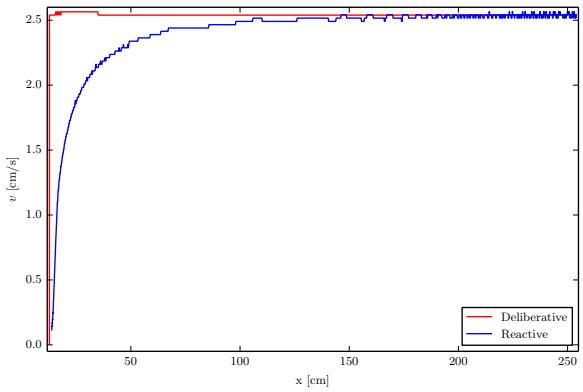

Fig. 10. Total Velocity Profile for a Deliberative and a Reactive Exact Laser Time Trial

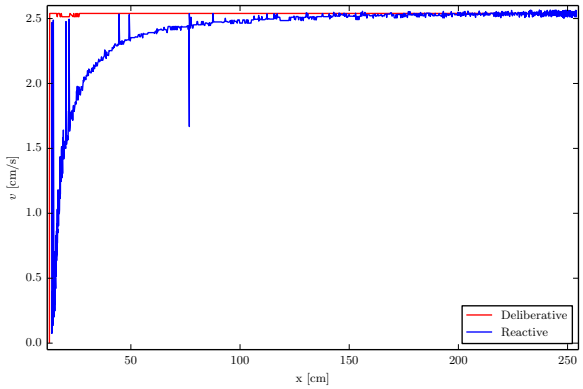

Fig. 11. Total Velocity Profile for a Deliberative and a Reactive Exact Laser Time Trial

means also increases in the same order, first with an increase in distance, then with an increase in sensor error. For all of these values, the initial conditions for y and z offsets were set to 0 cm, refer to Table III.

Insert analysis paragraph

C. Deliberative vs. Reactive Architecture

As seen in Tables I through III, the deliberative algorithm exhibits greater efficiency than the reactive algorithm for both the laser range finder and the computer vision sensors. For the average time to completion, the reactive algorithm exceeds the deliberative method's time by about 10 and 18 seconds for the laser (noisy and exact) and CV sensors, respectively. However, its standard deviation differs from the deliberative method's standard deviation by less than one second, indicating that significant outliers were not responsible for increasing the average time to completion. As a result of this longer flight time, the reactive method exhibits a significant loss in fuel efficiency. This inefficiency can be seen in Table I where the exact laser is the best performing sensor under the reactive method, utilizing an average total fuel of approximately 49.946 cm/s – almost 5 times larger than the deliberative's comparable average fuel use of 10.124 cm/s. Furthermore, despite the reactive method's higher average fuel usage, the reactive method guides the satellite to an average final displacement that varies between 0.303 cm and 0.032 cm away from the target using the CV and laser sensors, respectively. These final displacements have a significant amount of error when compared to the small average displacements of 0.0018 cm for both deliberative sensors, reinforcing the final conclusion that the deliberative method exhibited greater time and fuel efficiency.

To compare the two architecture methods for a single trial run, refer to figures 8, 9, 10 and 11 where the system has an initial displacement of 254 cm, 127 cm, and 93.98 cm in the x, y, and z directions, respectively. As seen in Figures 8 and 9, the deliberative x-velocity profile maintains a relatively constant velocity of 1.27 cm/s as the satellite moves towards the docking port. When the satellite is approximately 2.54 cm from the target, the ΔV burn goes to zero, and the satellite drifts into its final 0.0018 cm displacement. In comparison, the reactive x-velocity profile exhibits a dynamic response, with large velocity spikes under the noisy laser for the first 50 cm traveled along the x-direction. As the system continues to move closer, large velocity spikes occur intermittently for approximately every 50 cm between the 203 cm and 100 cm displacement points. Once the satellite reaches approximately 75 cm from the docking port, large ΔV burns along the x-direction increase in frequency, with a final velocity spike at approximately 2.54 cm from target. As expected, the exact laser's x-velocity profile is a smoothed version of the noisy laser, omitting the large spikes in velocity since there was no added noise to the system.

If the total velocity profiles for each architecture is closely examined, the constant deliberative and the dynamic reactive velocity characteristics are reflected in figures 10 and 11. While the deliberative architecture appears to maintain its constant velocity profile and respective drop off at 2.54 cm, the reactive architecture's total velocity resembles a smooth and noisy exponential decay curve as the satellite moves in from its initial displacement under the exact and noisy lasers, respectively. As previously seen in the reactive's x-velocity profile, the system exhibits periodic large velocity spikes under the noisy laser, with a final velocity surge when the system is approximately 2.54 cm from target.

These results essentially reinforce the aforementioned claim that the reactive method does not take fuel efficiency into consideration when it computes its ΔV burns due to its lack of planning. In order to compensate for the reactive method's position error, large correction burns are performed to redirect the satellite, and ultimately sacrifices the system's fuel efficiency in order to prevent a target collision. Therefore, while the reactive method is slightly faster to implement computationally, the deliberative method proves to have an increase in both overall performance and efficiency, and is thus the optimal choice for this simulation.

VIII. Summary

Deliberative and reactive architectures were implemented to test autonomous rendezvous docking algorithms through

a virtual simulation environment. A simplified SPHERES VERTIGO image-processing algorithm was created and tested using OpenCV and EDGE. Several trials were conducted to compare these two architectures, and the system's initial conditions and sensors were varied to investigate their effects on fuel usage and time to capture.

While the naive reactive method had no concept of the orbital mechanics governing the simulation, it was still able to perform similarly to the deliberate method for the case of no error and small distances. The CV algorithm was able to run in real time without any knowledge of the actual state of the simulation. Additionally, the CV sensor performed better than the laser sensor for some initial conditions.

The deliberative architecture performed much better than the reactive architecture with respect to fuel usage, and slightly better with regards to time to capture. Sensors with larger error tended to cause their architectures to perform worse. As there was no filtering done to any signal inputs, both architectures had large variations due to sensor noise. The deliberate architecture mostly varied based off the magnitude of the sensor error, while the reactive architecture did worse with large variations in sensor error.

IX. Future Work

There are two different areas of the system that can be vastly improved upon. The first is within the sense step. It can be achieved by using both sensors simultaneously and combining their measurements into one best estimate for the position. The second can be achieved by creating a hybrid controller.

The optimal way to combine the two sensors would be to use a Kalman filter to mitigate the error in the position error. With better position estimates, the performance could be greatly improved. Since the Sense method is the same for all architectures, the improvements will be seen with a deliberative, reactive, or hybrid control architecture. More accurate position estimates will lead to smoother paths and increased efficiency. This may not improve accuracy in all cases, but will indeed give the best possible result for tested conditions.

Since different aspects of the two controllers were tested, a single hybrid controller can be designed that will implement characteristics of the two architectures to gain the advantages of both methods. For example, instead of trying to follow the orbital path of the host, an optimal path can be calculated using the same Clohessy-Wiltshire model used in the deliberative algorithm. Once a hybrid controller algorithm is developed, it must be tested to see if any unforeseen disadvantages arise.

Reactive sub-behaviors could be further split up to calculate burn weights for each axis individually. This would fix such problems as path keeping burns being outweighed by the move closer behavior at large distances from the target. Once the division of the axes is complete, the weighting equations would then be optimized. A genetic algorithm can be implemented to test various combinations of weight amplitudes and equations. Many simulations must be run at the various initial condition and noise levels to fully test each setting.

With an optimized hybrid control architecture and Kalman filter to combine both laser rangefinder and computer vision measurements, physical trials could be performed. Further optimization of both the Kalman filter and control architecture must be performed after the system is applied to a physical system to ensure that the optimal behavior is still achieved.

X. Notes

look back to modify the question
fix the tables
consistent 2 decimal places on numbers

References

[1] R. Nugent, R. Munakata, A. Chin, R. Coelho, and D. J. Puig-Suari, "The cubesat: The picosatellite standard for research and education," 2008. [Online]. Available: http://gwww.cubesat.org/images/More_Papers/cps2008.pdf

[2] K. Dismukes, "AERCam Sprint," 2003. [Online]. Available: http://spaceflight.nasa.gov/station/assembly/sprint/

[3] NASA, "NASA Johnson Space Center's Miniature Autonomous Extravehicular Robotic Camera (Mini AERCam)," 2002. [Online]. Available: aercam.jsc.nasa.gov_aercam.pdf

[4] H. J. Kramer, "ISS utilization: SPHERES (Synchronized Position Hold Engage Reorient Experiment Satellites)," 2002. [Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/i/iss-spheres

[5] B. E. Tweddle, E. Muggler, A. Saenz-Otero, and D. W. Miller, "The SPHERES VERTIGO Goggles: Vision based mapping and localization onboard the International Space Station," in *Proceedings of i-SAIRAS (International Symposium on Artificial Intelligence, Robotics and Automation in Space)*, 2012, Conference Proceedings. [Online]. Available: http://ssl.mit.edu/spheres/library/tweddle-isairas12.pdf

[6] M. Machula and G. Sandhoo, "Rendezvous and docking for space exploration," January 30 - Februrary 1, 2005 2005.

[7] Darpa.mil, "Developing technologies for more flexible, cost-effective satellite operations in geo," 2015. [Online]. Available: http://www.darpa.mil/Our_Work/TTO/Programs/Phoenix.aspx

[8] J. W. Thomas Guay, "International space station - quantifying and reducing risk following orbital debris penetration," September 24-26, 1996, 2008 1996.

[9] NASA.gov, "Astronauts complete spacewalk to repair ammonia leak, station changes command," 2013. [Online]. Available: http://www.nasa.gov/mission_pages/station/expeditions/expedition35/e35_051113_eva.html

[10] S. E. Frederickson, S. Duran, and J. D. Mitchell, "Mini AERCam inspection robot for human space missions," 28-30 Sept. 2004 2004.

[11] D. Fourie, B. E. Tweddle, S. Ulrich, and A. Saenz-Otero, "Vision-based relative navigation and control for autonomous spacecraft inspection of an unknown object," August 19-22, 2013 2013.

[12] ——, "Flight results of vision-based navigation for autonomous spacecraft inspection of unknown objects," *Journal of Spacecraft and Rockets*, vol. 51, no. 6, p. 10, 2014.

[13] M. Williams, "Google's Project Tango headed to International Space Station," 2014. [Online]. Available: http://www.cio.com/article/2377746/smartphones/google-s-project-tango-headed-to-international-space-station.html

[14] E. M. Kong, A. Saenz-Otero, S. Nolet, D. S. Berkovitz, and D. W. Miller, "SPHERES as a formation flight algorithm development and validation testbed: Current progress and beyond," 2004. [Online]. Available: http://ssl.mit.edu/spheres/library/SPHERESFFPaper72.pdf

[15] A. Fejzic, "Results of SPHERES microgravity autonomous docking experiments in the presence of anomalies," in *Proceeding of the 59th IAC (International Astronautical Congress)*, 2008, Conference Proceedings. [Online]. Available: http://ssl.mit.edu/spheres/library/IAC_2008_AmerFejzic.pdf

[16] A. Saenz-Otero, G. Aoude, M. M. Jefrey, S. Mohan, A. Fejzic, J. Katz, C. Edwards, and D. W. Miller, "Distributed satellite systems algorithm maturation with SPHERES aboard the ISS," in *Proceedings of the 59th IAC (International Astronautical Congress)*, 2008, Conference Proceedings. [Online]. Available: http://ssl.mit.edu/spheres/library/SPHERES_IAC2008.pdf

[17] A. F. Velasquez, G. Marani, T. Evans, M. R. Napolitano, J. A. Christian, and G. Doretto, "Virtual simulator for testing a vision based pose estimation system for autonomous capture of satellites with interface rings," pp. 1597–1602, 2013 2013. [Online]. Available: http://www.csee.wvu.edu/~gidoretto/publications/velasquezMENCD13med.pdf

[18] B. E. T. Miller and D. W., "Computer vision based navigation for spacecraft proximity operations," Thesis, Massachusetts Insitute of Technology, 2010.

[19] B. E. Tweddle, "Relative computer vision based navigation for small inspection spacecraft," 2011. [Online]. Available: http://ssl.mit.edu/spheres/library/tweddle_aiaa_gnc_2011.pdf

[20] B. E. Tweddle, J. McClellan, G. Vulikh, J. Francis, and D. Miller, "Relative vision based navigation and control for the Mars sample return mission: Capturing the Orbiting Sample," May 18-20, 2011 May 2011. [Online]. Available: http://ssl.mit.edu/spheres/library/tweddle_MOSR_SFFMT_2011.pdf

[21] Fujinon, "Fujinon yf2.8a-2 wide angle 2.8mm f/1.3 cctv lens with manual iris and focus, for 1/3-inch and 1/4-inch ccd cameras with cs-mount," 2015. [Online]. Available: http://www.bhphotovideo.com/c/product/404186-REG/Fujinon_YF28A2_YF2_8A_2_Wide_Angle_2_8.html

[22] NASA.gov, "Apollo 15, day 4: Lunar encounter," 1998. [Online]. Available: http://history.nasa.gov/ap15fj/09day4_lunar_encounter.htm

[23] Ocampo, "Clohessy-Wiltshire equations," 2015. [Online]. Available: http://courses.ae.utexas.edu/ase366k/cw_equations.pdf

[24] NASA, "Enhanced graphics tools for advanced 3D simulations," 2015. [Online]. Available: http://www.nasa.gov/centers/johnson/techtransfer/technology/MSC-24663-1-doug-edge.html

[25] Opencv.org, "Opencv," 2015. [Online]. Available: http://opencv.org

[26] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[27] S. Joshi, "MAE252. class lecture, topic: Autonomous robotics lecture," 2015.

[28] F. Hoots and J. Thomas F. Starchville, "Debris risk assesment process," August 18-21, 2008 2008.

[29] E. M. B. Walter, C. R. Price, and L. M., "Track and capture of the orbiter with the space station remote manipulator system," National Aeronautics and Space Administration, Johnson Space Center, Report, 1987.