AIAA Guidance, Navigation and Control Conference and Exhibit
20 - 23 August 2007, Hilton Head, South Carolina

AIAA 2007-6323

Visual Target Recognition and Tracking for Autonomous Manipulation Tasks

Michael P. Naylor,* Ella M. Atkins,† and Stephen Roderick‡

Increased levels of robotic system autonomy will enable scientific exploration in previously unreachable destinations. Dexterous manipulator systems can be coupled with machine vision systems to increase perceptive capabilities and provide greater opportunities for scientific return through environmental interaction. Reliable autonomous vision systems are often specific to a certain task and quickly become complex as new operational tasks are added. This paper describes the development of a flexible stereovision system coupled with the University of Maryland's Ranger manipulator system to identify, track, and sample arbitrary targets within the manipulator's workspace. The vision system is designed to be deployed on an autonomous underwater vehicle (AUV) that will conduct deep-sea sampling missions autonomously, where lighting and visibility constraints are formidable. Laboratory testing demonstrates the vision system's ability to repeatedly and reliably select desired targets and provide target position data to the manipulator during autonomous sampling tasks. Visual servoing results demonstrate closed-loop tracking of static and moving targets.

I. Introduction

Reliable and capable autonomous manipulation systems are in great demand for exploration in harsh, inaccessible environments. Development of such systems will allow for greater scientific return on missions where ground support, communications, and operator workload are prohibitive in terms of cost and factors such as time delay or communication bandwidth constraints. Enhanced robotic perception of the environment is a key enabler to reduced human interaction. Tasks utilizing robotic manipulators are notorious for the strain placed on human operators, both mentally and physically. Lack of sufficient camera views during teleoperation, hand strain from long-term use of hand controllers, and mental stress associated with difficult teleoperation tasks are all challenges that can be mitigated through effective automation.

The University of Maryland Space Systems Lab (SSL) has developed multiple dexterous robotic manipulator systems. Their long history of working with teleoperated systems provides a strong foundation for ongoing research in autonomous robotics. The Ranger telerobotic manipulator system has been utilized extensively in 1-G and neutral buoyancy environments to simulate on-orbit servicing of spacecraft and satellites, as shown in Figure 1. By visually characterizing the environment with accurate 3-D position data, this work augments the Ranger system to autonomously manipulate targets within its workspace. This extension will significantly reduce the workload placed on the operator while simultaneously reducing discrepancies caused by time delays, operator error, and fatigue.

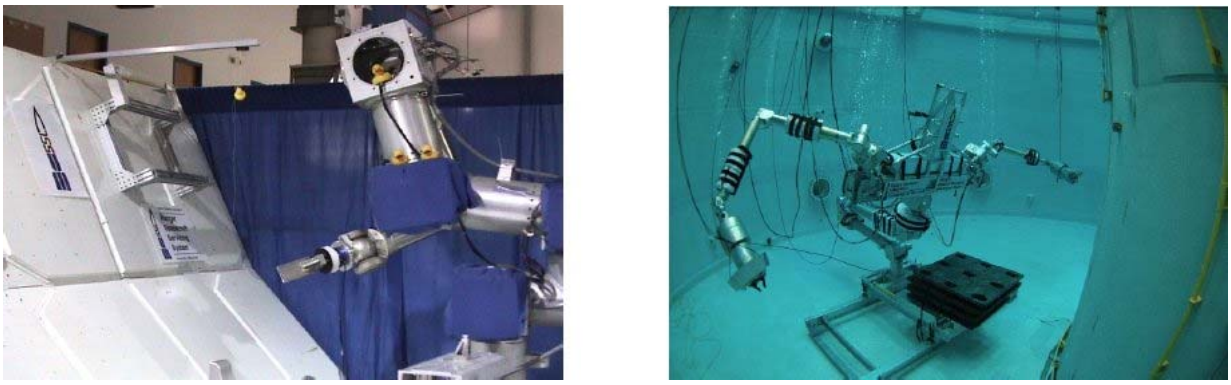


Figure 1: Ranger Configuration in 1-G (left) and Neutral Buoyancy (right).

* Engineer, Accuray, Sunnyvale, CA 94089, Member, email: mnaylor@accuray.com.
† Associate Professor, Aerospace Engineering Dept., University of Michigan, Associate Fellow, email: ematkins@umich.edu.
‡ Faculty Research Assistant, Aerospace Engineering Dept., University of Maryland, email: roderick@ssl.umd.edu.

This paper describes the development and testing of the Autonomous Vision Application for Target Acquisition and Ranging (AVATAR) system, extending our previous work[1] to enable more precise target identification and tracking over a variety of lighting conditions and target motions. System advancements allow tracking of multiple targets in the field of view as well as tracking objects of multiple types. Targets can also be tracked over time, providing the data necessary to statistically characterize localization accuracy for static targets or to follow moving targets. Underlying the vision system is a complex suite of software and hardware, augmented as a system through formal development methods. The academic result is a vision system with sufficient accuracy and precision to identify and track static and moving targets in real-time. The practical outcome of this work is a fully-integrated system that enables fully-autonomous manipulation.

AVATAR has thus far been implemented and tested on the Ranger manipulation system. In the future, this technology will transition to a lightweight dexterous manipulator attached to an autonomous underwater vehicle (AUV) developed by the Woods Hole Oceanographic Institution (WHOI). This system will be tasked with identifying and retrieving deep-sea biologic and geologic samples, requiring algorithms to account for adverse lighting conditions and potentially manipulator base (AUV) motion. This mission, as part of NASA's Astrobiology, Science and Technology for Exploring Planets (ASTEP) program, serves as an analogy to future space exploration where human teleoperation is impossible, with analogue to the proposed Europa under-ice mission.

This paper describes and evaluates the AVATAR software architecture and machine vision algorithms as well as their integration in the Ranger manipulation system. Below, the system architecture is first overviewed, focusing on communication and data transfer between modules. Next machine vision algorithms are presented, including a practical discussion of their implementation and validation in an embedded real-time computing framework. As will be shown, algorithms were designed to maximize simplicity and reliability while meeting target recognition requirements. The experimental hardware and test plan are described, followed by a presentation of test results and a brief conclusion.

II. System Architecture

The system architecture is defined in the context of a fully-autonomous deep-sea AUV-manipulator, for which remote teleoperation is not an option. This system (Figure 2) will consist of the WHOI AUV and its computer, the manipulator and its computer, the stereo camera pair, plus a dedicated vision system computer. These systems are supported by the embedded AUV and manipulator sensors, actuators, and auxiliary equipment (e.g., strobe lights, batteries) to enable robust autonomous deep-sea operation. The work described in this paper focuses on vision and manipulator subsystems. Each of these is further segmented into physical system hardware and control computer. The AVATAR software is executed on a Target Acquisition Unit (TAU) computer that connects directly to the stereo camera pair. The software supports any Firewire bus cameras. When running with Ranger, TAU shares target data through Ranger's communication protocols. Ranger is controlled via software running on the Data Management Unit (DMU) computer. While operating Ranger in teleoperation mode, the DMU interfaces directly with the operator through a set of hand controllers, for resolved rate control, or through a command interface that provides direct joint-by-joint control and access to a run-time trajectory generator.

The vision system interfaces to the DMU through a "target" trajectory item communicated through TAU during the system control loop. The visual servo trajectory implementation, described further below, uses traditional trajectory controllers (joint-by-joint or

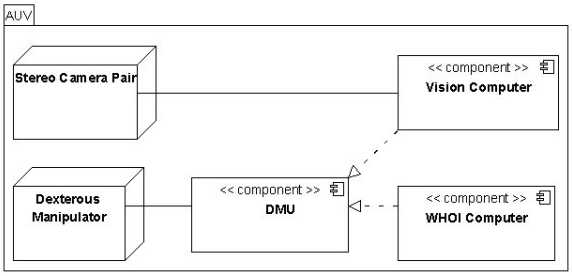


Figure 2: AUV-Manipulator Computer System.

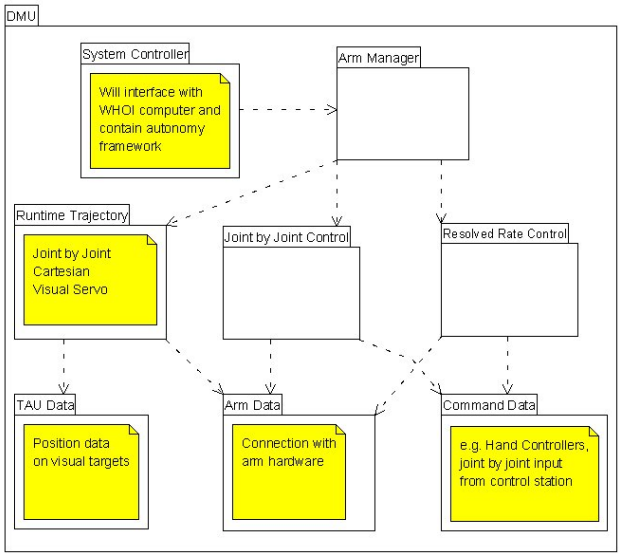


Figure 3: DMU Software Architecture.

Cartesian) to determine incremental joint angles for the arm based on the desired motion calculated by the vision system. By using Ranger's communication protocols for TAU, the implementation was straightforward. Figure 3 illustrates the components of the DMU architecture.

III. Vision Algorithms

An essential requirement for a machine vision system is an accurate set of calibration parameters. Three phases of calibration must be completed to accurately identify and sample targets from stereo camera feedback. First is the intrinsic calibration of each camera. The second step is determining the geometry of the stereo camera system (extrinsic calibration). Finally, registration between the vision system and robotic manipulator must be performed to determine the transformation between vision system and manipulator base frames. Once these three calibration steps have been completed, the software system is capable of determining 3-D locations of any target visible in both cameras' fields of view and then transforming the coordinates into a manipulator frame of reference.

The first calibration step is to mathematically estimate intrinsic camera parameters based on the correlation of unique features between 2-D image plane coordinates and known 3-D world coordinates. As is standard practice in the vision community, a planar checkerboard pattern is used to provide a matrix of readily distinguished corner features. The checkerboard is presented to each camera at a series of different orientations and positions to provide a three-dimensional set of points for calibration. The Camera Calibration Toolbox for Matlab, free software available over the Internet (http://www.vision.caltech.edu/bouguetj/calib_doc/), is used to determine both intrinsic and extrinsic calibration parameters.

The final step of the calibration is to determine the registration between the camera system and the manipulator base frame of reference. An automated registration process was developed based on an iterative minimization algorithm that determines the desired transformation from a set of corresponding points, ${}^{V}\mathbf{P}$ and ${}^{M}\mathbf{P}$, in the vision and manipulator frames of reference, respectively. First, the point cloud center positions ${}^{V}C$ and ${}^{M}C$ are calculated by averaging all points:

$$ {}^{V}C = \frac{\sum_{i=1}^{n} {}^{V}\mathbf{P}[i]}{n}, {}^{M}C = \frac{\sum_{i=1}^{n} {}^{M}\mathbf{P}[i]}{n} \tag{1} $$

Next, each point list is normalized by its center point to align both point clouds about the same center point.

$$ {}^{V}\overline{\mathbf{P}} = {}^{V}\mathbf{P} - {}^{V}C, {}^{M}\overline{\mathbf{P}} = {}^{M}\mathbf{P} - {}^{M}C \tag{2} $$

Within an iterative loop, a series of rotations about single axes are applied to align the two point clouds. During each iteration, sequential rotations about the x-axis, y-axis, and z-axis are applied. Intuitively, this algorithm is iteratively applying rotations to "reverse" the point set rotation so that the "unrotated" point sets are as close to coincident as possible. Equations (3)-(7) describe the initial rotation γ in the Y-Z plane, where initially ${}^{M}_{V}R$ is set to the 3x3 identity matrix. Similar calculations are performed for the other two rotations, β about the y-axis and α about the z-axis.

$$ \overline{m}_{V}{}^{2} \cos\gamma = \sum_{j=1}^{n} {}^{V}\overline{\mathbf{P}}[j].y \; {}^{M}\overline{\mathbf{P}}[j].y + {}^{V}\overline{\mathbf{P}}[j].z \; {}^{M}\overline{\mathbf{P}}[j].z \tag{3} $$

$$ \overline{m}_{V}{}^{2} \sin\gamma = \sum_{j=1}^{n} -\left({}^{V}\overline{\mathbf{P}}[j].z \; {}^{M}\overline{\mathbf{P}}[j].y \right) + {}^{V}\overline{\mathbf{P}}[j].y \; {}^{M}\overline{\mathbf{P}}[j].z \tag{4} $$

$$ \overline{m}_{V}{}^{2} = \sqrt{\left(\overline{m}_{V}{}^{2} \cos\gamma\right)^{2} + \left(\overline{m}_{V}{}^{2} \sin\gamma\right)^{2}} \tag{5} $$

$$ \cos\gamma = \frac{\overline{m}_{V}{}^{2} \cos\gamma}{\overline{m}_{V}{}^{2}}, \sin\gamma = \frac{\overline{m}_{V}{}^{2} \sin\gamma}{\overline{m}_{V}{}^{2}} \tag{6} $$

$$ R_{\gamma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} $$

$$ {}^{M}_{V}R = R_{\gamma} \, {}^{M}_{V}R $$

$$ {}^{V}\overline{\mathbf{P}} = R_{\gamma} \, {}^{V}\overline{\mathbf{P}} \tag{7} $$

Once the rotations have been determined, translation vector ${}^{M}\mathbf{t}_{V}$ is computed yielding final registration matrix ${}^{M}_{V}T$:

$$ {}^{M}\mathbf{t}_{V} = \frac{\sum_{i=1}^{n} {}^{M}\mathbf{P}[i] - {}^{M}_{V}\mathbf{R}\,{}^{V}\mathbf{P}[i]}{n} \tag{8} $$

$$ {}^{M}_{V}T = \begin{bmatrix} {}^{M}_{V}R & {}^{M}\mathbf{t}_{V} \\ 0 & 1 \end{bmatrix} \tag{9} $$

Due to lack of ambient light at depth and stringent power requirements for a deep-sea AUV, a lighting correction algorithm was developed to provide near true-color images rather than the blue/green images often seen in underwater photography. The algorithm used in this work is loosely based on the results from a more advanced but computationally-intensive algorithm developed at WHOI.[2,3] Our algorithm has two stages: (1) Correction of the lighting pattern created by the strobe, an intense light source brightest at the center of the image with significant radial attenuation, and (2) Correction of each color channel based on previously-calculated gains. Comparing uncorrected and corrected images using results from WHOI's procedure, the lighting pattern can be estimated and fit to exponential curves based on fit parameters c and k for each channel. In Equation (10), for each pixel location (i, j) the distance d from the center of the lighting pattern (o_x, o_y) is first calculated, then new values are supplied based on the pre-determined gains in Equation (11).

$$ d_{(i,j)} = \sqrt{(i - o_x)^2 + (j - o_y)^2} \tag{10} $$

$$ R_{(i,j)} = c_{red} e^{k_{red} d_{(i,j)}}, G_{(i,j)} = c_{green} e^{k_{green} d_{(i,j)}}, B_{(i,j)} = c_{blue} e^{k_{blue} d_{(i,j)}} \tag{11} $$

Color attenuation correction is based on the ratio of average change from an uncorrected versus a corrected image and is a simple multiplication operation for each pixel.

The remainder of AVATAR's vision process requires extracting image segments that resemble desired targets, matching these targets between corresponding images, and then utilizing stereo algorithms that calculate the 3-D location of the targets in the scene. The feature extraction process is primarily based on red-green-blue (RGB) color information and pixel area. The system functions under the assumption that a human operator provides information relevant to desired target color and size. Separate tools were developed to facilitate the estimation of filter parameters. Once appropriate parameters are determined, the remainder of the system functions autonomously. Use of the OpenCV library, another open-source tool (http://www.intel.com/technology/computing/opencv/), facilitated basic image handling as well as use of common image processing algorithms, e.g. erosion and dilation.

The parameters used for feature extraction consist of the overall magnitude M of each pixel and color channel ratios RvB, RvG, and BvG. Equations (12)-(13) show how these parameters are calculated at a given pixel $\mathbf{p}$.

$$ M = \mathbf{p}.red + \mathbf{p}.green + \mathbf{p}.blue \tag{12} $$

$$ RvG = \frac{\mathbf{p}.red}{\mathbf{p}.green}, RvB = \frac{\mathbf{p}.red}{\mathbf{p}.blue}, BvG = \frac{\mathbf{p}.blue}{\mathbf{p}.green} \tag{13} $$

In the case where an operator specifies filter parameters, a number of points are selected then all values are calculated and plotted for visual analysis. Figure 4 shows example plots of target values (red x's) vs. background values (blue o's). During operation, values from Equations (12)-(13) are calculated at each pixel location. Based on color and size reference parameters, if a pixel matches all criteria it will remain after the filter. Otherwise pixel RGB intensities are set to 0. In the case where targets are not easily distinguishable from the background, it may be necessary to also apply an erosion operator to remove noisy pixels or small features that remain after filtering. Use of a feature-AND operator[4] restores desired targets to their original state. This filter and denoise sequence is illustrated for an undersea image in Figure 5.

After extracting feature "blob" pixel values, another filter is applied based on each feature's image plane geometric properties. Feature area, aspect ratio, and area ratio are used to determine whether or not the feature is a match for the target the vision system is attempting to identify. Equations (14)-(16) are based on a list of points $\mathbf{f}$ for a single image feature. $boxArea$ represents the area of a bounding box surrounding the feature. By dividing the feature area $size(\mathbf{f})$ by $boxArea$, the area ratio is calculated. This area ratio denotes pixel packing density within the feature, which can help identify whether or not the feature matches desired target parameters.

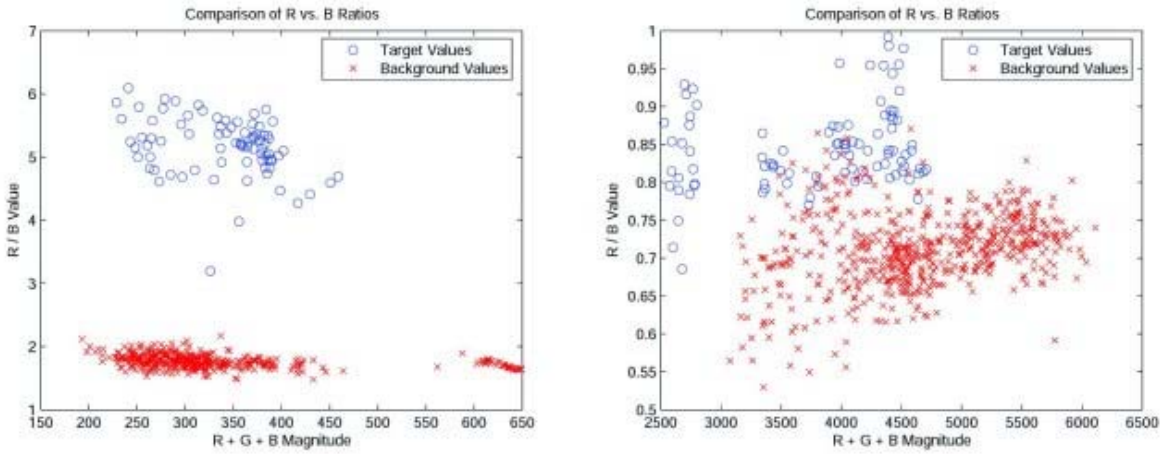
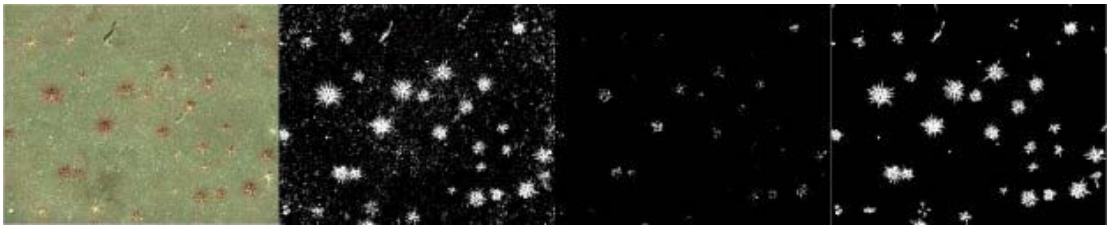


Figure 4: Plots showing color ratio data for a sample target.



(a) (b) (c) (d)

Figure 5: Image filter process (a) Original image (courtesy WHOI), (b) Filtered image based on RGB ratios, (c) Eroded image to remove noise, and (d) Image restored with feature-AND operation.

$$aspectRatio = \frac{X_{\max} - X_{\min}}{Y_{\max} - Y_{\min}} \tag{14}$$

$$boxArea = (X_{\max} - X_{\min})(Y_{\max} - Y_{\min}) \tag{15}$$

$$areaRatio = \frac{size(\mathbf{f})}{boxArea} \tag{16}$$

Once all candidate features have been extracted from both images, these features must be accurately matched across the camera pair to enable stereo triangulation. This process uses a number of points located around the perimeter of the feature to estimate shape. All perturbations of the local and global, minimum and maximum, *x* and *y* values as well as the centroid are tabulated into a list of nine points for each feature. For each of the nine points a magnitude vector **M** and direction (shape) vector **S** are calculated. Minimizing errors between features in both images, a list of good shape matches is calculated. Equations (17)-(18) show these calculations, where each **p** represents an ordered pair of (*x*,*y*) values with $i \neq j$.

$$\mathbf{M}[i][j] = \sqrt{(\mathbf{p}[i].x - \mathbf{p}[j].x)^2 + (\mathbf{p}[i].y - \mathbf{p}[j])^2} \tag{17}$$

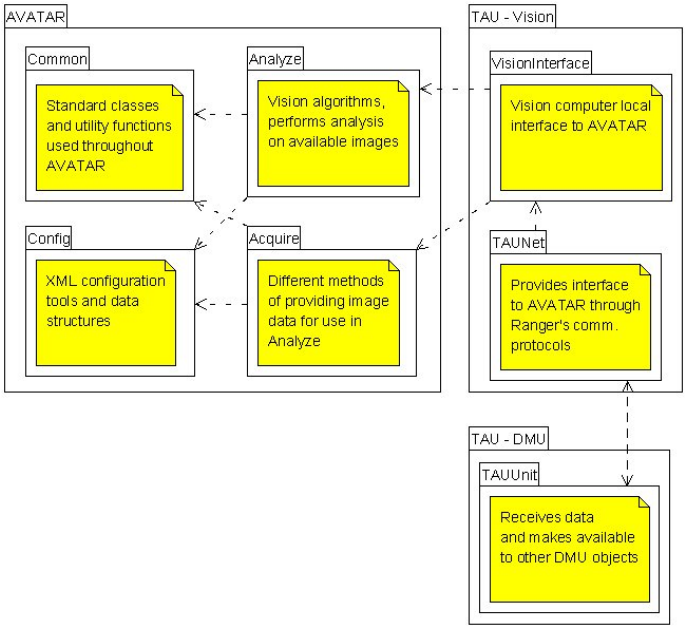
$$\mathbf{S}[i][j] = \left(\frac{\mathbf{p}[i].x - \mathbf{p}[j].x}{\mathbf{M}[i][j]}, \frac{\mathbf{p}[i].y - \mathbf{p}[j].y}{\mathbf{M}[i][j]} \right) \tag{18}$$

After the list of features has been comprehensively tested for matches between corresponding images, a stereo triangulation is performed utilizing the centroid pixel values and the camera calibration parameters. The stereo triangulation algorithm was translated to C from a function in the Camera Calibration Toolbox for Matlab.

IV. Software Architecture

Shown in Figure 6, the vision software is separated into two modules – AVATAR, the main vision processing algorithms, and TAU, the interface and communication software. AVATAR has four components: `Analyze` executes the above vision algorithms, `Acquire` handles image acquisition, `Common` provides a library of shared data structures and functions, and `Config` reads XML configuration files that describe system behavior. TAU is composed of an interface to AVATAR, `VisionInterface`, which provides access through `TAUNet`, and `TAUUnit`, used by the DMU to communicate with `TAUNet`.

The `Common` module contains the low-level building blocks for AVATAR. Each `StereoImagePair` may contain a pair of OpenCV images or a pair of virtual images mapped to raw memory on the camera bus. This class allows a corresponding pair of images to be transmitted anywhere in the system without additional overhead to distinguish left vs. right. `Common` also includes a suite of utility functions that perform tasks ranging from swapping endian values for 16-bit images to converting Bayer pattern images to RGB format. Throughout, `Common` maintains a system-wide data log.



The `Acquire` module, shown in the Figure 7 UML diagram, is split into five classes, four of which implement camera-specific methods for acquiring images and inherit common functionality from the `AcquireStereoImagePair` class. The fifth class, `AvatarCameras`, provides access to camera hardware over a firewire bus.

The `AcquireStereoImagePair` class enables unique identification of each set of acquired images. Two values, group number and member number, identify each set of acquired images. Member number is incremented every cycle, while group number is incremented when a system configuration change occurs. Thus, when either the member number reaches the group size, or a configuration change occurs, the group number is incremented and the member number is reset. `AcquireSIPFileLoader` reads data from stored 8-bit image files rather than acquiring real-time image data, facilitating offline software tests. The `RawDataAcquire` class handles raw images of any bit-depth as specified by the configuration file. The current file naming scheme contain all information required to reconstruct the correct image size and bit-depth from the raw memory block. This acquire method is used for recreating a previous test.

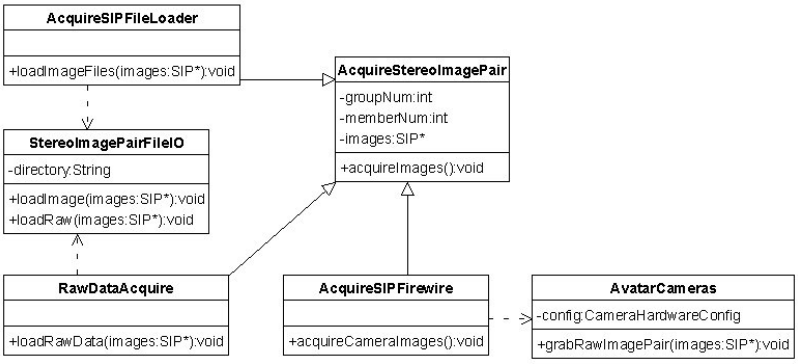


Figure 7: Class Diagram for AVATAR Acquire Module.

Once the raw data is in memory, two possible transformations may occur. The first is an endian swap; the second is a Bayer pattern correction for camera images output in a grayscale Bayer pattern rather than RGB format. `AcquireSIPFirewire` is the final derived class, enabling the user to dynamically adjust camera physical properties, e.g. exposure or white balance, to account for changing light conditions, environment changes, etc. The `AvatarCameras` class interfaces with the kernel modules, camera hardware, and the rest of `Acquire`. It includes a wrapper driver for libdc1394 that provides a configuration and acquisition interface to the cameras. `grabRawImagePair` pings the cameras then copies the appropriate image memory buffer.

The `Analyze` module (Figure 8) performs all image processing actions. The `AnalyzeStereoImagePair`, class performs target acquisition and coordinates each subsequent processing step, managing all discovered features, matching features between images, calculating the final target coordinates, and storing the data for retrieval.

To minimize execution time, certain processing algorithms are only executed depending on environment lighting and visual uniqueness of the target. For instance, with a uniquely-colored target (e.g., a yellow rubber duck) in a lab environment, color correction, erosion, and feature-AND are not required. If used, all lighting correction and color filtering algorithms have complexity *O(mxn)*, where *m* is image height and *n* is image width. Though computationally-tractable, removing unnecessary calculations decreases the per-pixel processing time. Tests performed on an Intel Core-Duo Mac Mini running at 1.8GHz with 1GB RAM had an average execution time of 1.22s for the `Analyze` routine without any form of lighting correction. The addition of both correction routines increased this execution time to 1.36s. As a high-level execution trace, the `SIPColorFilter` first sequentially performs the `ImageColorFilter` processing then extracts image features (`SIPFeatureLocator, ImageFeatureLocator`). Next, features are matched between left and right images (`SIPFeatureMatcher`). The final image processing task (`StereoCoordinateCalculator`) calculates target coordinates based on the intrinsic and extrinsic calibration parameters of the stereo vision system. Completion of the `StereoCoordinateCalculator` process populates a `VisionTargets` object, with all target data needed external to AVATAR, such as 3-D position, centroid coordinates, and object area. By providing sole access to AVATAR data through this `VisionTargets` object, AVATAR is isolated from all other interfaces.

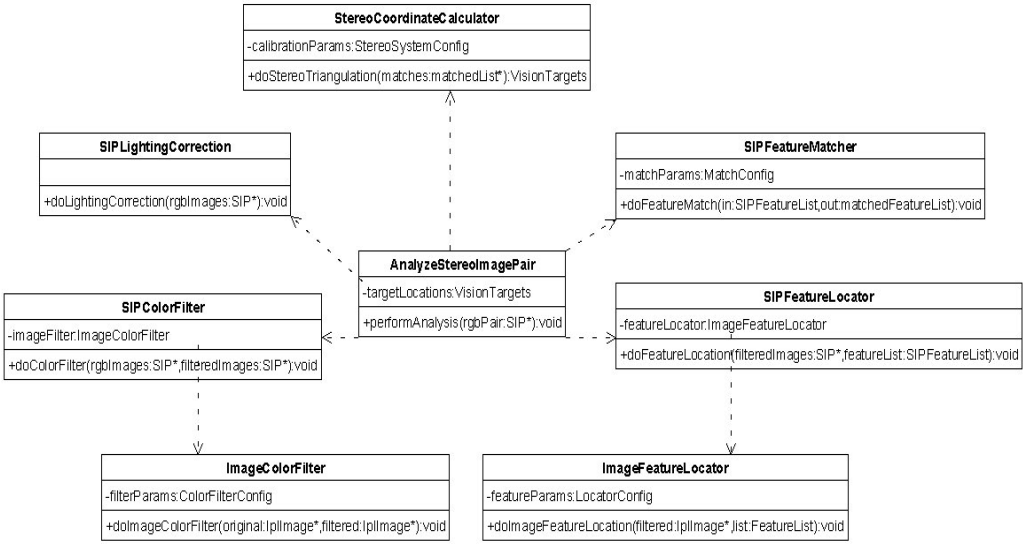


Figure 8: Class Diagram for the AVATAR Analyze Module.

Both `Acquire` and `Analyze` modules contain numerous parameters that must be modified to reflect changes in the operating environment, target properties, and camera systems. The `Config` module facilitates changing these parameters via an external public interface. An XML configuration file contains all relative information to completely fill the necessary values for the AVATAR system. The handling of the XML files is done using an XML Config tool based on TinyXml and Boost, a set of open-source C++ libraries.

TAU

The Target Acquisition Unit, TAU, interfaces the manipulator systems to AVATAR. The `VisionInterface` class provides public data access through the `TAUNet` application. By providing a standard interface across all vision algorithms and hardware, higher levels of TAU can incorporate any method. The two currently implemented versions of `VisionInterface` are `TAU_1394` and `TAU_Raw`, corresponding to `AcquireSIPFirewire` and `RawDataAcquire`, respectively. The initialization of the firewire cameras in `TAU_1394` is achieved as described above. `TAU_Raw` parses log files to ensure all images, log files and configuration files exist and are in the correct directories. `TAUNet` executes continuously on the vision computer, waiting to receive commands over the network to start/stop target acquisition. To minimize translation overhead, `TAUNet` utilizes Ranger's communication protocol for all message handling activities. The actions required for nominal vision system operation consist of starting or stopping a continuous search for targets, performing a single "snapshot" of the current view to determine target locations, and retrieving the target coordinates from the most recent analysis. If the designated task is based on a single system configuration, only these operations are necessary for the entire mission.

The remaining TAUNet tasks provide capabilities for more complicated sampling tasks. A configuration interface enables changing any aspect of the system – camera parameters if lighting is different than expected, the image filter parameters to search for a different target type(s), or modifications to the camera or stereo system properties, among others. TAUNet can retrieve any desired set of images from AVATAR, such as the original images, filtered images, or images marked with located and matched features.

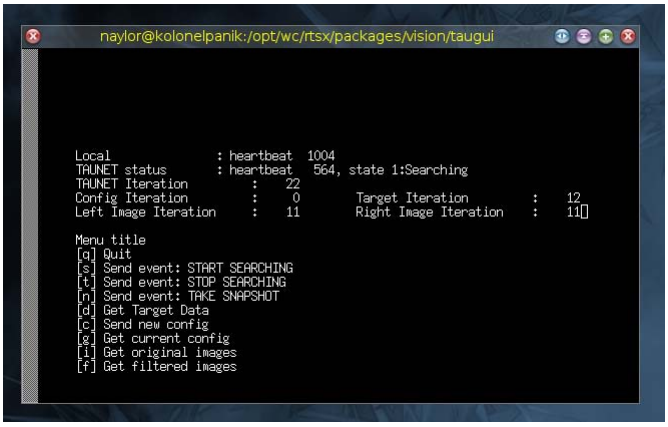


Figure 9: TAUTUI Interface to AVATAR.

The next level of interface is the TAUUnit class, providing message transmission (TX) and receipt (RX) capabilities. While TAUNet must be executed locally on the computer connected to the stereo camera pair (or data files), a TAUUnit class can be created in any executable, such as the DMU, then invoked to send commands or retrieve data from the vision system. In addition to providing the public interface to TAUNet, the TAUUnit class also maintains a log of action execution timing information. The highest-level interface to the vision system is the TAUUnit_GUI. There are two executable programs for this class: TAUTUI, providing a text-based interface (Figure 9), and TAUGUI, providing a wx-widgets GUI.

Software Engineering

One requirement of the system is that any component can be fully tested regardless of whether or not the rest of the hardware is available. This enables the vision software to be tested with archived images, and enables an operator to test that the control software to acquire and process vision data. Simulation software provides the vision data to the DMU in the same data structure over the same communication protocols. Attention was given to const-correctness when dealing with references, ensuring zero memory leaks, proper use of inheritance and polymorphism of classes, and many other common C++ issues where problems can easily arise. Comprehensive documentation including defect tracking, logging of programmatic state and internal data during execution, frequent system-wide unit testing and coverage analysis in addition to continuous integration are all built into the software system to further validate and accurately profile all software. We employed a suite of software engineering tools in this process, including Doxygen for documentation, Gentleware's Poseidon for UML diagramming, Cxxtest and gcov for thorough testing and identifying frequently-executed code to be optimized, and Subversion coupled with Trac and CruiseControl for version control, defect tracking, and automatic system builds.

V. Experimental Platform and Test Setup

A series of experiments was performed with a stereo camera pair affixed to the Ranger manipulator system. Two sets of cameras were used during AVATAR testing. Initial tests were performed using analog Sony XC-999 cameras.[1] Due to poor image quality issues related to these cameras, higher resolution cameras were purchased for the next phase of testing – Point Grey Scorpion 14SO cameras that run on a Firewire bus. The Point Grey cameras were selected to meet undersea operational requirements: 16-bit depth per channel to maximize data available in each image, ability to be integrated within deep-sea-rated housings, and project budget constraints. These cameras have 1280x960 resolution with 16-bits per channel operating up to 19 frames per second (FPS). Each image is stored in a Bayer pattern, meaning the CCD is organized with alternating elements sensitive to different wavelengths of light. Computar model H3Z4512 lenses are attached to the Scorpion cameras. The H3Z4512 are vari-focal cs-ir 4.5-12.5mm F1.2 TV lenses. They are recommended for operation at ranges from 0.5 m - 10 m, corresponding to the expected operational range for approaching undersea sites and conducting sample collection activities.

With cameras, lenses, and housings ready for testing, the next issue was camera placement – both placement of the camera pair with respect to the manipulator and placement of the cameras relative to each other. Placement with respect to the manipulator is driven primarily by occlusion considerations – the cameras should be maximally capable of viewing sampling target workspace before and during manipulation activities. The camera/lens has a 75-degree field of view (FOV) in air, but when placed underwater in our housing, the FOV drops to 55 degrees. This affects the range of possible stereo baselines. If an overlap of 75% is desired at a distance of 1m, a good range for

use with the Ranger dexterous manipulator, then the maximum baseline between the cameras is limited to 0.26m. A large overlap is necessary to maximize the possible sampling area, so keeping the cameras close is important.

After acquiring test images with the cameras placed as close as possible while inside the deep-water housings, it was evident that the high resolution of the Scorpion cameras provides sufficient pixel disparity to accurately locate features, even with a short baseline. With the minimum baseline of 10cm, constrained by the underwater housings, a single pixel offset with a target located at a distance of 1.2m is only 2cm. All testing was performed with this baseline, as extending it will only increase accuracy of the system, until there is insufficient image overlap.

Ranger Manipulator System

The Space Systems Lab's Ranger manipulator was utilized for hardware testing in this research. Ranger is a 10DOF manipulator with eight revolute joints and two torque-driven tool drives. Kinematically, Ranger is segmented at the wrist into two four degree of freedom sections for mathematical simplification. Since Ranger has eight degrees of freedom rather than the traditional six, it has a relatively complex mechanical and kinematic design, but also possesses more capabilities due to the redundant degrees of freedom. Previous research has been performed to analyze and characterize the additional manipulator capabilities.[5,6] While the dexterous workspace of the manipulator increases substantially given the extra degrees of freedom, singularities are more frequent but are also more easily avoided. The redundancy of the manipulator is controlled in two segments – a four DOF upper arm segment and a four DOF wrist. The upper arm segment control is in the form of the roll angle of the shoulder-elbow-wrist (SEW) plane. Through the use of the SEW angle, the upper arm joint angles can be computed independent of the wrist joint angles, while the arm still possesses an additional DOF for avoiding the wrist singularity.[5] Fully extended, the Ranger manipulator has a reach of approximately 1.3 meters. However, singularities exist in fully extended joint configurations. Additionally, large, sometimes prohibitive, torques are required to hold the arm straight in 1-G given its native neutral buoyancy design environment, further limiting the dexterous manipulator workspace. Due to the redundant degrees of freedom, and the resulting capabilities to avoid singularities, the dexterous workspace is almost as large as the reachable workspace. Through correct orientation of the skew angle wrist (SEW), there are configurations that avoid nearly all singularities in the reachable workspace.

For this work, we drive the end effector toward the visually-identified sampling target. This requires real-time inverse kinematic analysis to compute joint-space trajectories. To simplify the analysis for Ranger, the manipulator is broken into two segments, joints 1-4 in the upper arm, and joints 5-8 in the wrist. A different method is used to solve the inverse kinematics of each section. Joints 1-4 use the Extended Jacobian Method based on the wrist location and SEW angle. Joints 5-8 use the General Inverse Method that finds a locally optimal solution for joint velocities specific wrist orientations and additional constraints imposed by tool and forearm orientations. This design was motivated by the kinematic redundancies in the skew wrist that causes singularities prohibiting use of the Extended Jacobian Method. [5,6]

Previous testing of Ranger was performed to characterize both static and dynamic performance characteristics of the manipulator, an important consideration when autonomously commanding the manipulator to grasp a sampling target. These tests were performed in compliance with ANSI standards ANSI/RIA R15.05-1-1990 (R1999) for point-to-point static performance characteristics and ANSI/RIA R15.05-2-1990 (R1999) for path-related and dynamic performance characteristics. Results show Ranger is statically accurate to about 2 cm, while having a static repeatability of about 0.5 mm and a static compliance no worse than 0.4 mm/kg of applied force at maximum reach. For path following, Ranger has an average Cartesian accuracy of 1 mm, a Cartesian repeatability of 1 mm and a Cartesian path cornering radius of about 1 cm.[7,8] In terms of this research, the most important number is static accuracy due to the use of Ranger's Cartesian position estimates for camera-manipulator registration. This 2cm error primarily results from startup routines that initialize the high-resolution incremental encoders from the lower-resolution absolute encoders thus can be mitigated by visual servoing. In 1-g, deflections due to gravity play a secondary role in this error.

Test Sequence

Comprehensive testing of the fully-integrated AVATAR system required a sequence of steps. First, the stereo camera pair was calibrated. Next, camera-manipulator registration was determined. To test the baseline sampling capabilities, a visually-distinct object was placed in the camera FOV and manipulator workspace and autonomous end-to-end sampling trials were performed. Results from this initial testing were previously reported for a low-resolution Sony camera pair, validating sampling ability in 1-g and underwater test environments.[1]

Camera calibration throughout testing has required user input. Each time the relative position of cameras changed, a new extrinsic calibration was performed, but to be conservative intrinsic parameters were also recomputed. The use of a checkerboard in numerous poses requires human interaction both moving the

checkerboard and validating images. However, once the images have been acquired and validated, the remainder of the calibration procedure is straightforward and can be handled automatically. The camera-manipulator registration procedure defined previously was used to locate the manipulator with respect to the camera system. Small errors, especially with the rotation matrix, can cause large errors when applied to points at the extremities of the manipulator workspace. In the final implementation, a hand-eye registration was obtained autonomously from a visually-distinct segment on Ranger's wrist, eliminating manual measurement error. Accuracy and precision were evaluated as well. As will be described below, once the system was accurately calibrated and registered, tests focused on visual servoing and vision system ability to accurately identify targets of interest in visually-complex fields over a variety of lighting conditions.

VI. Test Results

A sequence of vision-based experiments was conducted with the high-resolution Scorpion camera pair and the Ranger manipulator system. These results complement our previous results in which Ranger successfully grasped a visually-distinct object (a yellow rubber duck) in 1-G and underwater environments with low-resolution cameras.[1]

Registration, Precision, and Accuracy

Intrinsic and extrinsic camera calibration parameters were computed, followed by automated hand-eye registration. This registration procedure was performed multiple times with different sets of points to evaluate precision. After the registration parameters have been chosen, they were used to transform a set of independently recorded points from vision frame into manipulator frame to evaluate accuracy between the two sets. Tables 1 and 2 show the calibration data used for this phase of testing – parameters were similar for subsequent test phases. Parameters include focal length (f_x, f_y), principle point (c_x, c_y), as well as relative orientation (ω_x ω_y ω_z) and translation offset (t_x, t_y, t_z).

Table 1: Intrinsic Calibration Parameters.

	f_x (mm)	+/-	f_y (mm)	+/-	c_x (px)	+/-	c_y (px)	+/-
Left Camera	1049.13	0.67	1050.38	0.68	712.28	1.13	564.45	1.18
Right Camera	1047.20	0.67	1048.66	0.69	702.74	1.22	561.13	1.16

Table 2: Extrinsic Calibration Parameters.

Rotation (rad)	ω_x	+/-	ω_y	+/-	ω_z	+/-
	0.002	0.001	0.003	0.001	0.047	0.000
Translation (mm)	t_x	+/-	t_y	+/-	t_z	+/-
	-106.53	0.08	-2.58	0.07	0.82	0.29

Vision system precision and accuracy were evaluated to characterize the overall system. The Ranger wrist was identified by the Interchangeable End Effector Mechanism (IEEM), a gold puck on the wrist used as the visual registration target. Precision evaluation was performed with two procedures. First, the precision of the vision system was calculated based on multiple sets of static images to ensure consistent results. Second, the automated registration procedure was repeated three times with two different sets of points to assess overall precision. Accuracy measurements were based on a comparison of perceived arm position with instantaneous arm telemetry. By taking a random set of points within the manipulator workspace and applying the hand-eye registration on the vision data, the difference between this result and the arm telemetry indicated relative accuracy.

To evaluate automated registration, data was recorded for seven random points within the Ranger workspace at distance magnitudes ranging from 0.65m to 0.95m from the cameras. A 3-D reconstruction from fresh images was performed at each point five distinct times then analyzed for repeatability. Table 3 shows the results from this experiment in the form of standard deviation for all three dimensions and the corresponding magnitude. The substantial improvement from the earlier system is evident, as the maximum standard deviation magnitude over all three dimensions is less than 4mm. The seventh point is not listed as all five analyses resulted in an identical 3-D reconstruction. Another test was performed with two rubber duck targets placed approximately 2.5m from the

cameras. Thirty tests were performed, and in every case the pixel centroids were measured at precisely the same value, thus there was zero change in reconstructed position.

Table 3: Vision System Precision with High Resolution Cameras.

σ x (cm)	σ y (cm)	σ z (cm)	magnitude
0.13	0.14	0.27	0.33
0.10	0.05	0.36	0.38
0.03	0.13	0.26	0.29
0.01	0.09	0.24	0.26
0.01	0.07	0.29	0.30
0.00	0.04	0.01	0.05

This test data gave confidence that AVATAR itself provides extremely consistent results. The next task was to demonstrate that when combined with Ranger to determine hand-eye registration, results were once again consistent. Two different sets of points within the workspace were chosen for this experiment. For each set, three iterations of the registration procedure were performed. Test results summarized in Table 4 show excellent precision for this process. Each of the Euler angles has error less than 0.005 radians, while the overall magnitude of the standard deviation for the translation vector is slightly over than 5mm. This indicates that vision system precision is maintained through the registration process.

With automated hand-eye registration, the transformation was applied to the points used to determine vision system precision. When each of these points was recorded, arm telemetry at that point was also logged. By transforming the vision data into the arm frame of reference, the relative accuracy between transformed data and arm telemetry data was calculated. Table 5 shows a summary of standard deviations and maximum offsets from these transformations, indicating the updated system performs adequately but with potential for improvement. Multiple sources of calibration and measurement error factor into these calculations, thus the relative accuracy achieved here is encouraging. However, this error is still appreciable, motivating use of a visual servo algorithm.

Table 4: Hand-Eye Registration Precision Results.

Euler Angles		
σ α (rad)	σ β (rad)	σ γ (rad)
0.0049	0.0042	0.0024

Translation Vector			
σ x (cm)	σ y (cm)	σ z (cm)	magnitude
0.33	0.23	0.33	0.52

Table 5: Difference between Arm Telemetry and Transformed Vision Coordinates.

Standard Deviation of Offsets			
σ x (cm)	σ y (cm)	σ z (cm)	magnitude
0.28	0.92	0.34	1.02
Maximum Offsets			
x(cm)	y(cm)	z(cm)	magnitude
1.02	2.87	1.37	3.34

Target Identification with Realistic Visual Environments

The next test series evaluated AVATAR performance when transitioned from the laboratory to a real-world environment with less visually-distinct targets and sub-optimal lighting conditions. Based on both deep-water color attenuated imagery from WHOI, as well as cluttered target fields created specifically to stress AVATAR algorithms, output from the initial stages of the vision system was obtained with desired sampling targets extracted. For laboratory testing, a sample target field was created within view of the Scorpion camera pair. The targets in the simulated sampling field consisted of different rocks, as well as a starfish and sand dollar, providing an "undersea" scene for the vision system. Feature extraction tests were performed with both lights on and lights off to simulate a dark environment. Using a combination of the MATLAB filter creation process and the `filtergui` program, separate filters were created for both cases. In the *lights on* case, the focus was on extracting the rock targets, while in the *lights off* case, an attempt was made to extract whatever targets were distinguishable.

Output from the MATLAB filter creation program is displayed in Figure 10. The range of target data versus background data has a clearly distinguishable separation, which translated to clear image filtering parameters. As shown in Figure 11, the filtering process easily segmented desired targets for further analysis. No further image processing was necessary for successful localization in these tests. With lights off, the difficulty of the filtering process increases. The data from these tests, shown in Figure 12, is still somewhat separated into two clusters, but the two regions now overlap significantly. The data is now separated by a diagonal line rather than a horizontal line, which requires a more complex filter to model ratio variance over RGB magnitude.

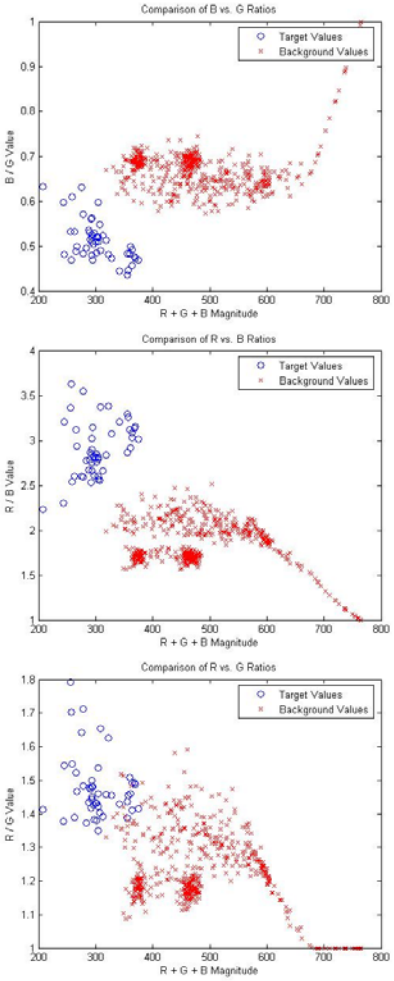


Figure 10: RGB Ratios for Lighted Targets.

To achieve accurate results, a feature-AND operator was applied to eroded versions of the original image to restore degraded features. Figure 13 shows the results from this test – the original image is dark, with results translated to binary values to delineate segmented features. In this test, the starfish, a single rock, and the sand dollar were all extracted with sufficient quality for further processing. Edges of the background image used to simulate sand also appear, but would be ignored through aspect ratio constraints. Note that the lead weights in the background also appear due to their similar color properties.

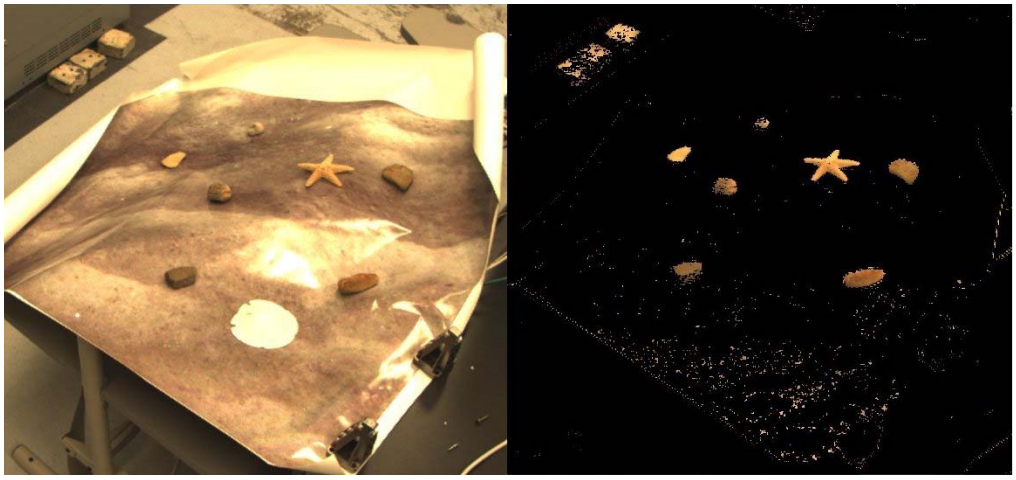


Figure 11: Realistic Targets Segmented in a Lighted Laboratory Environment.

With full color correction applied to raw undersea imagery provided by WHOI, any clearly distinct objects can be segmented. The focus of this test was to show that without full color correction, or even application of the simple frame averaging algorithm, targets can be cleanly extracted from the color-attenuated images. At first, the feature extraction did not function suitably, as the difference in lighting from the center of the image to the edges drastically changed the RGB values at each pixel, due to the attenuated data. By applying the lighting correction algorithm described above, a more homogenous image was created that provided more useful filter output. Once again, the filter creation algorithm was applied to the raw imagery to estimate ratio parameters for the filtering process. Figure 14 shows the results. Due to color attenuation, the Blue vs. Green chart is uninformative. These plots are similar to the low-light results from the lab tests, but the lighting correction creates sufficient distinction to extract the majority of the sand dollars. Figure 15 shows the lighting-corrected image and extracted features side by side.

Figure 13: Targets extracted in darkened laboratory environment.

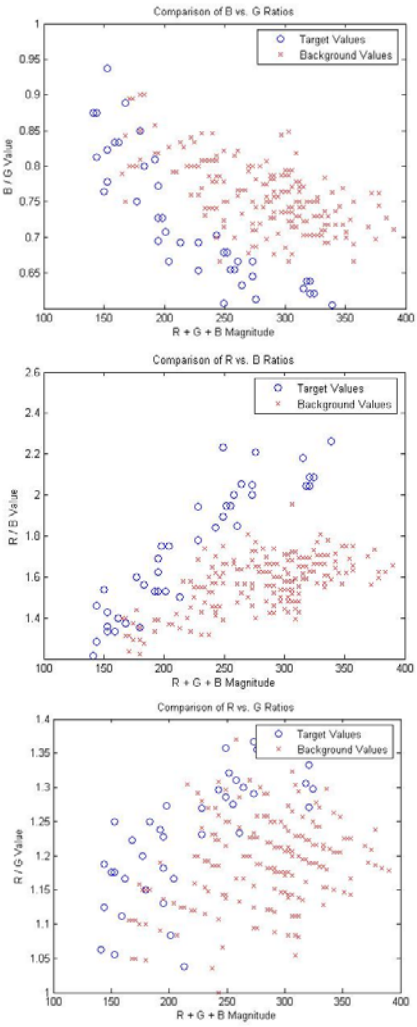


Figure 12: RGB Ratios for Unlighted Targets.

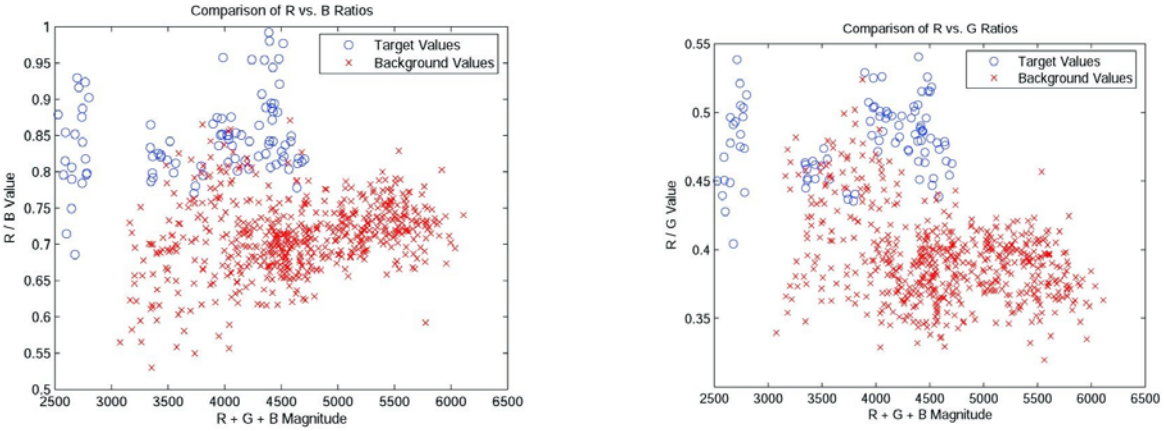


Figure 14: RGB Ratios for Undersea Sand Dollar Images.

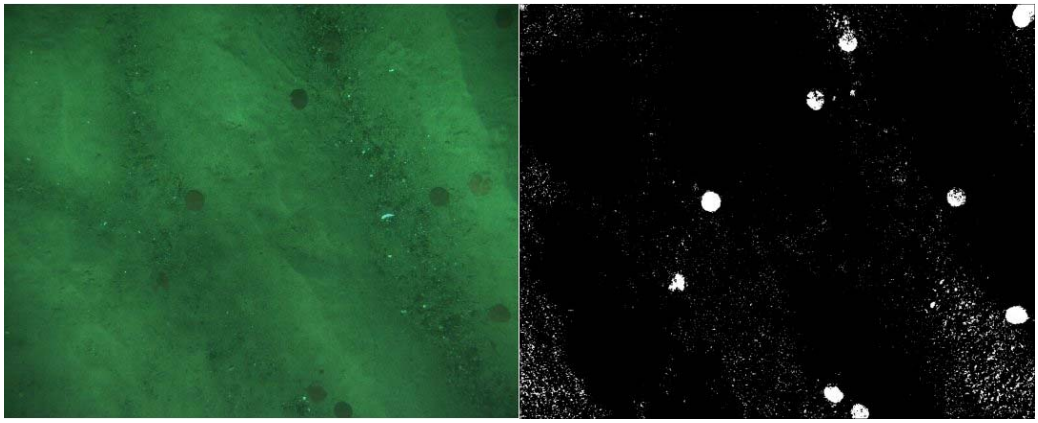


Figure 15: Sand Dollar Targets Extracted from a Color-Attenuated Image.

Visual Servo Testing

The majority of the tests conducted with Ranger were a two-stage process in which AVATAR located the (stationary) target, and then the manipulator traveled to this target. Given the ideal lab environment in which manipulator base and target were truly stationary with respect to each other, this two-step "open-loop" sequence was sufficient. However, the application of AVATAR is undersea target acquisition on an AUV, where it is likely that both the target and the vehicle (thus manipulator base) will at least slowly drift over time. To account for such motion, a visual servoing algorithm was implemented to close the loop between manipulator end effector and sampling target. For these tests AVATAR was configured to track both the manipulator wrist IEEM as well as a yellow rubber duck sampling target. Once positions were estimated for both targets, a motion vector was calculated from the IEEM position to the target, and then a linear Cartesian move was executed along that vector. The magnitude of motion for each step is an option configurable prior to starting the system. Initially the visual servo system was developed to handle errors with camera calibration or unforeseeable problems when human interaction would be impossible, e.g., at 6000m depth. By calculating the manipulator motion vector as the position error between end effector and sampling target in the image plane, the sampling process is robust to appreciable calibration and registration errors. Figure 16 shows sample left and right camera images from a visual servo test in which the sampling target (duck) and manipulator target (IEEM) are tracked.

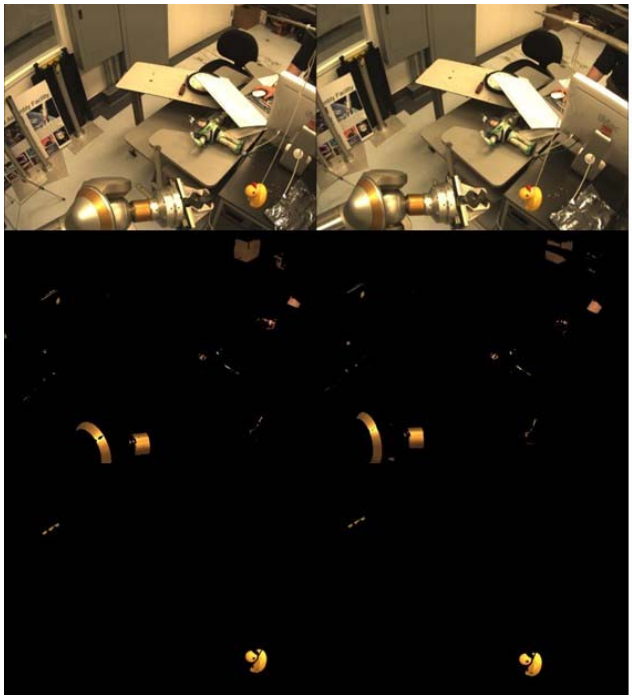


Figure 16: Visual Servo Images: Raw (top); Filtered for IEEM (middle); Filtered for target (bottom).

Results from testing the visual servo system with a stationary target and manipulator base matched previous "open-loop" results: the manipulator dependably and accurately reached the sampling target. The benefit of the visual servo system became apparent, however, when the sampling target was no longer stationary. For these tests the target was moved slowly away from the manipulator. Figure 17 shows a plot of Cartesian position of the end effector versus controller iteration (125Hz). Figure 18 shows a plot of joint angles versus controller iteration. The motion of Ranger followed a linear trajectory in joint space, with a magnitude of 5cm motion per visual servo iteration. This data shows approximately 22 seconds execution time representing 15 visual servoing cycles. Although not shown in the Figures, the rubber duck followed a profile that approximately matches that of the manipulator wrist, indicating the manipulator through visual servoing was able to match its motion to that of its target.

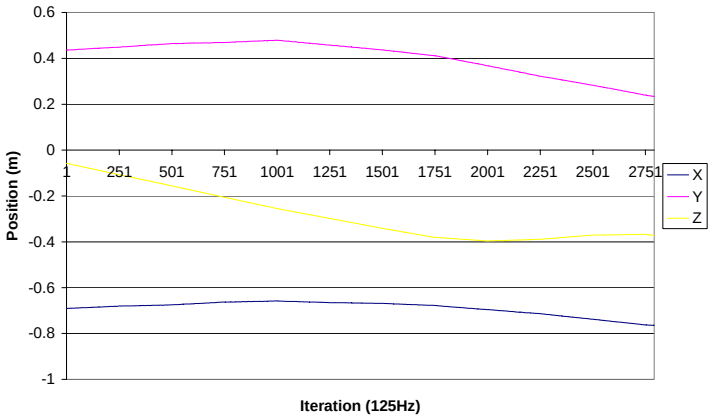


Figure 17: End Effector Cartesian Position for a Moving-Target Visual Servo Sequence.

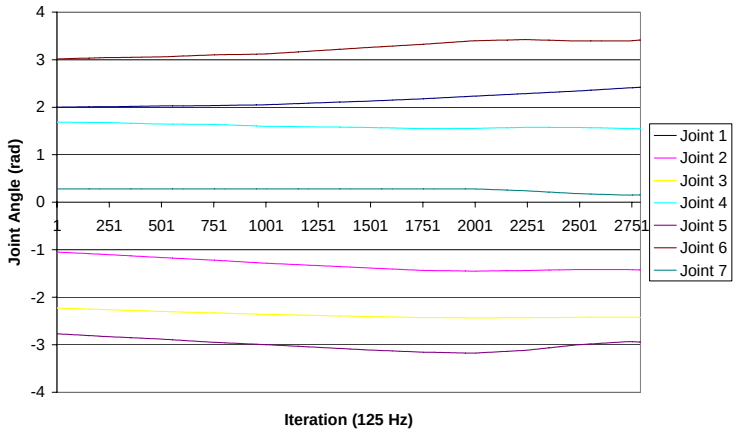


Figure 18: Joint Angles for a Moving-Target Visual Servo Sequence.

VII. Conclusions and Future Work

This paper has described the development and implementation of a fully autonomous vision system used to provide 3-D localization of sampling targets for a robotic manipulator. Three major focus areas are pursued: development of the vision algorithms to perform feature segmentation and 3-D reconstruction, design of a logical, modular software structure, and hardware integration with a robotic manipulator and subsequent sampling tests. The overall system is capable of visually tracking both sampling targets and the manipulator, providing position data in the correct frame of reference to allow the manipulator to accurately approach a sampling target. The current set of vision algorithms provide the necessary capabilities to sample targets with sufficiently distinct color properties. Although the algorithms remain simple from a mathematical standpoint, this simplicity minimizes computational complexity and facilitates intuitive understanding, both important for an environment where processing resources are highly constrained and where users are neither fluent in nor accepting of computer vision and autonomy.

The success of sampling trials shows that the current system is capable of autonomously sampling a desired target within the manipulator's workspace, both statically and with a moving target. Testing with low-resolution cameras demonstrated visually-distinct targets could be retrieved in both 1-G and underwater environments.[1] With an increase in camera quality as well as software capability and reliability, subsequent accuracy and precision data indicate the system is precise, accurate, and dependable, key prerequisites for long-term operation. Task execution, aside from the initial calibration of the cameras and tuning of the target filters, is fully-autonomous.

The final set of tests utilized a visual servo system developed to increase target sampling robustness for an underwater environment where the manipulator base (AUV) and/or the target may not be stationary. Since the system was calibrated accurately and Ranger was operated in a lab setting, visual servoing for stationary target tests exhibited similarly-robust sampling ability to earlier single image (open-loop) tests. Moving-target test, requiring the visual servo approach, were successful, with the end effector able to follow the target throughout the dexterous manipulator workspace provided target speed is equal to or less than manipulator traversal speed constraints.

While the AVATAR system provides the machine vision capability necessary to autonomously sample an undersea target, there are numerous avenues for future work. The modular nature of the software design allows quick and easy integration of new algorithms or other modifications, facilitating extension. The major vision-related area to be augmented is feature segmentation beyond the presented RGB ratio method. By implementing multiple methods that complement the current scheme, from an initial undersea image survey a user could select and tune the most capable algorithm for each target class. Another useful enhancement would be matching features extracted from one image, based on color data, with grayscale features in the other image. Most real-time computer vision applications use grayscale imagery since the one data channel can be analyzed more quickly than multi-color data. Many underwater systems have, for science purposes, one color camera and one grayscale camera, as color provides visually attractive pictures while the grayscale cameras have higher bit depth.

VIII. Acknowledgments

The authors would like to thank Craig Carignan, Dave Akin, and the remaining University of Maryland Space Systems Lab students and staff for their support during setup and testing with the Ranger manipulator system. We also thank Hanu Singh and the Woods Hole Oceanographic Institute team for providing undersea images, references, and baseline Matlab software for undersea image lighting and color correction.

IX. References

[1] M. Naylor, N. Scott, E. Atkins and S. Roderick, "Toward Autonomous Sampling and Servicing with the Ranger Dexterous Manipulator". *Proc. AIAA Infotech@Aerospace Conference*, Crystal City, VA, September 2005.

[2] H. Singh, R. Armstrong, F. Gilbes, R. Eustice, C. Roman, O. Pizarro, J. Torres, "Imaging Coral I: Imaging Coral Habitats with the SeaBED AUV," *Journal for Subsurface Sensing Technologies and Applications*, pp. 25-42, vol 5, no 1, 2004.

[3] H. Singh, A. Can, R. Eustice, S. Lerner, N. McPhee, O. Pizarro, C. Roman, "SeaBED AUV Offers New Platform for High-Resolution Imaging," EOS, *Transactions of the AGU*, vol 85, no 31, pp 289,294-295, August 2004.

[4] J. Russ, *The Image Processing Handbook.* CRC Press, Boca Raton, Florida, 1992.

[5] C. Carignan and R. Howard, "A Partitioned Redundancy Management Scheme for an Eight-Joint Revolute Manipulator," *Journal of Robotic Systems*, 17(9):453-468, September 2000.

[6] C. Carignan and R. Howard, "A Skew-Axis Design for a 4-Joint Revolute Wrist," *Proc. IEEE Int. Conf. on Robotics and Automation*, Washington, 3636-3642, May 2002.

[7] S. Roderick or W. Smith, *DT21-0039 Test Report: Ranger Static Performance Measurements*, Space Systems Laboratory internal document #06-005.

[8] S. Roderick or W. Smith, *DT21-0041 Test Report: Ranger Dynamic Performance Measurements*, Space Systems Laboratory internal document #06-006.