TOWARD PERFORMING A FILTER-VACUUMING PROCEDURE USING A HUMANOID ROBOT

Seth Gee, Michael W. Lanighan, Barrett Ames, Robert R. Burridge

1 TRACLabs Inc., Houston, TX, USA, E-mail: {seth, burridge}@traclabs.com
2 University of Massachusetts Amherst, Amherst, MA, USA, E-mail: lanighan@cs.umass.edu
3 Duke University, Durham, NC, USA, E-mail: cbames@cs.duke.edu

ABSTRACT

As humanoid robotic systems advance, there exist opportunities to further demonstrate their semi-autonomous and autonomous capabilities on human-designed procedural tasks. Robonaut 2 (R2), currently aboard the International Space Station (ISS), provides a testbed for robot execution of tasks designed for crewmembers to perform periodically. R2 is physically capable of dextrous tasks with bimanual tool use, but lacks the sensory, control, and planning algorithms required to handle any of the current maintenance procedures. This paper details a case study extending the library of skills available to R2 by examining the task of vacuuming an ISS bacteria air filter.

1 INTRODUCTION

There are many well-defined procedures describing routine maintenance tasks on the International Space Station (ISS) that crewmembers must perform periodically. In examining this list, it became apparent that the current state of humanoid robots, and specifically Robonaut 2 (R2), is not ready to perform fully autonomous interpretation and execution of these procedures. However, there are subtasks within these procedures that offer opportunities for further investigation as candidates for robotic assistance and execution. R2 on the ISS [1] provides a testbed for the execution and evaluation of subtasks in human-designed procedures. R2 has demonstrated that it is physically capable of many dexterous tasks, specifically bimanual tool use [2]. Unfortunately, it lacks the software capabilities to fully utilize its hardware [3]. In order to develop more software capabilities, a set of tasks was examined for both utility and feasibility. Vacuuming air filters was selected from this list as both feasible and useful to the crew. Cleaning bacteria air filters is a difficult bimanual task for R2 because it depends on visual localization, collision-free motion planning of high degree of freedom arms, and precise force control. In addition, these bacteria air filters play a very important role in the safety and sanitation of the station. As such, it is vital to ensure both that the filter is cleaned thoroughly and that no damage is done to the filter during cleaning. Thus, the techniques developed must be robust and well behaved in the event of error. By accomplishing this task and others like it with R2, ISS crew time can be freed up for scientific work while simultaneously extending the capabilities of R2.

2 SYSTEM TECHNIQUES

The techniques required to successfully vacuum an air filter are split between perception, planning, and control. Each technique was developed independently and then integrated for the demonstration. The following sections detail how each system component was developed, including techniques that were explored on the path to a viable solution.

2.1 Perception Localization Techniques

R2’s head contains a Swissranger rangefinder and two Prosilica cameras behind a visor. A number of vision filters and classifiers were investigated to detect and localize the air filter and nozzle before finally settling on a 3D model-based object tracker called BLORT [4], which provided the most reliability.

2.1.1 Early Localization Techniques

The first stage of detection and localization was image-based object detection. Image segmentation was the first technique explored on the belief that finding a feature to determine which groups of segments were positive would be easier than finding features in the whole image. Two methods of image segmentation were explored. The first was a graph-based image segmentation algorithm that uses local information between pixels to determine boundaries [5]. The results were unsatisfactory because the algorithm does not do well with subtle boundaries and was much slower than expected. The other image segmentation algorithm tested is based on the application of quaternions to color, which allows for inter- and intra-color channels to be measured by the same representation.
This technique represents texture as a quaternion. The algorithm involved transforming the image to the quaternion color representation, applying principal component analysis (QPCA) to the transformed image, and grouping areas with similar low rank QPCA representations using a k-means classifier [6]. Though the QPCA segmentation algorithm works better than the graph-based segmentation algorithm, the texture information is not enough to segment the hose from the background, thus ending the pursuit of image segmentation algorithms.

Next, frequency-based Gabor filters were explored. Gabor filters offer a way to find repeated patterns in the spatial domain [7]. A plain Gabor filter has a very narrow bandwidth, but by using Log Gabor filters, it is possible to increase the bandwidth, and thus detect a larger window of spatial frequencies. The Log Gabor filter was more robust to changes in the scale of the hose. Unfortunately, noise from the floor created many false-positives that matched the nozzle. Although the environment could have been engineered to avoid this artifact, the brittleness discouraged further exploration.

The third kind of object detection algorithm explored was based on cascade classifiers. Three different feature types were tested: Haar, Histogram of Gradients (HoG), and Local Binary Patterns (LBP). The most successful was the HoG approach, which is not surprising as it was specifically designed for detecting shapes in larger images [8], whereas LBP was developed to find patterns [9], and Haar is only capable of categorizing small subsections of an image based on intensity [10]. The cascade classifiers performed well in environments in which they were trained, but as the environment changed they struggled with different lighting or novel scenery.

Next, general stereo vision algorithms were explored. Unfortunately, these algorithms were exceedingly slow and the output had a low frequency oscillation in it. In order to reduce the computation time to a reasonable duration, the images were downsampled to a quarter of the original resolution. At that point the nozzle representation became very thin and the algorithm failed to provide adequate depth estimates for the images provided.

A new approach was investigated using the aforementioned cascade classifier with the stereo algorithms. Initial results showed the nozzle detected in both the left and the right images. The detected locations were then combined with the disparity for the stereo estimation. This produced several pose estimates of the nozzle. These poses were further filtered based on their proximity to the hand based on the assumption that the nozzle will be held by the hand. The results were repeatable and reliable for the nozzle, but failed for the filter because the repetitive nature of the filter led to a high variance in the disparity calculation.

### 2.1.2 BLORT

After extensive testing and continued inconsistent results with the above techniques, a model-based method known as BLORT [4] was explored. BLORT was selected for its 3D CAD model-based approach and its use of only a single camera instead of stereo. Initially, BLORT was working at 1 frame per second, but the CAD model (see Figure 1(a)) contained two orders of magnitude more vertices than any model used by the authors [4]. To overcome this, a simplified CAD model was created. This improved both frame rate and tracking accuracy. Next, BLORT was trained with live footage of the air filter. BLORT selects SIFT features from the video and applies them to the CAD model, as seen in Figure 1(b).

This method performed well, as it could leverage both optical and model based information, even with low fidelity models. BLORT’s detection phase uses model information and associated SIFT information to find an initial pose for the object that is being tracked. Next, the estimate is fed into an edge-based particle tracker that uses approximately 150 particles. Unfortunately, the radial symmetry of the nozzle still caused significant errors. For example, if a complete nozzle model was trained then the nozzle would rotate freely about the longitudinal axis of the nozzle. In order to avoid this error only a partial model was trained. This prevented BLORT from rotating freely about an axis, and still returned satisfactory pose estimates. This allowed BLORT to perform both nozzle and air filter detection.

There are downsides to BLORT, however. One is that it requires a GPU, as the model is inspected using OpenGL. Another is the relative sensitivity of the SIFT features that are used to populate the geometric model. This means that flipping lights on and off in the lab often necessitated retraining the model. Lastly, the software stability is still in flux. There are several hard-coded parameters that, if not correctly set for the particular type of image that is being used, will crash the entire system. Overall, BLORT provided an adequate pose of the nozzle and the filter, as shown in Figure 1(c).

### 2.2 Force Control Techniques

R2 has force-torque sensors located in each shoulder, each forearm, and each phalange of the hands [2]. In this work, we utilized the force-torque sensor located in R2’s forearms to provide feedback for force control. Force control was used to determine when light contact was made between the vacuum nozzle and the filter.
Light contact was critical to prevent damage to the filter but also to ensure that the vacuum nozzle made contact with the filter to ensure it was cleaned.

To understand and implement force feedback, a Jacobian transpose was chosen, which defines the relationship between forces at the end effector and torques along the arm [11]. This relationship is defined as

$$\tau = J^T f.$$  

Here, $\tau$ represents the vector of motor torques, $f$ the forces at the end effector, and $J^T$ is the transpose of the manipulator Jacobian. It is assumed that an underlying joint PD servo controller exists which relates a change in joint positions, $\Delta q$, to changes in joint torques through

$$\Delta \tau = k \Delta q,$$  

where $k$ is a gain matrix. Using these equations, a relationship between changes in forces and changes in joint positions can be specified using

$$\Delta q = k^{-1} J^T (f_{ref} - f_{obs}),$$  

where $f_{ref}$ and $f_{obs}$ are reference and observed forces respectively. In the final implementation, the Moore-Penrose Pseudoinverse of the Jacobian was used in place of the transpose. The inverse Jacobian controller operates in task space, which led to a more linear prodding motion.

The existing gravity compensation for the JR3 forearm force torque sensors was unreliable, as it drifted significantly over the workspace. Tare functionality was introduced that allows the controller to work in the neighborhood of the current joint configuration. This approach was sufficient for small prods since R2 was positioned in front of the filter before force control was called. If the existing model-based gravity compensation was improved then the force control developed should be more reliable and robust across the workspace and not just near the current joint configuration. The controller could be used in other force-sensitive tasks, such as button pressing or lever switching.

### 2.3 Trajectory Planning Techniques

The goal of vacuuming an ISS bacteria filter is to achieve complete coverage of the entire filter. Naturally, a back and forth motion over the filter using the nozzle was the default approach. Assuming the location of the filter is known, a grid of waypoint goals covering the area of the filter can be laid out and an arm trajectory can then be generated that covers the waypoints. To successfully cover the area of the filter and to complete the generated arm trajectories, the kinematic limitations of the robot, potential collisions with objects and an ability to change localized workspaces must be considered.

#### 2.3.1 Waypoints and Reachability

Early development focused on providing R2 with successive waypoint goals. These waypoints were overlaid on an internal model of the filter. Each waypoint became a target pose that was generated using the localized nozzle and applying that transform back to R2’s tooltip, which was the palm grasping the nozzle. The target pose’s frame to solve for was the palm frame. Tests were done that translated and rotated the filter and waypoint goals. As the goals were
translated and rotated, the inverse kinematics (IK) were solved at each waypoint as a test of the default kinematic solver, KDL.

Concurrently, TRACLabs developed a convex optimization-based solver for general IK solving known as TRAC-IK [12]. TRAC-IK and R2’s KDL solver were tested against one another using the aforementioned tests. One of the major differences between TRAC-IK and KDL was the increased solve speed. KDL could take 2.5x as long as TRAC-IK to solve for all of the waypoints. Another major factor in switching to TRAC-IK over KDL was TRAC-IK’s solve rate was much higher leading to more trajectory waypoints solutions. TRAC-IK became the default inverse kinematics solver for the filter coverage leading to a completely solvable trajectory every time it was tested.

Having access to a fast IK solver introduced a new technique of preplanning of waypoints. The IK solver has a way to ask for an IK solution from some start pose to some goal pose. With this method, an entire trajectory of waypoints could be preplanned without moving the robot. This created a fast way to check waypoints for reachability by R2’s arms, leading to an understanding of the reachable workspace for the filter. This technique became vital throughout the vacuuming process.

Another benefit of using TRAC-IK is the ability to specify bounds and constraints on the rotational axes. By loosening the rotational bounds and constraints, more waypoints were consistently solved for in the trajectory; however, doing so inevitably led to poor control of the nozzle. These lenient rotational constraints on the palm caused errors that were compounded at the nozzle. In order to address this imprecision, a dynamic tooltip frame is added to the kinematics chain. This allows new tooltips to be defined given some transform between the new tooltip and the palm. In this case, the end of the chain moved from the palm to the localized nozzle. Now, solving for the nozzle instead of the palm provided more control at the nozzle tip. Finally, the looser rotational constraints allowed TRAC-IK to find a solution while continuing to maintain an appropriate distance from the filter.

2.3.2 Trajectory Following

The back and forth vacuuming motion over the filter’s surface was the primary trajectory-following action to complete the vacuuming coverage. Three methods were explored in hopes of finding a generalized filter vacuuming technique.

The first method consisted of holding the filter in one hand and holding the nozzle the other hand. The vacuuming hand was to cover as many of the waypoints as possible until the reachable space was exhausted. R2 then shifted the filter into another part of the reachable workspace and R2 would continue to cover waypoints until the reachable space is exhausted. This process was repeated until every waypoint was covered.

The second method again consisted of holding the filter in one hand and holding the nozzle the other hand. The vacuuming hand again covered as many of the waypoints as possible until the reachable space was exhausted. The nozzle hand then remained stationary while the filter hand moved the filter under the nozzle tip. This method maintained the trajectories originally intended for the nozzle relative to the filter. The filter hand would continue to follow the trajectories matching the nozzle waypoints until the reachable space was exhausted. This process would repeat until every waypoint was covered.

The third method consisted of the filter remaining stationary and covering as much of the filter as possible with one hand, then transferring the nozzle from one hand to the other to complete the remaining waypoints.

All methods were first tested in the simulator and then the test was replicated on R2C6. The third method was chosen for its speed, efficiency, and limited workspace in a potentially crowded environment, such as that on the ISS.

2.3.3 Hand-to-Hand Transfer of Tools

One of the long-term goals of the project was to achieve hand-to-hand transfer of tools. Though most of the vacuuming points could be reached using a combination of the waist and a single arm, some vacuuming waypoints could only be reached when the nozzle was in the other hand. Various hand-to-hand tool transfer techniques were explored to enable autonomous hand-to-hand transfers. Considerations while exploring the techniques included deciding how to orient the palms during transfer and narrowing down an appropriate workspace within which to perform the transfer. A fully autonomous hand-to-hand tool transfer followed the steps outlined in Procedure 2.3.3.

Figure 3 demonstrates the hand-to-hand tool transfer sequence from the right to left hand of the robot.

2.3.4 Collision Detection Using MoveIt!

During initial development, an obstacle-free environment has been a key assumption in the operation of the robot. In order to make this task and other potential tasks safe, obstacle-aware trajectories needed to be generated. The MoveIt! planning package [13] was determined to be the best approach to obstacle avoid-
Figure 3: Hand-to-hand transfer of tools sequence

Procedure 1 A generalized approach for hand-to-hand tool transfers

Require: Nozzle is held in hand 1, hand 2 is empty
1: Hand 1 is positioned in front of R2
2: Nozzle is localized and a Cartesian goal is created for hand 2
3: Once the nozzle is in hand 2's reach, hand 2 grasps the nozzle
4: Hand 1 releases the nozzle and is moved back to a ready pose
5: The nozzle, now grasped in hand 2, is centered in front of R2
6: Nozzle is re-localized

ance because the needed configuration and launch files already exist for R2 and because of MoveIt!'s integration with depth sensors like the one in R2's head. MoveIt! is a planning suite that includes access to multiple kinematics solvers as well as various sampling based planners. It also includes obstacle-aware trajectory generation, which uses input from sensors to build an environment that includes any obstacles that may be in the robot's workspace. One challenge when introducing MoveIt! into the planning architecture was programmatically using MoveIt's software suite through an API instead of the traditional user interaction through RViz which is fairly complex and very rich with features.

Building on the technique of using TRAC-IK to presolve a Cartesian pose before any execution, if TRAC-IK returned a valid joint set for the pose, the joint set was passed to MoveIt! as a joint trajectory goal. MoveIt! then searched for an obstacle, and if one was found to be blocking a trajectory point making the goal unreachable then the waypoint was skipped and the next waypoint was evaluated until a valid point was found. The next valid point would provide a path around the obstacle, and allow R2 to continue executing its task. Figure 4 demonstrates the technique described above.

3 SYSTEM INTEGRATION AND DEMONSTRATION

Throughout development, all individual components of the system were tested using a publicly-maintained physics-based Gazebo [14] simulation of R2 within the ISS environment. After successful execution in Gazebo, system techniques were tested and demonstrated on-site at NASA/JSC on R2C6, which is one of the ground-based versions of R2. The final demonstration was performed with a mockup of the bacteria air filter found on the ISS.

3.1 Integrating System Techniques

The system was integrated through NASA's Robot Task Commander (RTC) [15], an interface that uses ROS [16] to interface with the robot. Integrating all of the system techniques began with the perception localization. BLORT provides poses of the nozzle and air filter to the RTC block, which then updates the tooltip transform and the pose of the air filter. These localized values are then used to calculate the trajectory and overlay the waypoints on the filter. The overlaid trajectory waypoints on the ISS bacteria air filter mockup can be seen in Figure 5.

Access to the ISS bacteria air filter mockup necessitated a change in the trajectory following strategy. The change in strategy was to switch from a dragging motion to a prodding motion. The prodding motion required each waypoint goal in the trajectory to have an approach pose two centimeters from the desired final goal pose. Having this approach point allowed R2 to reach for the air filter, using tactile servoing, and then back away from the air filter after a successful touch. R2 then moved on to the next point. Utilizing this approach-and-retreat strategy avoids dragging the hose along the uneven air filter and reduces the risk of accidentally applying too much force to the
filter during translation. Cartesian position control is used for all motions except the tactile servoing.

Once the force control had been integrated into the general trajectory-following algorithm, MoveIt!’s obstacle-aware trajectory generation was included with the control. For each waypoint, TRAC-IK computed the IK and returned the joint goals. Upon success, the joint goal was then sent through MoveIt! which includes information on any obstacles that are added into the planning environment. If there are no obstacles present in the planning environment then the resulting trajectory looked similar to the previous non-obstacle-aware trajectories coming from TRAC-IK. While the MoveIt! results were successfully integrated and tested in the simulator, including obstacles in the mockup vacuuming environment was deemed unsafe as some of the plans that MoveIt! generated included many twists of the arm and waist to achieve the goal.

3.2 Demonstration
R2 was able to successfully localize the filter and nozzle, proceed to vacuum the filter using a mix of Cartesian and force control, and transfer the nozzle between hands when necessary in order to complete filter coverage. In addition, R2 did not apply forces to the filter that would have damaged it during the task execution. The final, fully autonomous demonstration followed the steps outlined in Procedure 2.

Procedure 2 ISS bacteria air filter vacuuming
Require: Nozzle is in the left hand of the robot.
1: Move the left hand in front of R2
2: Localize the nozzle
3: Generate trajectory waypoints for vacuuming
4: while valid waypoints remain do
5: for each hand do
6: for all waypoints in reach of hand do
7: Find IK solution to waypoint
8: Generate a collision-free trajectory
9: if no trajectory available then
10: Mark waypoint as not valid
11: else
12: Execute generated trajectory
13: Force control to filter surface
14: Retreat back to the waypoint
15: end if
16: end for
17: Execute Procedure 2.3.3
18: end for
19: end while

3.3 Issues Encountered
Immediate testing on R2C6 demonstrated that Cartesian pose control was not as smooth as idealized by Gazebo. Another difference when moving from Gazebo to R2C6 was that IK failure could be ignored in Gazebo, but failure on R2C6 usually caused the robot to park its joints. Once that happened, R2C6 would have to reset and begin the trajectory again from the start. To handle the robot failures, the pre-planning technique mentioned in section 2.3.1 was created to catch potential failures and allow for some other action.

4 CONCLUSION
In order for R2 to vacuum an ISS bacteria air filter, model-based object recognition, trajectory planning
and following with assistance from force feedback control, hand-to-hand transfer of the nozzle, and MoveIt!’s obstacle awareness had to be integrated together using RTC forming a cohesive and successful demonstration. This paper discussed the methodologies and techniques required to develop the skills necessary to integrate the system together. The skills developed are now available to R2 in its library and open up potential building blocks for other skills to be learned and developed. As humanoid robotic systems continue to advance and become more autonomous, these new skills will provide a foundation for automating other tasks required by the ISS crewmembers.

Acknowledgment

This work was supported by NASA grant NNX12CE96P. We would like to thank Julia Badger, Dustin Gooding, Josh Mehling, Vienny Nguyen, and Allison Thackston for their assistance with this project.

References


