Hardware Testbed for Relative Navigation of Unmanned Vehicles Using Visual Servoing

Mark J. Monda

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Aerospace Engineering

Hanspeter Schaub, Chair

Christopher D. Hall

Craig A. Woolsey

April 27, 2006

Blacksburg, Virginia

Keywords: Hardware Testbed, Relative Motion Simulation, Visual Snakes, Visual Servoing,

Parameter Estimation, Aerial Refueling, Forced Perspective

Hardware Testbed for Relative Navigation of Unmanned Vehicles Using Visual Servoing

ABSTRACT

Mark J. Monda

Future generations of unmanned spacecraft, aircraft, ground, and submersible vehicles will require precise relative navigation capabilities to accomplish missions such as formation operations and autonomous rendezvous and docking. The development of relative navigation sensing and control techniques is quite challenging, in part because of the difficulty of accurately simulating the physical relative navigation problems in which the control systems are designed to operate. A hardware testbed that can simulate the complex relative motion of many different relative navigation problems is being developed. This testbed simulates near-planar relative motion by using software to prescribe the motion of an unmanned ground vehicle and provides the attached sensor packages with realistic relative motion. This testbed is designed to operate over a wide variety of conditions in both indoor and outdoor environments, at short and long ranges, and its modular design allows it to easily test many different sensing and control technologies.

The first relative navigation sensing technique evaluated on the testbed is a visual sensor. The visual sensor uses a digital camera and a color statistical pressure snake algorithm running in real-time on an onboard computer to track a target image and determine the relative position and orientation of the target. Non-linear proportional and proportional-integral

feedback control laws are developed for the visual servoing problem. The effectiveness of the sensing and control techniques are evaluated on the hardware testbed, and results are shown for both collaborative and non-collaborative targets in real-world test conditions.

This research was supported by Sandia National Laboratories under Contract Number # 307618. We would also like to thank ORION International Technologies for providing a license to the UMBRA software framework.

Contents

1 Introduction **1**

1.1 Hardware Simulation 2

1.2 Visual Sensing 6

1.3 Visual Servoing 9

2 Hardware Simulation of Relative Motion **11**

2.1 Relative Motion Simulation 12

2.2 The UMBRA Simulation Framework 15

2.3 Simulation Hardware 17

2.3.1 Unmanned Ground Vehicle 17

2.3.2 Pan-and-Tilt Unit 20

2.3.3 Visual Sensing Components 21

3 Statistical Pressure Snake Algorithms **23**

3.1 Statistical Pressure Snakes 24

3.2 Extracting Target Image Features 30

3.2.1 Moments of a Parametric Curve 30

3.2.2 Shape Eigenfactors 31

3.2.3 Principal Axis Dimensions 34

3.3 Visual Snake Performance 37

4 Constant Gain Proportional-Integral Visual Servoing **43**

4.1 Relative Heading and Range Determination 44

4.2 Proportional Feedback Control 46

4.3 Proportional-Integral Feedback Control 50

4.4 Hardware Visual Servoing Results 52

4.4.1 Fixed-Target Range Servoing Experiment 54

4.4.2 2D Visual Tracking of a Moving Target 56

4.5 UMBRA Visual Servoing Module Implementation 59

5 Variable Gain Proportional Visual Servoing **61**

5.1 Proportional Feedback Control with Time Dependent Gain 62

5.2 Hardware Application of Time Dependent Feedback Gains 65

6 Visual Servoing with Parameter Estimation **69**

6.1 Parameter Estimation 70

6.2 Hardware Implementation 74

6.3 Hardware Visual Parameter Estimation Results 76

6.3.1 Long Range Test 76

6.3.2 Close Range Test 79

6.3.3 Changing Target Size 79

7 Visual Sensing for Aerial Refueling **83**

7.1 Forced Perspective Target Setup 85

7.2 Sensitivity Analysis 88

7.3 Simulation Results 90

8 Conclusions and Future Work **95**

List of Figures

1.1 Simulated Spacecraft Hovering on a Polished Granite Table at Stanford University's Aerospace Robotics Lab.[14] 3

1.2 6 DOF Spacecraft Simulator at the U.S. Naval Research Laboratory.[6] 4

1.3 Pioneer 3-DX Robotic Ground Vehicle Used for Relative Motion Simulation at the Virginia Tech AVS Lab. 5

1.4 Block Diagram of the AVS Lab Relative Motion Hardware Testbed. 6

2.1 Illustration of a Free-Flying Robotic Assistant Visually Servoing on a Passive Optical Target on the Space Station. 12

2.2 Testbed Illustration Where an Unmanned Ground Vehicle is Used to Mimic Near-Planar Spacecraft Motion Relative to a Target Vehicle. 14

2.3 ActivMedia Robotics Pioneer 3-DX Robotic Ground Vehicle with a Digital Camera System on a PTU. 18

2.4 Onboard PC-104 Computer Inside the Pioneer 3-DX UGV. 19

2.5 PTU Mounted on the Pioneer 3-DX UGV. 21

2.6 Block Diagram of UMBRA Visual Sensing Modules Used in Hardware Visual Servoing. 22

3.1 Color Statistical Pressure Snake Tracking a Visual Target. 24

3.2 Conic Illustration of the Hue-Saturation-Value (HSV) Color Space. 26

3.3 Examples of the Visual Snake Tracking Different Targets. 28

3.4 Visual Snake Tracking an Elliptical Target. 36

3.5 Zoomed View of a Target Edge for an Ideal Test Image and a Camera Image. 38

3.6 Calculated X COM Error from the Visual Snake Algorithm with an Ideal Test Image. 39

3.7 Principal Axis Length Error from the Visual Snake Algorithm with an Ideal Test Image. 40

3.8 COM and Principal Axis Error Percentage for an Elliptical Target from the Visual Snake Algorithm with an Ideal Test Image. 41

3.9 Histogram of X COM Measurement Error from the Visual Snake Algorithm with an Elliptical Ideal Test Image. 42

4.1 Simplified Illustration of the Pin-Hole Camera Model Used for Depth Extraction 44

4.2 Non-linear Smoothly Saturating Control Law with $f(x) = \arctan(x)$ 47

4.3 Block Diagram of UMBRA Modules Used in Hardware Visual Servoing. . . . 53

4.4 Illustration of the UGV Following a Rectangular Target Moved Manually in a Laboratory. 54

4.5 Visual Range Servoing Results Where the Commanded Range Is Changed in Discrete Steps. 55

4.6 Experimental Results from the 2D Tracking Test of a Randomly Moving Visual Target in a Visually Cluttered Laboratory Environment. 58

5.1 Illustration of Range Coordinates and Tracking Errors. 63

5.2 Sample Screenshots from Short and Long Range Visual Servoing Tests . . . 65

5.3 Position Data from Depth-Dependent and Constant Gain Long Range Visual Servoing Tests. 67

6.1 Illustration of a UGV Servoing on a Non-Collaborative Visual Target. 70

6.2 Long Range Visual Servoing Maneuver with Parameter Estimation. 78

6.3 Close-Range Visual Servoing Maneuver with Visual Calibration Parameter Estimation. 80

6.4 Changing Target Size Visual Servoing Maneuver with Parameter Estimation. 82

7.1 Illustration of Probe-and-Drogue Aerial Refueling Method 84

7.2 Flying Boom Aerial Refueling Method 85

7.3 Illustration of Forced Perspective Showing Visual Targets as Seen by the Tanker and as Painted on the Receiver. 87

7.4 Aircraft Coordinate System Used in Visual Snake Aerial Refueling Simulations. 89

7.5 Position Error Magnitude for Aerial Refueling Visual Position Sensing Simulation 91

7.6 Range Error, Heading Error, and Heading Position Error for Aerial Refueling Visual Position Sensing Simulation. 92

7.7 Exaggerated Illustration of the Shape of the Range (Red) and Heading Errors (Green) from Aerial Refueling Visual Position Sensing Simulation. 94

List of Tables

3.1 Statistically Averaged Snake Performance for an Elliptical Target of Size 200 Pixels 42

4.1 Simulation Parameters for Proportional Range Regulation Experiment. . . . 56

4.2 Simulation Parameters for 2D Tracking Experiment of a Moving Target. . . 57

5.1 Simulation Parameters for Depth-Dependent Gain Tests. 68

6.1 Simulation Parameters for Hardware Visual Parameter Estimation Tests. . . 77

7.1 Range Error and Heading Error Sensitivity to Perturbations from Nominal Position in the Aerial Refueling Visual Position Sensing Simulation 89

7.2 Range Error and Heading Error Sensitivity to Perturbations from Nominal Orientation in the Aerial Refueling Visual Position Sensing Simulation . . . 90

7.3 Error Magnitude, Range Error, and Heading Error Data from Aerial Refueling Visual Position Sensing Simulation. 93

Chapter 1

Introduction

Future generations of unmanned underwater, ground, air, and space vehicles offer many benefits over their manned counterparts including reduced size, cost, complexity, and danger for human operators. Many of these planned vehicle systems will operate in environments that require extremely precise relative navigation between vehicles. For example, a small, unmanned, free-flying spacecraft that is able to maintain precise formation with a larger vehicle, such as the Space Shuttle or the International Space Station, could fly around the larger craft, inspecting for damage or unusual wear and tear. It could also act as a helper vehicle, assisting astronauts performing space construction projects, or conducting repair or deployment operations in proximity to the larger vehicle. Other examples of relative navigation problems include formation flight or aerial refueling of aircraft and formation operation of ground or submersible vehicles.

In all of these relative navigation problems, the absolute position of a vehicle is less important than its position relative to another vehicle. However, precise, centimeter-level relative position and orientation, or pose, measurements between the vehicles are required for many applications such as rendezvous and docking. Relative navigation problems are particularly challenging for several reasons. Measuring the pose between two vehicles with sufficient accuracy can be extremely difficult. But more generally, testing the required sensing and control techniques in hardware under realistic operating conditions is difficult due to the challenges in accurately simulating the relative navigation problems for which they are designed.

1.1 Hardware Simulation

To truly understand the capabilities and limitations of relative navigation sensing and control techniques, they should be developed not only in software simulation, but also with hardware simulation and testing. Hardware simulation is important because it captures non-idealities that may not be known or are not included in software simulation. These hardware tests should provide the sensors with realistic relative motion. Thus, a hardware testbed that can accurately simulate relative motion problems is vital for the development and evaluation of relative motion sensing and control techniques.

In the past decades, several different approaches have been used to simulate relative navigation problems, and Reference 39 contains an excellent survey of spacecraft simulators. Planar motion systems typically involve two translational and one rotational degree of freedom and

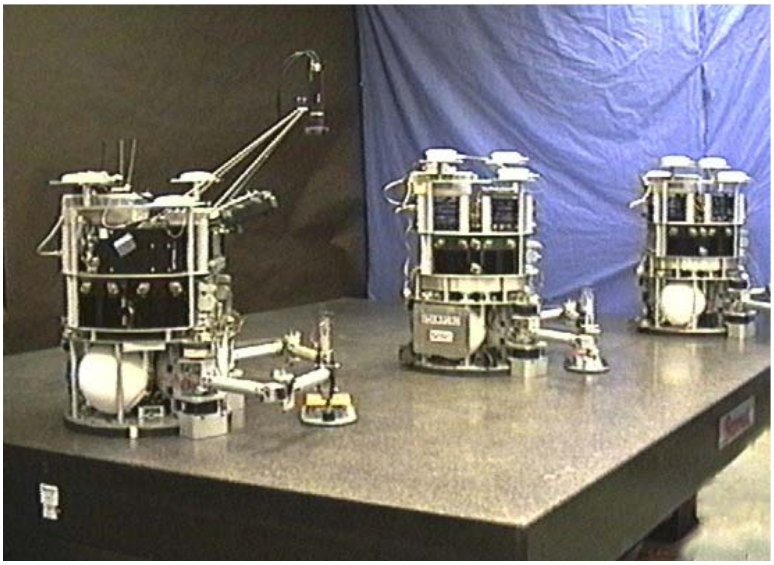


Figure 1.1: Simulated Spacecraft Hovering on a Polished Granite Table at Stanford University's Aerospace Robotics Lab.[14]

are of interest to rendezvous and docking operations, as well as flexible manipulator research. Simulators discussed in References 14, 20, 21, and 38 float test hardware on polished flat surfaces, and an example is seen in Figure 1.1. These planar simulators mimic the frictionless space environment to produce realistic force-free motion. However, these simulators are not typically able to simulate the relative orbital dynamics between formations of vehicles.

In contrast to the planar systems, another approach is the use of cranes or robotic manipulators to simulate the full 6 DOF motion of a vehicle. The U.S. Naval Research Laboratory's Spacecraft and Robotics Engineering and Controls Lab, seen in Figure 1.2, uses a software-controlled gantry crane to simulate the 6 DOF motion of a full-size spacecraft.[3] However, both this approach and planar float tables have rather limited ranges of motion. In addition, it is difficult to accurately simulate environmental conditions such as natural sunlight.

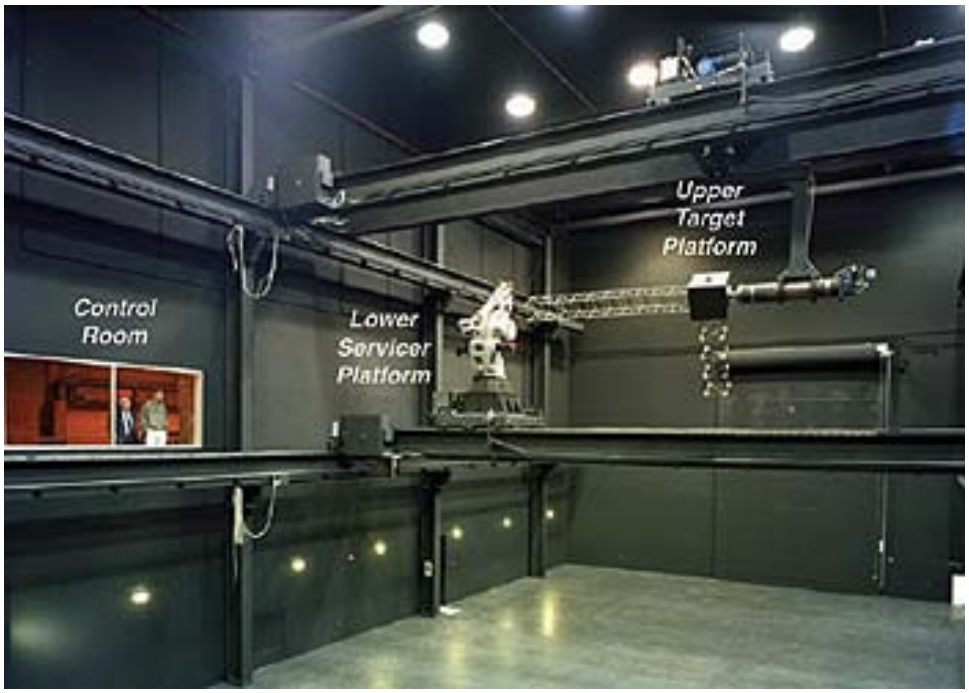


Figure 1.2: 6 DOF Spacecraft Simulator at the U.S. Naval Research Laboratory.[6]

The new testbed under development at Virginia Tech's Autonomous Vehicle Systems (AVS) Laboratory combines aspects from each of these approaches, using a software-controlled unmanned ground vehicle (UGV), seen in Figure 1.3, to simulate near-planar motion. The goal of the testbed is not to mimic frictionless planar motion, which is only an approximation of the coupled orbital motion a satellite would actually experience in flight. Rather, the UGV motion is prescribed by numerical software modules to provide the sensor package with realistic physical motion. This approach, illustrated in Figure 1.4, is quite similar to the approach used in modern aircraft flight simulators. The use of UGVs for motion simulation has several advantages. It allows relative motion tests to be conducted both indoors and outdoors in a variety of lighting and environmental conditions. Moreover, tests

Figure 1.3: Pioneer 3-DX Robotic Ground Vehicle Used for Relative Motion Simulation at the Virginia Tech AVS Lab.

can be conducted over very long ranges of up to hundreds of meters. Such flexibility is impossible with float tables or crane structures.

The AVS Lab's hardware testbed uses a software framework called UMBRA[13] which allows various hardware and software components to be combined seamlessly into a simulation. In this manner, it is similar to the Marshall Space Flight Center's Flight Robotics Laboratory,[29] a sophisticated testing facility in which test hardware interacts through networking. The AVS testbed is much smaller in scale and differs in the use of unmanned ground vehicles to simulate vehicle motion, providing a broader base of applications ranging from underwater, ground, aerial and space-based vehicles. Further, the use of UMBRA ensures compatibility with other research facilities such as Sandia National Laboratories.

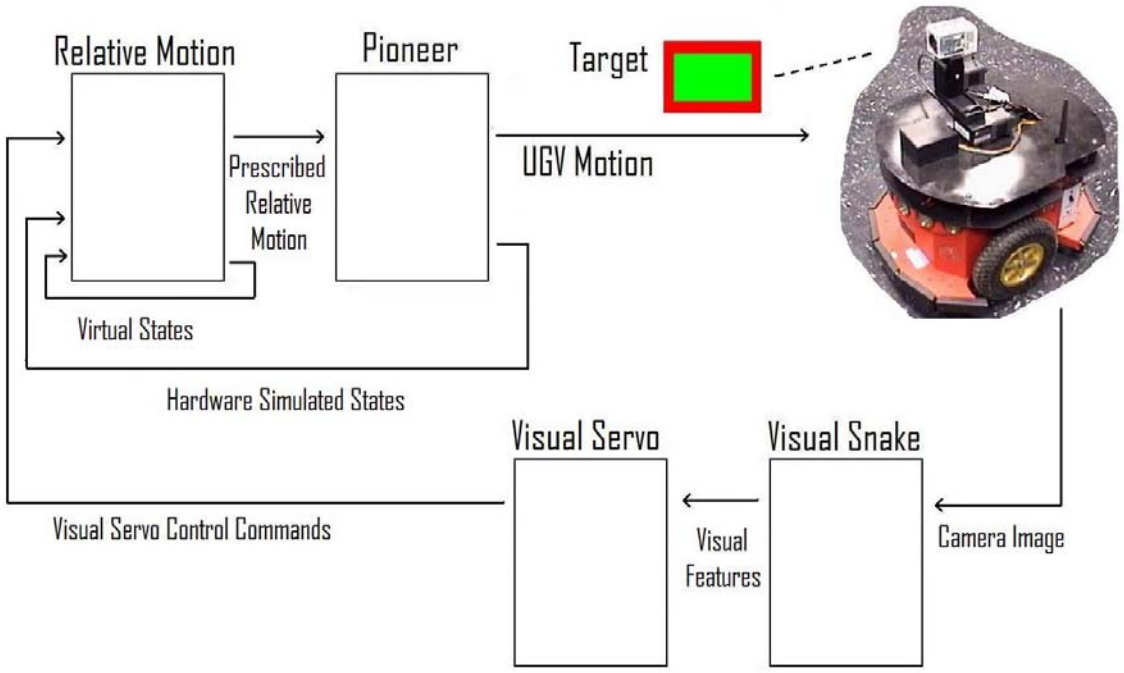


Figure 1.4: Block Diagram of the AVS Lab Relative Motion Hardware Testbed.

1.2 Visual Sensing

Future unmanned vehicle systems will require relative navigation sensors with centimeter-level accuracy at close ranges to accomplish their missions.[45] Many technologies have been suggested for achieving this type of precision navigation. Differential GPS, optical beacon systems, and radio triangulation all show promise in certain applications, but also have drawbacks. For example, differential GPS would be unable to guide vehicles in transit to the Moon or Mars. In Low Earth Orbits (LEOs), the differential GPS solution would also be susceptible to multi-path errors due to the close proximity of the vehicles. Optical beacon systems or radio triangulation could function at close ranges, but require the addition of substantial hardware to both vehicles, and are dependent on active communication between

them.[10]

One alternative relative navigation sensing technique is the processing of visual information. Humans rely heavily on vision for information about the surrounding environment, and in particular during relative navigation problems. For example, when performing an aerial refueling operation, the pilot's vision gives extremely accurate information about how the aircraft must be maneuvered relative to the tanker. However, it is difficult to endow unmanned vehicles with the same visual processing capabilities.

A novel visual sensing technique uses color statistical pressure snakes,[9,25,35,42] or visual snakes, to track a target in an image. The visual snake algorithm itself has been developed previously and is not the focus of this work. Rather, the performance of the visual snakes in relative navigation problems is evaluated. A typical visual sensing application involves equipping a vehicle with a digital camera and a computer for image processing. The relative pose between the vehicles is determined by tracking a visual target with the statistical pressure snakes.

This sensing technique has several exciting benefits. First, it does not require any additional hardware on the target vehicle, nor any active communication between the vehicles. In addition, visual sensing works regardless of location (orbiting about a planet or underway between planets). Finally, visual sensing tends to be most accurate at close ranges, when precise relative navigation is most important.

However, visual sensing techniques have drawbacks as well. Developing methods that work

well in changing lighting conditions is quite difficult. Therefore, extracting precise relative navigation information when varying lighting conditions cause irregular reflection and high noise levels, especially near the target's edges and boundaries, is quite difficult.[28] In addition, visual sensing is usually computationally intensive and difficult to use for real-time control applications.

Much work in the field of computer vision has focused on developing relative motion sensing techniques. Combining concepts from Kalman filtering with visual sensing algorithms has proven useful for mitigating the effects of lighting variation and image noise, and providing smoother, more continuous information with lower noise and error levels.[2,40] Many existing algorithms, though, are designed to analyze the motion of a moving object in images taken by a stationary observer.[43] Matthies, Szeliski, and Kanade have shown that range to a target can be estimated by prescribing a known camera motion.[19] However, in this application, the opposite is desired: given images taken from a moving vehicle, the pose between the vehicles is sought.

In this work, the use of an existing visual snake algorithm to extract relative navigation information from images is explored. A non-communicative target image fixed to a target vehicle is considered. The visual snake is used to track the target and calculate properties such as center of mass, area, and principal axis lengths. These outputs can be used for visual feedback control. Many possible applications involve a collaborative target of known size and shape. However, in applications where the target size is unknown, a Kalman filter is developed to estimate the unknown visual calibration parameter during maneuvers.

1.3 Visual Servoing

Using relative navigation information from the visual snake, velocity-based visual feedback control (or visual servoing) laws are developed. Building on the developments in Reference 33, smoothly-saturating non-linear constant-gain proportional-integral and variable-gain proportional control laws are developed. Related vision-based control of spacecraft is discussed in References 1 and 15, where 6 DOF, zero-gravity experiments are performed in a neutral buoyancy water tank, but ranges are limited to several meters. The visual tracking employs statistical sampling of the entire image for target colors, but such methods only provide fast tracking if the target is the only object of a particular color in the image. The AVS Lab testbed will provide a greater range of relative motion that can be simulated, and better lighting control by avoiding the need to suspend the test craft in water. Further, the visual snakes are more robust than statistical sampling in the presence of non-unique target colors.

Examples of real-time visual control of ground vehicles are given in References 4 and 5. A modular test setup is created using an omni-directional camera for sophisticated formation control. Targets are identified through statistical color models in the YUV color space for robustness to lighting variations, similar to the visual snake's use of the HSV color space. However, image gradients are used to determine objects of interest, assuming a flat monocolor background. The visual snakes will operate in unstructured environments without such assumptions.

The initial goal of the AVS Lab's hardware testbed is to evaluate visual sensing and visual

servoing of vehicles in near-planar motion. Chapter 2 describes the hardware testbed in detail. Chapter 3 discusses the visual sensing algorithm that has been implemented in real-time on the UGV. Chapters 4 and 5 present smoothly-saturating non-linear, constant-gain proportional-integral and variable-gain proportional feedback control laws. In Chapter 6, a non-collaborative target is considered, and a method to estimate the unknown visual calibration parameter is presented. Hardware results from the UGV testbed are presented for all the visual servoing problems. Finally, Chapter 7 presents a novel use of visual sensing for automated aerial refueling operations.

Chapter 2

Hardware Simulation of Relative Motion

Consider a free-flying robotic assistant maintaining its position relative to the International Space Station by looking at visual features on the station's structure. Without requiring complex radio or optical beacons, the vehicle is able to maintain its position using only a stream of images from a video camera. Figure 2.1 illustrates such a concept. Developing this system is a complex and challenging undertaking. In particular, for this space-based visual servoing example, performing hardware proof-of-concept tests includes simulating the complex relative orbital motion, simulating the three-dimensional attitude, and duplicating the intricate lighting, reflection and shadow variations seen in space. Physically simulating this problem is a daunting challenge. However, using modern software and communication developments, a novel spacecraft simulation approach is being developed that blends the

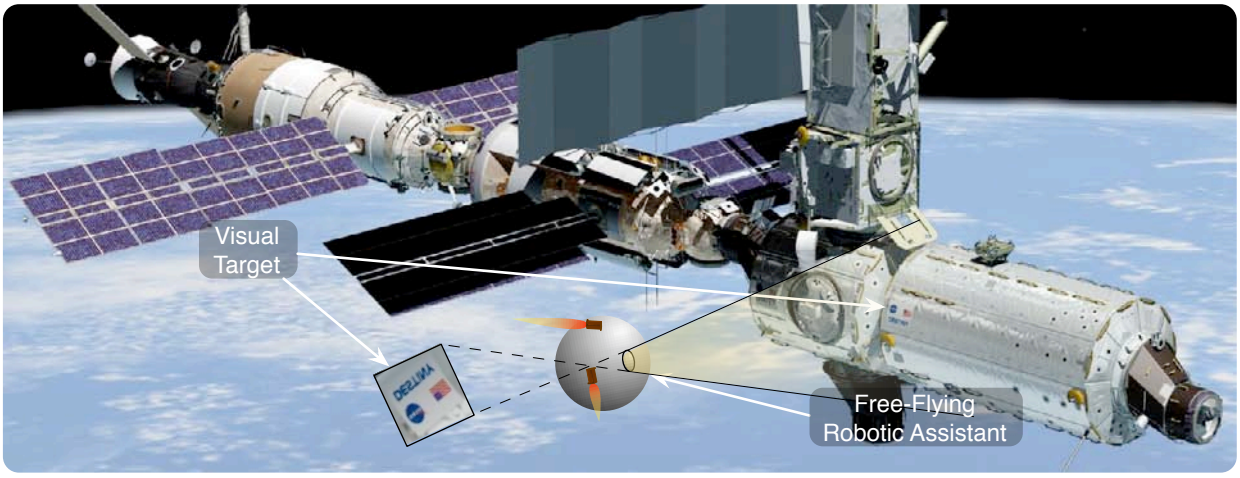


Figure 2.1: Illustration of a Free-Flying Robotic Assistant Visually Servoing on a Passive Optical Target on the Space Station.

boundaries between hardware and simulated components.

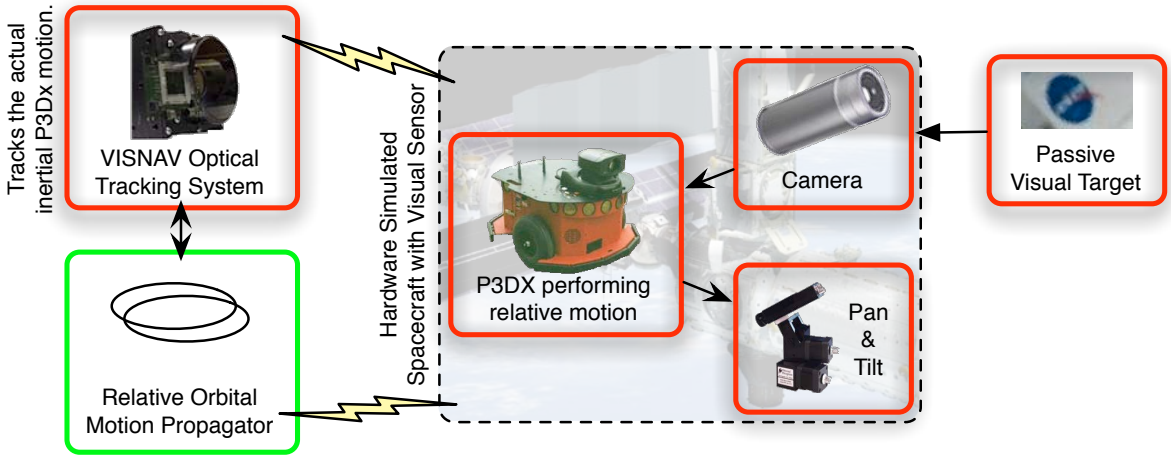
2.1 Relative Motion Simulation

Numerical simulations are commonly employed to model new navigational sensing and control concepts. While such simulations provide important initial studies, they are rarely detailed enough to fully model the problem and sufficiently develop relative navigation sensing and control techniques. Instead, hardware tests with critical sensing, communication and mechanical behaviors are typically required to fully develop these sensing and control methods. In practice, the complexity of many research projects requires combinations of numerical simulation and hardware testing at an early stage. A hybrid hardware and software simulation environment is being developed at the Virginia Tech Autonomous Vehicle Systems

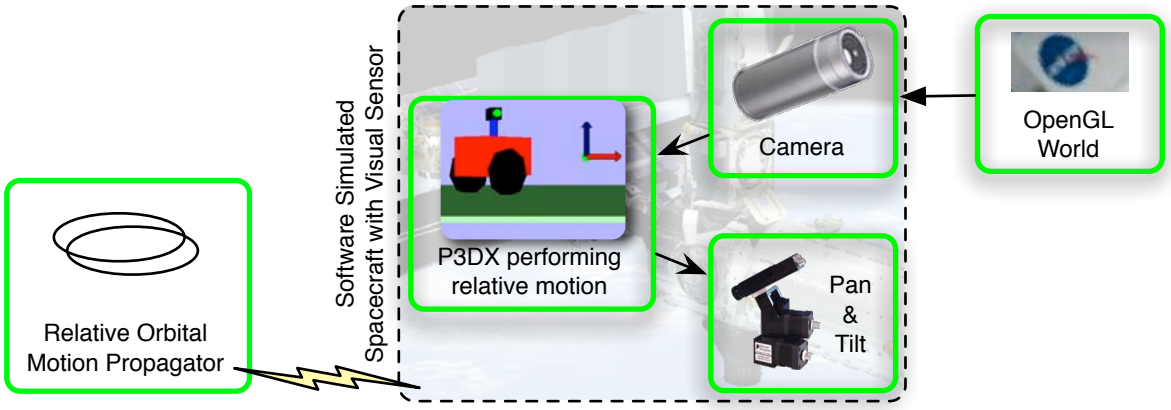
(AVS) Lab to investigate relative navigation sensing and control techniques for unmanned vehicle systems.

Instead of a costly testbed that simulates all environmental, physical, and communication aspects of an autonomous vehicle, a cost-effective relative motion testbed is being developed for scenarios where the relative motion dynamics are well understood. For example, an unmanned ground vehicle (UGV) is programmed to move relative to a target as if it were in orbit in proximity to that target vehicle. The UGV thus acts as a sensor platform, providing the relative navigation sensor with realistic relative motion. With onboard processing capabilities, the sensor data can be analyzed in real-time and a corresponding control solution computed. The response to the simulated control actuation of the modeled vehicle is then computed and simulated in software and the resulting physical vehicle relative motion is prescribed to the UGV through dedicated servo routines.

Figure 2.2(a) illustrates a free-flying robotic assistant visually servoing on another spacecraft in a simulated hybrid hardware and software environment. A physical camera is mounted on the UGV, and images are captured with a frame-grabber card. After processing the image and computing target tracking errors, the station-keeping control solution is sent to the simulated spacecraft module which computes the relative motion response. The vehicle motion will provide planar (x, y) motion, but cannot provide independent camera heading angles. By adding a pan-and-tilt unit (PTU) to the vehicle, the camera can be pointed independently from the vehicle heading. Further, using the tilt mode, small out-of-plane attitude motions can also be simulated.



(a) Physical Vehicle Simulating Relative Motion of Aerospace Vehicle



(b) Software Simulated Vehicles Servoing in a Virtual Environment

Figure 2.2: Testbed Illustration Where an Unmanned Ground Vehicle is Used to Mimic Near-Planar Spacecraft Motion Relative to a Target Vehicle.

To compute the spacecraft orbital response due to a control input, the actual inertial position of the UGV must be known. An optical tracking system called VISNAV[10] provided by StarVision Inc.[8] will be used. This system is capable of tracking active optical beacons both indoors and outdoors at ranges of up to 50 meters. The VISNAV system is expected to be delivered to the AVS Lab in late 2006.

The current hardware test setup consists of the operational UGV, a PTU that allows for independent control of the vehicle direction of travel and camera heading, a camera, and an onboard computer. Elementary wireless communication has been implemented so the vehicle can operate completely untethered. The current hardware testbed is capable of visually servoing the UGV on a target without any communication, video or power connections. This wireless operational mode is an important milestone to developing the full unmanned vehicle simulation capability.

2.2 The UMBRA Simulation Framework

The UMBRA simulation framework,[13] developed at Sandia National Laboratories, is a key component of the hardware testbed at the AVS Lab. UMBRA is a modular software environment based on C++ and Tcl/Tk with an integrated OpenGL graphics environment. Originally designed for large-scale software simulations of complex systems, it allows diverse behaviors such as vehicle dynamics, sensor modeling, communication infrastructures, vehicle path planning, *etc.*, to be encapsulated into compiled modules. These modules can then be

invoked at run-time and interact with other modules through flexible connection protocols. Its modularity makes it ideal for quickly and easily connecting various software modules and passing data between them.

The AVS Lab's use of UMBRA to control hardware devices and unite hardware and software simulation is a novel extension on the original capabilities of the software package. UMBRA modules have been created to encapsulate both the hardware and software components of the testbed. Examples include modules that interface with and control hardware devices like the UGV, PTU, or frame-grabber card, as well as modules that implement the visual snake processing algorithm, the visual servo control routine, and modules that calculate prescribed relative motion. It is noted, however, that UMBRA is not designed as a real-time operating system for hardware control, and its use as such presents some difficulties that must be accounted for.

One of the most valuable features of the UMBRA software environment is its ability to tie hardware and software simulations together seamlessly. Figure 2.2(b) illustrates a parallel development of the hardware relative motion testbed. As described in Reference 30, the testbed components, including the UGV, PTU, and camera, are all modeled with virtual software modules. This software simulation of all the components of the hardware testbed will enable researchers to first test sensing or control strategies on the simulated virtual devices before implementing them in hardware. However, because great effort has been made to keep the interface and behavior of the hardware and simulated devices identical, the transition from software to hardware is nearly seamless. As a result, the exact same

visual processing or visual servo module can control either the physical or simulated vehicle simply by redirecting the servo module output. Hybrid simulation scenarios are already being tested where real camera images are used to drive virtual vehicles. In the future, it will be possible to run simulations involving combinations of real and virtual devices interacting in real or simulated environments.

A final benefit of the UMBRA environment is that many complex models, such as vehicle dynamics or terrain models, have already been developed by other researchers. The modular nature of UMBRA allows existing code to be reused and incorporated into new simulations using both hardware and software quite easily. This collaboration among various research groups throughout the country allows for rapid advancement and assimilation of new ideas.

2.3 Simulation Hardware

2.3.1 Unmanned Ground Vehicle

The AVS Lab's relative navigation hardware testbed is based on the ActivMedia Robotics Pioneer 3-DX[17] mobile robotic ground vehicle, shown in Figure 2.3. It is a three-wheeled vehicle featuring two independently-driven, differential-steer drive wheels and a third caster for balance. This platform features a serial interface that can be commanded from an onboard PC-104 computer, or an externally mounted computer. Several types of sensors interface directly with the vehicle's microprocessor, and are then accessed through the serial

Figure 2.3: ActivMedia Robotics Pioneer 3-DX Robotic Ground Vehicle with a Digital Camera System on a PTU.

communication protocol. These sensors include encoders on each of the differential-steer drive wheels, bump sensors, sonar sensors (5-meter maximum range), and a digital 3-axis compass/roll/pitch magnetometer. Specific details about the capabilities, performance, and software interface of the P3-DX UGV are given in References 16 and 22.

The onboard PC-104 computer is shown in Figure 2.4. It is based on an 800 MHz Pentium III processor, and is currently equipped with a wireless ethernet card, a frame grabber card, and a Firewire card. The wireless LAN card allows the PC-104 to be networked with any wireless ethernet network. This allows for easy communication between the vehicle and a base station computer, and UMBRA modules have been created to transmit data between the UGV and a workstation. Almost every type of computer card is available in the PC-104

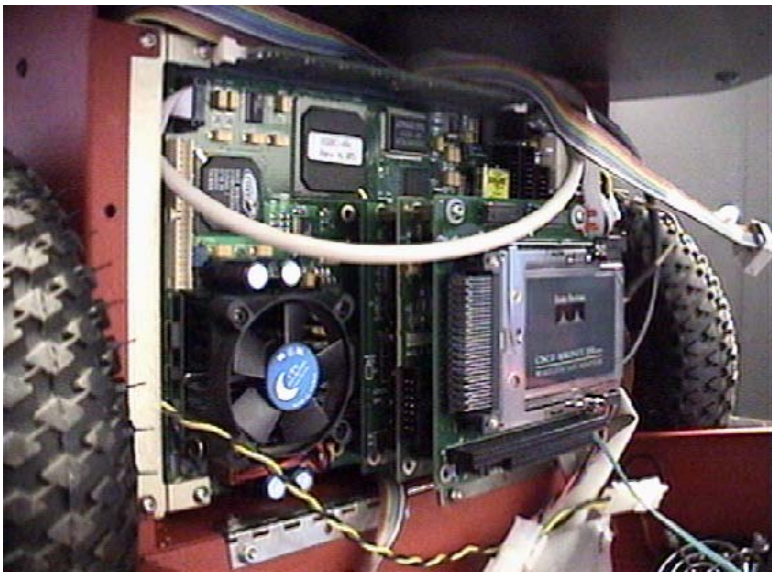


Figure 2.4: Onboard PC-104 Computer Inside the Pioneer 3-DX UGV.

form factor, and therefore many additional cards and capabilities could be added (such as a GPS receiver, a data acquisition card, etc.). The PC-104 communicates with the vehicle's control system serially, using the same protocol that an external computer would use.

An UMBRA module called "Pioneer" has been developed to interface with the vehicle and control its functionality. This module makes use of the ARIA vehicle control libraries provided by ActivMedia with the UGV, and interfaces directly with the low-level vehicle control microprocessor. Great effort was expended to develop the UMBRA module so that it could compile directly with the ARIA library without customization. In addition, the Umbra software simulation of the UGV, called "Pioneer_sim", uses a simulated version of the ARIA libraries, and the modules are input/output identical. This ensures that there is a consistent interface and repeatable behavior between the hardware and virtual software simulations of the UGV, and allows for direct transition between the hardware and simulated UGV

modules.

Sensing the actual inertial motion of the UGV is crucial if the vehicle is to simulate the motion of an aerospace vehicle. For example, spacecraft relative motion dynamics depend on the location of a spacecraft relative to a target. The onboard wheel encoders provide a reasonably accurate measure of the vehicle motion assuming it is operating on a surface where the wheels do not slip. For more precise measurements, a differential GPS receiver can be added to the UGV, but this would only operate outdoors. An alternative that can work both indoors and outdoors is the VISNAV system made by StarVision.[10] VISNAV uses a position sensitive photo-detector (PSD) to detect incoming light and determine a relative heading and range. Compared to using charge-coupled device (CCD) cameras at rates of only tens of Hertz, update rates of hundreds of Hertz are possible with a PSD-based device.

2.3.2 Pan-and-Tilt Unit

A Directed Perception PTU-D46[7] Pan-and-Tilt Unit (PTU), seen in Figure 2.5, is mounted to the vehicle. The PTU is controlled by an UMBRA module, and can be operated in either position or velocity modes. Using the PTU in conjunction with the UGV, the heading of the camera can be controlled independently of the direction of travel of the vehicle, which allows for the simulation of near-planar relative motion. For example, when performing rendezvous and docking operations in space, or performing general close-proximity operations, the orbit plane motion is strongly coupled. The hardware testbed described in this chapter is able to

Figure 2.5: PTU Mounted on the Pioneer 3-DX UGV.

reasonably mimic this motion with its two translational and independent rotational degrees of freedom. Future testbed expansions could add hardware that would rotate or lift the camera to provide additional relative motion simulation capabilities.

2.3.3 Visual Sensing Components

The visual sensor used in these relative navigation problems consists of the digital camera, the frame-grabber card, and the onboard PC-104 computer. UMBRA modules have been developed to interface with the frame-grabber card and pass the resulting images to the UMBRA visual snake module. This module processes the images using a recent real-time statistical pressure snake algorithm[35] described in detail in Chapter 3. Figure 2.6 shows the connections between the UMBRA hardware and software visual sensing components used to

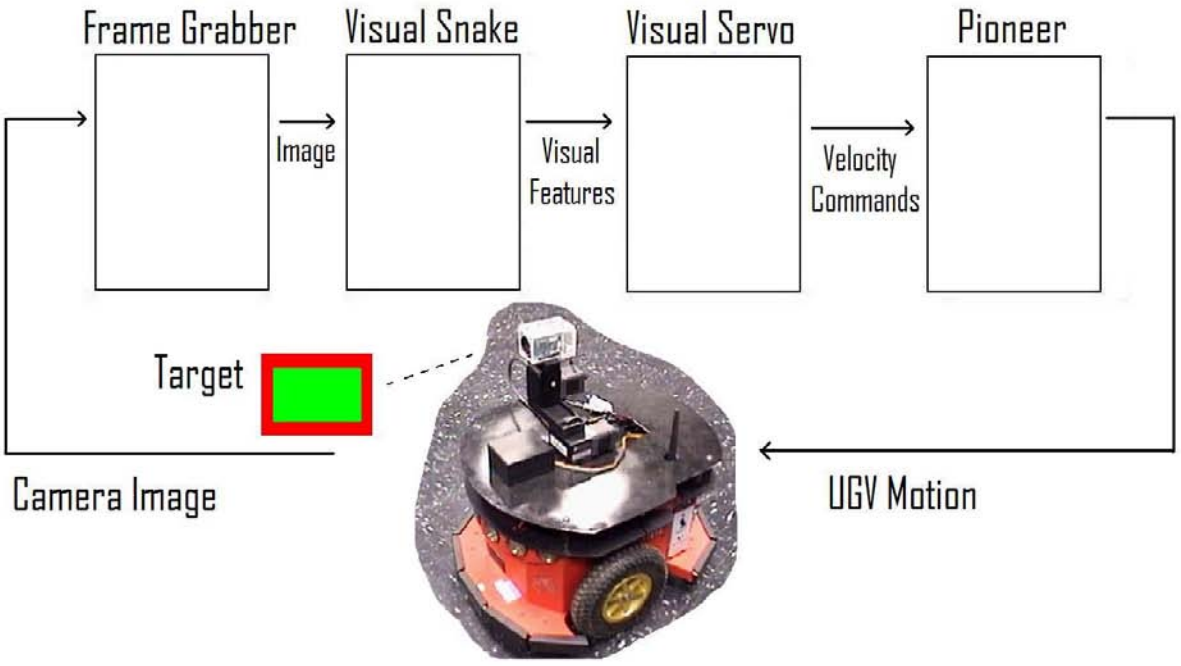


Figure 2.6: Block Diagram of UMBRA Visual Sensing Modules Used in Hardware Visual Servoing.

implement visual servoing behavior.

Chapter 3

Statistical Pressure Snake Algorithms

The hardware relative motion testbed presented in Chapter 2 is designed to be modular and generic, so that it is not limited to investigating only certain types of relative motion sensors. However, one of the primary goals for the testbed is to investigate visual sensing techniques based on statistical pressure snake algorithms for relative motion control. Statistical pressure snake algorithms, commonly referred to as visual snakes, are relatively recent additions the field of computer vision.[9,25,35] Generally speaking, visual snake algorithms form a closed contour around a target, which can be identified by many different properties, including color, intensity, or texture. This work only considers an existing color statistical pressure snake algorithm developed by Smith, Perrin, and Schaub[25,32] that is currently used in the Virginia Tech AVS Lab. An example of the color visual snake is shown in Figure 3.1. While the general theory of statistical pressure snakes is presented for completeness, the development of or improvements to this visual snake implementation are not the focus of

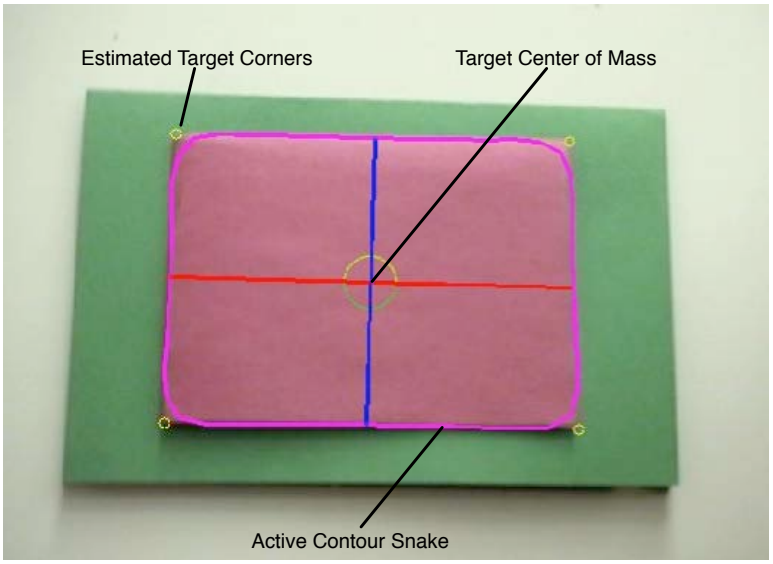


Figure 3.1: Color Statistical Pressure Snake Tracking a Visual Target.

this work. Instead, this work focuses on the performance of this particular implementation (Section 3.3), and its possible applications in relative motion problems (Chapters 4-7).

3.1 Statistical Pressure Snakes

Tracking a target in an image is a segmentation problem. Given the $N \times M$ pixels of the image, the task is to find a contour which will outline the desired target in each frame. How to perform this action in an unstructured environment where the lighting conditions can vary is a challenging problem that is actively being researched in the computer science imaging community.

In 1987 Kass *et al.* proposed the original active deformable model to track targets within

an image stream.[11] Also referred to as a visual snake, the parametric curve is of the form

$$S(u) = I(x(u), y(u))', \qquad u = [0, 1] \tag{3.1}$$

where I is the stored image. This curve is then placed into an image gradient-derived potential field and allowed to change its shape and position to minimize the energy E along the length of the curve $S(u)$. The energy function is expressed as:[11]

$$E = \int_0^1 \Big[E_{\text{int}}(S(u)) + E_{\text{img}}(S(u), I) \Big] \mathrm{d}u \tag{3.2}$$

where E_{int} is the internal energy defined as

$$E_{\text{int}} = \frac{\alpha}{2} \left| \frac{\partial}{\partial u} S(u) \right|^2 + \frac{\beta}{2} \left| \frac{\partial^2}{\partial u^2} S(u) \right|^2 \mathrm{d}u \tag{3.3}$$

and E_{img} is the image pressure function. The free weighting parameters α and β enforce tension and curvature requirements of the curve $S(u)$.

Active deformable models can be divided into two groups:[27] parametric models (snakes)[11,26] and level-set models (geometric contours).[18] The original Kass snake formulation is a parametric snake solution. However, it is difficult to tune and has several well-documented limitations. For example, the target contour tends to implode in the presence of weak gradients. While level-set models show excellent segmentation and robustness capabilities, they remain challenging to implement in real-time applications. Instead, the color visual snake considered here uses a modified parametric snake formulations proposed by Ivins and Porrill.[9] Here a pressure function is introduced that computes the statistical similarity of pixel values around a control point to create a pressure force that drives the snake towards the

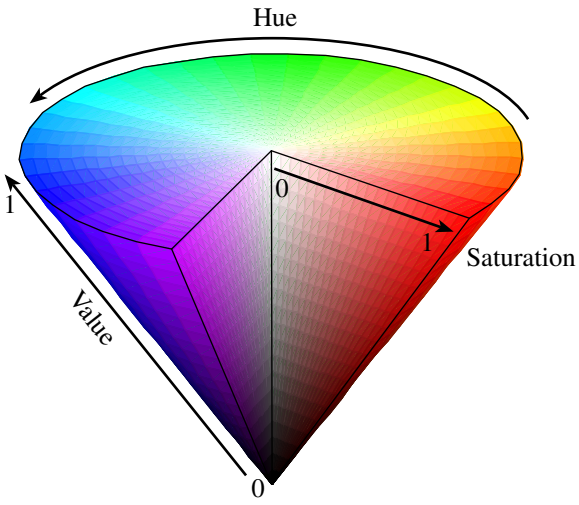


Figure 3.2: Conic Illustration of the Hue-Saturation-Value (HSV) Color Space.

target boundaries. The new energy function is given by

$$E = \int_0^1 \Big[E_{\text{int}}(S(u)) + E_{\text{pres}}(S(u)) \Big] \mathrm{d}u \tag{3.4}$$

where the pressure energy function E_{pres} is

$$E_{\text{pres}} = \rho \left(\partial S / \partial u \right)^{\perp} (\epsilon - 1) \tag{3.5}$$

and ϵ is the statistical error measure of the curve $S(u)$ covering the target. Perrin and Smith suggest replacing the E_{int} expression with a single term that maintains a constant third derivative.[26] This simplified formulation includes an even snake point spacing constraint. The resulting algorithm does not contain the difficult-to-tune tension and curvature force terms, yielding an easier to use and more efficient parametric snake algorithm.

Numerical efficiency is critical when using visual snakes to control an unmanned vehicle. A fast snake point cross-over check algorithm, which ensures that the snake forms a single

closed contour around the target, is implemented,and yields significant speed improvements for large sets of snake points.[42]

To provide additional robustness to lighting variations, Schaub and Smith proposed a new image error function:[32]

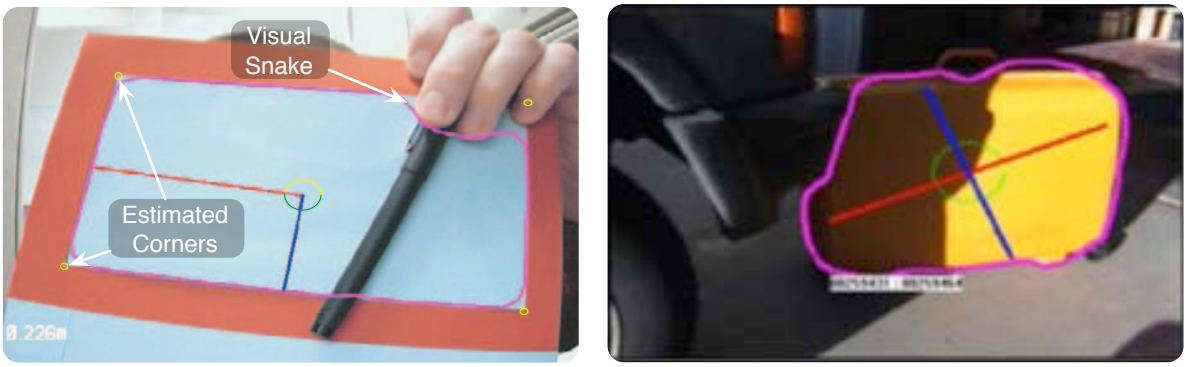
$$\epsilon = \sqrt{\left(\frac{p_1 - \tau_1}{k_1\sigma_1}\right)^2 + \left(\frac{p_2 - \tau_2}{k_2\sigma_2}\right)^2 + \left(\frac{p_3 - \tau_3}{k_3\sigma_3}\right)^2} \tag{3.6}$$

where p_i are local average pixel color channel values, τ_i are the target color channel values and σ_i are the target color channel standard deviations. The gains k_i are free to be chosen. The image RGB colors are mapped into the Hue-Saturation-Value color space illustrated in Figure 3.2. By choosing appropriate gains k_i, the visual snake can track targets with significant variations in target saturation and shading.

The implementation that is operational in the AVS Lab has several modes that utilize different gain settings. The "Hue Only" mode, for example, has small gains on hue but infinite gains on saturation and value. This mode is extremely robust to lighting variations, but less robust in unstructured environments where there may be background hues similar to the target hue. The "HSV" mode uses more evenly balanced gains, but is still most sensitive to hue. It is less robust to lighting variations, but more robust to non-unique target colors.

The resulting visual snake implementation is referred to as the color statistical pressure snake and is currently operational in the Virginia Tech AVS Lab. The promising tracking performance illustrated in Figure 3.3 has also been independently verified in Reference 37.

Figure 3.3 shows two image captures where the same visual snake algorithm is tracking



(a) Visual Snake Tracking a Partially Obscured Target and Estimating Corner Locations[36]

(b) Visual Snake Tracking a Yellow Suitcase Outdoors with Severe Lighting Variations[35]

Figure 3.3: Examples of the Visual Snake Tracking Different Targets.

different types of targets. In Figure 3.3(a) the operator selected the blue target square and the snake (purple outline) snaps to the visual boundaries. This tracking can be performed with up to 100 snake points at the full frame-grabber frame rate of 30 Hz on a workstation with a 2.8 GHz processor, and at 25-30 Hz with 50 snake points even on the relatively slow 800 MHz processor onboard the UGV. No specialized hardware or DSP chips are employed. Note that the visual snake routine is not confused by the image being partially obscured by a pen or fingers. A traditional corner or straight-line detection algorithm would have difficulties tracking this target. Further, additional target features such as target centroid location,[32] principal image axes size and angle, and even target corners[36] can be extracted from the visual snake contour. This is discussed further in Section 3.2.

An open research question involves efficient automatic snake gain parameter k_i selection. Gain selection is difficult because the HSV color space has singularities with near-gray-scale colors. The implemented automatic gain selection algorithm provides reasonable target se-

lection performance, but enhancements to the gain selection process are possible. While gain selection algorithm improvements are not the focus of this work, this area should be pursued in the future, and performance enhancements can be found. The current implementation is able to select targets when the user double-clicks on the computer screen. From this double-click, the gains and standard deviations k_i and σ_i in Equation (3.6) are automatically selected. This extremely simple and intuitive activation method is promising for applications where the end user is not extensively trained in computer vision, or in semi-autonomous applications.

Given a robust snake solution, the target contour can be exploited to yield more information than simply the target image centroid. Besides rapidly determining the target principal axis size and orientation,[31] the contour can also be used to determine corner points, as demonstrated in Figure 3.3(a).[36] These snake features can be exploited in filtering and estimation algorithms to determine additional relative motion information besides the typical target centroid heading.

This statistical pressure snake implementation has proved to be sufficiently fast and robust to control the motion of the UGV. However, as discussed in Chapter 2, it should be noted that the AVS Lab hardware testbed is not tied to the color statistical pressure snake algorithm presented here. As active deformable model algorithms mature and become more efficient, other segmentation routines can be used. Moreover, additional relative motion sensing techniques, such as differential GPS or VISNAV,[10] can be used in place of, or in conjunction with, this statistical pressure snake algorithm.

3.2 Extracting Target Image Features

The statistical pressure snake algorithm discussed above provides a parametric contour of the target image. However, the contour itself provides little useful information for the relative navigation sensing problem. More useful are image features like target area, center of mass (COM), and principal axis sizes and orientation. This section discusses how those features are extracted from the contour.

3.2.1 Moments of a Parametric Curve

To extract the desired image features, the area moments M_{ij} of the parameterized curve are computed. Given these moments, the target area, COM, and shape eigenfactors can be calculated. The snake is assumed to form a closed contour and is roughly outlining a shape as shown in Figure 3.1.

The ij-th area moments of the shape are defined as

$$M_{ij} = \iint x^i y^j \, \mathrm{d}x \, \mathrm{d}y \tag{3.7}$$

where the area integral is taken over the area defined by the closed snake curve. If the area A were defined through a set of pixel coordinates (x_n, y_m), then it would be trivial to compute the various area moments

$$M_{ij} = \sum_{n=1}^{N} \sum_{m=1}^{M} x_n^i y_m^j \tag{3.8}$$

In this case, however, the area A is defined through a parametric curve outlining this area.

Recall Green's theorem, which relates an area integral to a line integral through

$$\iint \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) \mathrm{d}x \, \mathrm{d}y = \oint P \mathrm{d}x + Q \mathrm{d}y \tag{3.9}$$

Using this theorem, the lengthy area integral is avoided, and the moments are calculated directly from the discrete points of the snake contour. Optimized formulas for computing the target moments are discussed and presented in Reference 44.

Note that the area A of the target shape is the 00-th area moment:

$$A = M_{00} = \iint \mathrm{d}x \, \mathrm{d}y \tag{3.10}$$

The center of mass (x_c, y_c) of the shape is computed using the first area moments M_{10} and M_{01}:

$$x_c = \frac{M_{10}}{A} = \frac{1}{A} \iint x \, \mathrm{d}x \, \mathrm{d}y \tag{3.11a}$$

$$y_c = \frac{M_{01}}{A} = \frac{1}{A} \iint y \, \mathrm{d}x \, \mathrm{d}y \tag{3.11b}$$

Because an area integral is used to compute the target COM, this computation is rather insensitive to minor deviations between the snake contour and the actual target contour. Noise and small errors along the snake almost cancel each other and yield a steady and accurate estimate of the COM.

3.2.2 Shape Eigenfactors

Given the target area and COM, we would like to determine the orientation of the target and its size. These features are computed by evaluating the 2^{nd} order moments. Here we

assume that the coordinate system origin has been translated to coincide with the computed target COM. Let the inertia-like matrix $[I]$ be defined as

$$[I] = \begin{bmatrix} M_{20} & M_{11} \\ M_{11} & M_{02} \end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \tag{3.12}$$

Note that $[I]$ is symmetric and positive definite. The principal orientation angle θ is the coordinate system rotation angle that yields a diagonal image inertia matrix $[I']$. Let $[C]$ be the direction cosine matrix (rotation matrix)[34] that maps the coordinate axis orientations between the two coordinate frames:

$$[C] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{3.13}$$

The rotation matrix is defined such that the coordinate transformation

$$[C][I][C]^T = [I'] \tag{3.14}$$

or

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} = \underbrace{\begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}}_{[I']} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{3.15}$$

results in a diagonal inertia matrix $[I']$. The inertias I_1 and I_2 of the diagonal matrix are called the principal inertias. The angle θ is defined here as the orientation of the I_1 inertia axis. Note that I_1 and I_2 are the eigenvalues of the matrix $[I]$, while the corresponding eigenvectors determine the principal inertia axis orientation. Since $[I]$ is symmetric and positive definite, the eigenvectors are guaranteed to be orthogonal.

The eigenvalues I_i of $[I]$ are computed as the roots of the quadratic equation

$$\det\left(\begin{bmatrix} I_{xx}-I_i & I_{xy} \\ I_{xy} & I_{yy}-I_i \end{bmatrix}\right) = (I_{xx}-I_i)(I_{yy}-I_i) - I_{xy}^2 = 0 \tag{3.16}$$

The principal inertias I_1 and I_2 are therefore given by:

$$I_1 = \frac{1}{2}\left(I_{xx}+I_{yy}+\sqrt{(I_{xx}-I_{yy})^2+4I_{xy}^2}\right) \tag{3.17a}$$

$$I_2 = \frac{1}{2}\left(I_{xx}+I_{yy}-\sqrt{(I_{xx}-I_{yy})^2+4I_{xy}^2}\right) \tag{3.17b}$$

Note that we define $I_1 \geq I_2$. Thus we can assume that I_1 is always the largest principal inertia of the target.

To determine the eigenvectors of $[I]$ we could solve the standard eigenvector problem. However, since we know that for a symmetric, positive definite matrix the eigenvectors are orthogonal unit vectors, we can simply solve for the angle θ in Eq. (3.15). Carrying out the matrix multiplications leads to the four equations

$$\cos\theta I_{xx} + \sin\theta I_{xy} = I_1\cos\theta \tag{3.18a}$$

$$\cos\theta I_{xy} + \sin\theta I_{yy} = I_1\sin\theta \tag{3.18b}$$

$$-\sin\theta I_{xx} + \cos\theta I_{xy} = -I_2\sin\theta \tag{3.18c}$$

$$-\sin\theta I_{xy} + \cos\theta I_{yy} = I_2\cos\theta \tag{3.18d}$$

Multiplying Eq. (3.18a) by $\sin\theta$, Eq. (3.18b) by $\cos\theta$, and subtracting one from the other leads to

$$\sin\theta\cos\theta(I_{xx}-I_{yy}) = (\cos^2\theta - \sin^2\theta)I_{xy} \tag{3.19}$$

This equation can be solved for the desired angle θ in terms of the given inertias I_{xx}, I_{yy} and I_{xy}:

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2I_{xy}}{I_{xx} - I_{yy}}\right) \tag{3.20}$$

When numerically evaluating the $\tan^{-1}()$ function, it is important to use the `atan2()` function which takes both the numerator and denominator as arguments, and returns the angle θ in the proper quadrant without singularities.

Given the principal axis orientation angle θ, the desired eigenvectors (principal axis unit direction vectors) are computed using:

$$\boldsymbol{v}_1 = \begin{pmatrix} \cos\theta \\ \\ \sin\theta \end{pmatrix} \tag{3.21a}$$

$$\boldsymbol{v}_2 = \begin{pmatrix} -\sin\theta \\ \\ \cos\theta \end{pmatrix} \tag{3.21b}$$

Note that the eigenvector directions are only unique to within a sign. Both θ and $\theta + \pi$ yield the proper eigenaxis orientation angle. The snake algorithm currently keeps track of the previous eigenaxis angle to ensure that switching between possible solutions does not occur.

3.2.3 Principal Axis Dimensions

Given the principal inertias I_1 and I_2 of the target, it is possible to estimate the size of the target if assumptions can be made about the target shape. For example, if the the target is assumed to be a rectangle with a half-height h and half-length l, the principal inertias I_i are

then related to the box dimensions h and l through

$$I_1 = \frac{A}{3} l^2 \tag{3.22}$$

$$I_2 = \frac{A}{3} h^2 \tag{3.23}$$

where $A = 4lh$ is the box area. Solving for the desired target dimensions,

$$l = \sqrt{\frac{3I_1}{A}} \tag{3.24}$$

$$h = \sqrt{\frac{3I_2}{A}} \tag{3.25}$$

Note that we assume that $l \geq h$ and that in practice, A is calculated as the 00-th area moment. If the true target shape is not a perfect rectangle, this routine approximates the equivalent box dimensions from the principal inertias. The computed principal axes, though not perfectly correct, still provide information about the target image size, which can be used to estimate the depth of the target. This information is important when tracking non-collaborative visual targets with unknown shapes, and is discussed in more depth in Chapters 6 and 7.

Figure 3.1 illustrates the rectangular box dimensions estimated from the principal inertias. The primary and secondary inertia axes are shown in red and blue, respectively. The target COM is highlighted with a green circle, and the yellow arc on the green circle illustrates the heading angle θ. Note that even though the snake rounds off the box corners, the box dimensions are estimated quite accurately. This computation is robust to small shape errors because the area integral is only secondarily affected by these snake-tracking errors. Performance of this snake algorithm is discussed in more detail in Section 3.3.

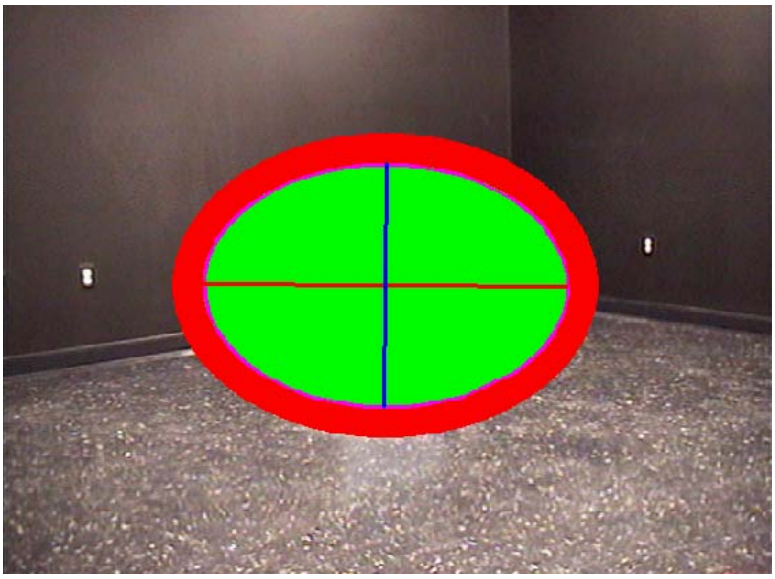


Figure 3.4: Visual Snake Tracking an Elliptical Target.

This approach for estimating the target principal axis sizes can be used with any pre-defined shape that is parameterized through two coordinates, as long as an analytical solution exists to extract the shape parameter from the principal inertias. For example, if the target shape is assumed to be an ellipse, the semi-major axis a and semi-minor axis b are computed as

$$a = \sqrt{\frac{4I_1}{A}} \tag{3.26}$$

$$b = \sqrt{\frac{4I_2}{A}} \tag{3.27}$$

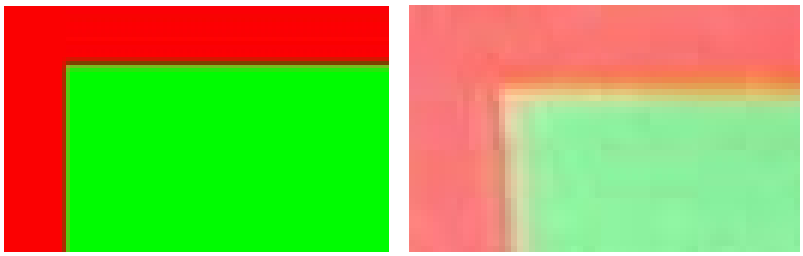
where $A = ab\pi$ is the ellipse area. An example of a visual snake tracking an elliptical target is seen in Figure 3.4. However, if the shape cannot be parameterized with two parameters (like a trapezoid), or an analytical mapping between the parameters and the principal inertias cannot be determined, this method breaks down.

3.3 Visual Snake Performance

This section discusses the performance of the visual snake algorithm as a relative navigation sensing technique. The accuracy of this sensing method is determined primarily by the accuracy of the target area, COM, and principal axis length measurements. We therefore seek to compare the measured values for these parameters with the true values. However, determining the true values in real world test conditions is extremely challenging. Moreover, due to issues related to target colors, pixelation at the target image boundary, and lens distortion specific to a particular camera/lens system, the performance would only be indicative of a particular test case, rather than the algorithm as a whole. We therefore confine this discussion to an ideal test case that shows the performance of the algorithm itself. This ideal test case represents an upper bound on performance of the snake algorithm as a visual sensor.

To construct the ideal test case, a "perfect" target of known size, shape, location, and pure color is drawn on the video image frame before processing with the visual snake. An example frame shot at high magnification is seen in Figure 3.5. Note the perfectly crisp color boundaries in the ideal test image, in contrast to the boundaries seen in an image taken with a real camera.

Performance data is taken for both a rectangular and an elliptical target of 5 different sizes. For each case, the visual snake is started 20 times and a total of 5000 image frames are captured. The transients associated with the snake first converging to the target are



(a) Ideal Target Image Corner (b) Camera Image Target Corner

Figure 3.5: Zoomed View of a Target Edge for an Ideal Test Image and a Camera Image.

removed, so the remaining data represents "steady-state" performance. The remaining data is averaged and plotted as a function of the target image principal axis size.

Figure 3.6 illustrates the advantages of tracking an elliptical target. The calculated X COM is far more repeatable for the ellipse than the rectangle because the snake is able to track the entire ellipse without rounding off the corners, as seen in Figures 3.1 and 3.4. More accurate COM measurements translate into better relative heading measurements. It is noted that a "bias-like" constant offset is seen in both the X and Y COM locations for both target shapes. This is an unexpected result, and the cause has not been definitively established. However, these "bias-like" errors are easily corrected. Results for the Y COM, not shown, show similar trends.

Figure 3.7 shows the calculated principal axis length, and confirms the superior performance associated with an elliptical target. Again, the results for the ellipse are far more repeatable, and the bias error is easily corrected. Since principal axis length is used to determine range to a target, more accurate principal axis lengths imply better depth measurements. Therefore, the visual snake provides more accurate heading and range measurements with an elliptical

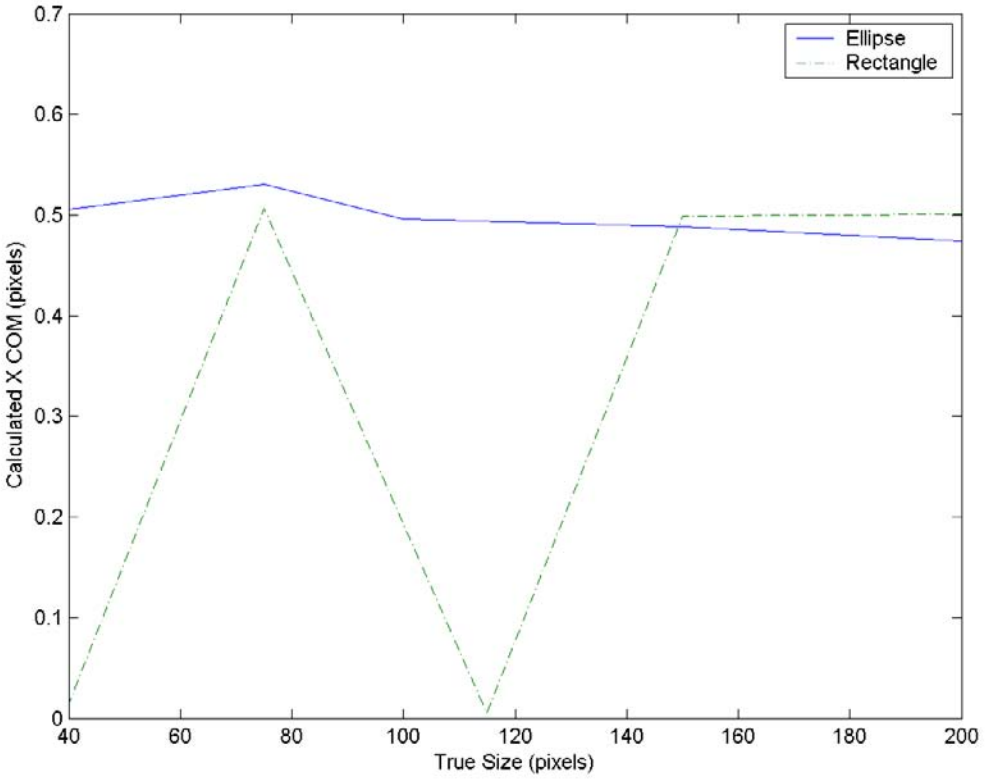


Figure 3.6: Calculated X COM Error from the Visual Snake Algorithm with an Ideal Test Image.

target than a rectangular target.

Figure 3.8 shows the X COM and principal axis length errors as a percentage of the true size. Both increase dramatically as the target image decreases in size, implying that the visual snake measurements are less accurate for smaller target images. It is therefore recommended that, wherever possible, the camera focal length and target image size be adjusted so that the target has a length of at least 100 pixels in the camera image.

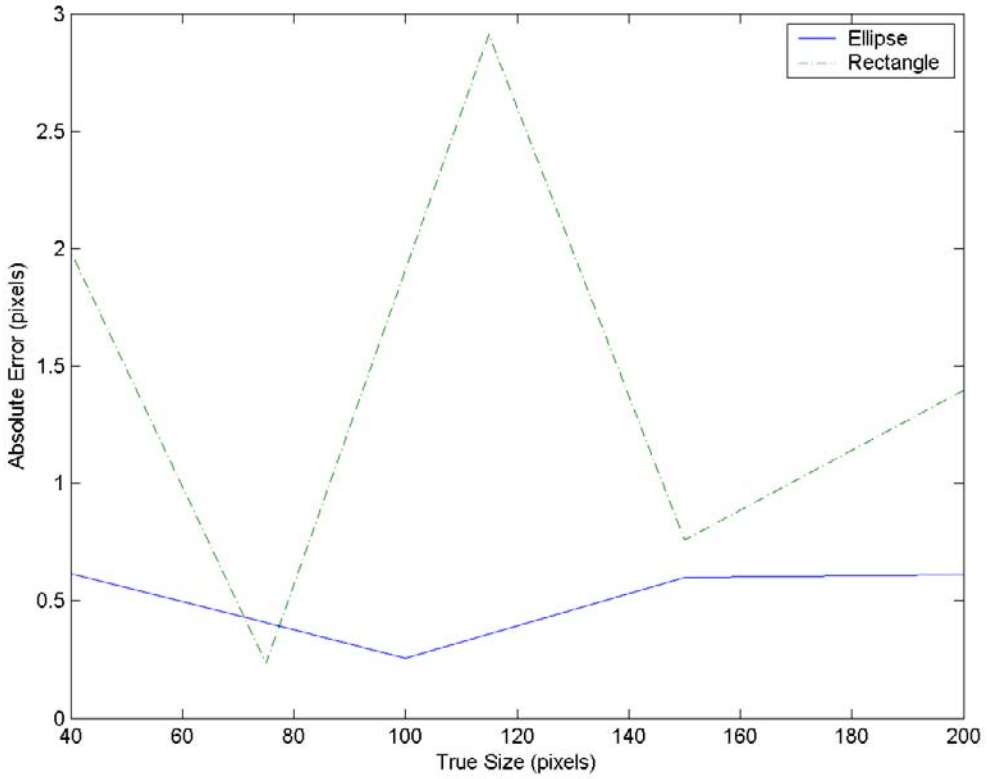


Figure 3.7: Principal Axis Length Error from the Visual Snake Algorithm with an Ideal Test Image.

In visual servoing applications where the vehicle attempts to maintain a fixed range to a constant-size target, the target image size should remain close to some nominal value. If the visual snake is "calibrated" about this nominal target image size, better performance can be obtained. This is discussed more in Chapter 7. Table 3.1 shows the performance for an elliptical target at an image size of 200 pixels. The bias errors are corrected so that the mean values match the true values for this image size. The values in Table 3.1 represent an upper-bound on the performance of this visual snake algorithm as a relative pose sensor.

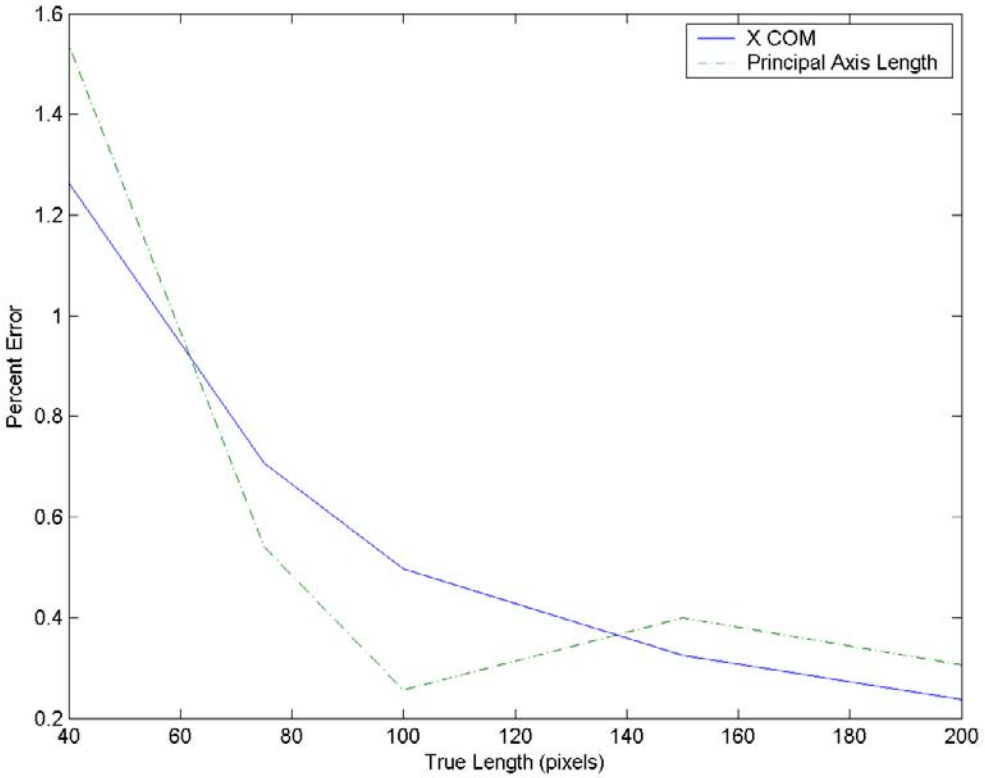


Figure 3.8: COM and Principal Axis Error Percentage for an Elliptical Target from the Visual Snake Algorithm with an Ideal Test Image.

Finally, the COM and principal axis length measurement errors resulting from the visual snake are approximately Gaussian, as seen in Figure 3.9. This implies that combining the visual snake with a Kalman filter might enhance the accuracy of the measurements. The application of estimation theory to this relative navigation sensing technique is an interesting area that should be explored further.

Table 3.1: Statistically Averaged Snake Performance for an Elliptical Target of Size 200 Pixels

Description	Pixels	Percentage
$\sigma_{\mathrm{COM_x}}$	0.1088	0.0544%
σ_{Length}	0.1347	0.0674%

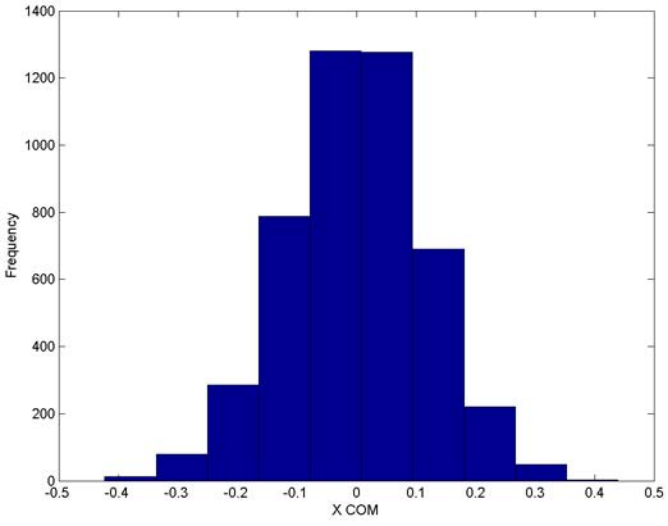


Figure 3.9: Histogram of X COM Measurement Error from the Visual Snake Algorithm with an Elliptical Ideal Test Image.

Chapter 4

Constant Gain Proportional-Integral Visual Servoing

Feedback control based on the visual snake output, or visual servoing, has an extremely broad range of applications, and most relative navigation problems can be formulated as a visual servoing problems. The AVS Lab hardware testbed UGV is currently capable of tracking and following a passive collaborative or non-collaborative target using the visual snakes discussed in Chapter 3. The operator first selects the target on the screen by double-clicking on it. The tracking control mode is then engaged and the vehicle attempts to maintain a fixed heading and distance relative to the target. A useful terrestrial application of this particular behavior is the "follower problem" where a UGV must follow another vehicle through visual servoing. However, as discussed in Chapter 2, many other complex visual servoing problems and behaviors pertaining to UAVs or spacecraft can be simulated as well.

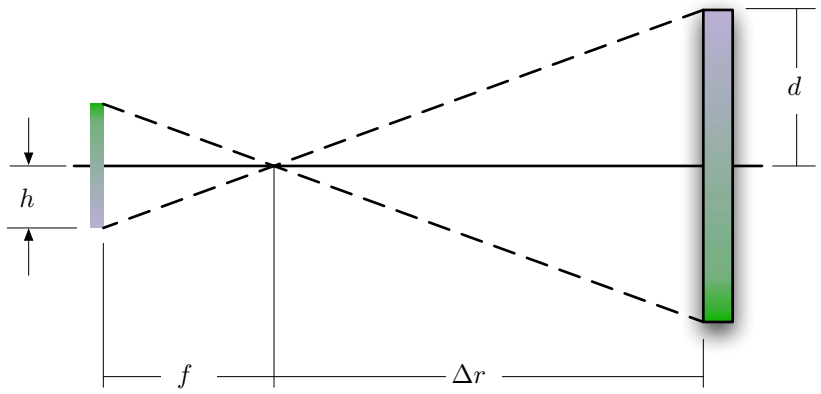


Figure 4.1: Simplified Illustration of the Pin-Hole Camera Model Used for Depth Extraction

4.1 Relative Heading and Range Determination

Given the target contour provided by the visual snake algorithm, the target area, center of mass (COM), and principal axes sizes are extracted using the methods discussed in Section 3.2. The target relative heading is determined from the COM location in the field of view. For the UGV problem with only one rotational degree of freedom, the target heading is computed using the horizontal target centroid coordinate x_c. The target range is computed indirectly from the size of the principal axes, which are calculated from the principal inertias I_1 and I_2 and the target area. In the UGV problem, the relative tilt angle between the vehicles is more constant than the relative heading angle. For example, as a target vehicle turns a corner, the apparent horizontal principal axis varies greatly. Therefore the vertical principal axis should be used to calculate range. More generally though, combinations of both principal axes can be used, depending on the application.

Figure 4.1 illustrates a simplified model of a pin-hole camera where f is the focal length and

Δr is the range to the target. Using the geometric law of similar triangles, we find that

$$\frac{h}{f} = \frac{d}{\Delta r} \tag{4.1}$$

must be satisfied. If the snake algorithm provides the principal axis image size h, then the range to the target is determined using

$$\Delta r(h) = \frac{df}{h} = \frac{1}{\alpha h} \tag{4.2}$$

where $\alpha = 1/(df)$ is visual calibration parameter for a particular camera focal length and target size. If d and f are not known *a priori*, α is determined through a simple one-click calibration procedure. By placing the target a known distance Δr_0 away from the camera and measuring the corresponding pixel dimension h_0, the calibration parameter α is found through

$$\alpha = \frac{1}{df} = \frac{1}{h_0 \Delta r_0} \tag{4.3}$$

This depth measurement strategy using a single camera is commonly used in the robotic vision community and has well-known limitations. The pin-hole camera model is sufficiently accurate about its calibration depth, but accuracy suffers at large deviations from the calibration point. In addition, if the target plane is not perpendicular to the camera bore-sight axis, the target has a perceived foreshortening, and the calculated target range is greater than the true value. However, this axis foreshortening appears only if higher terms are included, making this distance measurement relatively insensitive to target plane orientation misalignments.[31]

If the visual calibration parameter α is not known and cannot be determined through a calibration, the vehicle can still maintain the current separation distance. This mode has many practical applications when it is not necessary to maintain a specific separation distance. For example, an operator might command a UGV to follow a target and maintain the current range, even if that current range is unknown.

4.2 Proportional Feedback Control

Kinematic-based velocity steering laws are developed to control the motion of the UGV relative to the visual target. For the follower problem where the UGV tracks a target heading and range, the two target errors can be controlled individually. Thus, to simplify the following control discussion, x_d represents either a desired heading or separation distance target state. The state $\delta x = x - x_d$ is the visual target tracking error. The nonlinear velocity steering command $\dot{x}$ is

$$\dot{x} = -\gamma f(\delta x) \tag{4.4}$$

where $f()$ is an odd, smooth bounding function, $\gamma > 0$ is a constant feedback gain, and $f(\delta x)$ must satisfy

$$\lim_{\delta x \to \pm\infty} f(\delta x) = \pm c \tag{4.5a}$$

$$f(0) = 0 \tag{4.5b}$$

$$f(\delta x)\delta x > 0 \tag{4.5c}$$

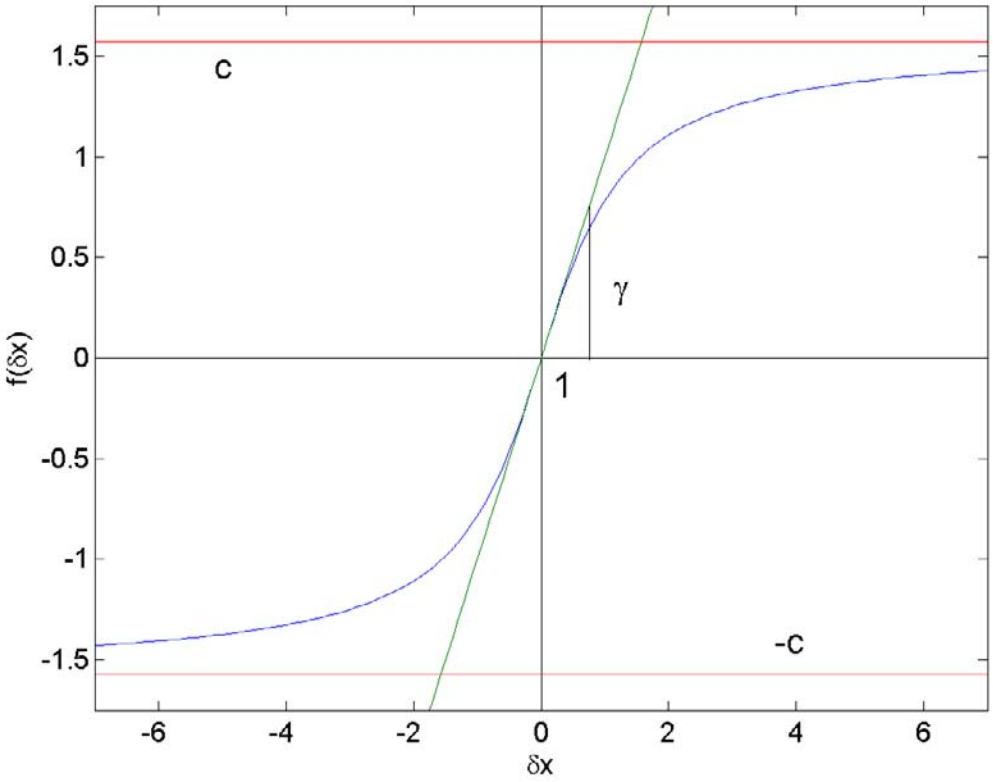


Figure 4.2: Non-linear Smoothly Saturating Control Law with $f(x) = \arctan(x)$

with c a finite constant. For example, $f(x) = \arctan(x)$ could be used, as is illustrated in Figure 4.2. The purpose of $f()$ is to smoothly bound the UGV speed. If the target state error δx is large, the servo should command a maximum response (saturated control). Once δx is reduced, the control will smoothly reduce the velocity command.

This simple velocity-based control law is essentially a smoothly saturated proportional feedback control. The two kinematic control law gains γ and c have simple interpretations. The gain γ, which controls the end-game convergence, should be set to provide the desired regulator stiffness. The constant parameter c, which determines the maximum commanded

speed, should be set such that γc is the maximum desired control command velocity.

This control strategy is very simple to implement. Compared to position-based visual servo control methods, which require extensive camera calibration before use, this velocity-based steering law only requires a simple calibration to determine the visual parameter α. Even without any calibration, this control is able to maintain an initial separation distance.

The purpose of this velocity-based visual control strategy is to regulate tracking errors δx and drive them to zero. This steering law is effective in regulating tracking errors relative to a stationary target. However, if the target is moving, asymptotic convergence does not occur from what is essentially a proportional feedback control.

To analyze stability of this steering law, let us define the candidate Lyapunov function

$$V(\delta x) = \frac{\delta x^2}{2} \tag{4.6}$$

If the target is stationary, then $\dot{x}_d = 0$ and the Lyapunov rate

$$\dot{V} = -\gamma \delta x f(\delta x) \leq 0 \tag{4.7}$$

is negative definite in δx. Thus, for inertially fixed targets, the visual servoing law in Eq. (4.4) provides asymptotic convergence for δx.

However, if the target is moving, the visual servo strategy in Eq. (4.4) will not yield asymptotic convergence. If the target moves with $\dot{x}_d = \text{constant}$, the closed-loop tracking dynamics are

$$\delta \dot{x} + \gamma f(\delta x) = -\dot{x}_d \tag{4.8}$$

With $\dot{x}_d$ constant, the $\dot{V}$ expression is no longer negative definite and the visual servoing control strategy is not asymptotically stable. For finite $\dot{x}_d$ values and assuming

$$\lim_{\delta x \to \infty} f(\delta x) > \dot{x}_d \tag{4.9}$$

$\dot{V}$ shows that the δx tracking errors remain finite. Because the tracking error δx is bounded, so is $V(\delta x)$. Thus, the integral of $\dot{V}$ from $t = 0 \rightarrow \infty$ exists and is bounded:

$$\int_0^\infty \dot{V} \mathrm{d}t = V(t_\infty) - V(t_0) = \text{ finite} \tag{4.10}$$

Barbalat's theorem[12] states that if Eq. (4.10) holds and $\dot{V}$ is uniformly continuous on $[0, \infty)$, then $\dot{V} \rightarrow 0$ as $t \rightarrow \infty$. To prove uniform continuity of $\dot{V}$ it is sufficient to show that $\ddot{V}$ is bounded.[41] Taking the derivative of $\dot{V}$ we find

$$\ddot{V} = (\gamma f(\delta x) + \dot{x}_d)\left(\gamma\left(f(\delta x) + f'(\delta x)\delta x\right) + \dot{x}_d\right) \tag{4.11}$$

with $f'(\delta x) \equiv \partial f / \partial(\delta x)$. Thus, $\ddot{V}$ is bounded if $f'(\delta x)$ is bounded. Using the $f()$ conditions in Eq. (4.5), the bound is satisfied and we conclude that $\dot{V} \rightarrow 0$ as $t \rightarrow \infty$.

The condition on $\dot{V}$ is satisfied by

$$\dot{V} = 0 \quad \begin{cases} \delta x = 0 \\ \\ \gamma f(\delta x) + \dot{x}_d = 0 \end{cases} \tag{4.12}$$

Looking at the closed loop tracking dynamics in Eq. (4.8), $\delta x = 0$ cannot remain true for a finite amount of time with a constant target rate $\dot{x}_d$. The steady-state tracking error is determined through

$$\gamma f(\delta x_{ss}) + \dot{x}_d = 0 \tag{4.13}$$

Solving for the steady-state tracking error due to a constant target motion $\dot{x}_d$, we find

$$\delta x_{ss} = -f^{-1}\left(\frac{\dot{x}_d}{\gamma}\right) \tag{4.14}$$

Thus, for constantly moving targets the tracking errors will have a finite value.

4.3 Proportional-Integral Feedback Control

Consider the scenario where the target is not stationary, but moving at a constant rate:

$$\ddot{x}_d = 0 \qquad \dot{x}_d = \text{constant}$$

We seek a modified version of the velocity-based visual servo law in Eq. (4.4) that asymptotically tracks a linear target motion without knowledge of the inertial motion of either the target or follower vehicles. The nonlinear tracking error feedback control is modified to include an integral feedback term through

$$\dot{x} = -\gamma f(\delta x) - \beta \int_0^t \delta x \, \mathrm{d}\tau \tag{4.15}$$

with $\beta > 0$. Note that the same smooth bounded function defined in Eq. (4.5) is used to bound the δx feedback term. Assuming a perfect servo system is implementing the commanded vehicle velocities, the vehicle acceleration is given by

$$\ddot{x} = -\beta \delta x - \gamma f'(\delta x)\delta \dot{x} \tag{4.16}$$

The tracking error acceleration equations are

$$\delta \ddot{x} = \ddot{x} - \ddot{x}_d = -\beta \delta x - \gamma f'(\delta x)\delta \dot{x} \tag{4.17}$$

To prove that the tracking errors δx asymptotically converge to zero for a constant target motion, a new candidate Lyapunov function V is defined as

$$V = \frac{\beta}{2}\delta x^2 + \frac{1}{2}\delta \dot{x}^2 \tag{4.18}$$

Using $\delta x = x - x_d$, the Lyapunov rate is expressed as

$$\dot{V} = -\gamma f'(\delta x)\delta \dot{x}^2 \leq 0 \tag{4.19}$$

Because $f(\delta x)$ is defined such that $f'(\delta x) > 0$ is bounded, the velocity steering law in Eq. (4.15) is globally stabilizing and $\delta \dot{x} \to 0$. To show that the tracking steering law is also asymptotically stabilizing with $\delta x \to 0$, the higher order derivatives of the Lyapunov function V are evaluated on the set where $\dot{V} = 0$. Clearly, $\dot{V} = 0$ if $\delta \dot{x} = 0$. The second derivative of V is

$$\ddot{V} = -\gamma f''(\delta x)\delta \dot{x}^3 - 2\gamma f'(\delta x)\delta \dot{x}\, \delta \ddot{x} \tag{4.20}$$

Because $f \in C^3$ and $\delta \ddot{x}$ is bounded we find that

$$\ddot{V}(\delta \dot{x} = 0) = 0 \tag{4.21}$$

Taking the third time derivative of V yields

$$\dddot{V} = -\gamma f'''(\delta x)\delta \dot{x}^4 - 5\gamma f''(\delta x)\delta \dot{x}^2 \delta \ddot{x} - 2\gamma f'(\delta x)\left(\delta \ddot{x}^2 + \delta \dot{x}\delta \dddot{x}\right) \tag{4.22}$$

Making use of $f \in C^3$, evaluating V at $\delta \dot{x} = 0$ results in

$$\dddot{V}(\delta \dot{x} = 0) = -2\gamma \beta^2 f'(\delta x)\delta x^2 \leq 0 \tag{4.23}$$

Because $\dddot{V}(\delta\dot{x} = 0)$ is negative definite with respect to the tracking error δx, and the first non-zero higher order derivative of V evaluated on $\delta\dot{x} = 0$ is of odd order, the control in Eq. (4.15) asymptotically stabilizes the tracking errors δx.

4.4 Hardware Visual Servoing Results

This section presents hardware visual tracking results that use the visual servo control strategy outlined in Sections 4.2 and 4.3. The scenarios presented could represent a UGV follower problem, or the free inertial motion of a floating robotic assistant spacecraft visually servoing on a mother spacecraft in interplanetary space.

Given the visual tracking errors calculated by the visual snake, the UMBRA servo module drives the vehicle. A block diagram of the UMBRA modules and their connections in these servoing experiments is seen in Figure 4.3. In spacecraft applications, it is assumed that an onboard IMU measures the actual inertial response and a velocity sub-servo system implements the velocity commands. In the future, IMUs could be added to the hardware testbed to measure the vehicle motion and close the velocity command sub-servo loop. This would simulate the required spacecraft motion sensing.

The visual target in these test cases is a green rectangle surrounded by a red border as seen in Figure 4.4. The visual calibration parameter α is determined at a range of 1 meter. Thus, the pin-hole camera-based range measurement will be most accurate around a range of 1 meter. This scenario is referred to as a collaborative, passive visual target. The visual

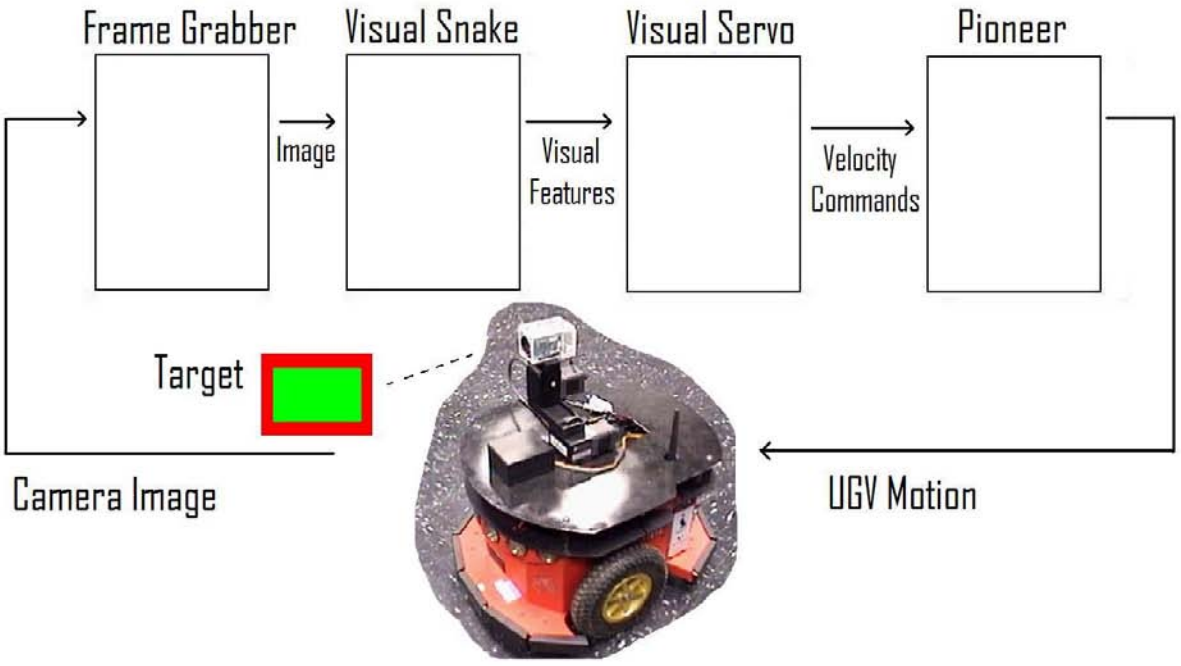


Figure 4.3: Block Diagram of UMBRA Modules Used in Hardware Visual Servoing.

target is collaborative because it has a known size and shape, but passive because no active communication occurs between the target and the visual sensor.

The visual snake automatic gain selection is set to only track hue colors. This mode provides the largest robustness to color intensity variation, but also makes the visual snake more susceptible to spilling over onto other green areas in the image background. The red border around the green target is provided to help contain the snake. If an arbitrary visual target is tracked, a blend of gains across the HSV space is used. In comparison to the "hue only" mode, the HSV mode is less easily fooled by other image components with similar hue, but more sensitive to lighting intensity variations.

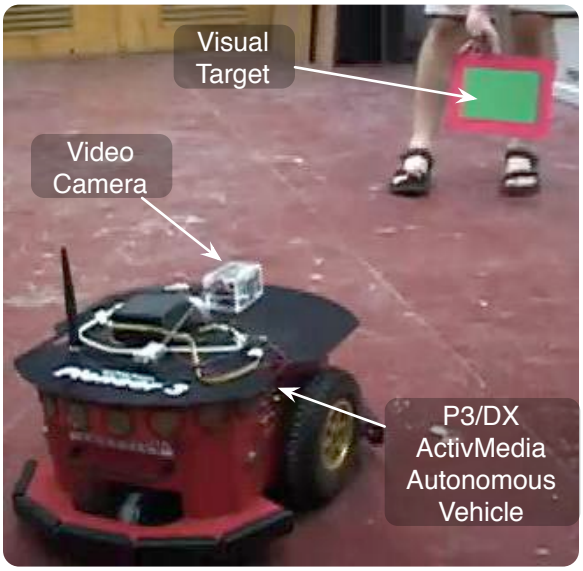
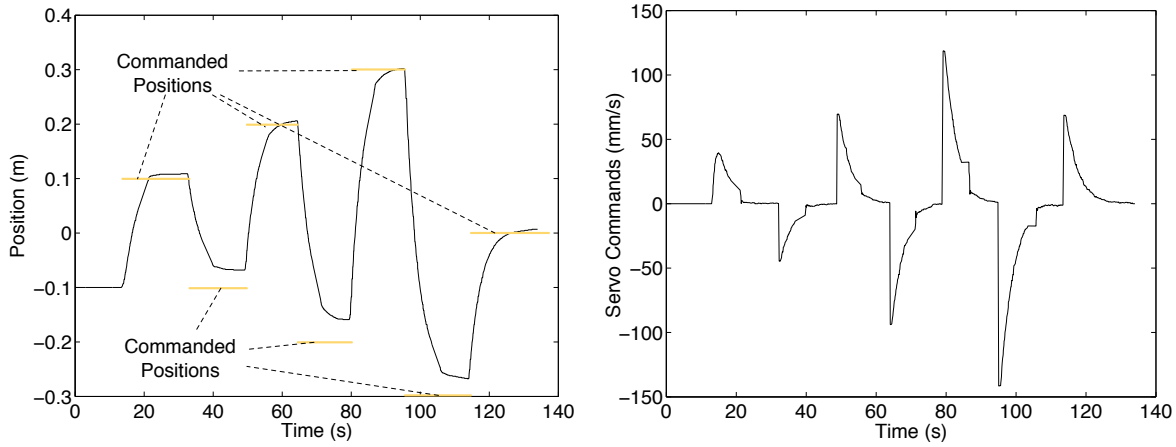


Figure 4.4: Illustration of the UGV Following a Rectangular Target Moved Manually in a Laboratory.

4.4.1 Fixed-Target Range Servoing Experiment

To test the range servoing capability of the proportional visual servo strategy outlined in Equation 4.4, the green target is attached to a wall and the visual parameter α is calibrated at a 1 meter range. After engaging the visual servo, the commanded target range is changed at discrete intervals. The vehicle moves forwards or backwards to achieve the desired range to the visual target.

Figure 4.5(a) shows the change in vehicle position relative to the nominal 1 meter separation distance, with positive changes indicating that the vehicle is moving forward. The vehicle motion is determined from the wheel encoders. Because the vehicle is performing straight-line motion on a non-slippery surface, the encoder position measurements are sufficiently



(a) Change in Actual Vehicle Position Relative to Calibration Position

(b) P3-DX Forward/Backward Servo Commands

Figure 4.5: Visual Range Servoing Results Where the Commanded Range Is Changed in Discrete Steps.

accurate.

The initial position is 1.1 meters, or -0.1 meters from the calibration distance. Then separation distances of 0.9, 1.1, 0.8, 1.2, 0.7, 1.3 and 1.0 meters are commanded. Each time the vehicle responds quickly and approaches the desired range. However, note that as the separation distances differ increasingly from the 1 meter calibration point, the measured vehicle separation distance is less accurate. Computing range using the principal axis is less accurate far from the calibration point because the pin-hole camera model is less valid. At the end of the experiment, the original calibration separation distance is commanded, and the vehicle converges to this desired range quite well.

The commanded vehicle servo speeds are shown in Figure 4.5(b). Note the discontinuities seen in the servo command signal within several of the re-positioning maneuvers (for example,

Table 4.1: Simulation Parameters for Proportional Range Regulation Experiment.

Parameter	Value
γ	1000 mm/sec
c	0.5

at 90 and 105 seconds). These are a result of visual snake measurement errors. It is noted that even in the presence of large discontinuous error signals, the control strategy shows quite reasonable behavior.

Table 4.1 shows the parameter values used in this experiment. The gain γ is selected to provide the desired regulator stiffness, and c is selected such that γc gives the desired maximum velocity command of 500 mm/sec.

This hardware test demonstrates that the visual servoing strategy can successfully track a calibrated separation distance. Separation distances other than the calibration distance can also be commanded, but with less accurate final positioning. In a follower problem scenario where a UGV must track a vehicle at a given separation distance, this visual servoing control shows much promise.

4.4.2 2D Visual Tracking of a Moving Target

In this hardware visual servoing experiment, the UGV tracks a visual target that is smoothly moved by a human through a laboratory as shown in Figure 4.4. The human mimics a lead

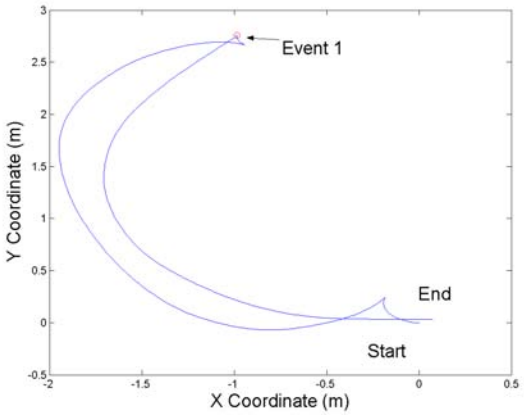
Table 4.2: Simulation Parameters for 2D Tracking Experiment of a Moving Target.

Parameter	Value
γ_{range}	800 mm/sec
c_{range}	0.625
β_{range}	250 $1/\text{sec}^2$
γ_{heading}	40 deg/sec
c_{heading}	1.5
β_{heading}	0.75 $1/\text{sec}^2$

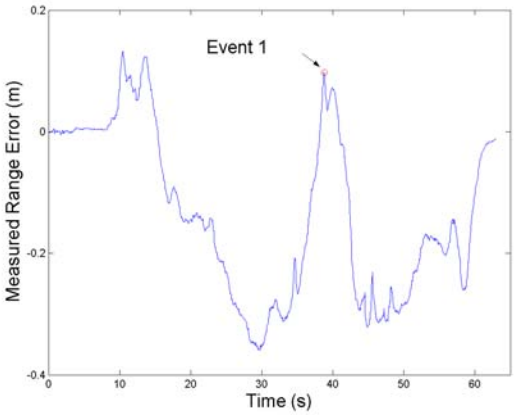
vehicle, and the UGV must maintain a 1-meter separation distance while pointing directly at the target. The visual servo control supplies both heading rate and translational speed commands to the UGV. If the target moves at a constant rate, the visual servo control law presented in Eq. (4.15) should drive the tracking errors to zero.

Experimental results are shown in Figure 4.6, and the parameter values used are seen in Table 4.2. The vehicle motion is again computed using the wheel encoders. Figure 4.6(a) shows the resulting trajectory. The range and heading tracking errors, as computed by the visual snake, are shown in Figures 4.6(b) and 4.6(c).

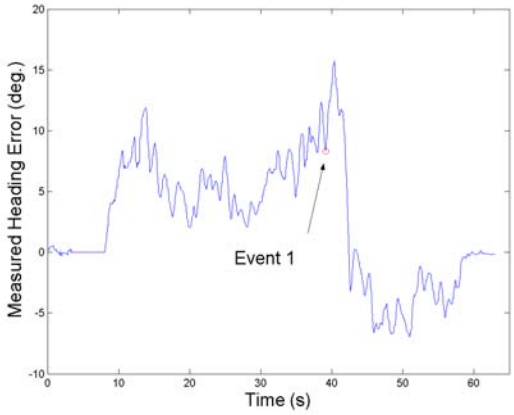
The start and stop points are marked on the trajectory plot in Figure 4.6(a), as well as a special event. At Event 1, the visual target is rapidly removed from the UGV's field of view. The visual snake therefore looses the target, and the visual servo routine automatically transitions into a stand-by mode. When the target is re-introduced into the UGV field of



(a) UGV (x, y) Trajectory



(b) Visual-Snake Computed Target Separation Distance Errors



(c) Visual-Snake Computed Target Heading Errors

Figure 4.6: Experimental Results from the 2D Tracking Test of a Randomly Moving Visual Target in a Visually Cluttered Laboratory Environment.

view, the visual snake reacquires the target and the servo control resumes. As the snake reacquires the target, the relatively slow computer onboard the UGV takes several frames to fully track the visual target, causing erroneous separation distance transients visible in Figure 4.6(b). This test illustrate the effectiveness and robustness of the current visual servo implementation in moving target tacking problems in unstructured environments.

4.5 UMBRA Visual Servoing Module Implementation

The visual servo control discussed above is implemented as an UMBRA module, and can be used to track a target with either the UGV or a pan-and-tilt unit. The control scheme is essentially that seen in Equation (4.15), but integral windup protection is added to improve performance on actual hardware in real-world test conditions. Integral windup protection limits the size of the integral term to ensure that it does not dominate the proportional term in the case of a quickly accelerating target.

The servo module's behavior is selected by the operator. The default mode points the UGV directly at the target and maintains a desired range. However, an alternate mode maintains any arbitrary, or the initial, relative heading between the target and the vehicle. This mode allows for following behavior reminiscent of ducks flying in a V-formation instead of simple straight-line following behavior. Likewise, range can be maintained at the current range instead of being driven to some pre-selected value. This option is particularly useful for non-collaborative targets or cases where the visual calibration parameter is unknown and

cannot be determined through a calibration procedure.

Chapter 5

Variable Gain Proportional Visual Servoing

In Chapter 4, constant gain visual servoing feedback control laws are developed. In many applications though, the ability to adjust the feedback gains could significantly improve the performance of the control. For example, if the visual snake pose measurements are very uncertain and noisy, decreasing the feedback gains would prevent the amplification of the measurement noise. This chapter presents a time-dependent-gain feedback control and shows results from the hardware testbed demonstrating the performance enhancements of this control strategy.

5.1 Proportional Feedback Control with Time Dependent Gain

Assume a vehicle is to move towards a visual target with piecewise-constant or slowly changing visual features, and stop at a prescribed separation distance Δr_r. Using the visual snakes as the tracking method, the target range is determined from the principal axis using the pinhole camera model described in Section 4.1.[31] Recall that the target range is computed from the image principal axis h using

$$\Delta r = \frac{df}{h} = \frac{1}{\alpha h} \tag{5.1}$$

where

$$\alpha = \frac{1}{df} \tag{5.2}$$

Let α be the true value for a given camera and target combination. Using a calibration routine, an estimated value $\hat{\alpha}$ is obtained. The corresponding estimated range is

$$\Delta \hat{r} = \frac{1}{\hat{\alpha} h} \tag{5.3}$$

Assume for now that α is a known constant. Let Δr_r be the desired distance between the camera and the target as shown in Figure 5.1. The target position relative to an inertial reference frame is r_T, and the inertial camera position is r. The current separation distance and rate between camera and target are

$$\Delta r = r_T - r > 0 \tag{5.4}$$

$$\Delta \dot{r} = \dot{r}_T - \dot{r} \tag{5.5}$$

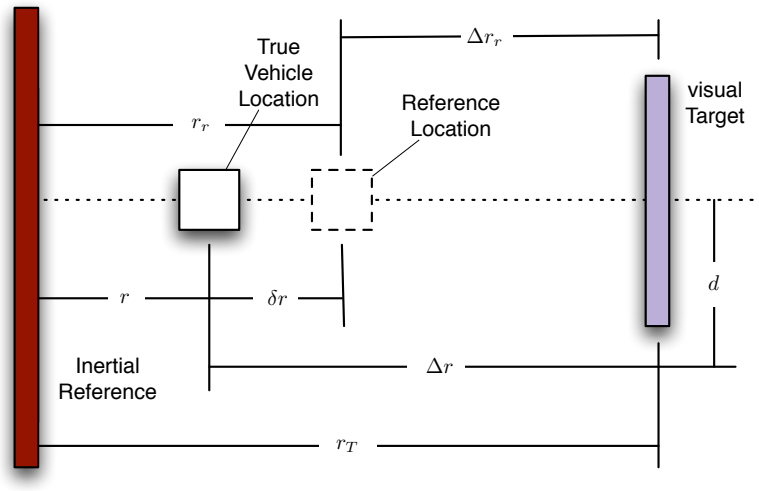


Figure 5.1: Illustration of Range Coordinates and Tracking Errors.

The range tracking error is defined as

$$\delta r = \Delta r_r - \Delta r \tag{5.6}$$

$$\delta \dot{r} = \Delta \dot{r}_r - \Delta \dot{r} \tag{5.7}$$

Thus δr is the visual target tracking error in the separation distance control. Building on Section 4.2 and Reference 33, the commanded vehicle velocity $\dot{r}$ is

$$\dot{r} = -\gamma(t) f(\delta r) \tag{5.8}$$

where $f()$ is an odd, smooth bounding function and $\gamma(t)$ is a time-varying feedback gain. The bounding function $f(\delta r)$ must satisfy the conditions in Equation (4.5) For example, $f(x) = \arctan(x)$ could be used. Recall that this non-linear steering law approximates a smoothly saturating proportional feedback control, and the gain $\gamma(t)$ controls the regulator stiffness. Note that the time-dependent gain $\gamma(t)$ makes the closed-loop dynamics time

dependent, and thus the system is non-autonomous. To analyze the stability of this kinematic steering law, let us define the candidate Lyapunov function

$$V(\delta r) = \frac{\delta r^2}{2} \tag{5.9}$$

Because this Lyapunov function does not explicitly depend on time, it is trivial to find time-independent functions $W_1(\delta r)$ and $W_2(\delta r)$ such that $W_1 \leq V \leq W_2$. Thus V is a positive definite function to study a non-autonomous system.

Assume that the target is inertially stationary and $\dot{r}_T = 0$. Taking the time derivative, the Lyapunov rate is

$$\dot{V} = \delta\dot{r}\delta r = -\gamma(t)\,\delta r\, f(\delta r) \tag{5.10}$$

To show that the steering law $\dot{r}$ in Eq. (5.8) is asymptotically stabilizing for a non-autonomous system, a positive definite function $W_3(\delta r)$ must be found such that[12]

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial r}\dot{r} \leq -W_3(\delta r) \tag{5.11}$$

The term $\partial V/\partial t$ is zero because $V(\delta r)$ does not depend on time. If the time-dependent feedback gain satisfies the lower-bound condition

$$\gamma(t) \geq \epsilon > 0 \tag{5.12}$$

with ϵ a small positive constant, then setting

$$W_3(\delta r) = \epsilon\,\delta r\, f(\delta r) \tag{5.13}$$

satisfies the condition in Eq. (5.11), and the proposed range steering law $\dot{r}$ is asymptotically stabilizing. The gain can vary with time, and convergence is guaranteed as long as the gain remains positive.

(a) Short Range Visual Servoing Case (b) Long Range Visual Servoing Case

Figure 5.2: Sample Screenshots from Short and Long Range Visual Servoing Tests

To implement this control, the true α parameter is not available. Using the estimated $\hat{\alpha}$ value, the steering law becomes

$$\dot{r} = -\gamma(t) f \left(\frac{1}{\hat{\alpha} h} - \Delta r_r \right) \tag{5.14}$$

If $\hat{\alpha} \neq \alpha$, the steering law is not asymptotically stable. The control is still stabilizing, but only the estimated tracking errors $\delta\hat{r} = \Delta r_r - \Delta\hat{r}$ will go to zero, not the actual tracking error δr.

5.2 Hardware Application of Time Dependent Feedback Gains

In Section 4.4.1, target distance servoing experiments at relatively close ranges are performed, and the UGV is seen to smoothly approach a new position and come to rest. However, during close range experiments, the visual target always is always of substantial size in the UGV

field of view, as seen in Figure 5.2(a).

Keeping the same camera focal length (not adjusting the camera zoom) and target size, the visual target image decreases in size as the commanded range increases, as seen in Equation (5.3). With a focal length of 4.2 mm and a target size of 22 cm, a range of 4.5 meters results in a target image size of approximately 15 pixels. This scenario is illustrated in Figure 5.2(b). At this long range, small errors in the principal axis measurement (Section 3.3) translate into large calculated range errors. If the same feedback gains used in close-range servoing are kept, the UGV erratically oscillates with high-frequency small-amplitude motion during long-range servoing tests. The resulting behavior is seen in Figure 5.3.

In many applications, precise high-bandwidth position control at a relatively long range of 4.5 meters is unnecessary, and a smoother positioning behavior is far more desirable. A modification to implement this desired behavior is to decrease the servoing gain at large commanded ranges by

$$\gamma(t) = \gamma_n \left(1 - \kappa_d \frac{\Delta r_r}{\Delta r_m}\right) \tag{5.15}$$

where γ_n is a nominal gain, Δr_m is a maximum commanded range, and κ_d controls the gain decrease with $0 < \kappa_d < \left(1 - \frac{\epsilon}{\gamma_n}\right)$ with ϵ a small positive constant. This bound on κ_d ensures the conditions on $\gamma(t)$ given in Equation (5.12) are satisfied.

As seen in Figure 5.3, this modification greatly reduces the erratic high-frequency motion, and the resulting behavior is much more desirable. However, this modification does not enhance the servoing position accuracy. If higher range precision at long ranges is required,

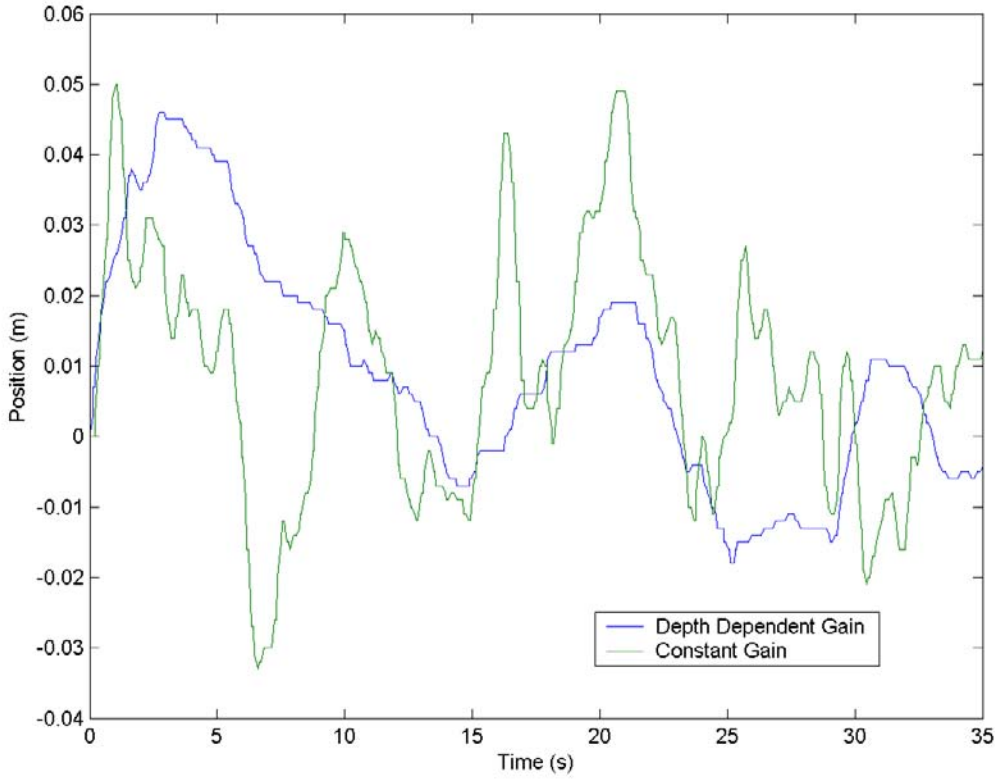


Figure 5.3: Position Data from Depth-Dependent and Constant Gain Long Range Visual Servoing Tests.

the target size or camera focal length should be changed so that the target image is of substantial size in field of view at the desired range. The parameters used in this experiment are seen in Table 5.1.

Table 5.1: Simulation Parameters for Depth-Dependent Gain Tests.

Parameter	Value
Δr_r	4.5 meters
Δr_m	4.5 meters
γ_n	1000 mm/sec
κ_d	0.75
c	0.5

Chapter 6

Visual Servoing with Parameter Estimation

Chapters 4 and 5 consider visual servoing with a collaborative target, in which the target size is known *a priori* or is determined with a simple off-line calibration procedure. However, non-collaborative targets present new challenges in the visual servoing problem. With a non-collaborative target, its size is unknown, and the visual target calibration parameter α cannot be determined through an off-line calibration. This chapter presents a Kalman filter-based method for estimating the visual calibration parameter on-line.

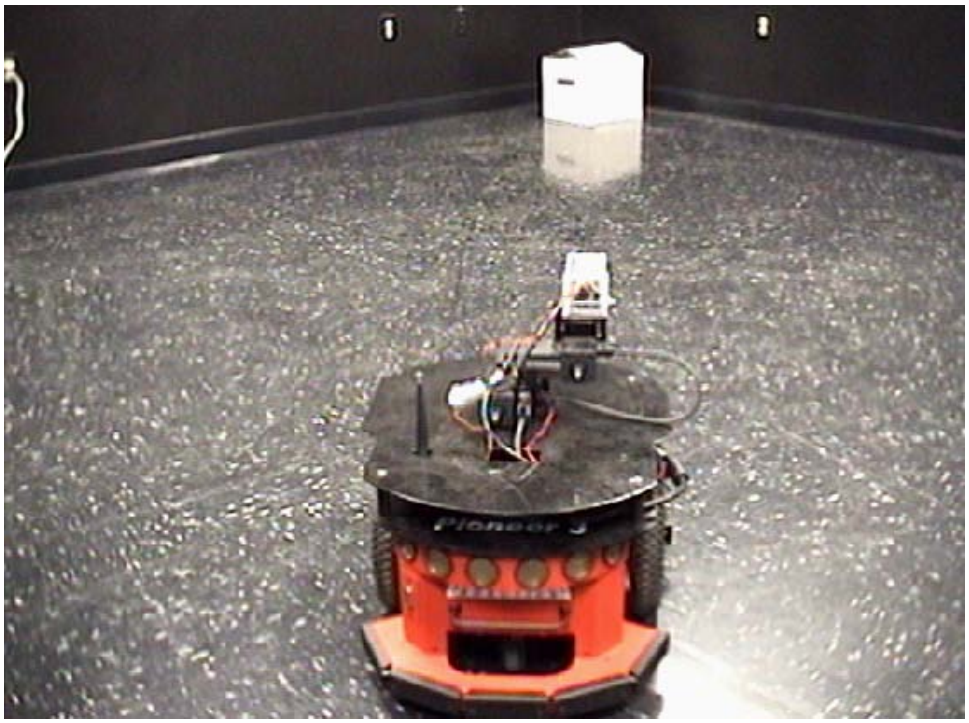


Figure 6.1: Illustration of a UGV Servoing on a Non-Collaborative Visual Target.

6.1 Parameter Estimation

If the target has an unknown or changing size (or the camera focal length is not precisely known), we wish to estimate the visual calibration parameter α on-line. This scenario, illustrated in Figure 6.1, could represent a visual servoing problem where a UGV is used to approach and investigate a suspicious object, or a spacecraft approaches an unknown visual target. The goal in this servoing problem is not to achieve high position accuracy, but to develop a robust strategy to approach an unknown target in an unstructured environment. Once the vehicle is in proximity to the target, a human operator provides further guidance. The craft is assumed to perform a straight-on approach to a target of unknown, but piecewise

constant or slowly varying physical size.

To estimate α, the Kalman update equation is used.[2] This is essentially a Kalman filter without parameter dynamics ($\dot{\alpha} = 0$) because α is assumed to be piecewise constant. Let $\boldsymbol{y}$ be the vector of M measured states, and $\boldsymbol{x}$ be the vector of N estimated parameters. The following notation is used. The $\hat{\boldsymbol{x}}$ symbol is used for estimated states, and hatless symbols $\boldsymbol{x}$ are the true states. The right subscript denotes the current discrete time value of the estimate, while the right superscript $\pm$ symbol indicates if a measurement correction has been applied (+) or not (-). Thus, $\hat{\boldsymbol{x}}_k^-$ is the estimate of the state vector $\boldsymbol{x}$ at time step k before any measurement corrections are applied.

To estimate the parameter α, with a slight abuse of notation we set

$$\boldsymbol{x} = \alpha \tag{6.1}$$

To estimate α using the Kalman update equation, α must appear linearly in the measurement vector. The target principal axis d is measured through the image length h. This measurement and α are related through

$$\frac{1}{h} = \alpha \Delta r \tag{6.2}$$

where the desired parameter α appears linearly. If the distance Δr between the vehicle and the target is known, this equation can be used as the measurement equation. However, the actual range to the target is an unknown state to be determined with the visual snake. Instead of using Eq. (6.2) directly, we write this equation at two different time steps k and

$k-1$:

$$\frac{1}{h_k} = \alpha \Delta r_k \tag{6.3a}$$

$$\frac{1}{h_{k-1}} = \alpha \Delta r_{k-1} \tag{6.3b}$$

Writing the M-dimensional measurement vector $\boldsymbol{y}_k$ as

$$\boldsymbol{y}_k = \frac{1}{h_k} - \frac{1}{h_{k-1}} = \alpha \left(\Delta r_k - \Delta r_{k-1} \right) \tag{6.4}$$

we no longer need the actual distance to the target, but the *distance traveled toward the target:* $\Delta r_k - \Delta r_{k-1}$. A similar technique for determining the visual calibration parameter by performing known motions is discussed in Reference 24, where it is applied to the visual 6 degree-of-freedom control of a manipulator end effector.

The distance traveled toward the target can be measured by wheel encoders or an IMU. For visual control of the unmanned ground vehicle in straight-line motion, the wheel encoders provide sufficient accuracy to effectively estimate the α parameter. The goal of this control is not to achieve high position accuracy, but rather, to achieve a robust autonomous strategy to approach an unknown target.

To use a Kalman filter to estimate α on-line, the following update equations are used. Let us define the $M \times N$ matrix $[H]$ as

$$[H_k] = [\Delta r_k - \Delta r_{k-1}] \tag{6.5}$$

The current observation vector is then

$$\boldsymbol{y}_k = [H_k] \boldsymbol{x}_k \tag{6.6}$$

Given a current state estimate $\hat{\boldsymbol{x}}_k^-$, applying the measurement update yields the corrected estimate:

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + [K_k]\left(\boldsymbol{y}_k - [H_k]\hat{\boldsymbol{x}}_k^-\right) \tag{6.7}$$

where $[K_k]$ is the current Kalman gain matrix

$$[K_k] = [P_k^-][H_{k]}^T\left([H_k][P_k^-][H_k]^T + [R_k]\right)^{-1} \tag{6.8}$$

The $N \times N$ matrix $[P_k^-]$ is the current, un-updated covariance matrix of the estimated states $\hat{\boldsymbol{x}}_k$, and the $M \times M$ $[R_k]$ is the covariance matrix of the measurement $\boldsymbol{y}_k$. To update the covariance matrix $[P_k]$ we use[2]

$$[P_k^+] = \Big([I_{N\times N}] - [K_k][H_k]\Big)[P_k^-] \tag{6.9}$$

In a Kalman filter, the states are updated between measurements using their differential equations. However, in this piecewise-constant parameter estimation problem, there are no state differential equations. The only quantity that must be updated between measurements is the covariance matrix. Let the $N \times N$ matrix $[Q_k]$ be the current process noise matrix. The covariance matrix $[P_k]$ is then updated using

$$[P_{k+1}^-] = [P_k^+] + [Q_k]\Delta t \tag{6.10}$$

where Δt is the time between measurement updates. With a non-zero $[Q_k]$, the uncertainty of the current estimate $[P_k]$ will increase over time without measurement updates.

6.2 Hardware Implementation

The following hardware implementation of the visual servoing strategy with parameter estimation is considered. The variable feedback gain $\gamma(t)$ presented in Chapter 5 is made dependent on the error covariance matrix such that a high covariance P (state estimation uncertainty) results in a low feedback gain. Thus, if the vehicle is uncertain about its α estimation, it proceeds "cautiously." As the confidence level of the $\hat{\alpha}$ estimate increases, so does the feedback gain and the vehicle responds more aggressively. The time dependent gain value is given by

$$\gamma(t) = \gamma_n \left(1 - \kappa_p \frac{P}{P_m}\right) \tag{6.11}$$

where P_m is a maximum allowable covariance value, γ_n is the nominal feedback gain, and $0 < \kappa_p < \left(1 - \frac{\epsilon}{\gamma_n}\right)$. Larger κ_p implies more conservative steering when the estimate uncertainty is large, which helps ensure that the UGV does not collide with a target because of a poor estimate of the visual calibration parameter. The gain modification presented in Section 5.2 can be combined with this gain formulation so that the gain depends on both the commanded servo range and the error covariance. Note that this $\gamma(t)$ definition satisfies the gain stability condition in Eq. (5.12).

To initialize the visual servo control, the $\hat{\alpha}$ covariance P is set to $P_{\max}$ because the visual calibration parameter is not known initially. Since the target size is unknown, the control cannot determine if the target is large and far away, or small and near. The parameter $\hat{\alpha}$ is

initially set to

$$\hat{\alpha} = \frac{5}{6}\frac{1}{\Delta r_r h} \tag{6.12}$$

so that the control's initial range estimation given by Eq. (5.3) is

$$\Delta\hat{r} = \frac{6}{5}\Delta r_r \tag{6.13}$$

This initialization value ensures that the control initially calculates its range to the target as 20% too large, and it begins to move toward the target cautiously.

On the first estimate update, Equation (6.4) is used to calculate $\hat{\alpha}$ directly instead of using the Kalman update equations. Although noisy, this first measurement is still significantly more accurate than the naive initialization value, and a rough estimate of the range to the target is obtained quickly. As the vehicle continues to move and the $\hat{\alpha}$ parameter is estimated with the Kalman filter, the vehicle accurately learns the visual calibration parameter and the resulting range to the the target.

As discussed in Section 3.3, the visual snake target feature measurements are more accurate when the target image is large. Therefore, the measurement covariance R_k is a function of the size of the target image so that the measurements are trusted less when the target image is small (long ranges).

Due to discretization errors with the encoders, the Kalman filter update routine is only called when the vehicle has traveled at least 2 cm since the last update. This greatly reduces the position measurement noise due to the encoders, and yields more accurate parameter estimates.

Finally, P and $\hat{\alpha}$ are bounded in software from growing too large. These bounds are particularly useful for creating a robust servoing strategy that performs well even if the control is started from a range inside the desired range or the visual target changes size while the UGV is approaching. This is discussed further in Sections 6.3.2 and 6.3.3.

6.3 Hardware Visual Parameter Estimation Results

Several hardware visual parameter estimation test cases are considered. These cases involve approaching a target that may change in size from various initial ranges. However, in all cases, the same parameters are used, and are shown in Table 6.1. The ability to achieve reasonable performance while using the same parameters in widely varying test cases shows the performance robustness of the this sensing and control technique.

6.3.1 Long Range Test

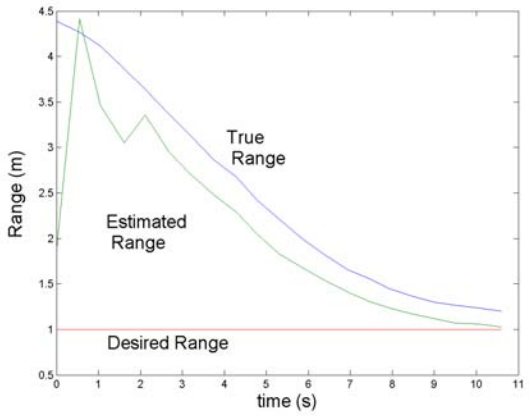
In this case, the vehicle starts from a range of 4.4 meters and approaches a target that remains a constant size throughout the test. Figure 6.2 illustrates the results of this test case. Because of the initialization of the the unknown $\hat{\alpha}$ calibration parameter and the error covariance matrix, the initial feedback errors and feedback gains are small, and the UGV approaches the target slowly and cautiously at first. As the vehicle begins to move, the calibration parameter estimate and calculated range are determined more accurately, resulting in large feedback errors and a maximum saturated response. In addition, the

Table 6.1: Simulation Parameters for Hardware Visual Parameter Estimation Tests.

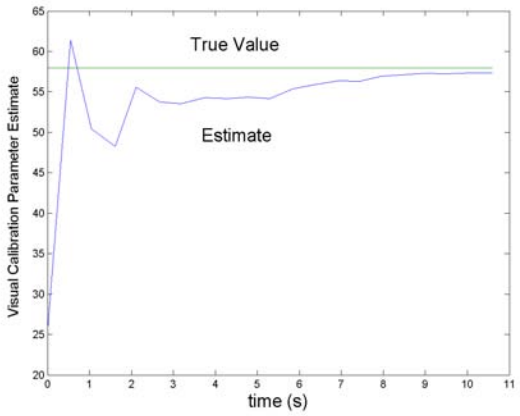
Description	Value
Δr_r	1.00 m
$P_{\max}$	30.0 m^{-4}
$\alpha_{\max}$	200.0 m^{-2}
Q_k	0.05 $\text{s}^{-1}\text{m}^{-4}$
R_k	2.0 m^{-2}
κ_p	0.75
γ_n	1000 mm/sec
c	0.5

covariance of the estimate decreases, providing a larger feedback gain. As a result, the vehicle quickly approaches the target at the maximum desired velocity until it gets much closer to the desired range. The resulting error dynamics after a few seconds of motion resemble an exponential decay.

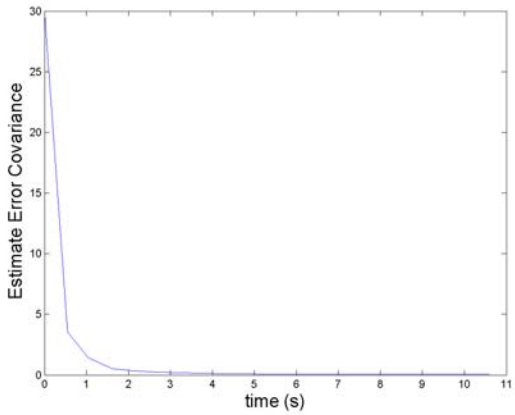
Note that $\hat{\alpha} \not\to \alpha$ as $t \to \infty$. This estimation technique does not provide any parameter convergence guarantees, and the final range tracking errors do not converge to zero. But the $\hat{\alpha}$ estimate is close to the actual α, and the tracking error is small. This servo control with automatic $\hat{\alpha}$ initialization provides the desired robust strategy, and the vehicle is able to approach an unknown target and identify the visual calibration parameter on-line and stop near the desired separation distance.



(a) Vehicle Range



(b) Estimated Parameter



(c) State Covariance Estimation

Figure 6.2: Long Range Visual Servoing Maneuver with Parameter Estimation.

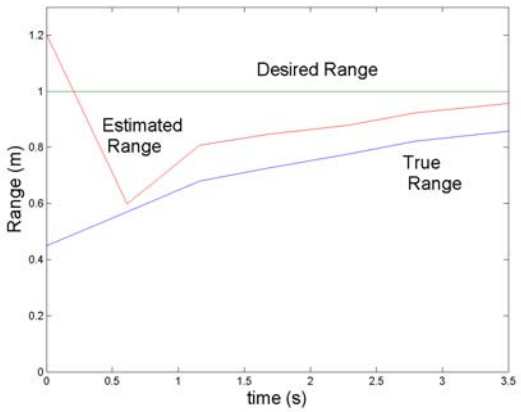
6.3.2 Close Range Test

In this case, the vehicle starts at a range of 0.45 meters, which is closer to the target than the desired 1.0 meter range. The parameter $\hat{\alpha}$ is initialized so that the control calculates a range of 1.2 meters, but the required $\hat{\alpha}$ to calculate this +20% range error is greater than the imposed $\hat{\alpha}_{max}$ of 200. With $\hat{\alpha}_{max}$, the control calculates a range inside the desrired range, and the vehicle initially moves backwards, as illustrated in Figure 6.3. This bounded control strategy is robust, and prevents the UGV from moving forward and colliding with the target if the servo is engaged at a very close range.

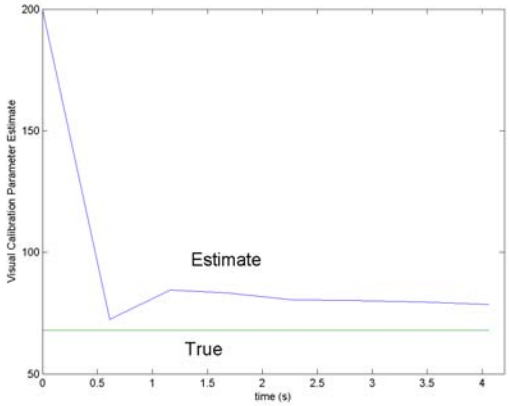
While $\hat{\alpha}$ approaches α, there is significantly more steady state range error in this case than in the long-range test case. For small vehicle motion, the visual calibration parameter is weakly observable, and the system is unable to obtain a good estimate. Future work could investigate the use of persistent motion excitation to improve the parameter estimate. However, even in this challenging case, the control results in reasonable vehicle motions with the desired conservative behavior.

6.3.3 Changing Target Size

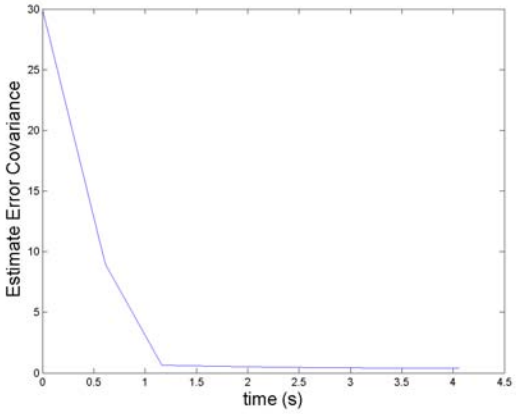
In this case, the vehicle starts at a range of 4.3 meters, but there is a step-change in target size during the test. As seen in Figure 6.4, the general behavior in both of the fixed-target-size segments matches that seen in the simpler case described in Section 6.3.1. At 2.7 seconds, the target area decreases by 45%. This drastic scenario might result from part of the target



(a) Vehicle Range



(b) Estimated Parameter



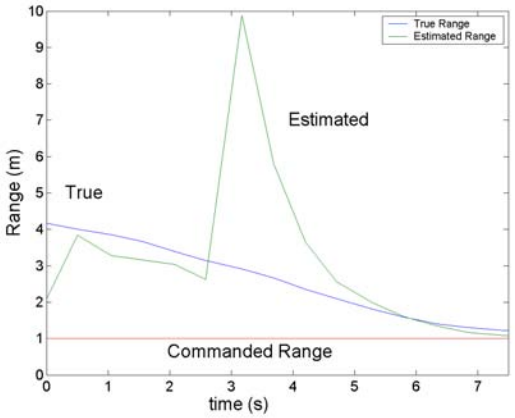
(c) State Covariance Estimation

Figure 6.3: Close-Range Visual Servoing Maneuver with Visual Calibration Parameter Estimation.

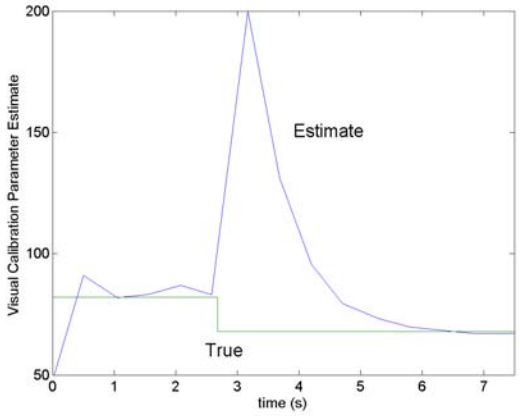
being obscured during the approach, or the target being rotated and projecting a smaller area.

Without estimation of the visual calibration parameter, the control calculates that the range to the target has drastically increased. As a result, the vehicle would move forward aggressively and could collide with the target. However, with the $\hat{\alpha}$ estimation, the control is able to tune its visual calibration parameter estimate in real time during the maneuver. The $\hat{\alpha}_{\max}$ bound is active for one estimate and the craft does accelerate slightly as the target size is changed. However, the craft also recomputes its $\hat{\alpha}$ estimate, which ensures that it does not run into the target.

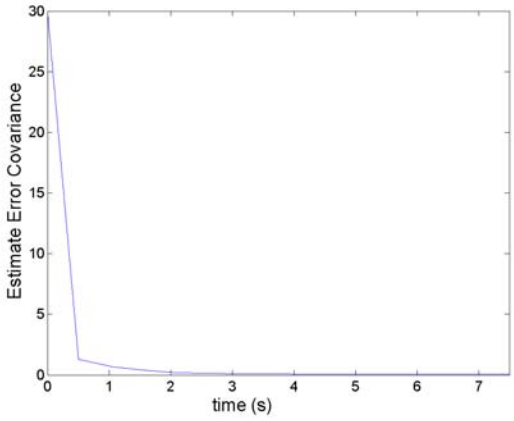
Similar to the results seen in the Section 6.3.2, the $\hat{\alpha}$ estimate does not converge to the true value due to the short range left to maneuver after the change in target size. However, the final range error is acceptable, and it is conservatively farther away from the target, rather than closer. Again, this control does not guarantee convergence of $\hat{\alpha}$. However, it does illustrate sophisticated visual servoing capabilities on unknown, non-collaborative targets in real-world applications where robust, conservative behavior is more important than final positioning accuracy.



(a) Vehicle Range



(b) Estimated Parameter



(c) State Covariance Estimation

Figure 6.4: Changing Target Size Visual Servoing Maneuver with Parameter Estimation.

Chapter 7

Visual Sensing for Aerial Refueling

In this chapter, the use of visual sensing as part of an automated aerial refueling system is considered. Aerial refueling is vital for the operation of the United States military, and as more unmanned aerial vehicle (UAV) systems become operational, automated aerial refueling systems will become increasingly important.

There are currently two approaches used for aerial refueling. In the probe-and-drogue refueling method, illustrated in Figure 7.1, the tanker trails a hose with a flexible, aerodynamically stabilized basket, called a drogue, at the end. The pilot of the receiver aircraft is responsible for maneuvering the probe into the drogue. This method is used for small, agile aircraft, and a human operator is not required on the tanker.

The United States Air Force uses the flying boom refueling method seen in Figure 7.2. The boom has aerodynamic control surfaces and is precisely "flown" into the receiver aircraft's

Figure 7.1: Illustration of Probe-and-Drogue Aerial Refueling Method

receptacle by a human operator onboard the tanker aircraft. The receiver aircraft must simply maintain a nominal reference position, and the precision control is left to the tanker operator. Due to the mechanical tolerances of the mating hardware, position errors in the three translational directions must be below 2 cm.[45]

Because the receiver aircraft is not required to perform precise relative navigation, the flying boom method is particularly attractive for the refueling of UAVs. In addition, it is desired that an automated control system replace the human operator onboard the tanker. However, an automated system would require improved relative navigation sensing techniques.

This chapter investigates the use of visual snakes as the relative navigation sensor for an automated aerial refueling system. Developing such a system is an extremely challenging and complex undertaking, and the work presented here is only a preliminary analysis of the use of visual snakes for this relative motion problem.

Figure 7.2: Flying Boom Aerial Refueling Method

7.1 Forced Perspective Target Setup

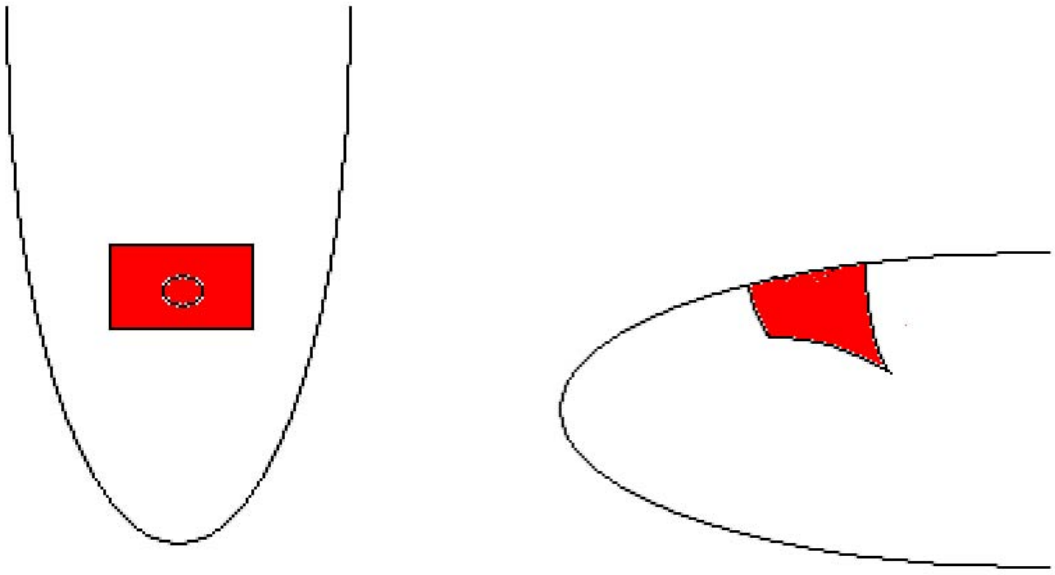
To use visual snakes as part of an aerial refueling system, a camera and a visual target must be placed on the tanker and receiver aircraft, respectively. It is initially suggested that the camera be placed in a similar position and orientation to that from which Figure 7.2 is taken. The visual target should be placed as close as possible to the receiver aircraft receptacle, which greatly reduces any position errors that might be introduced by the inability of the visual snake sensor to measure the full 3 DOF orientation of the receiver aircraft.

As discussed in Chapter 4, the target image COM location is used to determine the relative heading to the target, and the principal axis sizes are used to determine range. From these measurements, the relative position of the receptacle is determined. For particular target

shapes, the principal axis sizes can be determined from the target image moments. However, this only holds for target shapes parameterizable by two measurements and for which there is an analytical relationship between the those parameters and the moments. Examples include a rectangle, which is parameterized by its length and width, and an ellipse, parameterized by its semi-major and semi-minor axes. For an arbitrary target shape however, the relationship cannot be determined. Therefore, the target image should appear as a rectangle or an ellipse in the camera image plane.

However in general, the camera image plane is not parallel to the plane on which the visual target is drawn, which means that the target image appears skewed in the camera plane. For example, a rectangle painted on the aircraft appears as a trapezoid in the camera image plane. Moreover, it is not guaranteed that a planar surface can be found in proximity to the refueling receptacle. Therefore, simply painting a visual target of the desired shape on the aircraft is not a feasible solution.

To make the target image, which is painted on a curved surface, appear as a desired shape in the camera image plane, we suggest using forced perspective. This technique, often employed by artists, consists of painting the target image so that it appears "correct" from some desired viewing position and orientation. This is illustrated in Figure 7.3. It is noted that the image is only correct when viewed from the nominal pose, and it appears skewed when viewed from any other pose. However, in this aerial refueling application, this is not a significant problems, because the aerial refueling operation can only take place when the aircraft are at or very near their nominal positions. The visual snake measurement errors caused by slight



(a) Visual Target as Viewed by the Tanker Aircraft (b) Visual Target as Painted on the Receiver Aircraft

Figure 7.3: Illustration of Forced Perspective Showing Visual Targets as Seen by the Tanker and as Painted on the Receiver.

deviations from the nominal relative pose between the aircraft are analyzed and discussed in Section 7.2.

To find the shape that must be painted on the target to produce the desired camera image plane shape, rays are projected from the desired image shape on the camera plane through the focal point. The intersection of those rays and the receiver aircraft surface generates the contour that appears as the desired shape in the camera image plane.

7.2 Sensitivity Analysis

As discussed in the previous section, the use of forced perspective implies that the target image is only the "correct" shape when the relative pose between the aircraft is the nominal pose. Perturbations from the nominal pose skew the target image shape, and the resulting moments calculated from the snake contour change. The relative COM heading and range calculations are therefore corrupted when there are perturbations from the nominal pose.

A numerical simulation designed to identify the error between the visual snake-measured and true relative headings and ranges is developed. This simulation assumes that the visual target is coincident with the refueling receptacle. For this analysis, the visual snake is assumed to track the target perfectly. The calculated errors are due to the method of extracting the relative heading and range from a contour, not the visual snake tracking errors. Using this simulation, the sensitivity of the relative heading and range errors to small perturbations about the nominal position and orientation of the receiver aircraft are determined with finite-difference derivatives.

Tables 7.1 and 7.2 show the error sensitivity to position and orientation perturbations, respectively. Standard aircraft coordinate systems (X forward, Y towards the right wing, Z down as illustrated in Figure 7.4) and 3-2-1 Euler Angles are used. The nominal range between the camera and the visual target is 10.7 m. Because the visual snake measurement error is not included, these values are the sensitivity of the algorithm itself, and represent an upper bound on the performance of the entire visual sensing method.

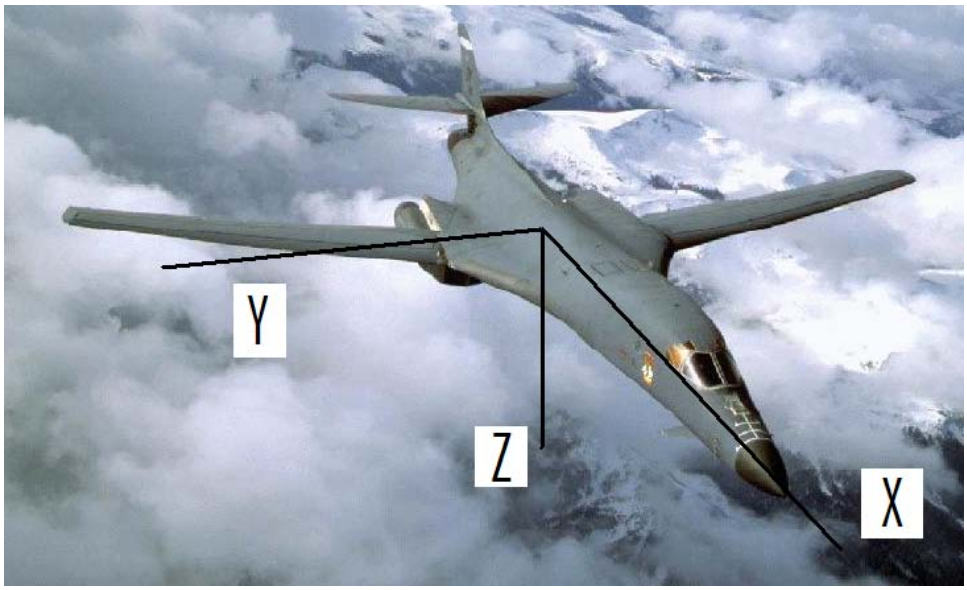


Figure 7.4: Aircraft Coordinate System Used in Visual Snake Aerial Refueling Simulations.

In Table 7.1, the sensitivities to Y position perturbations are much lower than the other axes. This is because the nominal position is assumed to be directly in line with the tanker aircraft, as seen in Figure 7.2. In Table 7.2, perturbations in pitch are seen to be strongly coupled with range errors, while roll and yaw perturbations are strongly coupled with heading errors.

Table 7.1: Range Error and Heading Error Sensitivity to Perturbations from Nominal Position in the Aerial Refueling Visual Position Sensing Simulation

Axis	Range Error Sensitivity (m/m)	Heading Error Sensitivity (deg./m)
X	0.8756	0.0569
Y	0.0169	0.0009
Z	-0.5232	0.0372

Table 7.2: Range Error and Heading Error Sensitivity to Perturbations from Nominal Orientation in the Aerial Refueling Visual Position Sensing Simulation

Angle	Range Error Sensitivity (m/deg.)	Heading Error Sensitivity (deg./deg.)
Yaw	0.0011	0.1606
Pitch	-0.1228	0.0460
Roll	4.761×10^{-4}	0.1405

7.3 Simulation Results

The visual snake tracking errors are introduced to the numerical simulation to simulate the true performance of the visual sensing system. The snake COM and principal axes size measurements are corrupted with Gaussian noise according to the characteristics determined in Section 3.3. Because those values represent an ideal case, they were further multiplied by a factor of two. These simulation results all assume that the aircraft are at the nominal relative orientation and range of 10.7 m. If this were not the case, these results would be further corrupted according to the sensitivities seen in Tables 7.1 and 7.2. Further details about the simulation are found in Reference 23.

Figure 7.5 shows the position error magnitude, and Figure 7.6 shows the errors resolved in the heading and range directions, as well as the position error resulting from the heading uncertainty at the nominal range. Table 7.3 shows the mean and standard deviations. From the data, it is seen that the measurement accuracy should easily be sufficient to meet the required 2 cm positioning tolerance. In addition, the error in range greatly dominates the error

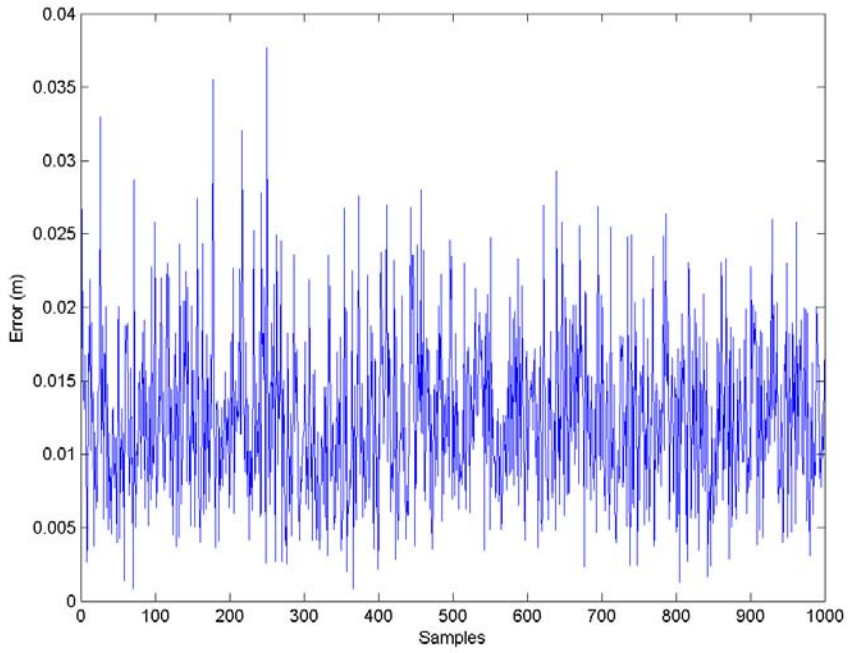
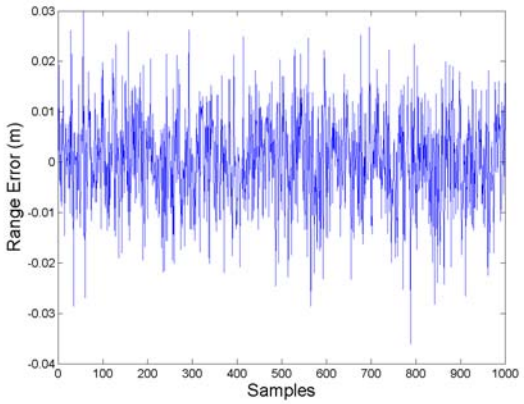


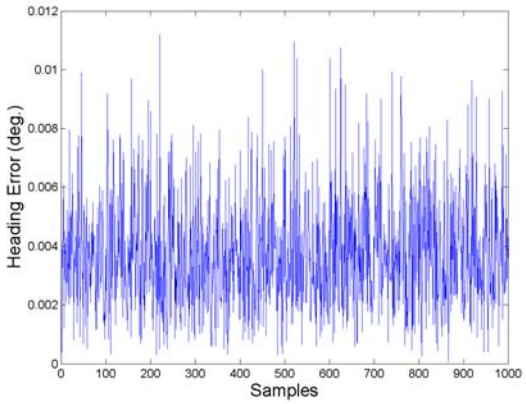
Figure 7.5: Position Error Magnitude for Aerial Refueling Visual Position Sensing Simulation

in heading. In other words, this visual sensing method determines the target COM heading much more accurately than it determines the range to the target. The resulting "measurement error envelope" looks like long thin tube, as illustrated in Figure 7.7. The green lines represent the cone defined by the heading uncertainty, and the red region corresponds to the depth uncertainty. Both regions are extremely exaggerated for effect.

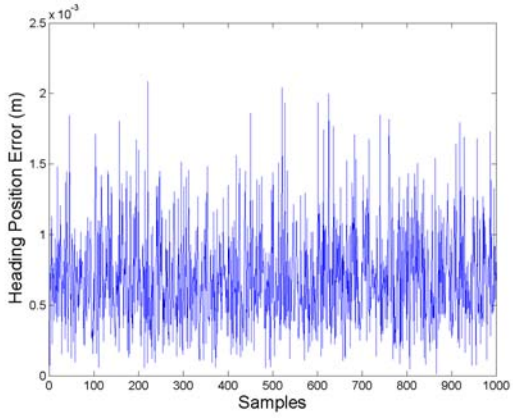
The use of visual sensing for aerial refueling is a promising application, and much work remains. Many assumptions, such as the visual target coincident with the receptacle, should be relaxed, and higher fidelity UMBRA simulations using the OpenGL world and visual snakes are planned. Wind gust and other disturbance models should be incorporated.



(a) Range Error



(b) Heading Error



(c) Position Error from Heading Uncertainty

Figure 7.6: Range Error, Heading Error, and Heading Position Error for Aerial Refueling Visual Position Sensing Simulation.

Table 7.3: Error Magnitude, Range Error, and Heading Error Data from Aerial Refueling Visual Position Sensing Simulation.

Quantity	Mean	Standard Deviation
Error Magnitude (m)	0.0124	0.0057
Range Error (m)	6.2919×10^{-5}	0.0103
Heading Error (deg.)	0.0037	0.0020
Position Error from Heading Uncertainty (m)	6.89×10^{-4}	3.72×10^{-4}

The use of multiple cameras should be investigated further. A second camera near the boom tip would help account for uncertainty in the shape of the flexible boom. In addition, because it is much closer to the receptacle, it would greatly increase the end-game boom tip relative position accuracy.

Figure 7.7: Exaggerated Illustration of the Shape of the Range (Red) and Heading Errors (Green) from Aerial Refueling Visual Position Sensing Simulation.

Chapter 8

Conclusions and Future Work

This work describes a hardware testbed being developed at the Virginia Tech Autonomous Vehicle Systems (AVS) Lab to investigate the performance of visual sensing and control techniques in relative navigation problems. Using software to command the motion of an unmanned ground vehicle (UGV), near-planar relative motion problems for underwater, ground, air, and space vehicles could be simulated in the future. By providing the attached sensor packages with realistic relative motion, the UGV-based testbed allows relative navigation sensing and control techniques to be evaluated in real-world test conditions, both indoors and outdoors, at ranges of up to hundreds of meters. The UMBRA simulation framework is used to control both the hardware and software components of the testbed, and its modular nature facilitates the interchange between real hardware and simulated software components. This modularity allows for rapid transition between testing virtual components and real hardware, and allows for large-scale simulations of combinations of real

and virtual components interacting in real or simulated environments.

A relative navigation sensor based on a color statistical pressure snake algorithm, or visual snake, is considered. The visual snake algorithm, which is designed to be robust to lighting variations and numerically efficient, is not developed in this work, but its performance as a relative navigation sensor is assessed. Methods for extracting the target image area, center of mass, principal axes sizes, and principal axis orientation from the visual snake contour are presented, and the accuracy with which these features are extracted is assessed.

Simple, effective and easy to tune non-linear proportional and proportional-integral feedback control laws are developed for the visual servoing problem. Hardware results from the UGV testbed operating in an unstructured environment are shown for a variety of cases using stationary and moving collaborative targets of known size. In addition, a stationary non-collaborative target is considered. A Kalman filter is developed to estimate the unknown visual target-depth calibration parameter. Hardware results from these visual servoing tests show promise for the use of visual sensing and control techniques for unmanned vehicle relative navigation problems. The use of visual servoing in an automated aerial refueling application is considered, and preliminary results are shown.

The AVS Lab relative motion hardware testbed has many exciting future applications, and much work remains. Vehicle tracking systems, like VISNAV or differential GPS, must be incorporated to determine the inertial position of the UGV. Relative motion simulation software modules, such as relative orbital motion predictors, can be developed for a wide variety of problems. The control of a pan-and-tilt unit attached to the UGV must be

integrated with the vehicle motion to provide independent heading and translation in relative motion problems.

In addition, much work in the development of the visual sensing and servoing algorithms remains. While not addressed here, improvements to the visual snake automatic gain selection algorithm can improve performance in widely varying lighting conditions. Improved schemes for determining depth from target image principal axes can be developed for different problems. Estimation theory can be applied to determine more accurate relative position and orientation measurements. In addition, the use of estimation to determine relative motion from image flows should be investigated, and the use of intelligent persistent excitation should be investigated as a means to increase the observability of certain estimation problems. Finally, particular visual servoing applications, such as automated aerial refueling, can be simulated with more realistic assumptions and higher fidelity using UMBRA software simulations as well as the hardware testbed. The AVS Lab's hybrid hardware and UMBRA-based software relative motion testbed presented here is an excellent foundation for the continued investigation of these and other unmanned vehicle relative motion problems.

Bibliography

[1] Ella M. Atkins, Jamie A. Lennon, and Rhiannon S. Peasco. Vision-based following for cooperative astronaut-robot operations. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, March 2002.

[2] John L. Crassidis and John L. Junkins. *Optimal Estimation of Dynamic Systems*. Chapman & Hall/CRC, Boca Raton, Florida, 2004.

[3] Glenn Creamer, Frank Pipitone, Charmaine Gilbreath, Dexter Bird, and Sam Hollander. NRL technologies for autonomous inter-space rendezvous and proximity operations. In *Advances in the Astronautical Sciences, The John L. Junkins Astrodynamics Symposium*, volume 115, pages 233–250. American Astronautical Society, 2003.

[4] Aveek K. Das, Rafael Fierro, R. Vijay Kumar, B. Southall, John R. Spletzer, and Camillo J. Taylor. Real-time vision-based control of a nonholonomic mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 1714–1719, Seoul, Korea, May 2001.

[5] Aveek K. Das, Rafael Fierro, Vijay Kumar, James P. Ostrowski, John Spletzer, and Camillo J. Taylor. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, 2002.

[6] U.S. Naval Research Laboratory Spacecraft Robotics Engineering and Controls Laboratory, 2006. `http://www.nrl.navy.mil/content.php?P=02REVIEW207`.

[7] Directed Perception Inc., 2006. `http://www.dperception.com/index.html`.

[8] StarVision Techologies Inc., 2005. `http://vesuvius.jsc.nasa.gov/er_er/html/sprint/`.

[9] Jim Ivins and John Porrill. Active region models for segmenting medical images. In *Proceedings of the IEEE International Conference on Image Processing*, pages 227–231, Austin, Texas, 1994.

[10] John L. Junkins, Declan H. Hughes, K. P. Wazni, and V. Pariyapong. Vision-based navigation for rendezvous, docking and proximity operations. In *AAS Guidance and Control Conference*, Breckenridge, Colorado, Feb. 3–7 1999. Paper No. AAS 99-021.

[11] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[12] Hassan K. Khalil. *Nonlinear Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, 3rd edition, 2002.

[13] Sandia National Laboratories. Analysis of robotics technology – UMBRA, 2005. `http://www.sandia.gov/isrc/umbra.html`.

[14] Stanford University Aerospace Robotics Laboratory, 2006. `http://sun-valley.stanford.edu/`.

[15] Jamie Lennon and Ella Atkins. Color-based vision tracking for an astronaut EVA assist vehicle. In *Proceedings of the SAE International Conference on Environmental Systems (ICES)*, Orlando, FL, July 2001.

[16] ActivMedia Robotics LLC. Pioneer 3 Operations Manual. Technical report, ActivMedia Robotics, LLC., Amherst, NH, September 2004.

[17] ActivMedia Robotics LLC., 2006. `http://www.activmedia.com`.

[18] R. Malladi, R. Kimmel, D. Adalsteinsson, G. Sapiro, V. Caselles, and J. A. Sethian. A geometric approach to segmentation and analysis of 3D medical images. In *Proceedings of Mathematical Methods in Biomedical Image Analysis Workshop*, San Francisco, June 21–22 1996.

[19] Larry Matthies, Richard Szeliski, and Takeo Kanade. Kalman filter-based algorithms for estimating depth from image sequences. Technical Report CMU-RI-TR-88-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1988.

[20] S. Matunaga, K. Yoshihara, T. Takahashi, S. Tsurumi, and K. Ui. Ground experiment system for dual-manipulation-based capture of damaged satellites. In *Proceedings*

of the International Conference on Intelligent Robots and Systems, pages 1847–1852, Piscataway, NJ, 2000. IEEE.

[21] David Miller, A. Saenz-Otero, and J. Wertz et. al. Spheres: A testbed for long duration satellite formation flying in micro-gravity conditions. In *Proceedings of the AAS/AIAA Spaceflight Mechanics Meeting*, pages 167–179, San Diego, CA, 2000. Univelt.

[22] Mark J. Monda. Pioneer 3-DX UGV and the UMBRA Pioneer Module. Technical report, Virginia Polytechnic Institute and State University, Blacksburg, VA, May 2006.

[23] Mark J. Monda. Visual snakes for relative sensing during aerial refueling. Technical report, Virginia Polytechnic Institute and State University, Blacksburg, VA, May 2006.

[24] Nikolaos P. Papanikolopoulos, Bradley J. Nelson, and Pradeep K. Khosla. Six degree-of-freedom hand/eye visual tracking with uncertain parameters. *IEEE Transactions on Robotics and Automation*, 11(5):725–732, 195.

[25] Douglas Perrin and Christopher E. Smith. Region-based strategies for active contour models. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[26] Douglas Perrin and Christopher E. Smith. Rethinking classical internal forces for active contour models. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 615–620, Dec. 8–14 2001.

[27] Douglas P. Perrin, Andrew M. Ladd, Lydia E. Kavraki, Robert D. Howe, and Jeremy W. Cannon. Fast intersection checking for parametric deformable models. In *SPIE Medical Imaging*, San Diego, CA, February 12–17 2005.

[28] J. Philip. Estimation three-dimensional motion of rigid objects from noisy observations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):61–66, 1991.

[29] F. D. Roe, D. W. Mitchel, B. M. Linner, and D. L. Kelly. Simulation techniques for avionics systems – an introduction to a world class facility. In *Proceedings of the Flight Simulation Technologies Conference*, pages 535–543, Reston, VA, 1996. AIAA.

[30] Christopher C. Romanelli. Software simulation of an unmanned vehicle performing relative spacecraft orbits. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, May 2006.

[31] Hanspeter Schaub. Extracting primary features of a statistical pressure snake. Technical Report SAND2004-1869, Sandia National Laboratories, Albuquerque, NM, 2004.

[32] Hanspeter Schaub. Statistical pressure snakes based on color images. Technical Report SAND2004-1867, Sandia National Laboratories, Albuquerque, NM, 2004.

[33] Hanspeter Schaub. Visual servoing using statistical pressure snakes. Technical Report SAND2004-1868, Sandia National Laboratories, Albuquerque, NM, 2004.

[34] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA, October 2003.

[35] Hanspeter Schaub and Christopher E. Smith. Color snakes for dynamic lighting conditions on mobile manipulation platforms. In *IEEE/RJS International Conference on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 2003.

[36] Hanspeter Schaub and Christopher Wilson. Matching a statistical pressure snake to a four-sided polygon and estimating the polygon corners. Technical Report SAND2004-1871, Sandia National Laboratories, Albuquerque, NM, 2003.

[37] Ernst Schermann and Thomas Ebersole. An implementation of color statistical pressure snakes using HSV color space and redefined internal energy terms. CS 766 Project, Fall 2004.

[38] H. Schubert and Jonathan How. Space construction: An experimental testbed to develop enabling technologies. In *Proceedings of the Conference on Telemanipulator and Telepresence Technologies IV*, pages 179–188, Piscataway, NJ, 1997. IEEE.

[39] Jana L. Schwartz, Mason A. Peck, and Christopher D. Hall. Historical review of air-bearing spacecraft simulators. *AIAA Journal of Guidance, Control and Dynamics*, 26(4):513–522, 2003.

[40] A. Singh. Incremental estimation of image-flow using a Kalman filter. *IEEE Proceedings of the IEEE Workshop on Visual Motion*, 9(9):36–43, 1991.

[41] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

[42] Christopher E. Smith and Hanspeter Schaub. Efficient polygonal intersection determination with applications to robotics and vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, Aug. 2–6 2005.

[43] J. Spetsakis, M.E.; Aloimonos. Optimal motion estimation. *IEEE Proceedings of the IEEE Workshop on Visual Motion*, 3(3):229–237, 1989.

[44] Carsten Steger. On the calculation of moments of polygons. Technical Report FGBV–96–04, Forschungsgruppe Bildverstehen (FG BV), Informatik IX, Technische Universität München, Aug. 1996.

[45] J. L. Stephenson. *The Aerial Refueling Receiver That Does Not Complain.* Ph.D. dissertation, School of Advanced Airpower Studies, June 1998.