Demonstrating Autonomous and Robust Sorting in a Glovebox Environment

Adam Allevato, Matthew W. Horn, and Mitch Pryor

Nuclear and Applied Robotics Group, The University of Texas at Austin, 10100 Burnet Rd, Austin, Texas 78758, USA
allevato@utexas.edu, mwhorn@utexas.edu, mpryor@utexas.edu

INTRODUCTION

Even with the widespread growth of robotics, glovebox automation systems still face significant challenges on their path to maturity and eventual adoption in the nuclear-industrial complex. National labs and other institutions have countless uses for a flexible, autonomous, in-place system capable of performing hazardous tasks, but so far the risks and challenges of such a system have outweighed the benefits.

The various hardware and software components in industrial robotics have advanced in recent years, allowing us to build systems that overcome challenges and approach practical usefulness. To prove the usefulness of these advancements, we chose to implement the open-ended task of sorting solid radioactive waste autonomously in a glovebox. Handling and characterization of solid radioactive waste is one of the most widespread tasks involving radioactive materials. According to the International Atomic Energy Agency (IAEA), several countries face the issue of how to process waste from temporary or interim facilities [1], and the agency also provides specific instructions for collection and packaging of solid wastes. The IAEA warns that "it may become necessary to combine the range of pretreatments, monitoring, dismantling, shredding and deconamination with the sorting and packing stages" [2], making sorting an multifaceted task that can show off of many capabilities of an automated system. By performing basic sorting autonomously, we show our system's ability to intelligently and safely perform a common glovebox task, and also build a foundation to demonstrate higher-level capabilities in the future, such as material reduction or intelligent packing.

To complete even a basic task involving hazardous materials in a glovebox poses many high level challenges. One of the first issues is installing a manipulator in a glovebox. While this may seem like a trivial challenge, the amount of logistics and certifications required to install a machine in a hazardous environment (such as a national lab) quickly becomes prohibitive. Cable-driven manipulators in hot cells, such as the M-2 ([3]), remain in place for years after deployment. A more robust automation system should have the ability to switch between various deployment configurations once installed. This will allow the system to adapt to the needs of the current task.

Safety is another paramount concern in gloveboxes, especially when materials such as plutonium are involved. A good automation system should involve safety checks at every level, to protect both the system itself as well as human operators.

To perform more open-ended or custom tasks, an automation system must also be extremely flexible, both in terms of hardware degrees of freedom as well as software capability. A good system should be able to complete generic tasks such as sorting, packing, interacting with other machines, and rearranging objects, all in an unconstrained environment. Finally, there must be a determination of the appropriate level of autonomy for the remote system, see Table

Level	Tele-operation → Autonomy…
1	Reduce or eliminate operator's need to manage the robot's internal configuration.
2	Reduce or eliminate the operator's responsibility for avoiding undesired contact with the environment.
3	Reduce or eliminate the operator's responsibility for moving the robot to locations of interest.
4	Reduce or eliminate the operator's responsibility for selecting a proper grasping configuration for retrieving selected objects.
5	Allow the operator to quickly direct the system to complete tasks that involve subtasks completed as directed in levels 3 & 4 (such as pick and place).
6	Reduce or eliminate the operator's responsibility to avoid threshold forces for contact tasks such as opening a door or lifting items exceeding the system's payload.
7	Reduce or eliminate the levels of detail that are necessary for the operator to communicate a task (or subtasks) to the robotic system.
8	Integrate capability to complete tasks that require high levels of precision and/or the control of a specific force profile.
9	Reduce or eliminate the need for the operator to be in the loop for tasks that respond to non-operator, independent external events (i.e. timer on oven, low battery notification, etc.).
10	Based on prior tasks completed, the system anticipates future tasks to be completed based on historical use.

TABLE I. Transitional levels of autonomy from Brabec et. al. [4]

In previous sorting efforts [1] [2] [5] [6] discussed below, the highest level of autonomy on a deployed system was Level 1, where the operator is responsible for every facet of the robot's movements.

Recent efforts at UT Austin and in the broader community make it possible to demonstrate relevant sorting tasks at a level 6-7 in Table

RELATED WORK

Automating radioactive waste sorting is not a new idea. In 2002, the United States' Department of Energy (DOE) began development of the Handling and Segregating System for 55-gallon drums of mixed waste, or HANDSS-55. Once implemented, the system would have consisted of multiple modules to unpack 55-gallon drums, characterize and sort the waste inside, and repackage the appropriate waste for disposal at a long-term storage site, such as the Waste Isolation Pilot Plant (WIPP) in New Mexico. By including automatic object detection and characterization and performing simple tasks for the operator [7], [6], the system would have achieved Level 4 on the autonomy scale. HANDSS-55 was designed to be deployed at the Savannah River Site by 2005, but the the DOE Robotics Crosscutting program was discontinued in 2002, halting progress on the project.

DOE is advancing levels of autonomy in facilities, with

one notable system being the Advanced Recovery and Integrated Extraction System (ARIES) at Los Alamos National Lab. The system includes a three-axis gantry robot for transporting materials [8]. ARIES operates at an autonomy level of 4 (reducing or eliminiating the operator's responsibility for grasp configurations), but its flexibility is extremely limited. This system has been in development for 20 years, which suggests that glovebox manipulation is a solved problem. However, the ARIES system is designed to handle an extremely specific subset of hazardous materials–plutonium weapon pits. We seek a more general, robust solution.

Current technology employed by nuclear facilities and waste-sorting facilities include remote-controlled diggers, teleoperated robotic arms, and remote grappling devices. Robotic Manipulation for Nuclear Sort and Segregation (RoMaNS) [9] is a recent effort put forth by the European Union for automating radiation sorting tasks. Ortenzi et al. [5], a member of RoMaNS, compares two different control schema for robot/environment contact tasks that should help improve future operation of radiation sorting. Specifically, in Minimization and Segregation of Radioactive Wastes published by the IAEA [2], several sorting tasks of materials that pose a large risk to operators must be done with rakes, pushers, or teleoperated robotic manipulators.

NRG has developed key technologies for robust glovebox automation as separate projects, and our manipulation system incorporates many recent advancements, producing safer and more robust results than previous attempts.

Brabec et al. installed various manipulators in gloveboxes to perform simple tasks, such as grasping objects, and also showed the ability to open cabinet doors using a 3D vision-based approach. Their results proved the concept of open-ended glovebox automation deployment, but did not perform start-to-finish tasks in a glovebox. [4].

Computer vision is a key tool in providing robust automation in the face of uncertainty. Allevato et al. have shown the ability to discern between 12 different classes of object in a glovebox, even when the objects have little or no surface texture and high specular reflectivity [10]. The code used to produce the results in their paper is currently used for object detection at NRG.

Once objects have been located, the next step is to generate motion plans and grasp strategies for manipulators. Horn and Sharp have developed a motion plan scoring system that randomly samples valid plans and executes the highest scoring plan based on learned user preferences. Brabec created a shape-primitive-based grasping strategy that generates valid grasp points on objects through visual data [11]. Both of these approaches have been tested in confined, hazardous environments successfully.

TASK OVERVIEW

We wish to sort objects of three different colors: red, green and blue. The objects vary in shape and size, e.g., pieces of hook-and-loop fastener, machine parts, and wooden blocks. The objects begin in randomized positions on the glovebox floor, and the number of objects in each color category can range from 0 to 5, for a maximum of 15 objects. The scene

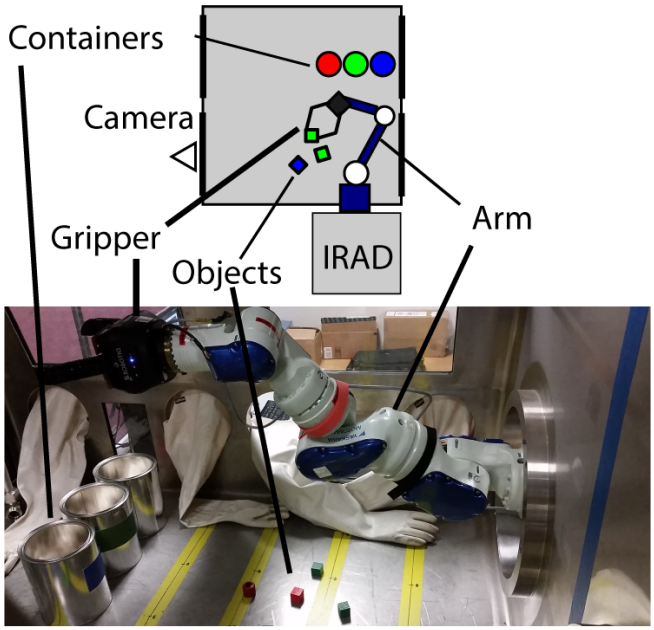

Fig. 1. Above: top view diagram of the major components of the glovebox automation system. Below: The inside workspace of the glovebox from the camera's viewpoint.

also includes three containers, one for each color of object (Refer to Figure

To simulate an actual glovebox manipulation task as closely as possible, the experiment was performed in NRG's cold glovebox, with windows and gloves installed. The floor of the glovebox has brightly-colored tape marking out distances (see Figure

IRAD's design includes an Agile Planet controller for the SIA5, as well as a Windows PC which communicates with the controller, allowing for manual or automated operation. In addition, our experimental setup uses an Ubuntu 14.04 machine for automating commands to the various system components.

By building the glovebox automation system out of many self-contained components, we have constructed a system that is flexible to different tasks and constraints. For example, we could install one arm in an upper gloveport and one in a lower gloveport (perhaps each with different end effectors) and move the camera to the opposite window, then recalibrate, and the system would be prepared to work on a different task.

SOFTWARE COMPONENTS

A system capable of autonomous sorting requires many complex components. While the top-level components of our system were developed at NRG, we list the underlying systems here for completeness.

System framework: Our main Ubuntu machine uses Robot Operating System (ROS) to orchestrate the many components of the experiment. ROS handles multithreading automatically for this complex application, and also provides the RViz visualization environment, which is the main operator interface for the demo. IRAD's controller commands the SIA5 as dictated by a dedicated Windows machine, which does not

run ROS. To allow the SIA5 to be controlled via ROS, we use a driver developed in-house that passes commands between the Ubuntu and Windows machines. We have recently extended this driver to allow both joint trajectory execution and simple directional jogging.

Motion planning: The ROS package MoveIt! plans motion trajectories for the manipulator using various motion planners included in the Open Motion Planning Library (OMPL) [12]. Different planners in MoveIt! produced varying results, and the best compromise between planning speed and resulting path quality was the RRTConnect planner [13], which uses rapidly-exploring random trees to generate motion paths.

Image Processing: We use three off-the-shelf libraries for computer vision. OpenNI2 acquires 3D image data from the Xtion sensor. Our custom vision library (see below) heavily utilizes Point Cloud Library (PCL) for 3D analysis and OpenCV for color analysis and histogram characterization.

Perception: For this experiment, we have extended our Object Recognition and Pose (ORP) library to discern between small colored objects. As in previous work [10], we perform various methods of filtering to the incoming 3D point clouds: voxel subsampling, spatial clipping, RANSAC plane removal, and Euclidean clustering convert the clouds from full scenes to clusters representing the individual objects on the glovebox floor. In previous ORP applications, we would then run a point cloud-based classifier on the data, but in this case we copy the point clouds into OpenCV objects. Then, using OpenCV, a new classifier developed specifically for this application generates a red/green/blue (RGB) histogram by summing the red, green and blue values of each point in the cloud. Finally, the strongest histogram channel is returned as the object's color classification.

By looking only at color information, we reduce the number of object classes to three, which raises the question of why complex 3D point cloud processing is necessary. Without point clouds, our OpenCV algorithm would not be able to distinguish between green tape on the glovebox floor and a green cube, but by removing the floor using 3D analysis, we can greatly improve the robustness of our (extremely fast) color detection technique. We also use the 3D point clouds to determine the 6DOF position of the object, reducing the amount of uncertainty and making manipulation easier.

Grasping: The installed Robotiq gripper includes its own ROS packages, but the library often exhibits unexpected behavior, making it unsafe to command directly. For example, sending an "open" command to the gripper before it has been activated causes the the activation procedure to execute, which depending on the robot configuration can result in collisions between the robot and the environment. To allow easier control and safer execution, we wrote a library that wraps the gripper's low-level inputs and outputs, handles activation intelligently, and provides easy access to its fault states. We also make use of our previously-developed custom RViz plugin for the Robotiq, which provides more accurate position feedback for visualizing the gripper.

Task Execution

The execution loop is the final software component of the experiment. Thanks to the number of self-contained libraries, modules, and small improvements we have made to our system, the main loop is straightforward, while still resulting in complex behavior during the demo. Pseudocode for the main application is shown below. Even with verbose formatting, the main application is under 400 lines of code.

```
1  detect objects
2  classify objects
3  while(object list is not empty):
4    try
5      move over object
6      grasp object
7      verify grasp
8      lift object
9      move over appropriate receptacle
10     release object
11     remove object from list //success, next
12   catch
13     if(object grasped):
14       put object back
15     remove object from list //failure, skip
16   move to home
17   wait for user confirmation
18 end
```

Lines 1 and 2 trigger an object recognition loop in ORP (see above). The loop runs for a short time to filter noise and get a good representation of the scene, then returns the list of objects in the scene along with their classification (red, green, or blue).

The loop beginning at line 3 goes through the objects one by one, and attempts to sort them into containers. Lines 5-10 primarily use ROS and MoveIt!, along with our custom gripper libraries, to perform the robot motion. Note that these liens could easily be combined into a single instruction if a mid-level task framework was created. After sorting an object, the robot gets out of the operator's way (line 16) so that they can inspect the workspace before proceeding.

Asking the user for confirmation between each sort (line 17) is only one layer of a multilevel security architecture built into the application. Because the system is so complex, it can fail in many ways. Some examples are: failure to create a valid motion plan, cable disconnection during robot motion, robot joint speed violation, and a missed grab. We built 6 levels of safety into the application, ordered here from least to most autonomy:

1. Emergency stop: at any time, the user can trigger a hard emergency stop, stopping all hardware immediately.

2. Software stop: at any time, the user can trigger a "soft" stop to either the Windows machine (stopping the physical robot), or the Ubuntu machine (stopping the entire application).

3. Operator confirmation: Before beginning the task, and after each object is sorted, the user is prompted for confirmation before continuing. Note that this safety layer can be disabled to allow autonomous sorting.

4. Joint torque limiting: the robot's controller automatically shuts off and locks the arm's servos if it detects a motor torque above a certain threshold.

5. Grasp validation: by monitoring the joint positions of the gripper, we are able to detect an invalid or missed grasp (line 7) and react accordingly, elevating to level 4.

6. Intelligent object skip: if object sorting should fail for any reason (usually unable to generate a robot motion plan), the object is first returned to its original location if it had been moved. Because the system has encounted a problem of some type while sorting this object, it is assumed to be a difficult-to-manipulate object, because of its location or otherwise, and is not revisited for sorting unless the user specifically requests it.

RESULTS

The system described in the previous section successfully ran multiple times per week over a period of 3 months. The strength of the system lies in the large codebase of self-contained components, which allows the main application code to be simple and generic. One can easily parse the pseudocode above with a different task in mind, e.g. packing different-sized objects into containers to minimize wasted space, sorting objects by their radiation signature instead of by their visual color, or sorting objects based on weight or other physical features. By abstracting the task as much as possible and changing out the various components, the application becomes a model for executing other glovebox tasks safely and reliably with a multitude of applications.

The histogram-based color analysis was 100% accurate, even when classifying objects that had specular reflections, that lied in shadow, or that were not entirely one color (see Figure

We observed 4 failure modes, which can be attributed to two issues with the underlying design: the calibration and the motion planner. The failure modes were:

- Failure to generate a plan
- Generated plan includes a collision with the environment
- Unplanned collision with environment
- Missed grab

The RRTConnect planner would often fail to produce a plan in the 5 seconds allotted to it. Increasing the timeout would result in more planning success, but the resulting plans would be complicated and involve unnecessary motions. But despite this shortcoming, the system still was able to sort the vast majority of objects, and if a plan failed, it often succeeded upon retrying because of the randomness of the path planner. The other more dangerous failure mode associated with the planner was when a motion path would cause a collision between the robot and the environment. This failure was caused by the planner not performing high-fidelity collision checking, and could be mostly mitigated by increasing the thickness of the 3D models in the virtual collision scene.

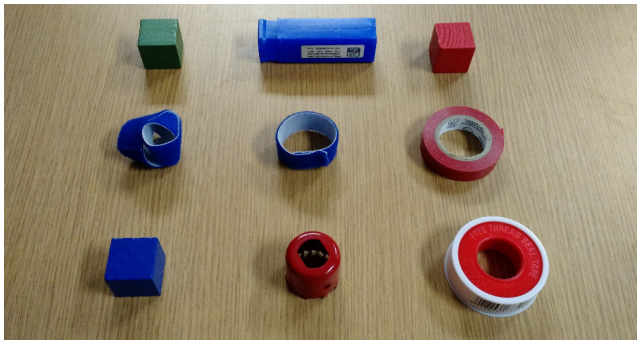

Fig. 2. Some of the objects sorted, all of which were correctly classified using histogram analysis.

Calibration issues accounted for the rest of the failures. Imperfect mounting solutions for the 3D camera often resulted in misalignment between the visual information and the virtual representation of the scene, causing missed grabs and environment collisions. For missed grabs, our safety architecture allowed the system to immediately recover and skip to the next object, with an option to return to the missed object in the future. In the case of environment collision, operator emergency stop was required, followed by correcting the calibration issue.

CONCLUSIONS AND FUTURE WORK

In this paper we showed the need for reliable and safe flexible in-glovebox manipulation, and demonstrated our approach to implementing one such system. Our system combines commercial hardware, open-source software, and custom solutions to integrate dynamic robotic manipulation into a useful framework. In addition, we added libraries to increase modularity and safety architectures to decrease operation risk. We explored several different components to include in the framework, and built a multi-layer set of safety checks. We have proven our system's viability by completing a color-cube sorting task reliably and safely, and demonstrated that the main application logic is generic and simple to understand. We have also proven our system's ability to adapt to different tasks and needs by exploring several different planners that require minimal changes to the system.

Future plans include adding improvements to the application code and supporting components to improve interoperability. As mentioned previously, many parts of our application were developed in parallel by NRG researchers, and as a result, active research is progressing on various application components. Future improvements to the ORP library will allow us to sort objects by multiple modes, such as shape [10] and radiation signatures. NRG is also working to improve motion planning using machine learning. Finally, NRG is currently testing a framework for custom force control, allowing complex moves such as peg-in-hole or surface following. We plan to use this framework to develop more complex actions,

such as placing samples in precise locations, or using special end-effectors to turn bolts and screws.

In addition to incorporating new work from other researchers, we will continue to iterate our system's design and generalize to more applications, including corobotics (human-robot interaction), bin packing, and interacting with centrifuges, valves, and other small devices in a glovebox. These changes will increase the level of autonomy for tasks completed in glovebox environments.

ACKNOWLEDGMENTS

This material is based upon work supported by Department of Energy Nuclear Energy University Programs Graduate Fellowships.

REFERENCES

1. IAEA, "Retrieval and Conditioning of Solid Radioactive Waste From Old Facilities," Tech. Rep. 456, International Atomic Energy Agency (2007).
2. IAEA, "Minimization and segregation of radioactive wastes," Tech. Rep. 652, International Atomic Energy Agency (1992).
3. J. HERNDON ET AL., "The State-of-the-Art Model M-2 Maintenance System," in "proceedings of the National Topical Meeting on Robotics and Remote Handling in Hostile Environments," (1984).
4. C. BRABEC ET AL., "Reducing the Operator's Burden During Teleoperations Involving Contact Tasks," in "3rd Int. Joint Topical Meeting on Emergency Preparedness and Response and Robotics and Remote Systems 2011, EPRRSD, and 13th Robotics and Remote Systems for Hazardous Environments," (2011).
5. V. ORTENZI ET AL., "Projected Inverse Dynamics Control and Optimal Control for Robots in Contact with the Environment: A Comparison," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS* (September 2015).
6. C. M. FRAZEE and M. E. BRENNAN, "HANDSS-55: A TRU Waste Repackaging System for the Savannah River Site," in "proceedings of the 2001 Waste Management Conference," (2001).
7. R. SMITH, "Helping HANDSS for Sorting Waste," *Radwaste Solutions*, **9**, *6*, 47–52 (November/December 2002), `www.ans.org/pubs/magazines/download/a_215` Accessed 1/8/2016.
8. T. E. SAMPSON and T. L. CREMERS, "Integrated Nondestructive Assay Solutions for Plutonium Measurement Problems of the 21st Century," Tech. rep., Los Alamos National Laboratory (1997).
9. "romans," `http://www.h2020romans.eu/`, accessed: Accessed 1/1/2015.
10. A. ALLEVATO ET AL., "Using a Depth Camera for Object Classification in Nuclear Gloveboxes," in "American Nuclear Society Student Conference," (2015).
11. C. L. BRABEC, "A shape primitive-based grasping strategy using visual object recognition in confined, hazardous environments," (2013).
12. I. A. ŞUCAN ET AL., "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, **19**, *4*, 72–82 (December 2012), `http://ompl.kavrakilab.org` Accessed 1/8/2016.
13. J. KUFFNER and S. LAVALLE, "RRT-connect: An efficient approach to single-query path planning," in "Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on," (2000), vol. 2, pp. 995–1001 vol.2.