

W
Z
O
N
O
W
R

Intelligent Visual Servoing for Robotic Control

Phase I Final Report

Document Number: 9424-REPT-001.1

Prepared for
Armament Research and Development Center
Picatinny Arsenal

19990722 004

August 17, 1995

Document Security Notice: UNRESTRICTED

REDZONE ROBOTICS INC. 2425 Liberty Avenue Pittsburgh, PA 15222-4639

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

Intelligent Visual Servoing for Robotic Control

Phase I Final Report

Document Number: 9424-REPT-001.1

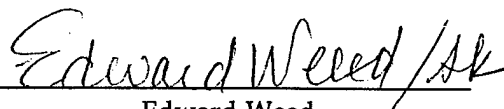
Prepared for
Armament Research and Development Center
Picatinny Arsenal

Release Date: August 17, 1995

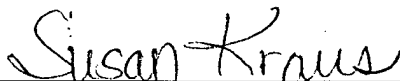
Prepared by
RedZone Robotics, Inc.
2425 Liberty Avenue
Pittsburgh, PA 15222-4639
412-765-3064



Alan D. Berger
Principal Investigator



Edward Weed
Machine Vision Engineer



Susan Kraus
Document QA/QC

Table of Contents

Table of Contents	i
List of Figures	iii
Chapter 1: Introduction	1
Chapter 2: Applications	4
Chapter 3: Prior Work	6
3.1. Eye-in-Hand Techniques	6
3.2. Stationary Camera Techniques	10
Chapter 4: Specification	11
Chapter 5: Technical Approach	18
5.1. Eye-in-hand vs. Static Camera	18
5.2. Potential Algorithms	19
5.3. Selection of Approach	26
5.4. Comparison with Competing Products	27
Chapter 6: Implementation of Optical Flow Based Visual Servoing	29
6.1. Technical Issues & Solutions	30
6.2. Hardware Implementation	34
Chapter 7: Development Plan	38
Task 1: Review Phase I Results	38

Task 2: Select Hardware	39
Task 3: Develop Application Software	40
Task 4: Develop User Interface	41
Task 5: Static Grasping Demonstration	41
Task 6: Task Decomposition System	42
Task 7: Distance Estimator Testing and Refinement	42
Task 8: Application to Mobile Robots	43
Task 9: Speed Improvements	43
Task 10: Integrated Demonstration	44
Task 11: Final Report/User's Manual	48
Chapter 8: Summary	49

List of Figures

Figure 1.	Wants Ranking Matrix	14
Figure 2.	Quantified Requirements	15
Figure 3.	Specification	17
Figure 4.	System Block Diagram	19
Figure 5.	Correlation Algorithm	21
Figure 6.	Edge Tracking Algorithm	22
Figure 7.	Optical Flow Algorithm	25
Figure 8.	Servoing Methods Comparison	27
Figure 9.	Competitive Technique Comparison	28
Figure 10.	Task Data Structure	31
Figure 11.	Task Decomposition	32
Figure 12.	Image Acquisition and Processing Boards	37
Figure 13.	Demonstration Work Cell	45
Figure 14.	Demonstration Flow Chart	46
Figure 15.	Control System Architecture	47

Chapter 1: Introduction

The Department of Defense Critical Technologies Plan (March 1990) targets machine intelligence and robotics as one of 20 "technologies with great promise of ensuring the long-term superiority of United States weapon systems." Robotic systems and technology are being developed and applied to the important requirement of keeping mobile weapons platforms such as tanks and artillery supplied with ammunition in combat situations. This application of robotic technology can both increase the efficiency of the ammunition handling and supply operations and, at the same time, reduce the exposure of military personnel to hazardous environments and missions.

RedZone Robotics, Inc. has completed a Phase I SBIR that addresses the needs of DOD SBIR solicitation A94-094, "Intelligent Sensor Based Robotic Control System Technology". This project has resulted in a design for intelligent vision-based servoing control of robotic and telerobotic manipulators that will enhance the capabilities, adaptability, robustness, and autonomy of existing robotic controllers for tasks that involve object tracking and manipulator positioning. This technology is modular in structure so that it can be added to existing robot control systems or integrated into new robotic and telerobotic control systems.

The visual servoing system incorporates novel video feedback algorithms, using a single camera mounted on the robot, to actively position a manipulator relative to target objects. Rather than relying on precise, pre-movement kinematic modeling, the technique continuously measures target position in a video image while the robot moves. Visual servoing relaxes conventional requirements for object tracking and manipulator positioning and will allow less expensive, faster setup of new and

existing robot systems. The ultimate result of this research will be new intelligent visual servocontrol commercial products for semi-autonomous robotic or telerobotic operations.

This report presents the results of the Phase I effort. The technical objectives of the Phase I effort have all been met or exceeded. The objectives are re-stated here, along a brief description of how they have been met:

Objective 1 Develop a workable technical approach to the problem of intelligent real-time visual servo control of robotic manipulators for ammunition handling and loading applications.

Status Three realistic approaches were developed and evaluated. Of these, one was found to be preferable for further development. Chapter 5 describes the approaches and the selection process.

Objective 2 Develop a mathematical framework for control system modeling and use it to develop a model of the presently proposed visual servo control scheme. Develop algorithms for robotic manipulator control functions and system behavior using visual servoing techniques.

Status A complete mathematical framework, along with experimental results, exist for the chosen approach. The details of the mathematics are available in the referenced literature. The algorithms for a complete system are presented in Chapter 6.

Objective 3 Determine the computational throughput required for real-time control using the proposed technical approach. Specify the control and image processing hardware performance requirements necessary for a real-time implementation of the proposed visual servo control system model. Also determine and specify the sensor hardware performance requirements for real-time operation of the system.

Status Computational needs are identified in Chapter 6. This chapter also presents a preliminary hardware analysis and design.

Objective 4 Provide an analysis of how the proposed visual servo control system is expected to rank in comparison to alternative and traditional (non-vision-based) control schemes in terms of speed, accuracy, robustness, adaptability, hardware requirements, and other criteria.

Status Visual servoing provides some unique advantages over currently available applications. This is shown via a formal analysis technique presented in Sections 5.3 and 5.4.

Objective 5 Develop the preliminary design for an intelligent visual servo control processor incorporating the technical approach selected by completing objectives 1-4 (above).

Status A preliminary design, incorporating enhancements to the basic framework, identified in Objective 3 has been completed. It is presented in Chapter 6. A plan for its development and demonstration is presented in Chapter 7.

The remainder of this document is organized as follows:

Chapter 2 describes robotic and other applications for visual servoing technology.

Chapter 3 reviews work done by others in the visual servoing field.

Chapter 4 develops a specification for the ultimate visual servoing product that would be sold at the completion of Phase 3.

Chapter 5 presents the technical approach for visual servoing and shows the process used to arrive at the chosen solution.

Chapter 6 describes detailed issues for implementation of the visual servoing system including technical hurdles and hardware.

Chapter 7 is a work plan for Phase II development.

Chapter 8 summarizes the report.

Chapter 2: Applications

There are numerous potential applications of visual servoing technology. Several promising applications are briefly presented in this chapter. It is not intended to be a comprehensive listing, but rather a starting point to illustrate the range of uses for visual servoing and image tracking technology.

1. **Replacement for pre-programmed motion.** Pre-Programmed algorithms, a non-visual feedback control paradigm requiring advance knowledge of the environment, calculate goal position before the manipulator is moved. Position estimates are updated as the manipulator moves by comparing (non-visual) sensor feedback to the world model; implementations assume that differences between the world model and the actual environment are negligible. Replacing pre-programmed algorithms with visual servoing relaxes these positioning requirements, allowing the use of lower accuracy (cheaper) robots and fixtures.
2. **Replacement for static vision.** Static Vision positioning, a non-feedback visual approach, requires a fixed target, *a priori* knowledge of camera-to-target relative position, a precise manipulator kinematic model, and calibration of camera-to-manipulator relative position. Goal positions are calculated before the manipulator is moved using target position measured from the camera image and the coordinate transformation between camera and end-effector. Replacing static vision algorithms with visual servoing relaxes the calibration requirements. This allows the use of lower accuracy (cheaper) robots and fixtures.

3. **Locating moving parts.** Static vision and point position sensors, such as proximity switches, cannot continuously monitor the position of moving parts. Continuous position measurement is necessary for all but the most simple manipulations. In fact, the biggest automation problem facing the US. auto industry today is using robots for final assembly of cars as they continuously move down the assembly line. The only practical approach available today is to stop the assembly line or build complex fixtures to hold the parts during assembly. Visual servoing solves this problem by allowing operations to be done on moving parts.
4. **Road following.** Road following is an active area of research both for the military (for autonomous vehicles and robotic convoying) and commercial sector (as a part of the intelligent highway effort). Typical methodologies employ neural nets to estimate the vehicle position in a well-defined lane. Visual servoing provides a deterministic alternative.
5. **Security.** Security and monitoring systems frequently attempt to detect moving objects in a secure area. Visual servoing can do this and automatically pan a camera to follow the moving object. This allows for a wider target area per camera, fewer cameras in an installation, and possibly fewer people to monitor those cameras.
6. **Traffic monitoring.** Traffic counters and sensors are relatively expensive and invasive devices. Most devices require modification to the existing pavement for installation. Visual servoing technology could provide for non-contact tracking of vehicles or pedestrians. Such a system would be more portable and potentially less expensive than current methods.
7. **Teleoperation aid.** Teleoperated robots are frequently used in dangerous unstructured environments as an extension of the operator. Operating these machines through a limited number of monocular video cameras is very time consuming and difficult. Visual servoing could provide semi-autonomous assistance for two applications. First, the operator could select a point or object on a video screen that he wants to move the robot to; visual servoing would automatically drive the robot to that location and stop when it reaches it, returning control back to the operator. Secondly, visual servoing could be used as an obstacle avoidance technique, avoiding specific objects rather than servoing to them.

Clearly, there is a wide range of practical applications that would benefit from the use of a cost-effective commercial visual servoing system.

Chapter 3: Prior Work

The area of computer vision known as active vision has been the subject of increasing academic attention in recent years. Visual servoing, in particular, has been studied by researchers for many years. However, the authors of this report are not aware of any operational industrial visual servoing applications at this time. In this chapter, a survey of the more recently published work is presented. The work is classified first by camera position, either eye-in-hand or stationary camera, and secondly by type of technique. The eye-in-hand approaches appear to be most compelling since they are applicable to a greater range of applications.

3.1. Eye-in-Hand Techniques

Eye-in-hand systems place the camera on the end effector of the robot. A wide range of techniques has been applied to eye-in-hand visual servoing. These include feature based, focus of expansion, learning, optical flow, and path planning. First, techniques to servo about specific extracted image features are examined.

3.1.1. Feature Based

Several authors, including Feddema¹² and Weiss³ have proposed solutions to the general case of both camera and object in motion. Both use adaptive estimation techniques to predict the location of the features in each new image, and compute errors in the feature space. This technique incorporates the image Jacobian, which describes the differential change in object pose with respect to the camera's frame of reference. Accurate calculation of the Jacobian requires some camera calibration. Also, the image features must be known in advance for this technique so that the goal can be presented to the controller. However, the technique could be used with almost any image derived feature. Feddema has also presented a smooth (continuous velocity, acceleration, and jerk) 7th order feature space trajectory generator.⁴

Hashimoto has developed a robust tracking system that operates on objects of known shape and size that dominate the camera's field of view⁵. Note that a tracking system simply follows a moving object at a constant distance, rather than approaching the object. The controller in this work is a hierarchical structure with a feature-based vision loop closed around a position based cartesian space controller. Most significantly, Hashimoto has shown through simulations and experiments that errors computed in the feature space lead to a more robust system than errors that are translated to cartesian space. This appears to be principally due to the inaccuracies inherent in computing object poses from measured data.

Yoshimi and Allen have demonstrated a technique in a class of specialized techniques that exploit specific characteristics of a specific problem⁶. The special case presented in their paper is a visual solution for the peg-in-hole problem. Given

¹Feddema J.T. and Lee C.S., *Adaptive Image Feature Prediction and Control for Visual Tracking with a Hand-Eye Coordinated Camera*, IEEE Transactions on Systems, Man, and Cybernetics, Vol 20, No 5, Sept/Oct 1990, pp 1172-1183.

²Feddema, et al, *Weighted Selection of Image Features for Resolved Rate Visual Feedback Control*, IEEE Transactions on Robotics and Automation, Vol 7, No 1, February 1991, pp 31-47.

³Weiss et al, *Dynamic Sensor-Based Control of Robots with Visual Feedback*, IEEE Journal of Robotics and Automation, Vol RA-3, No 5, October 1987, pp 404-417.

⁴Feddema J.T. and Mitchell O.R., *Vision-Guided Servoing with Feature-Based Trajectory Generation*, IEEE Transactions on Robotics and Automation, Vol 5, No 5, October 1989, pp 691-700.

⁵Hashimoto et al, *Manipulator Control with Image-Based Visual Servo*, Proceedings of the 1991 IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, April 1991, pp 2267-2271.

⁶Yoshimi B.H. and Allen P.K. , *Active, Uncalibrated Visual Servoing*, IEEE Int. Conf. on Robotics and Automation, 1994, pp 156-161.

known objects with circular symmetry, motion is planned to exploit this and derive estimates of position both parallel and normal to the image plane. One significant result of this approach is that it does not require camera calibration in spite of its use of the image Jacobian.

A theoretical framework focusing mostly on the control aspects of visual servoing is presented by Espiau et al⁷. In particular, the robustness and stability aspects of the problem are analyzed in the context of their servoing framework. They do, however, require known targets.

3.1.2. Focus of Expansion

Focus of Expansion (FOE) techniques exploit an image property where all features in a scene appear to expand radially from a particular point in the image. The location of this point in the 2D image plane is proportional to the camera velocity. There do not appear to be published results demonstrating application of these techniques in the feedback loop of a controller, but experimental results showing accurately measured motion have been published. The downfall of this technique for visual servoing is that the published formulations estimate camera motion relative to an entire scene, not just a portion of the scene. A good introduction to the techniques is provided by Ishiguro⁸. This paper shows how to estimate the relative positions of two panoramic views. Robust techniques for computing 3D camera motion are presented by Burger and Bhanu^{9,10}. In these papers, they introduce the concept of a fuzzy FOE, and describe techniques to extract an estimate of motion from a FOE region, rather than a specific point.

3.1.3. Learning Based Approaches

Learning techniques can avoid the need for camera calibration and/or for *a priori* information about the objects to be tracked. However, they typically require a series of preliminary motions (the learning stage) to observe the object from several different angles and positions. Once this is done, the system can actively track or

⁷Espiau et al, *A New Approach to Visual Servoing in Robotics*, IEEE Trans on Robotics and Automation, Vol 8, No 3, June 1992, pp 313-326.

⁸Ishiguro, et al, *Omnidirectional Visual Information for Navigating a Mobile Robot*, IEEE Int. Conf. on Robotics and Automation, 1993, pp 799-804.

⁹Burger and Bhanu, *A Geometric Constraint Method for Estimating 3-D Camera Motion*, IEEE Int. Conf. on Robotics and Automation, 1994, pp 1155-1160.

¹⁰Burger and Bhanu, *Estimating 3-D Egomotion from Perspective Image Sequences*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 12, No 11, November 1990, pp 1040-1058.

servo to the object. Nayar's work¹¹ is representative of such systems. Additional work in this area has been done by Chen and Weng¹²

3.1.4. Optical Flow

Optical flow seems like an intuitively obvious solution for visual servoing since it naturally produces camera to scene relative velocities. Papanikolopoulos et al have authored several papers on their technique^{13,14}, and other work is on-going at Carnegie-Mellon University. Papanikolopoulos' approach explicitly handles 3D motion of both the object and the camera for tracking applications. The optical flow computations follow the pyramidal Sum-of-Squares Differences approach. State estimators are used with an adaptive controller to reduce the reliance on camera and system calibration. Experimental results are presented and clearly show the performance advantages of the adaptive controller over simpler schemes.

3.1.5. Path Planning

Several techniques for path planning based on a eye-in-hand type system have been proposed^{15,16,17}. These are higher-level techniques than the visual servoing that we propose, but point to potential planning functions that could use the same hardware and data that the visual servoing system provides.

¹¹Nayar, et al, *Learning, Positioning, and Tracking Visual Appearance*, IEEE Int. Conf. on Robotics and Automation, 1994, pp 3237-3243.

¹²S. Chen and J Weng, *Autonomous Navigation Using Recursive Partition Tree*, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, August 1995, pp 130-135.

¹³Papanikolopoulos et al, *Six Degree-of-Freedom Hand/Eye Visual Tracking with Uncertain Parameters*, IEEE Int. Conf. on Robotics and Automation, 1994, pp 174-178.

¹⁴Papanikolopoulos et al, *Vision and Control Techniques for Robotic Visual Tracking*, Proc of the IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, 1991, pp 857-864.

¹⁵Sharma et al, *Dynamic Robot Manipulation Using Visual Tracking*, Proc IEEE Int. Conf. on Robotics and Automation, 1992, pp 1844-1849.

¹⁶Schrott, *Feature Based Camera Guided Grasping by an Eye-in -Hand Robot*, Proc IEEE Int. Conf. on Robotics and Automation, 1992, pp 1832-1837.

¹⁷Blake et al, *Visual Navigation Around Curved Obstacles*, Proc IEEE Int Conf on Robotics and Automation, April 1991, pp 2490-2495.

3.2. Stationary Camera Techniques

Stationary (also called static) camera techniques follow the same general classes of approaches as eye-in-hand systems, but the problem formulation and performance results are different. It is more likely that a stationary technique will require accurate camera calibration since the transformation between the camera and robot is an important part of the system. Further, camera positioning is tricky since the system could fail if the robot obscures the object or the gripper.

Stationary camera techniques equivalent to the eye-in-hand feature based systems are typified by those that control the robot so that the disparity between the gripper and the object¹⁸ is minimized, or control the end effector to a static point in the image frame¹⁹. The former approach uses stereo cameras, while the latter uses a single camera. Learning based approaches that require motion patterns of the robot have also been proposed²⁰.

Allen has presented an optical flow based approach²¹. This technique computes the optical flow from two cameras, and uses triangulation to find the location of the most significant motion in the image. This data is incorporated into the feedback loop through a predictive filter.

Finally, there is a class of techniques that are unique to stationary camera visual servoing^{22,23}. These techniques use the cameras to estimate robot state, rather than object state. While theoretically interesting, these are not relevant to practical manipulator systems with good direct measured joint positions.

¹⁸Hager G. D. et al, *Robot Feedback Control Based on Stereo Vision: Towards Calibration-Free Hand-Eye Coordination*, IEEE Int. Conf. on Robotics and Automation, 1994, pp 2850-2856

¹⁹Castano A. and Hutchinson S. , *Hybrid Vision/Position Servo Control of a Robotic Manipulator*, Proc IEEE Int. Conf. on Robotics and Automation, May 19923, pp 1264-1269.

²⁰Miller W.T., *Sensor-Based control of Robotic Manipulators Using a General Learning Algorithm*, IEEE Journal of Robotics and Automation, Vol RA-3, No 2, April 1987, pp 158-165.

²¹Allen P.K. et al, *Real-Time Visual Servoing*, Proc. IEEE Int. Conf. on Robotics and Automation, April 1991, pp 851-856.

²²Bishop et al, *On the Performance of State Estimation for Visual Servo Systems*, IEEE Int. Conf. on Robotics and Automation, 1994, pp 168-173.

²³Tendick, et al, *A Supervisory Telerobotic Control System Using Model-Based Vision Feedback*, Proc IEEE Int Conf on Robotics and Automation, April 1991, pp 2280-2285.

Chapter 4: Specification

A specification defines the engineering targets for a system that would be commercially viable at the completion of Phase III. The approach used to develop the specification is called Quality Function Deployment (QFD). QFD is simply a methodology that ensures logical inclusion of the important elements of a specification: weighted customer requirements, competitive comparison, and quantitative engineering targets. This chapter goes through the QFD process step by step, and ends with a concise specification for a visual servoing system.

Step 1: Customer Requirements. The QFD process starts with identifying customers and their requirements. There are three distinct customer groups. The first is the ARDEC robotics group at Picatinny Arsenal. The requirements listed below were extracted from discussions at the kick-off meeting. The second customer is RedZone management, and the third is the potential commercial end user. The requirements are listed below by customer:

Picatinny Arsenal ARDEC:

- System needs to be suitable as a research tool (open architecture)
- System should be able to function as a testbed for visual servoing techniques
- System must be portable to multiple robot platforms
- System should work with both mobile robots and manipulators
- Robust against variations in lighting

- Robust against background clutter
- PC compatible

RedZone Management:

- Inexpensive add-on to existing robot systems
- Preliminary design complete by 7/30/95
- Field prototype buildable in under 2 years
- Field prototype buildable for less than \$700K
- Uses proven technology wherever possible

Commercial End Users:

- Simple user interface
- Black-box (easy for non-experts to deploy)
- Easy to integrate into existing robots
- Wide range of objects that can be tracked
- Objects may be in motion
- Objects to be tracked may be selected from a video image

Step 2: Grouping of Requirements. In this step, the requirements are separated into groups by the type of requirement, such as performance, cost, etc. At this step, each requirement is also classified as either a Demand or a Want. Demands are those requirements that must be met, or else the product will fail. Wants are the requirements where there is a continuum of degrees to which the requirement could be met and still have a successful product. Some wants will still have a very high priority, and that will be accounted for in subsequent steps. Below, the requirements from step 1 are repeated and sorted into categories. For completeness, categories that don't currently have requirements are included. Demands are placed at the beginning of each list and are denoted by (D) at the end of the requirement. Wants are denoted by (W).

Performance

- System needs to be suitable as a research tool (open architecture) (D)
- PC compatible (D)
- System must be portable to multiple robot platforms (D)
- Objects to be tracked may be selected from a video image(D)
- System should be able to function as a testbed for visual servoing techniques (W)
- System should work with both mobile robots and manipulators (W)
- Robust against variations in lighting (W)
- Robust against background clutter (W)

- Simple user interface (W)
- Black-box (W)
- Easy to integrate into existing robots (W)
- Wide range of objects that can be tracked (W)
- Objects may be in motion (W)

Appearance

Time

- Preliminary design complete by 7/30/95 (D)
- Field prototype buildable in under 2 years (D)

Cost

- Field prototype buildable for less than \$700K (D)
- Inexpensive add-on to existing robot systems(W)

Manufacture/Assembly

Standards

- Uses proven technology wherever possible (W)

Safety

Environmental Issues

Other

Step 3: Ranking Matrix. A ranking matrix is used to compare the wants identified in step 2. Demands are absolute requirements, so they do not need to be ranked. To simplify the ranking, the wants are numbered below. In preparing the ranking matrix, each want is compared to all of the other wants. If the want is determined to be more important, a 1 is placed in the cell. For example, in Figure 1, a 1 has been placed in row 1, column 2. This means that want 1 (System should be able to function as a testbed for visual servoing techniques) is more important than want 2 (System should work with both mobile robots and manipulators). The right-most column of the table is the resulting weight of each want. This is calculated as a percentage of the total number of ones each want received. In reviewing this table, it is important to remember that this will be a guide for the design process, but absolute correctness to the percentage point will not have any affect on the resulting design.

1. System should be able to function as a testbed for visual servoing techniques (W)
2. System should work with both mobile robots and manipulators (W)
3. Robust against variations in lighting (W)

4. Robust against background clutter (W)
5. Simple user interface (W)
6. Black-box (W)
7. Easy to integrate into existing robots (W)
8. Wide range of objects that can be tracked (W)
9. Inexpensive add-on to existing systems(W)
10. Uses proven technology wherever possible (W)
11. Objects may be in motion (W)

Wants	1	2	3	4	5	6	7	8	9	10	11	Total	Weight	Rank
1		1	0	0	1	1	0	0	1	1	1	5	10	5
2	0		0	0	1	1	0	0	1	1	1	4	8	7
3	1	1		0	1	1	1	0	1	1	0	7	14	2
4	1	1	1		1	1	1	1	1	1	1	9	18	1
5	0	0	0	0		1	0	0	1	1	0	3	6	8
6	0	0	0	0	0		0	0	0	0	0	0	0	11
7	1	1	0	0	1	1		1	1	1	1	7	14	2
8	1	1	1	0	1	1	0		1	0	1	6	12	4
9	0	0	0	0	0	1	0	0		1	0	2	4	9
10	0	0	0	0	0	1	0	1	0		0	2	4	9
11	0	0	1	0	1	1	0	0	1	1		5	10	5
Grand total												50	100	

Figure 1. Wants Ranking Matrix

Thus, in order from most important to least the requirements are:

- Robust against background clutter (W)
- Robust against variations in lighting (W)
- Easy to integrate into existing robots (W)
- Wide range of objects that can be tracked (W)
- System should be able to function as a testbed for visual servoing techniques (W)
- Objects may be in motion (W)
- System should work with both mobile robots and manipulators (W)

- Simple user interface (W)
- Uses proven technology wherever possible (W)
- Inexpensive add-on to existing robot systems(W)
- Black-box (W)

Step 4: Quantify Requirements. Many of the requirements are subjective, and difficult to measure. The system that will be built needs more specific requirements as design targets wherever possible. Figure 2 shows the original requirement, and the quantified version that is used in the specification.

Original Requirement	Updated Requirement
Robust against background clutter	Servo without losing the desired object with up to 5 objects in the background
Simple user interface	No more than 3 operations to initiate servoing Process to initiate visual servoing must take less than 10 seconds for the operator to complete
Black box	Pre-defined interface to control system requiring no knowledge of visual servoing algorithms
Easy to integrate into existing robots	Must output cartesian or joint space commands Must take in standard NTSC or S-Video video signals
Wide range of objects that can be tracked	Must track at least two different types of features, and these features must be common to a large variety of objects such as round holes, cylindrical objects, pipes, vents, beams, etc.
Inexpensive add-on to existing robot systems	Manufacturing cost must be less than \$12K
Objects may be in motion	Must track objects that are moving at speeds up to 200 pixels/s parallel to the image plane.
System should work with both mobile robots and manipulators	Control algorithms must not depend on robot kinematics

Figure 2. Quantified Requirements

Step 5: Specification. The specification is shown in Figure 3 and is a compilation of the requirements that have been classified and analyzed in the preceding steps. The rankings between the various wants are not explicitly shown in the specification. However, the rankings are available for trade-off analyses, and are used in this way in Section 5.3. The classifications used in step 2 have been further refined in this step too. Performance requirements have been subdivided into the overall system

issues called System Requirements, User Interface Requirements, System Interface Requirements, and Performance Requirements. The new performance requirements category specifically defines internal functionality of the system. In addition, the time and cost categories from step 2 have been combined into a single category called Schedule and Budget.

Changes	D/W	Requirements
		1. System Requirements
	D	System needs to be suitable as a research tool (open architecture)
	D	PC compatible
	D	System must be portable to multiple robot platforms
		2. User Interface Requirements
	D	Objects to be tracked may be selected from a video image presented to the operator
	W	No more than 3 operations to initiate servoing
	W	Process to initiate visual servoing must take less than 10 seconds for the operator to complete
		3. System Interface Requirements
	W	Must take in standard NTSC or S-Video video signals
	W	Must output cartesian or joint space commands
	W	Control Algorithm must not depend on kinematics
	W	Pre-defined interface to control system requiring no knowledge of visual servoing algorithms
		4. Performance Requirements
	W	Robust against variations in lighting
	W	Servo without losing the desired object with up to 5 objects in the background
	W	Must be able to distinguish between features that are 20 or more pixels away from the target features
	W	Must track at least two different types of features, and these features must be common to a large variety of objects such as round holes, cylindrical objects, pipes, vents, beams, etc.
	W	Must track objects that are moving at speeds up to 200 pixels/s parallel to the image plane.
		5. Schedule and Budget
	D	Preliminary Design Complete by 7/30/95
	D	Field prototype must be built in under 2 years
	D	Field prototype development must cost less than \$700K
	W	Manufacturing cost must be less than \$12K

Figure 3. Specification

Chapter 5: Technical Approach

This chapter describes the process used in Phase I to develop a technical approach to the visual servoing problem. Both the literature review and the specification developed in the preceding chapters are used to guide this process. The literature review presented in Chapter 3 demonstrated that there are two major classes, and many subclasses of potential solutions. While the approaches identified in Chapter 3 may not represent the only solutions possible for visual servoing, they are typical of the types of solutions that are developed sufficiently enough to be commercially viable at this time. Since the taxonomy used in Chapter 3 presents two significantly different types of systems – eye-in-hand, or static camera, the first step is to pick which of these two approaches is appropriate for our specification. From there, the particular subclasses and in some instances several solutions within the subclass are evaluated against the specification. This determines the general approach that should be followed in Phase II.

5.1. Eye-in-hand vs. Static Camera

The camera position has a major impact on algorithm design, complexity, and performance. Eye-in-hand systems place the camera on the end effector of the robot allowing it to travel with the robot. They also reduce, or even eliminate, the need for calibration between the camera and the robot. Static cameras can be located so that they can see the robot approach an object simplifying the process of determining distance to the object. However, the location of the camera must be selected based on the task and the working environment.

Several of the specification requirements are relevant to making this tradeoff. In particular:

- The control algorithm must not depend on the robot kinematics
- The system must be portable to multiple robot platforms

Static camera methodologies fail for both of these requirements, while eye-in-hand may meet them. Static camera algorithms generally require an accurate calibration between the camera location and the robot base location. In addition, mobile platforms can easily move from a static camera's field of view. Since the camera is already located at the end effector, eye-in-hand systems do not suffer from either of these limitations. Therefore, the eye-in-hand class of visual servoing algorithms has been selected.

The eye-in-hand system is inserted into a typical robot control system as shown in Figure 4.

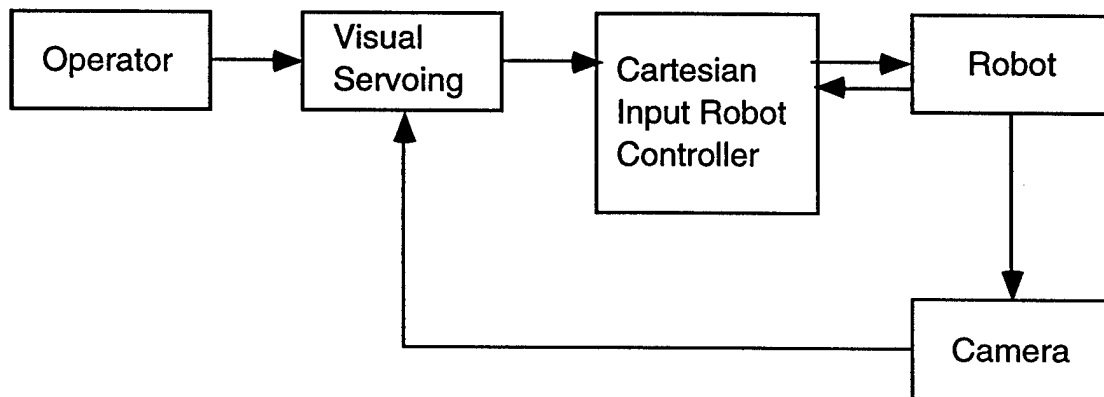


Figure 4. System Block Diagram

5.2. Potential Algorithms

The literature review in Chapter 2 addressed five types of eye-in-hand algorithms:

1. Feature-based
2. Focus of expansion (FOE)
3. Learning-based
4. Optical flow
5. Path planning

The path planning algorithms use vision to actively plan paths rather than for feedback control. This makes them unsuitable for tracking moving objects, so they

will not be considered further. The learning-based algorithms require the robot to make a series of motions to measure the parameters of the object prior to servoing to the object. These motions are typically a pre-computed set of arcs or straight-line trajectories nearly parallel to the image plane. While the specification does not specifically rule this out, the time taken to learn the object would negate many of the advantages that visual servoing has over other methodologies. Therefore, learning based approaches will not be considered further.

Next, FOE approaches are considered. There are two major problems with FOE. First, it cannot handle combined object and camera motion. The FOE principle depends on a single source of motion from which a single focus of expansion may be found. Motion of the target object would be difficult to distinguish from background noise, and it is likely that the manipulator would miss the moving object. Secondly, FOE is basically a velocity measurement tool. Therefore the position of the target object is not measured, making it difficult to home in on an object that occupies only a small portion of the field of view. While other techniques could be used to handle the position aspects, these approaches require the same results as the feature based techniques that can be used to solve the entire visual servoing problem. Therefore, FOE approaches will not be considered further.

This leaves feature-based and optical flow approaches. The current research in these areas is much more developed than for the other methods, so two feature-based approaches and one optical flow approach are considered in much more detail. The feature-based methods are correlation and edge detection and are adapted from existing research, while the optical flow method uses Sum of Squared Differences (SSD) to solve the correspondence problem and adaptive control algorithms for optimal tracking and depth measurement performance. Each of these techniques are described in greater detail in the ensuing paragraphs.

5.2.1. Correlation

Correlation is considered a feature based approach in the taxonomy used in this report since it uses the relationship between measurements of specific visual features from frame to frame. In this case, the features are rectangular groups of pixels that have a relatively unique intensity pattern. Thus, good features may be edges of objects, collections of objects, or portions of objects with patterns. A block diagram of a simplified algorithm is shown in Figure 5.

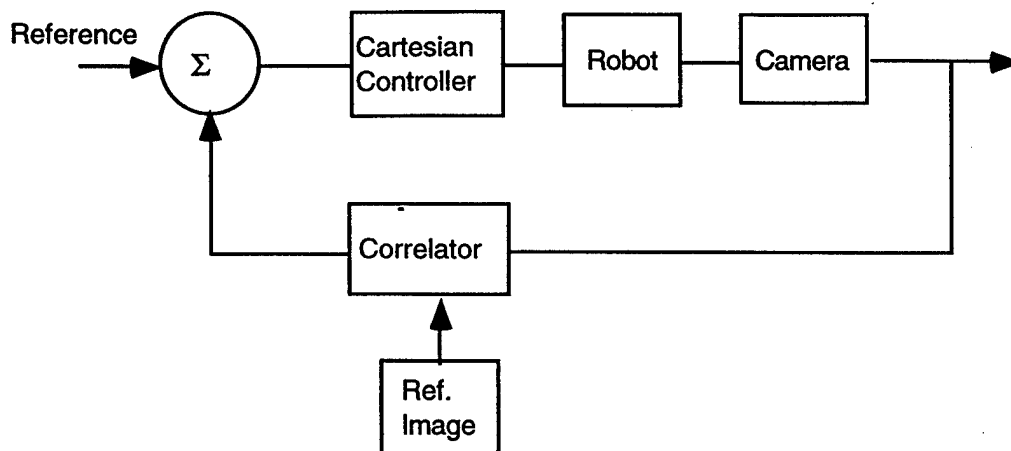


Figure 5. Correlation Algorithm

In this algorithm, a reference image of the feature to be tracked is stored prior to the initiation of servoing. On each cycle of the algorithm, this reference is convolved with the image. The peak value is assumed to correspond to the current position of the feature. The inverse perspective transform is applied to this result to provide an estimate of the position error parallel to the image plane. Error is computed in feature space since this has been shown to be more robust than cartesian space errors. Either an additional sensor or the operator would provide measurements of the distance to the object.

Enhancements could be made to the basic algorithm, such as:

1. Estimating the location of the target feature in the new image, and limiting the correlation to a neighborhood around that location
2. Updating the reference image over time to account for the changing appearance as the viewing angle and distance change.
3. Using some form of adaptive estimation to determine depth of object.

5.2.2. Edge Tracking

An edge feature tracker uses multiple edges on the object to be tracked as a guide in servoing and follows the approach presented by Berger for tactile servoing²⁴. One possible algorithm is shown in block diagram form in Figure 6.

²⁴A.D. Berger and P.K. Khosla, *A Methodology for Using a Tactile Sensor for Dynamic Feature Tracking*, In V. Hayward and O. Khatib Eds, *Lecture Notes in Control and Information Sciences: Experimental Robotics I*, Springer-Verlag, March 1990, Pages 476-596.

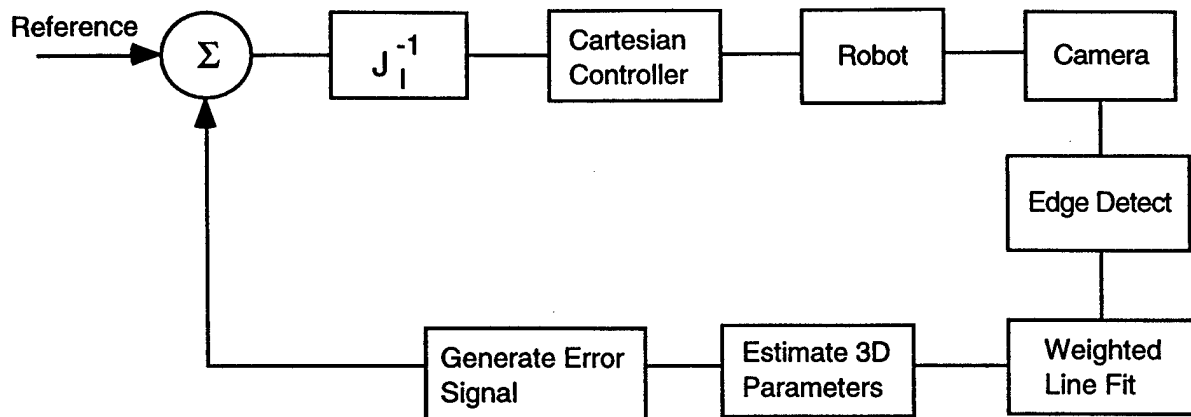


Figure 6. Edge Tracking Algorithm

Prior to initiation of servoing, an operator would select an object, and the system would perform an edge detection and line segment finding process to identify a series of line segments on the object to be tracked. On each cycle of the algorithm, a new image is obtained and a standard edge detector, such as Sobel, or Roberts²⁵, is run on the image. The resulting edge image is processed by a weighted least square line fitting algorithm. Weighting is used to ensure that only points near the expected location of the lines are used in deriving the new line equations. Sequences of images can provide estimates as to the full three dimensional equations of the lines, as shown in the next block in the diagram. The resulting line equations are used to generate a feature-space error, which is then transformed into Cartesian space via the inverse image Jacobian and supplied to the robot's end effector Cartesian space controller.

This algorithm is relatively simple from the theoretical standpoint and would work in many situations. However, it has never been demonstrated in the visual domain and is, therefore, unproven. Some potential limitations include:

1. Multiple, non-parallel edge segments are required in the images for good servo performance. These may not always be available.
2. Detected edges are not always true physical edges. For example, the sides of a pipe appear as edges in a 2D image, but the actual location of the detected edge moves depending on the viewing angle. Shadows and lighting variation can also create false edges that may move as the robot moves.
3. Estimation of 3D line parameters appears to be necessary to ensure stability of the control algorithms. This may be difficult to do accurately in practice since motion of the robot and the object must be estimated.

²⁵Faugeras, Three Dimensional Computer Vision, MIT Press, Cambridge Mass, 1993.

5.2.3. Optical Flow

The optical flow algorithm proposed here is based on the work of Papanikolopoulos et al²⁶. It consists of an optical flow velocity estimator (the imaging section), and an adaptive controller that provides a set of cartesian commands to the robot controller (the control section). An estimator within the controller also provides the distance to the object. Several rectangular intensity pattern features are used to track a single object. This is necessary since most features are not unique in all directions and for robustness. It has been shown experimentally that typical objects require 3 to 4 features for good tracking performance. The ensuing sections describe the imaging and control processes in more detail. Mathematical formulations of the optical flow and control models may be found in the references^{27,28}.

Imaging Section

In general, optical flow is a technique for measuring the velocity of objects in the field of view of a camera. Multiple images are used along with the mathematical model of the perspective transformation, describing the effect of object motion on object image motion. The result is an estimate of the object velocity in a plane parallel to that of the camera. Velocity perpendicular to the camera plane is not explicitly measured.

The challenge in implementing real optical flow systems is solving the correspondence problem. That is, "Where are the features in the new image?". For this system, this is solved via the Sum of Squared Differences (SSD) algorithm. This algorithm looks for regions in the image that most similarly resemble the desired feature. Specifically, it searches for the area that has the minimum (squared) difference between the feature and the image. The search is started at the expected location of the feature and spirals outward. Since this is a searching technique, it can become computationally intensive. Several optimizations are applied to minimize search time:

- Loop short circuiting stops the current SSD computation when the result exceeds the current minimum. This eliminates time that would be wasted computing a complete result for positions that are inferior to the current best match.

²⁶Ibid. 12

²⁷C. Smith, et al, *Eye-In-Hand Robotic Tasks in Uncalibrated Environments*, University of Minnesota AHPARC, Preprint 94-052.

²⁸N.P. Papanikolopoulos, *Controlled Active Vision and Vision-Based Control of Robotic Manipulators*, Proc. 3rd IEEE Mediterranean Symposium on New Directions in Control and Automation, Limassol, Cyprus, July 1995.

- Compute the SSD value in each position in a spiral manor, rather than the intuitive row-major method. This optimization takes advantage of the fact that most features have their highest intensity variation at the center. Therefore, the spiral maximizes the probability that the loop short circuiting optimization will stop the search early on.
- Dynamic pyramiding allows the use of a subsampled image when the features are accelerating. The subsampling allows a greater image area to be analyzed in the same time as a smaller unity sampled area. This approach sacrifices accuracy for speed when needed. Several levels of pyramiding are used, and the system picks the lowest level (highest accuracy) possible at any given moment.

Features may be automatically selected and re-selected in this methodology. This adds significantly to system robustness since new features are chosen as old ones become noisy or leave the field of view. The reselection process is run at a lower frequency than the rest of the visual servoing system since it is not expected that new features will be needed on each cycle. In the reselection process, new features within the bounding box of the current features (this helps to ensure that the new features are part of the object being tracked) are proposed. For the special case of objects with highly irregular shape against a cluttered background, the automatic feature selection procedure must be disabled. Each new feature is evaluated with a confidence measure that attempts to measure the quality of the feature. There are a variety of possible confidence measures for features.

One such measure evaluates how effective the SSD measure used during servoing will be. To do this, the auto-correlation technique is used. The SSD is computed for the feature applied to a neighborhood about itself, producing a SSD surface centered about the feature point. It is desired that the cross sections of this surface are parabolic in shape. This shape helps ensure that the SSD computations made during tracking will converge to single solutions quickly. Features that are superior to the current features are selected and used for continued tracking.

The overall result of the optical flow computation is a measure of the velocity of the features. This is the input to the visual servoing controller.

Control Section

The control section is a trajectory planner and adaptive controller that takes feature positions and computes new cartesian objectives for the robot. The trajectory planner determines the trajectories for each feature and provides a new desired trajectory location to the controller on each cycle. In general, the objective is to move the features towards the center of the image.

The system model is developed around the governing equations for optical flow. A unique aspect of the model is that only camera data sheet information is required to build it. No camera calibration is needed; rather, modeling inaccuracies are handled

as white noise. A control law is selected to minimize control change and error. Depth of the object at each feature point is implicit in this model. In the general (and in fact most useful) case(s), depth is unknown. To solve this problem, a standard adaptive recursive estimation scheme is employed²⁹.

The output of the control section is a command to the robot's cartesian space controller (in end effector coordinates).

Summary

A block diagram of the system is shown in Figure 7. Note that the structure of the feature selection mechanism would allow either a user or an automated feature detector to provide an initial tracking feature.

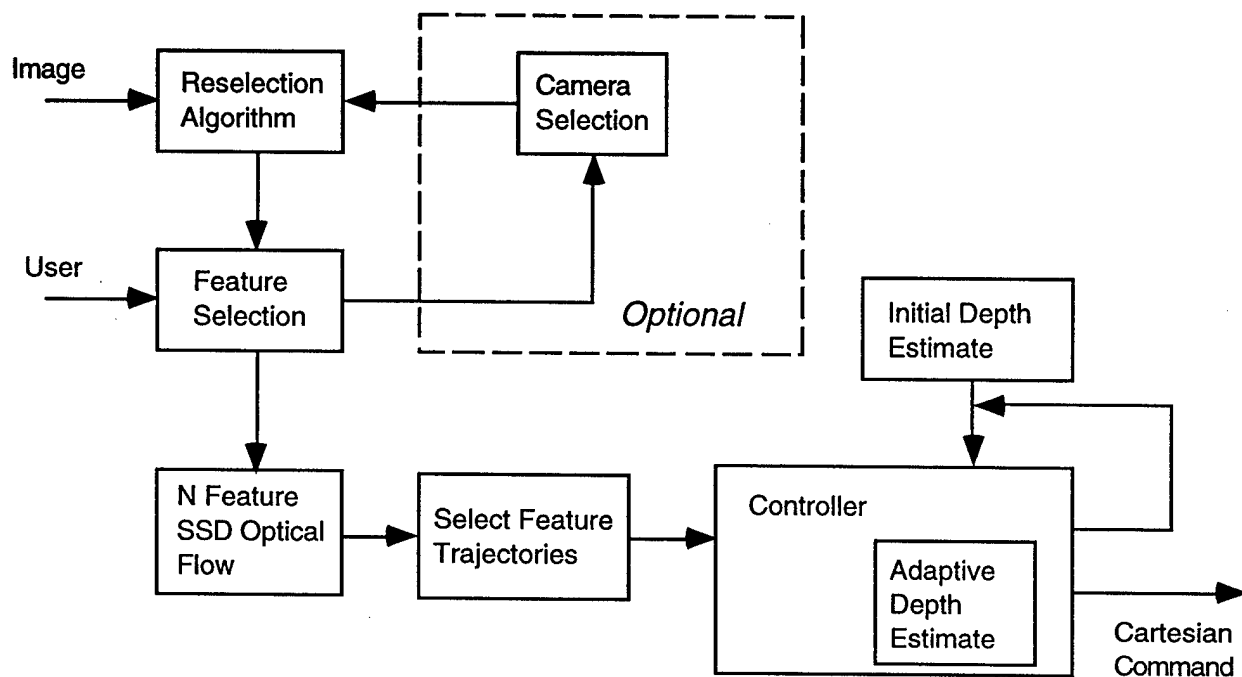


Figure 7. Optical Flow Algorithm

This particular approach to optical flow has been demonstrated both at Carnegie Mellon University and the University of Minnesota. It has been shown to be robust against lighting variations and background clutter. In addition, it provides for depth recovery and is able to work with moving objects. The most significant limitation is that it is computationally intensive and requires dedicated high-speed processors.

²⁹Ibid. 26

5.3. Selection of Approach

The previous section presented three visual servoing methods that are potential solutions for the system specified in Chapter 4. In this section, these three are compared, and the "best" approach is selected. First, it is noted that any of these techniques satisfy the demands (or firm requirements) listed in the specification, Figure 3. It is the subjective requirements, or wants, that differentiate the techniques. To evaluate the three approaches, the weighting matrix developed in Chapter 3 (Figure 1) is used. The rows in this matrix are the subjectively phrased wants. These are used, rather than the quantitative versions in the specification, because real experimental data is not available for all three methods. A simple rating scheme is used to highlight the differences – a "3" is given to the best approach, a "2" to the second best, and a "1" to the worst. The score on each item is multiplied by the weight from Figure 1, and the resulting values are totaled to obtain a total score for each technique. The normalized total corrects the results to a basis of 100 points.

Optical flow is the apparent preferred approach by this analysis. In addition, it has a feature that is not measured by the weighting matrix – it provides a relatively accurate estimate of the distance to the object. As proposed, neither of the other methods provide this. Therefore, optical flow is the preferred methodology for continued development. The next chapter discusses some of the more detailed issues related to a practical implementation of an optical flow based visual servoing system. However, before looking into the implementation details, the optical flow visual servoing technique is compared to the currently available commercial alternatives.

Requirement	Weight	Correlation	Edge	Optical Flow
System should be able to function as a testbed for visual servoing techniques	10	3	3	3
System should work with both mobile robots and manipulators	8	3	3	3
Robust against variations in lighting	14	1	2	3
Robust against background clutter	18	2	1	3
Simple user interface	6	3	3	3
Black-box	0	3	3	3
Easy to integrate into existing robots	14	3	3	3
Wide range of objects that can be tracked	12	3	2	3
Inexpensive add-on to existing systems	4	2	3	2
Uses proven technology wherever possible	4	2	2	3
Objects may be in motion	10	3	3	3
Raw Total		28	28	32
Weighted Total		246	234	296
Normalized Total		82	78	99

Figure 8. Servoing Methods Comparison

5.4. Comparison with Competing Products

It is important to verify that the proposed visual servoing system will be able to compete effectively in the marketplace. This is done here by comparing competing products to visual servoing. There are no known direct (same technology) competitors. However, there are alternate technologies that provide some aspects of the functionality, such as static vision, and pre-programmed motions. These are used as the competition for visual servoing. The ranking method used in Section 5.3 is used here. Note that this only compares the wants from the specification and not the demands. An analysis shows that the demands listed in Chapter 4 can be met by any of the methods under consideration. The wants are where the real differentiation lies. This comparison only considers the relative performance on a task that could conceivably be done by any of the three methods. There are other

classes of tasks, discussed in Chapter 2, that can only be automated with visual servoing.

Figure 9 shows the results of the comparison. Visual servoing is the apparent favorite using either the raw or weighted totals. While different users may have different weights and criteria, this analysis does show that a well-implemented visual servoing system could be competitive.

Requirement	Weight	Pre-programmed motion	Static Vision	Optical Flow
System should be able to function as a testbed for visual servoing techniques	10	2	2	3
System should work with both mobile robots and manipulators	8	3	3	3
Robust against variations in lighting	14	3	2	3
Robust against background clutter	18	2	1	3
Simple user interface	6	1	2	3
Black-box	0	3	3	3
Easy to integrate into existing robots	14	3	2	2
Wide range of objects that can be tracked	12	1	2	3
Inexpensive add-on to existing systems	4	3	2	1
Uses proven technology wherever possible	4	3	2	1
Objects may be in motion	10	1	2	3
Raw Total		25	23	28
Weighted Total		216	190	270
Normalized Total		72	63	90

Figure 9. Competitive Technique Comparison

Chapter 6: Implementation of Optical Flow Based Visual Servoing

The visual servoing system proposed here is based heavily on work currently being done at the University of Minnesota and has many good characteristics and competitive advantages over the alternatives. These have been discussed in the previous chapter and are summarized here in a concise form:

- Ability to use any unique intensity pattern as a feature
- Robust in varying lighting conditions
- Robust against background clutter
- Camera calibration is not necessary
- Distance to the object is estimated
- Can track and servo to static and moving objects

The major limitation is that the processing is computationally intensive. The amount of computer power necessary is related to the task. Static objects require the least computation, while moving objects and more features require increasing computer power. To minimize the impact of the computational burden, two versions of the system are envisioned:

1. **Static System.** The characteristic that distinguishes this system is that it can only guide a manipulator to objects that are stationary. Note that in this

context, "static" refers to the stationary target not the camera which moves with the robot. The main application of this is teleoperation. In this mode, the operator locates an object he wishes to get closer to and instructs the manipulator to servo to it. This system requires minimal computing power and is, therefore, the least expensive option.

2. **Dynamic System.** This version allows the target object to be in motion in a plane. This system is useful for autonomous operation of repetitive tasks, such as picking up objects as they move past the manipulator. Increased performance (within the limits of the manipulator, optical system, and image digitization speed) can be obtained by adding computing power to the system.

In the remainder of this Chapter, the technical issues and hardware involved in the implementation of this system are discussed.

6.1. Technical Issues & Solutions

There are a number of challenges associated with making the visual servoing system work in a real environment. Here, each one is described and potential strategies for solving the problems are presented.

- **Task Decomposition and Grasp Planning.** The visual servoing methodology described in the preceding chapter is a relatively low-level functionality in a robotic system. In order to take full advantage of its capabilities, the ability to decompose tasks into basic elements is necessary. In particular, there are several distinct steps involved in grasping objects. Many task decomposition schemes have been proposed in the literature. For purposes of the visual servoing system, a simple methodology is planned.

The key to the task decomposition method is the task data structure as shown in Figure 10. Each task consists of a linked list of *Actions*. An action-specific data structure is linked to each action element. The initial action types planned include:

- **Features:** A feature action describes a *set* of features that the system should center on and servo towards. The action-specific structure contains the initial features and their location relative to the centroid of the previous feature (if any).
- **Pre-grasp:** The pre-grasp action is a visual servoing action executed just before grasping. The major difference between pre-grasp and feature is that pre-grasp first centers the features and then approaches them, rather than performing both operations simultaneously. The pre-grasp action-specific data also adds a stopping distance from the object to the feature data.

- **Grasping:** The grasping action can be a blind operation (if necessary), and the data structure contains the offsets for grasping the object. Thus, the proper grasping position must be provided to the system at setup.

Additional action types may be added later to enhance the capabilities of the system, such as force control, blind move, etc. Actions also contain a termination condition. This condition tells the system when the current action is complete and the new one should be started. If a new action cannot be started, an error is raised and the system stops.

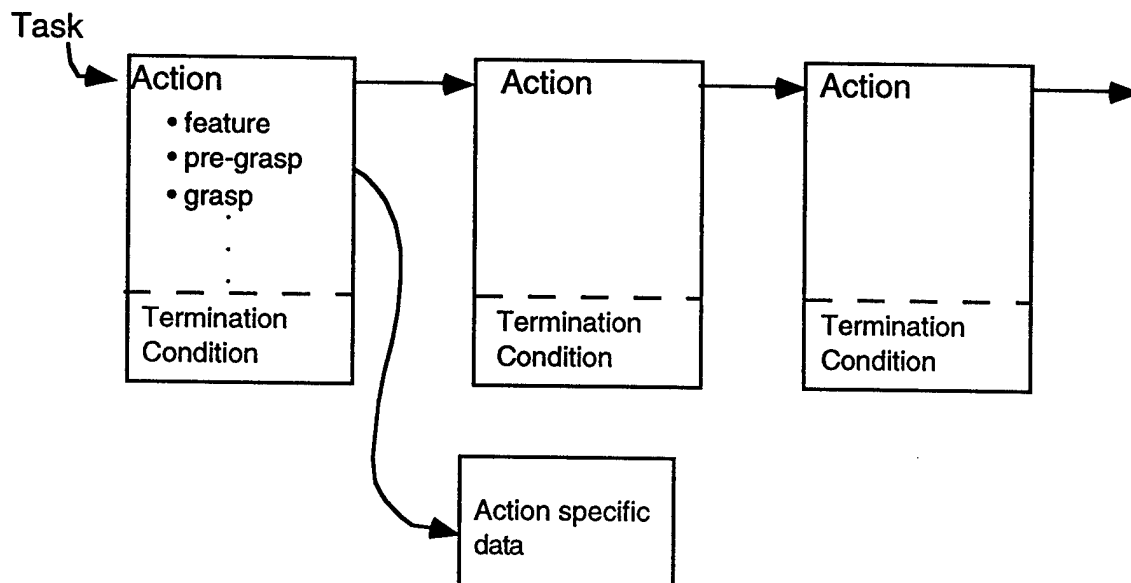


Figure 10. Task Data Structure

The actual task decomposition process is shown in Figure 11. It simply traverses the data structure and performs the steps associated with each action using the action-specific data to guide the input to the robot controller. This structure allows for flexible and sophisticated grasping strategies. As an example of task decomposition, consider how you decompose and execute the process of entering your house. First, you approach the house using either the whole house or some section as your guide for approach. Once you are near the house, you focus on the front door. Finally, when you are at the door, you focus on the door knob, ignoring the rest of the door. A similar set of strategies may be employed while visually servoing a robot. First, the system servos to a set of coarse features on the object. As those approach the edges of the image, a new finer set of features is used. Each set of features is represented by an action in the task decomposition scheme. Transition between feature actions is automatically handled by the system when the quality of the current features degrades to a lower total confidence level than the features associated with the next action. Other action types have different termination conditions, such as distance to the object, or grasping the object.

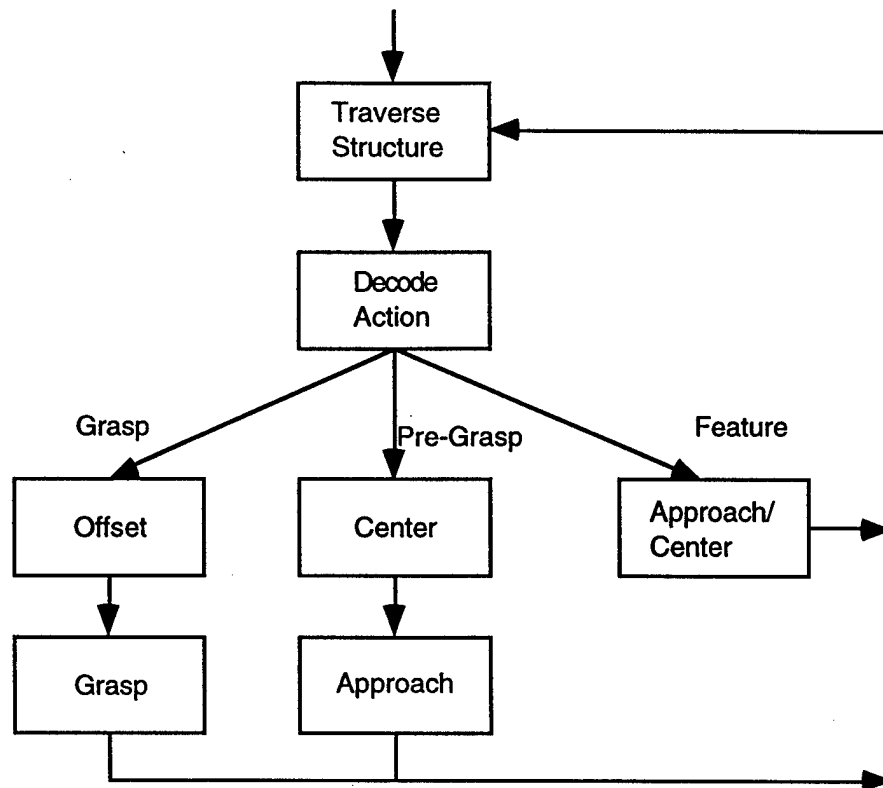


Figure 11. Task Decomposition

- **Delays.** Some robot controllers delay a cycle or more before acting on a new position command. This delay reduces the maximum possible tracking speed and destabilizes the system. Robustness in the control aspects of the visual servoing keeps the system stable. However, high tracking speeds require minimal control delay. This is a problem that cannot be solved within the visual servoing system and must be addressed in the robot controller. Thus, the visual servoing system may exhibit different performance limitations with different robot controllers.
- **Focal length limitations.** Certain applications, most notably teleoperation, require a large dynamic range in working distance. This presents an optical challenge. Both the proper focal length and focus change with varying distance to the object. Even if focal length is assumed to not be a problem, the focus may vary considerably over a long distance. Auto-focusing mechanisms could be employed, but they would introduce their own problems including aiming them at the desired object and synchronizing focus changes with image acquisition to avoid destabilizing the visual servoing system.

A simpler solution is to use two cameras and lenses. One is a medium to long focal length and the other a wide angle lens. The long lens is used at first, and the wider lens is used as the robot approaches the object. The challenge is in switching between the two cameras. First, the system must be able to detect the

same features in the wider view as it is seeing in the long view. This may be accomplished by synthetically generating features for the wide view from the original view. These new features are re-sampled (interpolated) versions of the original features. The re-sampling interval is determined by the ratio between the focal lengths of the two lenses. Once the features are detected in the wide view, confidence measures are compared for each feature in the two views. When the confidence measures in the wide view are higher, the system switches to that view for continued tracking.

- **Non-linearities.** The most significant non-linearity that affects the visual servoing system is the coupling of rotational and linear motions. In particular, pitch and yaw motions in cartesian space may produce translation in the end effector. Explicit handling of these non-linearities would increase the computational complexity tremendously. For the classes of applications that the current system is designed, cartesian pitch and yaw motions at the end effector are generally not needed while visually servoing to the object. The system would normally servo to the object with the optical axis of the camera (and presumably the end effector Z axis) perpendicular to the object. Any pitch or yaw that is necessary for manipulation would be performed after reaching the object. Roll, however, is easily handled by the system since it does not produce translation.
- **Distance Estimation Robustness.** Distance to the object is estimated by the control portion of the visual servoing system. This estimate is prone to slight variations over time, particularly when switching between features. Additional filtering may be needed to improve this. For static systems, a delayed version of the control output of the visual servoing system could be used as an input to the filter, since the distance to the object should only change by the distance that the robot moves. In the dynamic system, the same technique could be used, but it would not be able to filter as accurately since the object may be moving. For this situation, further tuning of the estimation scheme or possibly different estimation schemes will be necessary.
- **Integration of additional sensors.** Many tasks are best performed with the aid of multiple sensors. The general task control framework introduced above provides a mechanism for integrating actions that use different or multiple sensors. For example, a proximity sensor may provide a signal designed to satisfy the termination condition of a visually servoed pre-grasp action.
- **Exception handling.** The most significant exceptions that could occur in this system are errors in locating features. This could be due to a temporary occlusion of the object or an extreme lighting variation. Frequent computation of the feature confidence measures will help to detect these events. If the confidence is too low, the system will recognize that the feature is invalid. If the system knows where the object should be, it may attempt to automatically reselect features. If it does not, it would stop operation until the problem is

corrected. When practical (such as in teleoperation situations), an operator could also monitor the system's feature detection process and stop or help the system should problems arise.

- **Dynamically detecting the presence of moving objects.** Dynamic detection is a process whereby the computer automatically determines that the object that it wants to track is in the field of view. Possible approaches include:
 1. Correlation of the desired feature with the image
 2. Computing localized intensity variation rates to find target candidates that are moving at a rate significantly different than the entire image.
 3. An enhancement of the *figure/ground*³⁰ based technique that attempts to differentiate portions of the image that are moving at a different speed than the background. The *figure/ground* approach is one of a family of image-differencing algorithms. In the case of detection, it is helpful to view an image as a set of pixels that belong to one of two categories: *figure* or *ground*. *Figure* pixels are those that are believed to belong to one of several moving objects, while *ground* pixels belong to the surrounding environment. *Figures* are detected by subtracting new images from a stored *ground* image. The resulting image may then be thresholded and segmented to further isolate *figures*.

The enhancement would require slow selective updating of the background image via a formula such as $G[n+1] = G[n] + \alpha I[n]$, where G is the ground image and I is the new image. The parameter α controls how much that the new image influences the ground image.

The long-term plan for a visual servoing product includes the solution of all of these issues. However, only a subset will be handled during the Phase II work. The plan for Phase II is presented in detail in Chapter 7

6.2. Hardware Implementation

The hardware consists of a video camera and image processor. The video camera may be any moderate quality NTSC CCD video camera coupled with a lens appropriate to the application. The image processing hardware digitizes the video image and performs the optical flow computations. The remainder of this section discusses the options for the hardware.

³⁰C.A. Richards and N.P. Papanikolopoulos, *The Automatic Detection and Visual Tracking of Moving Objects by Eye-in-Hand Robotic Systems*, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, August 1995.

The image processing hardware will consist of a stand-alone PC and add-in boards. The add-in boards will provide image capture and high-speed image processing. In this configuration, the host PC will primarily serve as an interface to the user and robot and will perform little computation. Commercial PC compatible boards are available to provide the image acquisition and processing, so no custom hardware is anticipated.

Experimental results at the University of Minnesota have shown that the static system requires approximately 10 MFLOPs of general purpose processing, while a full featured dynamic system would require 80 – 100 MFLOPs. Both general purpose and specialized processors are available for the PC. The general purpose processors are generally Intel i860 or Texas Instrument TMS 320C family DSP chips. Specialized processors are optimized to provide convolution, histogram, or other common vision operations at accelerated throughput rates. Since the processing required by the visual servoing system is somewhat non-standard and can be optimized by altering the standard computational methods, general processors are the preferred choice. Specialized processors would limit the options available in implementation and might be appropriate in a later version of the system after the core development work is complete when product cost concerns override implementation flexibility.

General purpose add-in processors can interface with image acquisition boards either through the bus in the PC (either PCI or EISA) or through a proprietary bus. The EISA bus is tied to the host processor and is relatively inefficient. Therefore, processors that are considered communicate either over the PCI bus or a dedicated image transfer bus.

In evaluating the boards, the following properties are considered:

1. Performance. There are two critical parameters:
 - A. Image Transfer: Time from initiation of an image capture to when it is available for further processing. Some boards take considerably longer than a frame time to accomplish this. This figure is not typically published by manufacturers, but is critical to the performance of the visual servoing application.
 - B. Computational speed: The critical factor is the time it takes for a complete cycle of image processing and control computations. This can be evaluated by benchmarking the true sustained floating point computation speed of the processor.
2. Cost
3. Quality of function libraries
4. Bus availability (PCI, EISA, etc.)
5. Processor
6. Memory size
7. Ability to use multiple boards in parallel for increased processing power

8. Separate image buffer for superimposing graphics on the video image

Figure 12 shows several of the options currently on the market. Sharp's image processing board is not included in the table because a preliminary review of the board's capabilities indicate that while it is a very good processing board, it lacks the high speed general-purpose processing necessary for the specialized algorithms in visual servoing.

A price/performance ratio is computed in the third row of the table to show the relative value of each board combination. This computation does not take into account the image transfer time between the frame grabber and processing boards that is necessary for some of the options listed. For board combinations that require an intermediate data transfer between frame grabber and image processor, the image bus speed is listed. Based on the price/performance and the bus transfer time, it appears that either the Epix Model 12/COC402 or the Imaging Technology MVC IC combined with the Alacron FT200 would be best. The lowest price/performance combination, the Data Translation DT2867-LC/Alacron AL860AT, is hobbled by the 10MB/sec (~25 msec per image) image transfer rate. Since computing hardware choices change very rapidly, it is not appropriate to select the final hardware this far in advance of the Phase II work.

Frame grabber/image processor	Epix Model 12/COC402	Data Translation DT2867-LC/Alacron AL860AT	Data Translation DT2867-LC/DT2878	Imaging Technology MVC IC/Alacron FT200	Imaging Technology MVC 150/40 & CM-PA
Advertised Performance	50 MFLOP single processor 100 MFLOP dual processor	80 MFLOP	25 MFLOP	100 MFLOP single processor 200 MFLOP dual processor	25 MFLOP ³¹
Cost³²	\$8,000 - single \$12,000 - dual	\$7,000	\$7,500	12,000 single +\$15,000 dual	\$8500
Price/Performance (\$/MFLOP)	160 single 120 dual	87	300	120 single 75 dual	340
Function Libraries	Yes	Yes	Extensive	Yes	Yes w/graphical interface
Image bus speed	N/A	10 MB/sec	10 MB/sec	120 MB/sec over PCI bus	40 MB/sec
Buses	ISA or EISA	EISA (VME also available)	EISA	PCI (VME also available)	EISA/PCI(VME also available)
Processor	TMS320C25 plus single or dual 50 MHz TMS320C40	40 MHz i860	AT&T WE DWP32C	single or dual 50MHz i860	40 MHz TMS320C31 Multiple processors may be pipelined
Memory	4M, expandable to 256M	4M, expandable to 64M	4M, expandable to 8M	2-4M frame grabber 8-32M processor	3M, expandable
Multiple Boards	Up to 8	Via daisy chain	Via daisy chain	Yes	Yes
Graphics Buffer	Configurable	Yes	Yes	No	Yes

Figure 12. Image Acquisition and Processing Boards

³¹Estimated from benchmarks

³²Costs are for 4M or smallest memory configuration available, whichever is larger

Chapter 7: Development Plan

In Phase II, we propose to bring the development described in Chapter 6 to a full-scale demonstration. To do this, the following objectives must be achieved:

- Develop PC based visual servoing hardware and software
- Transfer University of Minnesota visual servoing technology to RedZone Robotics, Inc.
- Provide a robust user interface for using and monitoring the system progress
- Demonstrate the transportability of the visual servoing system to multiple robot platforms
- Improve speed and robustness of the system

The work plan, described in detail below, is a series of steps that build the elements necessary for an integrated ammunition handling demonstration.

Task 1: Review Phase I Results

This task will start with the project kick-off meeting at ARDEC. At this time, the status of the design, any applicable new technology, and ARDEC's objectives will be reviewed. These new objectives will be used to refine the work plan and schedule.

This time will also be used to update our knowledge of key components of the system and initiate a technology transfer mechanism with the University of

Minnesota. Much of the theoretical framework and mathematical formulation for our preferred technical approach to visual servoing has been worked out at the University of Minnesota. By leveraging this technology base and developing it for practical applications, RedZone can bring visual servoing to the market faster and with lower risk than if a complete ground-up development were undertaken.

The outcome of this task will be a refined set of objectives and an updated project schedule reflecting those objectives.

Task 2: Select Hardware

Hardware selection began in Phase I. The Phase I result proved that appropriate hardware is available and suggested some possible platforms and components. This task extends that work in a four step process that results in a final hardware design.

Revisit Phase I results

First, the Phase I results will be reviewed for use as guidance in hardware selection. Since it is likely that ~1 year will have passed between the Phase I work and this task, a brief market survey will be conducted to find any new hardware that has entered the market. In addition, the availability of the options found in Phase I will be confirmed. The same criteria used in Phase I will be used to identify the best hardware for further evaluation.

Benchmark hardware

The best 2 or 3 frame grabbers and accelerator cards will be obtained for benchmarking purposes. There are two critical performance benchmarks:

- Image Transfer: Time from initiation of an image capture to when it is available for further processing.
- Computational speed: The critical factor is the time it takes for a typical single cycle of image processing and control computations, disregarding the time it takes to acquire, store and transfer images. This can be evaluated by benchmarking the true sustained floating point computation speed of the processor.

In this task, appropriate benchmarks for these parameters will be developed or purchased and applied to each system under consideration.

Develop final hardware design

The final hardware selection and design will be a trade-off of the issues identified in Section 6.2. An evaluation will be made of each option against these and any other relevant criteria. This task will result in a system diagram and a bill of materials for

the hardware involved in this system. It is not anticipated that any custom hardware will be necessary.

Purchase and assemble hardware

Hardware selected in the previous subtask will be purchased and assembled in a PC at RedZone. Any necessary hardware configuration work will also be done in this task.

Task 3: Develop Application Software

The application software is the core of this project and includes all of the visual servoing imaging and control software. The imaging software handles the feature finding and optical flow computations, while the control software computes manipulator positions based on the desired trajectories of each feature.

Detailed Design

The software development task starts off with a detailed design of the software for the visual servoing controller. Block diagrams of the system were presented in Chapter 5. These will serve as the starting point for a more detailed design. Allowances will be made in the design for the planned addition of a user interface and task decomposition.

The detailed design will include a complete software block diagram showing all modules. In a separate document, the input and output for each module will be specified, along with the processing that is performed in the module. University of Minnesota researchers will assist in this design for those parts of the system that are directly related to their visual servoing work.

Port Software

In this task, the University of Minnesota software will be ported to the new PC based hardware at RedZone. This involves approximately 5000 lines of code that are currently running on the i860 processor in a DataCube Maxtower. It will be ported to the image accelerator processor selected in the previous task. All code will be written in C and will operate under the board manufacturer's operating system (if any). If appropriate, some code may run on the PC host processor (a Pentium or its most current replacement) for additional parallelism. It is expected that Windows NT will be the operating system used on the PC.

During the porting process, improved exception handling will be added to the system. Graduate students familiar with the code will assist RedZone engineers in this effort. This, along with the design task, will be the principal point for technology transfer from the U of M to RedZone. At the end of this task, not only

will there be a functional system at RedZone, but RedZone engineers will understand all fundamental aspects of the system.

Task 4: Develop User Interface

The user interface must provide several levels of functionality:

1. Monitoring display that shows the placement of features during autonomous operation.
2. Setup functions that allow the user to select features and associate them with particular objects and actions.
3. Teleoperation functions that allow the user to easily select an object and servo to it. This function is virtually a combination of the previous two functions.

The specification (Figure 3) dictates that the user interface should be easy to use, requiring no more than 10 seconds and 3 mouse operations to do any operation. These guidelines will be used to first design a user interface, and secondly implement it. The implementation will be on the host processor of the PC that houses the imaging cards. It is expected that the user interface software will be implemented in Visual Basic or C and will operate under Windows NT. These final implementation decisions will be made at the beginning of this task and will depend on the final imaging hardware selected in Task 2.

Task 5: Static Grasping Demonstration

A demonstration at RedZone's Pittsburgh facility will show the progress of the project so far. The demonstration will use the new visual servoing hardware and software developed in the preceding three tasks, and a Puma robot on loan from the ARDEC robotics laboratory. The demonstration will involve the Puma grasping the lid to a shell pallet (or similar object) under visual servoing control. Since this capability will be used again in the integrated demonstration (Task 10), this task serves as a building block for later work. Leading up to the demonstration itself, the following subtasks will occur:

1. Receiving and unpacking the Puma
2. Mounting the video camera on the Puma
3. Develop and debug the cartesian command interface to the Puma
4. Test and refine the visual servoing system

Task 6: Task Decomposition System

Practical application of the visual servoing system in an automation environment must allow the robot to grasp the object that it is servoing to. Since different objects have different grasping requirements, some form of intelligent planning is needed. Fully autonomous grasp planning based on geometric properties of objects derived from observation is still an active area of research, and implementing such a system is beyond the scope and budget constraints of this project. However, a practical methodology that will allow the system to function in real environments can be implemented within the task decomposition framework described in Chapter 6.

The control system progresses through all three *actions* described in Chapter 6:

1. One or more feature actions are used, centering and approaching the object simultaneously.
2. A pre-grasp action is executed, centering the tool before approaching to make sure that the gripper does not contact a portion of the object as it approaches.
3. A grasp action is used when the distance to the object reaches a pre-determined minima, the visual servoing process will end, and commands will be issued to the robot controller to position the gripper in the proper relationship to the servoed features. The robot will then complete the approach and grasp the object

A particular grasp is associated with each object. The grasp describes the type of gripper and the geometric relationship between the grasping points and the feature set that describes the object (a typical object will be tracked using 3-4 features). This approach is complicated by dynamic feature reselection. Since the dynamically selected features are not in pre-determined locations relative to the user determined features, the system must compute the relationship between any dynamically selected feature relative to the original feature set.

This sequence will be encapsulated in the task decomposition structure described in Section 6.1. The focus of this task will be the implementation of the task decomposition data structure traversal and action processing aspect of task decomposition. It will be implemented in the C programming language on the host PC to the image processing system. Testing will be done on the Puma test setup assembled in the previous task.

Task 7: Distance Estimator Testing and Refinement

Disturbances in the distance estimation scheme can cause uneven motion as a robot approaches an object. For static objects, additional filtering taking into account robot motion can be applied to the results of the distance estimator for smoothing. Dynamic objects will not be explicitly examined but will likely benefit from the

algorithm improvements. This task will examine estimation through the following steps:

1. Perform experiments on the Puma test system to measure the quality of the depth estimates on static objects. There are two significant contributors to measurement quality:
 - A. Absolute error in the depth estimate, as compared to a physically measured distance.
 - B. Sign errors in the slope of the depth estimate. This would occur when the estimated depth is changing in the opposite direction from the actual depth.
2. Propose additional filtering, if necessary.
3. Test and refine the filters.

Task 8: Application to Mobile Robots

Visual servoing will be used with Picatinny's indoor mobile robot for automated docking. This is considered a static application since the docking location will not be moving, and is similar to typical applications for teleoperated machines. Accurate depth estimation is an important part of a smooth docking procedure, so this task will serve as further verification of the results of the previous task.

The following steps are required:

1. Establish communications link between the visual servoing system and the robot controller.
2. Install the video camera on the robot. The video camera will be linked to the off-board visual servoing system via a tether.
3. Record the docking features and positions in the task data structure.
4. Verify and tune performance. Note that this will be done in a general way that enhances the overall system performance, and will not be specific tuning to the application.

Preparations for this task will be done at RedZone, but the actual installation and testing will be done by 1-2 RedZone engineers at the Picatinny facility.

Task 9: Speed Improvements

There are two types of speed improvements – processing speed and tracking speed. These two issues are partially, but not completely related. Improved processing speed will provide the computing power to do automatic feature selection which

will increase the system robustness in the presence of noise and occlusions. Automatic feature selection is a computationally intensive process that must compete with the servo computations for resources. Speed improvements will also allow for high system bandwidth, improving the maximum possible tracking speed.

Tracking speed may be limited by processing speed, continuous image acquisition rate, robot controller response, and camera positioning. Processing speed and continuous image acquisition rate affect how frequently a new image can be taken and used. If images are acquired infrequently, then objects can move too far across the field of view for efficient tracking. The ultimate objective is to hold processing to frame rates or a little better. The robot controller can cause a more serious bottleneck than the image processing, though. Typical robot controllers can only accept new position commands at pre-determined intervals. For Puma robots this is 28 msec. Generating new control commands faster than this does not help. In addition, sometimes there are additional delays in the robot controller before the command is acted on. These delays all limit the ultimate tracking speed obtainable. Finally, the apparent velocity of the feature in the image plane is dependent on the camera placement and optics. Objects that appear further away will also appear to move more slowly, so proper location of the camera can enhance tracking speed.

To address these issues, several improvements to the system are needed. First, a robot with a better control interface than the Puma is necessary. The Robotics Research arm installed in the ARDEC Robotics Laboratory can provide this. Secondly, processing speed must be optimized. This will be done through:

1. Careful allocation of tasks across the image processors and the host processor
2. Finding and implementing additional software optimization. The most likely place to find improvements is in the automatic feature selection process. Possibilities include adding loop short-circuiting to the auto-correlation methods and new search patterns similar to those used for the SSD computation.

Finally, careful placement of the camera on the Robotics Research arm for the planned tasks will reduce the apparent image speed.

To complete this task and verify the new maximum tracking speed, the visual servoing system will be integrated to the Robotics Research arm at Picatinny for testing and refinement.

Task 10: Integrated Demonstration

The integrated demonstration ties together all aspects of the project to date into one system that shows visual servoing operating in typical ammunition handling

applications. Figure 13 is an artist's sketch of the planned demonstration work cell.

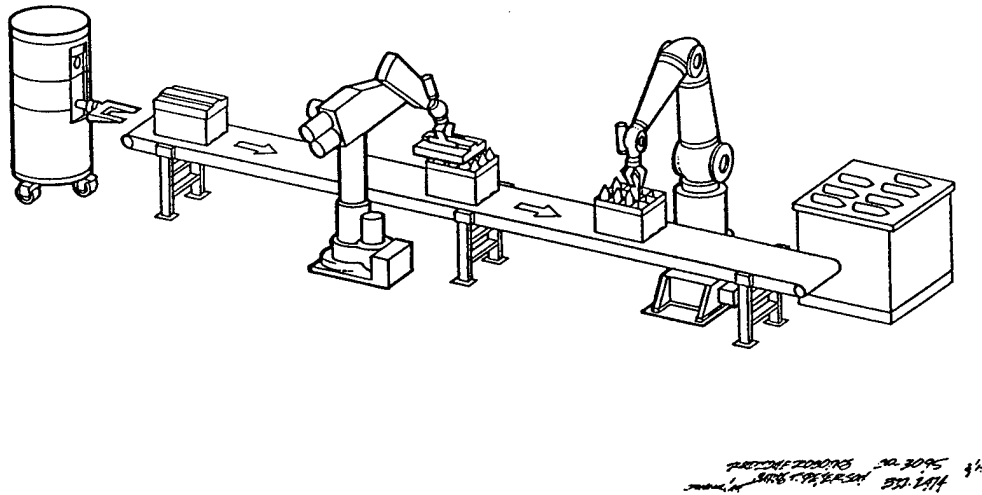


Figure 13. Demonstration Work Cell

The main elements of the system are:

1. The indoor mobile robot, operating as an autonomous pallet delivery system
2. A Puma arm to open pallets
3. A Robotics Research 7 DOF manipulator to manipulate shells
4. A conveyor belt to move the pallets past the manipulator
5. Pallets of shells
6. A rack for the fuzed shells

Figure 14 is a flow chart of the potential sequence of events in the work cell. While the sequence is not entirely realistic for any one particular application, it demonstrates the range of typical ammunition handling procedures in a compact fashion.

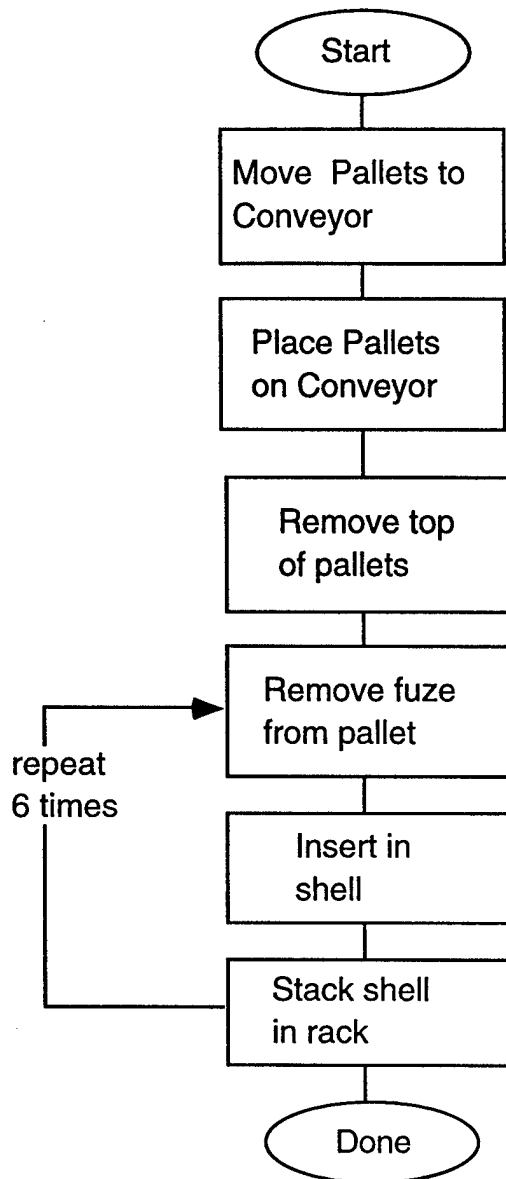


Figure 14. Demonstration Flow Chart

Each operation, except placing the pallets on the conveyor and stacking the shells, uses visual servoing as the main control function, either on the mobile robot or the Robotics Research arm. Different allocation of the tasks between the robots may be needed, depending on the results of previous tasks. A more detailed demonstration plan and scenario will be developed at the beginning of this task to reflect the current capabilities of all three robots.

This entire system will be controlled by an additional supervisory computer that coordinates the demonstration and decomposes the tasks into motion commands for each of the machines. It will communicate with the visual servoing computer via ethernet. A single visual servoing computer will switch between servoing the

robots; they will not visually servo at the same time. The hardware architecture for this system is shown in Figure 15.

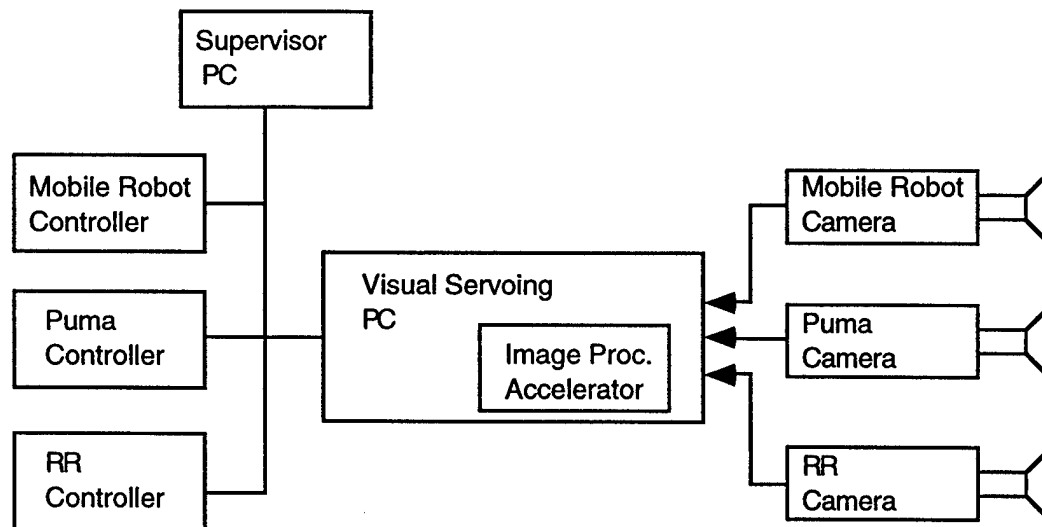


Figure 15. Control System Architecture

The following tasks lead up to the actual demonstration:

1. Modify visual servoing system to allow it to talk to multiple robot controllers
2. Develop supervisory software that follows the flowchart in Figure 14.
3. Acquire conveyor, supervisory PC, grippers, and any additional fixturing.
4. Set up the physical hardware
5. Integrate all of the elements
6. Test the system. The first test will occur at RedZone with just the Puma, the supervisor, and the conveyor. This will be used to get the initial bugs out of the system. A final test will be performed at Picatinny with the complete system.

Task 11: Final Report/User's Manual

The final report will summarize the results of the project and our plans for commercialization in Phase III. A section of the report will include a user's manual that describes procedures to setup and use the visual servoing system. Specific elements of the report will include:

- Hardware schematics
- Software design documents
- Program listings
- Test plans
- Test results
- Setup procedures
- Operation procedures
- Performance limitations
- Areas for enhancement
- Commercial follow-on work

Chapter 8: Summary

The results of this Phase I research effort indicate that the development of practical visual servoing systems is possible. Researchers have been working towards potential solutions for many years. Several techniques developed by researchers and the authors of this report were evaluated for application to real-world problems. An optical flow scheme was determined to be the preferred approach for its generality and the degree to which it has been developed. A series of enhancements were discussed in this report that will take the visual servoing scheme from the laboratory to the real world. Simple processing is the key underlying characteristic of the system and its proposed improvements. Simplicity allows for real-time speed with current computing hardware and helps to ensure robustness of the system.

The visual servoing technology we propose to develop and commercialize will have a significant impact on military and space applications, intelligent highways, manufacturing, and nuclear waste clean-up efforts. In these areas, new automated operations will be possible on moving objects, and improved price/performance of teleoperated and autonomous systems will be possible in existing applications.