ACE Project Report

December 10, 2007

Reid Simmons, Sanjiv Singh
Robotics Institute
Carnegie Mellon University
1. Introduction

This report covers the period from September 20, 2007 through December 10, 2007. During that time, we developed the task board for the project, ported the behaviors that we needed for doing the task, developed initial task procedures for doing the task, mounted and calibrated cameras, and performed an initial set of docking experiments. Each of these will be described in more detail in the subsequent sections.

2. Task Board

The task board is based on the rear section of an ’87 Firebird. During the past period, we built a holder for the task board out of 80-20 aluminum and casters (Figure 1). This enables us to easily adjust the height of the task board and to move it out of the way. It may also be useful next year, when we will do assembly on a moving task board. A fiducial was added on the task board to facilitate visual servoing.

![Figure 1. ACE task board, mounted on casters and with fiducial](image)

Assembly is accomplished by a mobile manipulator, a four wheeled robot combined with a 5 DOF manipulator arm (Figure 2). We designed a custom fixture mounted at the end of the arm to hold the clip that is to be inserted into the task board in place (Figure 3).
Figure 2. Bullwinkle, a mobile manipulator, is used to perform the assembly

Figure 3. Close up of the end effector and clip to be inserted into the task board

3. Ace Behaviors

The Ace behaviors are ported directly from the Trestle project [Sellner et.al. 2006], with minor modifications. The behaviors are implemented using a C++ version of the Skill Manager, as
documented in the 9/20/07 report. There are two main sets of behaviors needed by the Ace project (which are a subset of those used in Trestle) – control behaviors and tracking behaviors. The control behaviors (Figure 4) provide the ability to move (and query) the mobile manipulator (base and arm). They are relatively straightforward – most of the computation is done in the arm and vehicle controllers (which are independent processes that communicate to the blocks via IPC [Simmons & Whelan, 1997]).

- **Arm control block**: This block just passes commands along to the coordinated controller program. The coordinated controller program takes the end-effector command, calculates the desired velocities of the arm’s joints and the base’s wheels, and sends those velocity commands to the hardware. The coordinated control software monitors when the arm is done with the command and signals the arm control block. The arm control block then signals its calling task that the command is done.

- **Base control block**: This block passes commands through IPC to the base controller, which calculates the wheel velocities necessary to achieve the command and sends those to the hardware. The base control block receives pose updates from the low-level controller and determines when the base has achieved the desired pose, at which point it issues a signal (which can be handled by other blocks or tasks).

- **Go-to-goal block**: This block takes a desired base position. It calculates the desired base velocity and sends the command to the base control block. It waits to receive a signal from the base control block and, in turn, signals when the desired position has been reached.

The tracking behaviors enable the robot to determine its position with respect to fiducials in the environment. In the Ace project, fiducials are attached to the task board (Figure 1) and to the wrist of the robot (Figure 2). Tracking is fairly complex, and involves a number of behavioral blocks (Figure 5).

Tracking begins with the processing of images taken by a pair of stereo cameras. The **ImageGrabber** block coordinates with the image grabbing process and provides an abstraction independent of the particular cameras in use. It can also read a sequence of previously recorded images, which allows testing of different processing algorithms using precisely the same vision data. The images are provided to a block that detects fiducials within the images and determines their precise locations in space. This image-processing block is named after the type of fiducial it detects. The current system uses **FidDetectorStereoARTag** block to estimate the relative location of fiducials in the scene. The ARTag software is a freely available product [ARTag]). The **ImageAnnotation** block uses this information to mark the fiducial locations in the original image and display them to a user for debugging and system introspection. The fiducial pose information is also transformed by a generic utility block called **PoseTransform** block. Using additional information about the state and location of the robot carrying the cameras, the **PoseTransform** block transforms the fiducial poses from a camera-centric representation to a global coordinate frame for use in the rest of the system.
These fiducial poses may be obtained by other means--when running the system within the Gazebo simulator, fiducial poses are determined directly (without the step of generating synthetic images and running fiducial detection on them), and provided to the rest of the system (potentially with some added noise) by the \texttt{FidDetectorSim} block. The \texttt{FidDetectorFake} block can similarly provide data directly from log files.
Figure 5. Ace tracking behaviors for the assembly task.
Regardless of the source of the data, at this point in the process it consists of a list of fiducials detected and their 6 dimensional (x, y, z, roll, pitch, yaw) pose in a global coordinate frame. However, the true objects of interest are not the fiducials themselves, but the objects on which they are mounted. The objects, such as the robot’s gripper, the task board, or construction materials, are called bodies. The system already contains the knowledge of all fiducials attached to each body, and the precise transformation from each fiducial to an origin point on the attached body. Thus, the GroupFids block is able to organize the list of fiducial poses by grouping them according to the body on which they appear. Next, the RefinePoseEstimates block determines the 6 degree-of-freedom location of each body for which at least one fiducial is detected. When only a single fiducial is detected, this is a simple matter. When more than one fiducial is detected for a body, those fiducials are unlikely to be in perfect agreement as to the body’s location, simply due to imperfection in the vision system. In this case, the best estimate is determined by an averaging process designed specifically for 6 degree-of-freedom transformations.

The list of body poses is handed off to the BodySelector block, which limits the list to those bodies that are of interest to the rest of the system. For example, many construction and assembly scenarios are likely to contain duplicate instances of identical raw materials (this is not currently the case for Ace, but may be relevant in the future). This block may limit the system’s focus to the materials closest to the robot’s current position. The BuildHints block can use this limited list to direct the motion of a Pan-Tilt Unit (PTU) to better track the objects of interest.

Due to the high rate of processing desired and the relative size of the images, all of the processing to this point is generally performed on a single machine. Now that the information has been pared down to the poses of bodies and fiducials in a common coordinate frame, this information can be easily transmitted to other machines or robots in the scenario. In Figure 5, this state is marked by the “Potential Agent Boundary.”

With knowledge of body locations in a global frame, the BodyFilter block can determine the location of one body relative to another. This is precisely the information used by the VisualServo task in the executive layer (see next section). Similarly, a FidFilter block can determine the relative location of fiducials that need to be tracked. The VisualSearch task tries to ensure that these fiducials remain in the view of the cameras.

4. Ace Tasks

For the Ace project, we have had to develop a new set of task procedures. These are implemented using the Task Definition Language (TDL) [Simmons & Apfelbaum, 1998], as documented in the 9/20/07 report. Interior tasks (called “goals” in TDL) define subtasks and the relations between them; Leaf tasks (called “commands” in TDL) enable and disable behaviors, connect behaviors, and parameterize them.

The Ace task tree (Figure 6) consists of a set-up task (PreDock), a completion task (PostDock), and a VisualServo task, which performs most of the work. The PreDock and PostDock tasks each decompose into a sequence of MoveArm and GotoGoal tasks. The difference between
each instance of those tasks is how they are parameterized. The instance names in Figure 6 give a good indication of what the tasks are supposed to accomplish. For instance, the **PositionArmForDock** task moves the arm out and to the side, so that it is ready to start visual servoing (the wrist fiducial must be placed within the field of view of the cameras). A move-arm task sends its command to the **arm control** block (see Section 3), while a go-to-goal task simply enables a go-to-goal block.

The **visual servo** task (Figure 6) enables the visual servo block and the **visual waypoint monitor** block. The task also connects the output of the tracking system (see Section 3) to the input of the visual servo block. It is the visual servo block’s duty to take the positions of the end-effector and the plug hole, compute the relative end-effector pose command necessary to achieve the waypoint, and send that command to the arm control block. As the end effector moves, the visual servo block passes along the waypoint error (distance between the end-effector’s pose and the waypoint) to the visual waypoint monitor block. This block waits until the end-effector’s pose is within tolerance of the waypoint, pauses the visual servo, and signals the visual servo task that the visual servo is complete.

The top-level goal is to insert a plug into a specified hole on the task board. As described, this is done by visually servoing the plug relative to the hole. However, there is currently no means for directly sensing either the plug or the hole. Therefore, in order to perform the task, we have introduced a pair of fiducials into the scenario. One fiducial is attached to the task board and one is attached to the wrist of the manipulator. These fiducials allow us to extract full six degrees of freedom information about themselves and any rigidly attached bodies (see Section 3). Assuming we know, or can measure, the necessary transforms, the task can now be thought of as a visual servo of one fiducial relative to the other. In reality, we are still servoing the plug relative to the hole, but using the frame transformations allows us to use the fiducials that we
can actually sense in the environment. The transforms that we need are the task board fiducial to plug hole transform (taskFidToHole) and the wrist fiducial to plug transform (wristFidToPlug). However, since commands are sent in the end effector frame, we must decouple the wristFidToPlug transform into a wristFidToEndEffector transform and an endEffectorToPlug transform. Therefore, during every visual servo cycle, each fiducial is observed and the appropriate transforms are applied in order to determine the relative offset or error between the plug and the hole. Then, based on this error, an end effector command is determined by transforming the error into the end effector frame.

The challenge of this approach, and often a source of error, is in measuring the necessary transforms. For this particular scenario, the most challenging transform to measure was the taskFidToHole transform because of the curved shape of the task board. Therefore we chose to measure the wristFidToEndEffector and endEffectorToPlug transforms using rulers and calipers, and then used the tracking capabilities of our system to assist in determine the taskFidToHole. By placing the manipulator in the final docked configuration and then observing the relative location of the two fiducials, we can determine the fiducial to fiducial transform. By algebraically manipulating the chain of transforms we solve for the taskFidToHole transform. Through this process, we acquired all of the necessary transforms. Using these transforms in the actual docking experiments, we determined that our transforms were not quite good enough and we had to manually adjust the taskFidToHole transform by a few millimeters.

In addition to specifying all of the transform parameters, we also had to specify waypoint transforms. For the waypoints, we chose to specify two points: a pre-insertion waypoint and a docked waypoint. The pre-insertion waypoint was chosen such that the tip of the plug was axial with the plug hole and a few millimeters short of being coplanar with the hole. We specified a tolerance of 1 cm for this waypoint. Once this waypoint was achieved, our servo task moved toward the final waypoint which, when achieved, would ensure that the plug was fully inserted into the hole. Section 6 describes initial results of docking experiments.

5. Relative Pose Estimation

An important aspect of the system is to ensure that the cameras provide accurate pose estimates since the tolerance of the final insertion is on the order of a few millimeters. This is exacerbated by the fact that the fiducials are 10-20 cm away from the tip of the bodies to which they are attached, so that small errors in determining the fiducial angles can cause relatively large positional errors at the tips.

When determining the taskFidToHole transform, we noticed that the results were multi-modal. In particular, there were four distinctly separate, but tightly spaced, clusters of readings (Figure 7a). Further investigation revealed that there were two distinct modes for each fiducial (Figure 7b), which, when two fiducials were related to one another, led to the four separate clusters. Fortunately, similar behavior had been seen in the Trestle project. The fix was to have the cameras be closer to the fiducials (on the order of 75 cm, rather than 1 m in the original configuration). Due to the need to have the cameras see both fiducials all the time, placement was lim-
Ited. In the end, we decided to perform the docking with the base of the mobile manipulator parallel to the task board and with the arm off to the side (see Figure 8). This configuration led to a significant improvement in accuracy (Figure 7c– note the change in scale between this plot and those of 5a).

Figure 7. Pose errors between the arm and task board fiducials.
6. Clip insertion experiments

We have accomplished basic functionality for inserting a clip that holds a wiring harness into the task board (Figure 8). In these experiments the clip is loaded onto the end effector manually and from then the robot locates the task board using visual servoing and automatically performs the insertion. Successful clip insertion starting with the mobile manipulator at about 1.5 m from the task board takes approximately 1 minute at this point.

Figure 8. Bullwinkle performing the insertion task

Figure 9. Clip after insertion is complete
A preliminary analysis shows that clip insertion into the task board currently has a success rate of around 60%. Some of the common causes of failure include:

- In many cases, the tip of the plug is very close to the hole (approximately 1 mm) but does not go into the hole because our insertion process does not have sufficient strategies to deal with near misses. This will be addressed in the future, in large part, by using force sensing.

- A major source of missed docks appears to be too much gripper compliance because the spring in the gripper is not strong enough (possibly due to wear over repeated experiments) to hold the clip in exactly the same location repeatedly. We will look at tweaking the design of the gripper (which will likely still need to be used even after we get the Barret hand).

- Several insertions failed because of intermittent problems with the base and the arm. In some cases, the E-stop was invoked on its own (without a signal from the user). Causes for this are unknown, at this time.

We conducted a set of 15 docking experiments with slightly different starting conditions. In these tests, 9 completed successfully, 4 failed but were near misses and 2 failed because of system errors.

7. Future work

Our experience to date indicates that the problem of clip insertion requires high performance in several areas:

- **Dexterous manipulation is required**: we have substantiated that the 5DOF manipulator that we have currently is insufficient for the task of clip insertion, starting from arbitrary configurations. The lack of a yaw rotation at the wrist has restricted configurations of the mobile base and thus reduced the effectiveness of automatic insertion. We expect that this problem will be greatly reduced with the use of the new mobile manipulator.

- **Perception estimates must be filtered**: Docking requires sub-millimeter precision in estimating pose of the inserted clip relative to the task board. We need to ensure that pose estimates are filtered to reduce the effect of noise (which grows with the distance to the fiducials) in the visual pose estimation and that the configuration of the cameras allows high resolution imaging of the fiducials in the late stages of clip insertion. We expect that the location of the cameras on the new mobile manipulator will help mitigate this situation.

- **The process is currently prone to various failures**: While task execution currently is adequate for nominal behavior, it will have to be extended to detect and deal with the myriad of failures that can arise. To be commercially viable, failure rates of much less than 1% must be achieved.
- **Force control is necessary for insertion tasks**: Errors as small as a millimeter can cause an insertion to fail if the only means of feedback is based on geometric errors. We expect that force control at the tip will be necessary to speed up insertion and to make it more reliable.

- **Speed up of insertion**: Currently the process of insertion is slow for several reasons—the mechanism of our mobile manipulator can only move a few cm per second to guarantee smooth motion, relative pose estimation is performed at 2 Hz, and because we compensate for the lack of force control in part by pausing during the insertion process. We expect that several of these steps can be sped up with the use of the new mobile manipulator and faster computing that will be used.

**References**

[ARTag] http://www.artag.net/

