AIAA-94-1285-CP

# Real-time Tracking of Objects for a KC-135 Microgravity Experiment

Mark L. Littlefield
Intelligent Systems Department
Lockheed Engineering and Sciences Company
2400 Nasa Road 1
Houston, TX 77058
mll@lobo.jsc.nasa.gov

## Abstract

This paper outlines the design of a visual tracking system for use on the Extra-Vehicular Activity Helper/Retriever (EVAHR) autonomous robot during tests on board NASA's KC-135 Reduced Gravity Laboratory. Issues such as the laboratory environment, mission requirements, real-time constraints, and computational loads will be examined.

EVAHR is an autonomous robot designed to perform a number of tasks in an on-orbit microgravity environment. One of the critical tasks of EVAHR is the ability to grasp a freely translating and rotating object. This task is the current focus of investigation by the EVAHR development team. To perform this task, EVAHR must analyze range image generated by the primary visual sensor to locate and focus its sensors on the target so that an accurate set of object poses can be determined and a grasp strategy planned. This may involve positioning the sensor with its gimbal (pan/tilt) system and adjusting sensor parameters to provide the perception system with the best possible views of the target. The tracker must also provide the information that it extracts about the target to pose and state estimation modules. These tasks must be performed at a frame rate of ~9 frames per second.

## 1. Introduction

The EVA Helper/Retriever (EVAHR) is a prototype robot currently undergoing development at NASA's Johnson Space Center. The goal of the EVAHR project is to develop an autonomous robot which can carry out on-orbit tasks such as inspection, worksite preparation, Orbital Replacable Unit (ORU) changeout, and crew and equipment retrieval (CERS)[1].

The initial task chosen by the EVAHR team for investigation is the CERS task. The goal for this task is to autonomously grasp freely floating and rotating objects. Given the situation in which an object (crew member, tool, ORU, etc.) becomes unteathered and drifts away from a work area, the EVAHR must locate, rendezvous, grapple, and return with the object in a reasonable amount of time.

To accomplish this task in a realistic manner, a system consisting of a Robotics Research k-807i arm, a dexterous manipulator, a Perceptron laser range scanner mounted on a high-speed pan/tilt, and an instrument package consisting of gyroscopes and accelerometers (an Inertial Measurement Unit, or IMU) has been assembled, and a series of flights aboard NASA's Reduced Gravity Laboratory (KC-1 35 aircraft) has been scheduled[2]. Software for this task consists of an object tracker, a pose estimator, a set of state estimators (translational and rotational) and an arm/hand control module[3].

To prepare for this task, the software modules were tested and debugged against an on-orbit simulator[4] for testing the EVAHR software over an entire rendezvous and grasp. The simulator models the environment around a space station and maintains dynamics on the EVAHR and any other free-floating objects. The simulator also generates range images of the environment with the same general characteristics as the Perceptron laser range scanner. This simulation also has the ability to remove some modules and substitute perfect data to others, which provides an excellent testbed for debugging modules and verifying algorithms.

The second phase of investigation consists of a series of experiments aboard a KC-135 parabolic aircraft. There are four sets of flights scheduled. The first flight set consisted of a series of flights using the IMU to measure and record the aircraft motion during the parabolas. This information allowed the EVAHR team to characterize the aircraft motions and design the state estimators and the target release indicator needed for the following flights.

For the second flight, a simplified tracker program was flown with the laser scanner and pan/tilt device to evaluate the hardware performance and to collect data on object motion during the flight. The robot arm and hand were also flown but each ran a series of independent tests to ver-

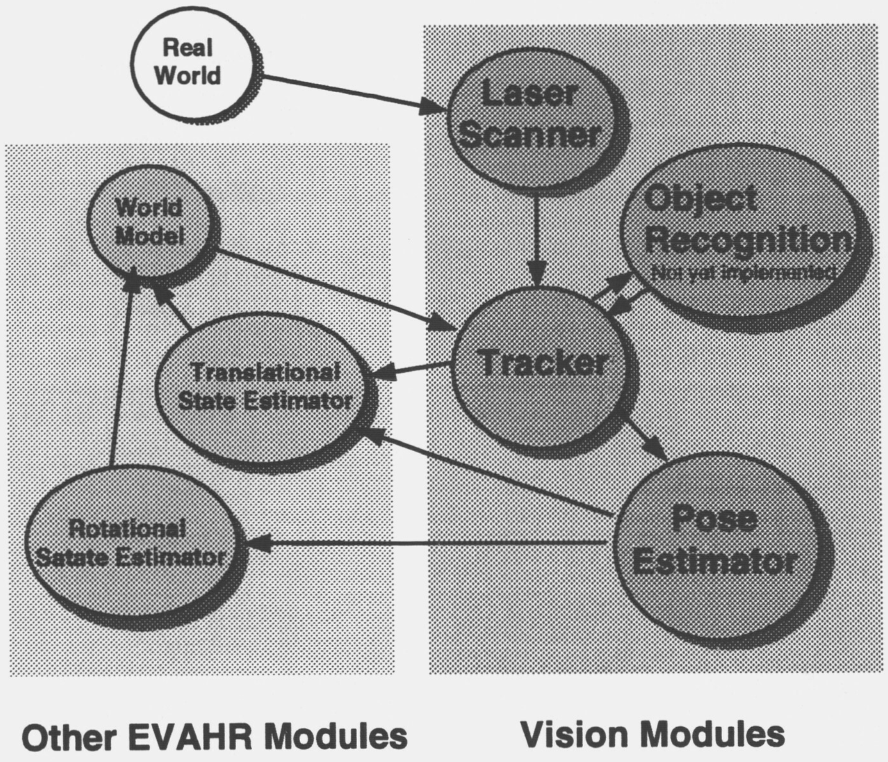

Figure 1: EVAHR perception modules and data flow

ify the hardware and software control module performance.

The third flight consists of flying the entire hardware configuration in a closed loop with the tracker, translational state estimator, and arm control modules, to grasp a ball during micro-g. Finally, in the fourth flight, the system will grasp a rotating polygonal target with the stage three system, combined with the pose estimator and rotational state estimator modules.

Phase I and flights 1 and 2 of phase II have been completed. Flight 3 is currently scheduled for February 1994 and flight 4 is scheduled for late Q1 1994[3].

The primary sensor for the EVAHR is a Perceptron laser range scanner mounted on a pan-tilt device. This scanner provides 12 bit range images over a 15 meter range. The scanner generates an image by sweeping a modulated infrared laser across the field of view using two mirrors, a spinning mirror (for each scan line) and a panning mirror (to move the scan lines up and down). The depth at each pixel is measured by comparing the phase of the returning laser signal.

The scanner has three speeds of operation (the slower the speed, the higher the spatial resolution) and a variable vertical field of view. At it's highest speed, the scanner produces 2.25 frames per second at 256x256 pixels, or 9 frames per second at 64x256 pixels. The latter is the setting sued in most of the EVAHR experiments. The scanner also provides a pixel registered 12 bit intensity image along with the range image.

There are three computational components that currently make up the perception system (figure 1). First, an object tracker performs a series of image processing tasks, and identifies the objects in the range images from frame to frame. The second module is the pose estimator[5], which calculates the object location and orientation (pose). The third module consists of the object state estimators[6], which maintain a real-time state of the object's rotation and translation. These three modules form a cross connected loop, in which each module provides information to the other modules as information becomes available.

This paper describes the tracker module of the EVAHR. Section 2 describes general design of the tracker module and the results of the initial test series in the on-orbit simulation. Section 3 describes the stage II/flight 2 tracker and the results of that flight. Section 4 describes the stage II/flight 3 tracker, focusing mainly on the lessons learned from flight 1. Section 5 discusses the added features required for the stage II/flight 4 tests and section 6 is a summary and a discussion of the future plans for the vision loop and the tracker module in particular.

## 2. System design and on-orbit simulations

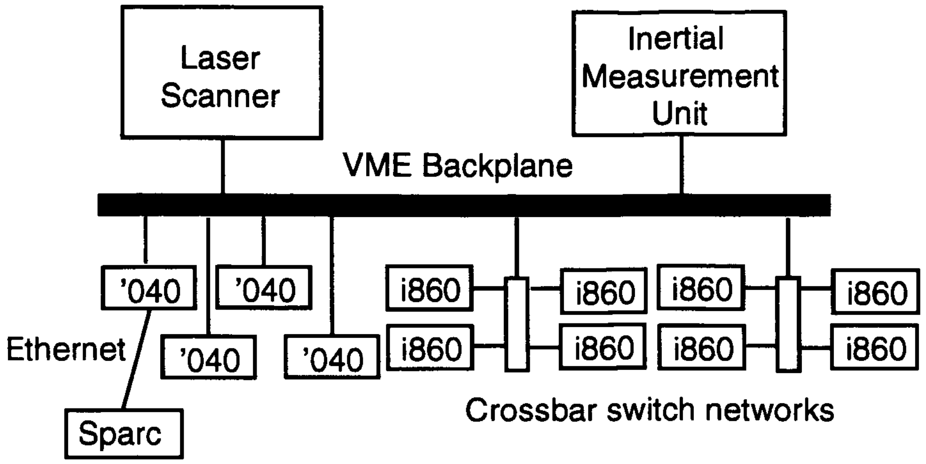

Figure 2: Physical connections for EVAHR percpetion hardware.

NASA's Reduced Gravity Laboratory is a KC-135 aircraft which is flown in a parabolic path to produce up to 25 seconds of freefall. Of this 25 seconds, roughly 9-15 seconds have less than 30 milli-g of Z axis accelerations. This is the period in which EVAHR must perform it's grasp.

To accomplish this task, the tracker module was designed as the first stage in the perception system. In the preliminary design, the tracker module had five tasks[7]:

1. Locate the target in each image. This also involves locating and marking the arm in the image.
2. Send the target position to the translational state estimator.
3. Send target contour data to the pose estimator.
4. Maintain an internal object database on target parameters (position, velocity, confidence, etc.).
5. Calculate the scanner configuration values and gimbal positions to maintain a lock on the target.

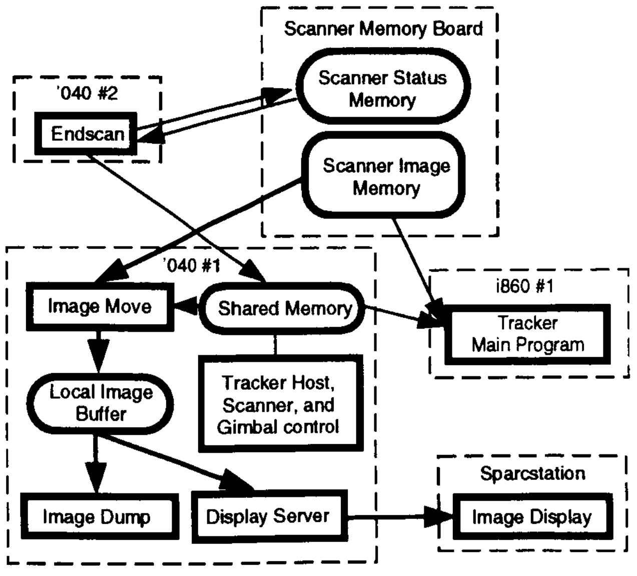

Figure 3: Sub-program organization and data flow in the tracker module.

The code to perform these tasks was tested extensively on an on-orbit simulator. Although the simulator provided an environment in which techniques could be developed and data structures could be ironed out, there were several shortcomings. First, the data provided from the scanner simulator was noise free. This made object segmentation trivial as pixels were either object, EVAHR arm, or background. Also, control of the scanner parameters and gimbal positions was instantaneous and took nothing more than a message to the scanner simulator. Finally, the scanner simulator did not perfectly simulate the Perceptron scanner, neither in the geometry of the image that it generated, nor in the way in which it could be configured.

For the flight test tracker module, several additional tasks were added:

6. Command and control of the scanner and gimbal.
7. Maintain a frame synchronization on the Perceptron scanner.
8. Optionally display or store images periodically for operator feedback or off-line analysis.

Each of these tasks, plus the ones listed above, must be performed in the 0.11 second cycle provided by the generation of the images.

To accomplish this, the tracker module is split into several concurrent sub-programs running on four separate computers: two 68040-based, vxWorks machines and a Mercury i860-based machine mounted on a VME backplane, and a Sun Sparcstation connected to one of the vxWorks machines via ethernet (figures 2 and 3). A host program, running on an '040 spawns the other programs, synchronizes them, and commands the scanner and gimbal. Also on this '040 are the image move, image display server, and image dump sub-programs. The image move sub-program moves images from scanner memory into a local buffer on the '040. The image display server sub-program connects to the image display sub-program running on the Sparcstation via ethernet and sends the locally stored images as fast as the bandwidth of the ethernet allows. The image dump sub-program takes locally stored image and stores them on a backplane-mounted hard disk.

The second '040 machine runs a sub-program (endscan) which performs two tasks: reset the scanner at the end of each image and maintain a data structure on the currently available image which records the address of the image in scanner memory and the direction of the scan. To maximize the speed that it collects images, the Perceptron scanner collects images on both the down scan and up scan of the mirror (there is no vertical retrace). This means that while the even images are collected top to bottom, the odd images are collected bottom to top, meaning that they will be "upside down" in memory. It is up to the controlling program to keep track of the direction of the scan for each image collected (there is no indicator of scan direction).

To collect data, the scanner must be commanded to 1) start the panning mirror to begin panning and 2) start collecting data[8]. This is done by setting a STOP ADDRESS for data collection, sending a "B" (for bi-directional scanning) over the serial port, and setting a FRAME REQUEST flag in the scanner status register. When the scanner begins storing data it asserts a FRAME BUSY bit in the status register and continues to assert it until the commanded number of pixels worth of data have been collected. The scanner also records the address of the current 4K block that it is writing to in the status register. At the end of the scan the scanner de-asserts the FRAME BUSY bit, but does not stop the panning mirror's motion. The endscan sub-program must poll the FRAME BUSY bit, and when it is de-asserted, set the STOP ADDRESS for the next frame and set the FRAME REQUEST flag before the mirror has reversed it's direction and started motion again. It typically takes between 2.15 and 10.77 milliseconds to perform this task. If endscan fails to perform this task, the FRAME REQUEST will not be recognized by the scanner and the upscan/downscan synchronization will be lost.

The main tracker sub-program runs on a Mercury i860. There were two reasons for the placing it on this platform. The first was the raw speed of the i860-based machine. The second was to facilitate communications between the tracker and the state and pose estimation modules. On the Mercury system, four i860 are connected by a crossbar switch, allowing shared memory access at internal memory access speeds.

The first task of the main tracker sub-program is to segment and identify each object blob found in each image from the laser range scanner. A unique object ID is given to each object that is found. As objects move relative to the scanner, their blobs move in the image frame. The tracker must maintain a proper object ID on each object in the field of view for each frame from the laser scanner, and send this information, along with 3D blob contour data, to the pose estimator. No object recognition is performed at this stage. To simplify the problem for the KC-135 experiments, it is assumed that a single object will be visible at the start of micro-g and that it is the target object. If, by chance, additional objects appear during micro-g, the tracker utilizes predicted location and object size to correspond the target object.

## 3. Stage II/Flight 2: hardware validation and data collection

The second flight in the planned EVAHR KC-135 experiments is to test the hardware on board the KC-135 during a flight to validate it's performance and to collect data on object motions during the micro-g potions of the flight. A simplified version of the tracker program tested in the on-orbit simulator was used to try and keep the scanner pointed and focused on the target (white ball) and to store images for off-line analysis. A micro-g indicator was also flown to inform the operator when to release the target.

During this flight, a crewmember released the target in front of the scanner at the start of micro-g. The simplified tracker then segmented any discrete blobs in each of the range images generated by the scanner, picked the blob that was "most circular", and made sure that the mapper was focused and pointed at the target that it located in the image. Meanwhile, the image dump sub-program saved images to the local hard disk. No information about the object was saved or used in subsequent images.

During preliminary tests aboard the KC-135, it was discovered that the first few seconds of micro-g are very noisy. That is, there are relatively large accelerations experienced during this time. To detect when this period is over and "clean" micro-g occurs, a release indicator was developed. The release indicator analyzes the data from the IMU and activates a light when the micro-g has stablized[9].

During some of the parabolas, the crewmember released the target when signalled by the release indicator while during others the crewmember used their own senses to estimate the proper time to release the target. In general, if the crewmember released the target when signalled by the release indicator rather than guessing on their own, there was less unwanted motion in the target object.

It was discovered during early testing that moving the gimbal while the scanner was scanning an image would interrupt the scan and disrupt the upscan/downscan synchronization. This is due to a safety device which shuts down the laser if the spinning mirror speed is not within a specified tolerance. To overcome this problem the tracker waits for a scan to complete, commands the scanner to stop scanning, moves the gimbal, then commands the scanner to start scanning again. This is a rather time consuming task, taking on the order of 0.75-1.0 seconds to complete.

To keep from gimballing unnecessarily, the tracker makes use of a feature of the scanner in which the vertical field of view can be adjusted to start and stop anywhere in the 60 degree overall vertical field of view. This means that when the scanner is set to focus on a target with, say, a 15 degree vertical field of view, and the target moves toward the top or bottom of the image, the scanner can be commanded to adjust the field of view up or down, thus maintaining the target in the center of the image. The tracker only moves the gimbal when the top or bottom of the vertical field of view moves outside the 60 degree overall field of view, or if the target moves close to the left or right sides of the image (a 60 degree field of view at the motor speeds we use during the flight).

During the off-line analysis, it was discovered that the tracker actually failed to track during most parabolas. There were two reasons for this:

1. Segmentation failed when the target moved too close to the walls or floor of the aircraft.
2. The target moved out of the field of view during scanner reconfiguration or gimbal motion.

Although data was successfully collected during the parabolas that the tracker functioned correctly, nearly all data was lost for the rest. For the following test flights, significant changes were made.

Several lessons were learned from flight 1:

1. Target segmentation using only the range image was inadequate.
2. Control of the scanner and gimbal was too slow.
3. Focus of the field of view was too tight.
4. There was no search mechanism for lost targets.
5. Having a crewmember inside the field of view complicated both the segmentation and target location.
6. Waiting until the release indicator signals to release the target reduces the amount of unwanted target motion dramatically.

## 4. Flight 3: Ball grasp

Flight 3 is the first flight where a grasp of an object is attempted. For this flight, it is imperative that the tracker maintain a lock on the target throughout the micro-g portions of the parabola.

Using what was learned in flight 2, changes were made to the tracker software. The elements of the on-orbit simulation version of the tracker that were left out for flight 2 were added in. These additions maintain a database of objects seen by the tracker and attempts to maintain a correspondence between objects located in each image, and the objects in the internal database. Code was also added to located and label the arm in each image[7].

Other changes were made to the tracker to address the issues raised after flight 2. These changes included:

1. Making the target white, the background black, and segmenting on the reflectance image generated by the scanner.
2. Increasing the speed of the commands to the scanner and the gimbal.
3. Adjusting the mechanism for focusing the field of view to be more conservative.
4. Adding a mechanism for searching for the object if it becomes lost.
5. Using a release mechanism to release the target, rather than a crewmember. This was actually planned for flight 3 to protect crewmembers from entering the workspace of the arm.
6. Planning to rely on the release indicator to signal the target release.

The most significant change to the tracker module is the change from segmenting based on the range image to segmenting from the reflectance image. This requires that the interior of the aircraft be covered in black fabric and the ball to be painted white. The arm and hand must remain their original white/aluminum colors, however, so locating the arm/hand in the image still must be performed.

Another problem that plagued the tracker during flight 2 was the loss of the target while setting scanner configurations or moving the gimbal. This problem was solved using three techniques. First, the communications between the tracker and the scanner and gimbal controller was speeded up. The time now needed to change the scanner settings is ~0.07-0.10 seconds, verses ~0.3 seconds during flight 2. Also, the time for gimbal motion has been cut from ~0.75-1.0 seconds, to <0.5 seconds.

The second technique for attacking the target loss problems is to adjust the vertical field of view to it's maximum following a gimbal move. This maximizes the chance that the target will be in the field of view. Finally, when a gimbal move is needed, the scanner is moved in both pan and tilt, pointing the scanner at the target. This is opposed to the method used in flight 2 in which the gimbal was moved only in pan if the target is at the right or left edge of the image and moving only in tilt if the target is at the top or bottom of the field of view.

During flight 2 there was no mechanism for recovering lost objects. If the target was lost, the tracker moved back to it's home position. To correct this the tracker uses a layered approach for reaquiring the target. If no target can be located in an image, the tracker commands the scanner to open the vertical field of view to it's widest. If, after the scanner change, no target can be found, the tracker commands the gimbal to point the scanner at the predicted location of the target (at the estimated time that the gimbal motion should be complete). This predicted location is generated either from the translational state estimator or the internal tracker estimate of the target position and velocity. If there is no target found after these two operations, the target is considered "lost". At this point the scanner and gimbal are reset to their starting positions and the tracker data structures are cleaned up and reset.

Although originally scheduled for Q4 1993, flight 2 has been postponed until Q1 1994 due to hardware and KC-135 scheduling problems.

## 5. Flight 4: Polygonal object grasp

For flight 4, the tracker software will remain generally the same, with the addition of the communications with the pose estimator module. This flight is currently scheduled for late Q1 1994.

When the pose estimator is ready for data, the tracker sends a contour of the target object, labeled with both depth information and which portions of the contour belong to the target (in the case of occlusion). Along with this information extracted from the image, the tracker passes along the state of the scanner and gimbal, and the estimated pose of the target (if it is available).

There are two competing issues that the tracker must deal with during the flight 4 object grasp:

1. The pose estimator takes a relatively long period of time to calculate the pose of an object without any prior knowledge of the target pose.
2. The rotational state estimator must have data very fast during it's initial start-up period in order to converge to a solution in a reasonable time.

To overcome these issues, the tracker queues up the first three images worth of data to pass to the pose estimator, whenever the pose estimator is free. If the rotational state estimator gets bad data from the state estimator, or if it cannot converge with the three images worth of data that it receives, it tells the tracker to re-initialize. The tracker then queues another three images worth of data and the process is repeated. This task is performed as part of the main tracker image processing cycle.

## 6. Summary

The tracker module performs the first stage of image processing as part of a general perception system for EVAHR. In addition to locating a target object in a series of range image, the tracker must transmit the position of the target to the translational state estimator, transmit range data from the target contour to the pose estimator, and correctly configure the scanner and point the gimbal so that the target stays inside the image. These tasks must me performed within the real-time constrains of the scanner frame rate and the environment inside NASA's Reduced Gravity Laboratory KC-135.

Of four sets of flights scheduled aboard the KC-135, two have been completed as of the writing of this paper. The first set of flights measured the motion of the aircraft with an inertial measurement unit (IMU). The second set of flights involved flying the laser scanner, the arm, and the hand in a series of independent tests. Also during these flights the tracker system stored images of a floating target to a hard disk for off-line analysis.

The third set of flights will involve using the perception system, along with the arm and hand systems, to grasp a freely floating ball. The fourth and final set of flights will incorporate a pose estimator and rotational state estimator to grasp a polygonal target.

The tracker module plays an important role in flights 2, 3, and 4. For flight 2, the tracker was tasked to maintain the scanner pointed toward a ball target throughout the micro-g portions of the flight. Although the tracker largely failed in it's task, important lessons were learned.

For flights 3 and 4, the tracker will maintain a history of each object that it detects in it's images. It uses this information to maintain an object correspondence between frames. The tracker also has a method for searching for the target if by chance the target moves outside the field of view of the scanner. This allows the tracker to recover from unexpected events such as an unexpected aircraft motion.

## 6. References

1. Crumrine, S.B., Dellenback, S.W., Fink, P.K., Franke, E.A., McFalls, D.S., *Requirements for an Extra Vehicular Activity (EVA) Robotic Assistant, Final Report,* Crew and Thermal Systems Division, NASA Johnson Space Center, Houston, 1987.

2. White, Linda G., *JSC Reduced Gravity Program Users Guide, JSC-22803*, Flight Crew Operations Directorate, Aircraft Operations Division, July 1991.

3. Grimm, K., Erickson, J.D., Norsworthy, R., Anderson, G., Hewgill, L., Littlefield, M., Chien, C.H., "An experiment in vision-based autonomous grasping within a reduced gravity environment," *SPIE Cooperative Intelligent Robotics in Space III,* Boston Massachusetts, 1992.

4. Norsworthy, R.S. and Merkel, L.E., *EVA Retriever and On-Orbit Environment Simulation Requirements and Design,* report number LESC-28938, Lockheed Engineering and Sciences Company, NASA Johnson Space Center, 1990.

5. Chien, C.H., "Multi-view based pose estimation from range images," *SPIE Cooperative Intelligent Robotics in Space III,* Boston Massachusetts, 1992.

6. Hewgill, L, "Motion estimation of a freely-rotating body in earth orbit," *SPIE Cooperative Intelligent Robotics in Space III,* Boston Massachusetts, 1992.

7. Littlefield, M.L., "Adaptive tracking of objects for a mobile robot using range images", *SPIE Cooperative Intelligent Robotics in Space III,* Boston Massachusetts, 1992.

8. *Perceptron LASAR Hardware Manual,* Perceptron, Inc. 1992.

9. Hewgill, L., "Motion estimation of objects in KC-135 micro-gravity", AIAA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS), Houston, Texas, March 1994.