Charlie Rides the Elevator–Integrating Vision, Navigation and Manipulation Towards Multi-Floor Robot Locomotion

Daniel Troniak, Junaed Sattar,
Ankur Gupta, & James J. Little
Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada.
{troniak,junaed,ankgupta,little}@cs.ubc.ca

Wesley Chan, Ergun Calisgan
Elizabeth Croft & Machiel Van der Loos
Department of Mechanical Engineering
University of British Columbia
Vancouver, B.C. Canada.
wesleyc@almuni.ubc.ca,ecalisgan@ieee.org,
{ecroft,vdl}@mech.ubc.ca

***Abstract*— This paper presents the design, implementation and experimental evaluation of a semi-humanoid robotic system for autonomous multi-floor navigation. This robot, a Personal Robot 2 named Charlie, is capable of operating an elevator to travel between rooms located on separate floors. Our goal is to create a robotic assistant capable of locating points of interest, manipulating objects, and navigating between rooms in a multi-storied environment equipped with an elevator. Taking the elevator requires the robot to (1) map and localize within its operating environment, (2) navigate to an elevator door, (3) press the up or down elevator call button, (4) enter the elevator, (5) press the control button associated with the target floor, and (6) exit the elevator at the correct floor. To that end, this work integrates the advanced sensorimotor capabilities of the robot - laser range finders, stereo and monocular vision systems, and robotic arms - into a complete, task-driven autonomous system. While the design and implementation of individual sensorimotor processing components is a challenge in and of itself, complete integration in intelligent systems design often presents an even greater challenge. This paper presents our approach towards designing the individual components, with focus on machine vision, manipulation, and systems integration. We present and discuss quantitative results of our live robotic system, discuss difficulties faced and expose potential pitfalls.**

I. Introduction

Service robots are becoming more commonplace occurrences, both in industrial and home-use scenarios. Such robots have been used for a variety of tasks, such as personal guidance, delivery, cleaning, assembly-line assistance, health-care and more [1][2][3]. In most cases, robots engaged in mobile service tasks (*i.e.*, where the robots need to be mobile to perform assigned tasks, as opposed to fixed-mount installations such as assembly-line robots) have one common requirement – the ability to navigate in their workspace robustly and accurately. A variety of such work environments exists, including but not limited to museums, hospitals, airports, warehouses and office buildings. In many cases, robots would need to navigate between floors in these installations to perform a number of tasks. Unless the robots are ambulatory (*i.e.*, using legged locomotion), climbing stairs is not an option. Even if the robot is able to climb stairs, there are significant challenges to overcome, particularly if there are tools to deliver, including load capacity while climbing stairs, balance maintenance, and manipulation of building objects such as doors while carrying a load. In most buildings with multiple floors, particularly commercial installations, there are elevators to facilitate the moving of people and goods alike. Elevators are accessible by both wheeled and legged robots, can be operated with a simple manipulator, and are simple locations to reach, given a map of the environment. The challenges to enable multi-floor autonomous navigation, thus, can be reduced to (1) robust navigation, (2) sensory detection of elevator controls and elevator state, and (3) manipulation of said controls.

Fig. 1: Charlie, our PR2 robot, using its projected light-enhanced stereo vision system to discover the 3-D location of an elevator call button.

This paper presents our work towards enabling multi-floor autonomous navigation on our semi-humanoid wheeled robot, called Charlie (see Fig. 1). Charlie is an instance of the Personal Robot version 2 class of robots, and is equipped with stereo and monocular vision, projected-light textured stereo capabilities, multiple laser range finders, inertial measurement units and two arms with grippers. For simple manipulation tasks such as pickup and push, the arms and grippers provide sufficient capabilities. Charlie operates using the Robot Operating System (ROS) suite [4], which is an open-source robot middleware system, providing interfaces between the low-level hardware controllers and higher-level applications. ROS also provides a number of

978-0-7695-4983-5/13 $26.00 © 2013 IEEE
DOI 10.1109/CRV.2013.12

pre-installed packages, which can be used as-is, or to provide enhanced capabilities beyond what is currently possible.

In the remainder of the paper, we present our approach to enable multi-floor navigation with Charlie. Tasks are classified primarily into navigation, visual sensing and manipulation. The different methods to achieve each of these individual tasks are described in detail. We also discuss our approach to integrate the individual components into a fully coherent system. For a personal robot working in a predominantly human-occupied environment, this work has enabled us to recognize a number of important issues that need to be addressed for successful deployments of robots. From the perspective of cognition, we highlight our experiences with multisensory perception, particularly when facing the challenges in a highly changeable operating environment. Furthermore, from a systems perspective, we discuss the importance of software design principles in creating pragmatic robotic programs, namely those of reusability, coherence and decoupling – direct benefits arising from the use of the ROS middleware. Finally, we present quantitative results to shed light on the system performance, in terms of speed and accuracy.

II. Related work

Our work is motivated by the scenario of personal robots working as human assistants in home and industrial settings, helping in a variety of tasks. For successful execution of such tasks, the robot is required to possess sophisticated abilities to (1) navigate, (2) perceive and manipulate objects of interest, and, (3) interact with human users. As such, our current work spans the domains of navigation, robot control, manipulation and vision-guided robotics. We briefly highlight some related work in these domains.

Simultaneous Localization and Mapping (SLAM) [5] is a rich area of robotics research, and a large body of literature exists to reflect that fact(see [6]). Of particular interest is the Rao-Blackwellized Grid Mapping technique [7], which is the algorithm used by our PR2 robot to localize and map.

Within the domain of robotic manipulation, our particular focus is on manipulation through pushing [8]. Manipulation in cluttered and in constrained spaces, particularly using vision as a sensor [9] is also relevant to us, as the robot has to move itself, and its manipulators to reach and push the correct button corresponding to the target floor.

Our robot Charlie is an instance of the PR2 robot created by Willow Garage (Menlo Park, CA, USA), which has been created to aid in research and development in personal robotics [10]. The PR2 is a semi-humanoid robot, on an omnidirectional wheeled base, with two manipulators, cameras in the head and arm joints, an extensible torso, and comes equipped with a number of sensors including LIDARs and inertial measurement units. The PR2 software stack is fully based on the Robot Operating System (ROS) [4], providing interfaces between low-level hardware controllers and user-level applications to provide intelligent autonomous behaviors. The robot can also be tele-operated using a joystick. A number of robotics research groups have developed intelligent autonomous capabilities for the PR2 (examples include [11] and [12]). The motivation of our work in particular stems from the work by Kunze et al. [13], where a PR2 robot is given the ability to carry out *find-and-fetch* tasks, for example to grab a sandwich from a kitchen or a food stall. In that work, the robot also has to operate an elevator and traverse multiple floors. Our work is similar in flavor, although our approach differs in the underlying techniques applied. The longer-term goal is to have not just a purely autonomous robotic system, but one that interacts with human users, receiving and asking for instructions as necessary to ensure successful task completion. The work presented in this paper thus can be considered one step towards that eventual goal.

III. Technical Approach

The overall goal of our work is to enable the PR2 to perform navigate-and-fetch tasks in a multi-floor environment that includes an elevator. As described previously, the primary task here is to find the elevator, operate it to reach a different floor, and navigate out of the elevator to reach the destination. This requires solving a number of issues in manipulation, perception, and navigation. The following sections provide further details about each of these sub-areas. We then discuss the integration stage which supports the complete, functional behavior. The entire codebase for this work is available under an open-source license, located at the `UBC-ROS-PKG`[1] SubVersion repository on SourceForge.

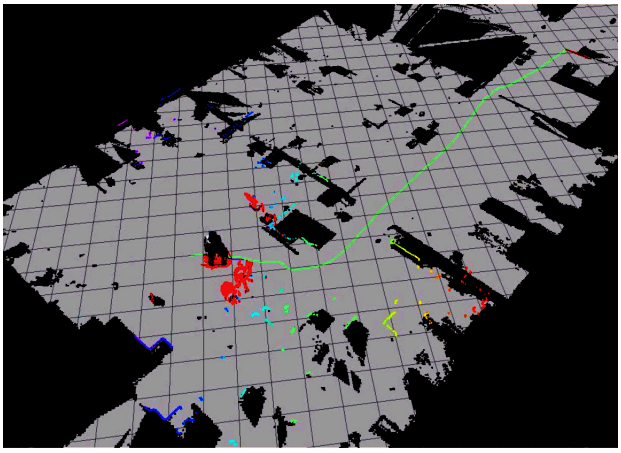


Fig. 2: Visualization of the robot within the 2-D floor map. The robot plans and navigates from an arbitrary start location to the elevator. The thin, green line indicates the planned path the robot intends to follow to reach the elevator.

A. Navigation

The navigation subtask requires planning paths from an arbitrary start location to the elevator, and then from the elevator to an arbitrary goal position. This task specification requires a SLAM algorithm, for efficient mapping and then

[1] https://ubc-ros-pkg.svn.sourceforge.net/svnroot/ubc-ros-pkg

localization in the created map. For the mapping and localization task, we relied on the built-in two-dimensional mapping and localization packages on the PR2. This approach uses the planar LIDAR scanner on the base of the robot to create a 2-D map of the environment. However, a number of issues further complicates the task, as described in the following paragraphs.

a) Map differences: As the two floors in question are not identically arranged, their maps do not precisely align. And since the robot only supports 2-D navigation, it was necessary for it to switch maps after taking the elevator. Finally, re-localization is a prerequisite for navigating in the new map since the robot is initially unaware of its precise location.

b) Presence of Doors: The presence of doors (*i.e.*, doors between rooms, not elevator doors) created a further complication, as the doors could be closed when the robot needs to transit through them to reach a given goal location (*e.g.*, the elevator). In our particular case, the force required to open doors exceeded the load ratings of the robot arms. Thus for the robot to continue on its navigation task (see Fig. 3), it is essential to either leave the doors open, or have a human hold the door open for the robot. Of course, mechanical solutions such as lighter doors or more powerful manipulators can substitute human intervention, but it was not feasible to adopt either of these approaches in the scope of the current work.

A more significant challenge is to move the robot into the elevator (or conversely, out of it) before the elevator door closed automatically. In our case, the elevator is equipped with an IR (infra-red) safety sensor to prevent door closing in case people are transiting in to or out of the elevator, and we aim to use these sensors to keep the door open; however, it is not a flawless approach, and in some cases, the robot fails to navigate into the elevator quickly enough, before the doors closed themselves.

c) Transparent and Reflective Surfaces: A number of walls in the operating environment are made of glass, which causes laser rays to go through, thus providing erroneous input to the mapping subsystem. In addition, the elevator interior and elevator doors are made of polished metal, which cause reflections and multi-path beam traversals for the laser beams, introducing further localization and mapping errors.

B. Wall and Button Detection

Humans effortlessly accomplish the task of taking an elevator because elevators are designed for ease of use by human operators, complete with visual landmarks and easily accessible buttons. Programming a semi-humanoid robot to accomplish this task, however, is surprisingly challenging. The visual task alone is nontrivial. Assuming the robot has localized and successfully navigated to an elevator, it must then detect the presence of elevator control buttons and calculate their three-dimensional coordinates within the 3-D workspace of the robot. Then, assuming the robot is able to press the button, the vision system can be used to verify that the button was successfully pushed by comparing the

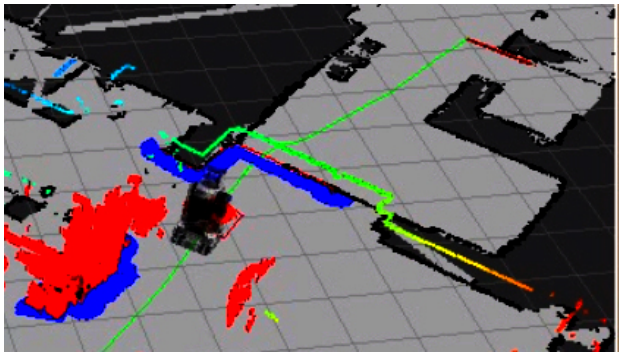
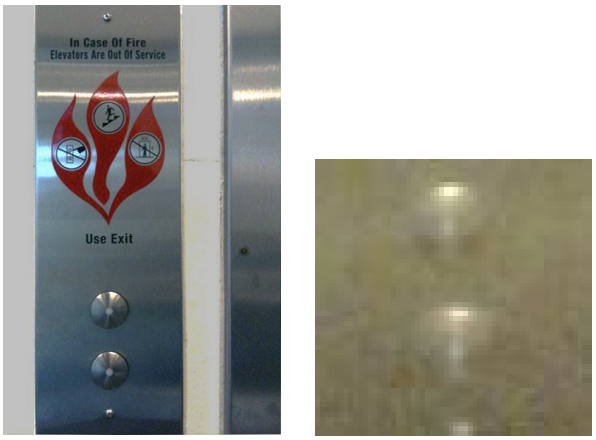


Fig. 3: Demonstrating the "closed-door" scenario. The path is computed to the elevator, but the closed door causes the robot to stop and wait, as there is no alternate path available.

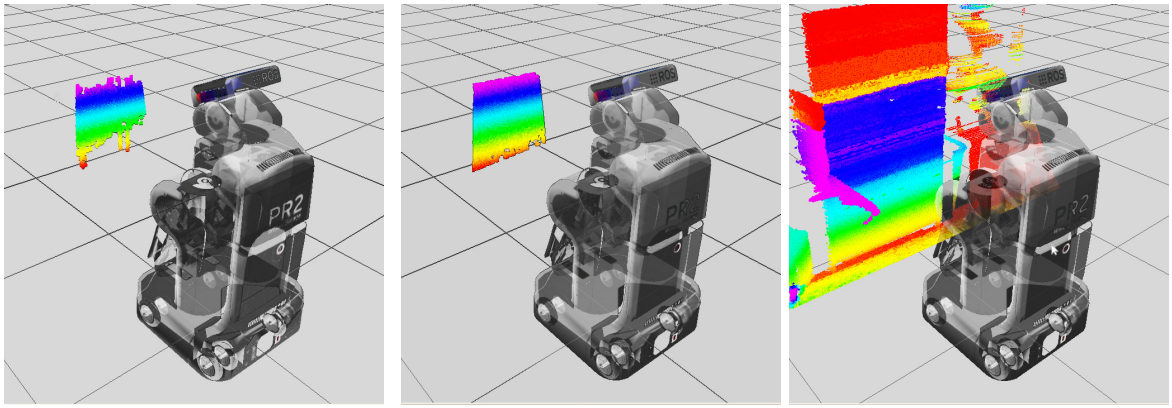


(a) (b)

Fig. 4: Elevator buttons (with wall and elevator door-frame visible) as seen from an external camera (4a) and a close-up view of the buttons via the on-board camera of the robot (4b). Note the lack of distinctive features: the buttons are silver, mounted on a silver backplate.

new appearance of the button to that of the "pushed-state" appearance of the button. Once inside the elevator, the robot must repeat this task but with the button associated with its desired floor. Finally, the robot verifies that it has stopped at the correct floor by identifying the number that appears on the floor number display panel, usually located above the control buttons.

1) Button Detection: One possible approach to detect the location of the elevator button in the camera image plane is to detect the buttons through a feature matching algorithm that is invariant to scale and viewing angle (such as SIFT [14] or SURF [15]). This is an appealing method, since the robot would approach the elevator from arbitrary distances and angles. Upon further investigation, however, template matching [11] was found to be a better-suited technique in vision-based localization for such manipulation tasks. Thus, we implement a template matching algorithm based on the *FastMatchTemplate* [16] technique. A high-level



(a) Wide-baseline stereo vision point cloud.

(b) Textured-light stereo point cloud.

(c) Tilting-laser point-cloud.

Fig. 5: Visualization of the robot perceiving 3-D point-clouds through various sensors. Note that stereo without projected light (5a) gives quite sparse readings compared to stereo under textured-light projection (5b). The tilting-laser scanner (5c) provides a much wider field of view, however the point cloud is spread out further, and has a higher degree of noise.

Algorithm 1 FastMatchTemplate algorithm for template matching. The algorithm accelerates standard template matching by first extracting regions of interest (ROIs) from shrunken images, effectively reducing the search space. *downPyramid* is a function that smooths and subsamples the image to a smaller size; *matchTemplate* is a template matching function built into OpenCV using the normalized correlation coefficients method; *extractROI* extracts a ROI defined as a rectangle; and *maxCorrelation* obtains the highest matching correlation among all matching regions.

```
1:  function FastMatchTemplate(source,target)
2:      sourceCopy ← downPyramid(source)
3:      targetCopy ← downPyramid(target)
4:      ROIList ← matchTemplate(sourceCopy,targetCopy)
5:      result ← φ
6:      for all ROI ∈ ROIList do
7:          searchImg ← extractROI(source, ROI)
8:          result ← result+
9:              matchTemplate(searchImg,target)
10:     end for
11:     return maxCorrelation(result)
12: end function
```

pseudo-code of this algorithm is presented in Algorithm 1. FastMatchTemplate has been shown to be efficient for on-board deployment on robotic platforms, and is also robust to small changes in the visual conditions, such as lighting and viewing angle. As we demonstrate in Sec. IV, the FastMatchTemplate algorithm is sufficiently scale-invariant for our needs as well.

FastMatchTemplate makes use of the *matchTemplate* function built into OpenCV [17]. The matchTemplate function comes with a configuration parameter, allowing the designer to select the matching method employed: (1) sum of squares or (2) cross correlation. In the case of FastMatchTemplate, the most sophisticated method, normalized correlation coefficients is the matching method of choice. Not unexpectedly, the more sophisticated the matching method, the higher the performance requirement. This additional performance penalty, however, has minimal impact on the overall system, partly owing to the preprocessing steps performed by the FastMatchTemplate algorithm.

Template matching is efficient and easy to implement; however, relative to more sophisticated routines, it is not particularly robust to noise or environmental variability. For example, *a priori* knowledge of the object to be detected must be available. If the object were to change in any way, the algorithm would fail to detect it. This dependence on *a priori* templates causes the current scheme to not act as a generalized button detector. Another disadvantage is that the standard template matching algorithm is not particularly scale or rotation invariant. In order to detect the buttons reliably, a predictable distance and angle to the buttons is required. While these limitations are non-issues with the PR2 functioning in the laboratory environment, it is crucial for the robot to be robust to these scenarios if it were to be deployed in the real world.

2) Button Localization: Once the location of the elevator button is discovered in the image, the next task is to obtain its corresponding location in the 3-D workspace of the robot. The first step is to use a pinhole camera model initialized with the intrinsic parameters of the camera that is used in the detection step. Given this, we use the ROS *image_geometry* library to map pixels in an image to three-dimensional rays in the world coordinate system of the camera. From there, we combine this 3-D ray with a depth measurement of the pixels in the image frame in order to determine at which point along the 3-D ray the object of interest is located. For obtaining depth measurements from the camera image plane, it is possible to use the following sensors on board the robot: (1) laser range data, (2) stereo vision data or (3)

textured-light augmented stereo vision data. Figure 5 shows a summary of these three approaches for use in plane detection and subsequent button finding.

The PR2 is equipped with a textured-light projector for enhanced depth extraction through stereo vision. This visible-spectrum textured light projector is attached to the robot head, located just to the left of its stereo camera pairs. Using the projector, the PR2 is able to augment the scene with artificial texture so that its stereo system can detect reliable features in both cameras. One approach to obtain depth information via projected light stereo is to take advantage of the fact that the elevator buttons are generally placed on walls, which are planar surfaces. From a point cloud of stereo data (composed of 3-tuples $< X, Y, Z >$), this approach attempts to extract the dominant plane fitting the maximum number of points. This scheme is also quite robust since an iterative best-match process is used to find the plane containing the greatest number of points in the stereo data. As a result, small levels of noise in the point-cloud have minimal impact on the overall result, equating to a robust, noise-insensitive method. Once the wall plane is obtained, solving for the location of the button on that plane is reduced to a geometric problem: discovering the point of intersection between the wall plane and the projected 3-D ray of the pixel. The solution is depicted in Fig. 6.

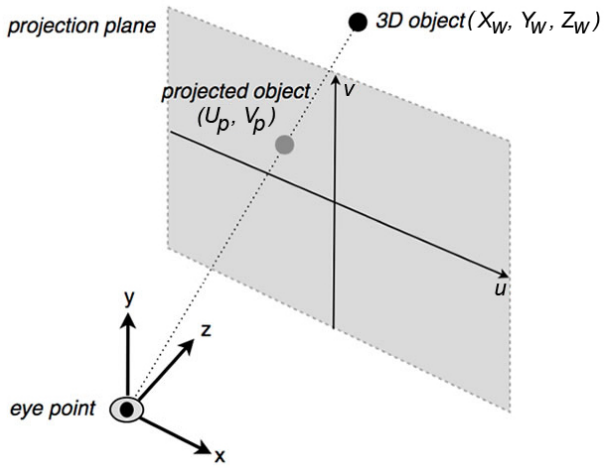


Fig. 6: Button detection through the plane-view ray intersection approach.

C. Pressing Buttons

For pressing the required button, we decompose the problem into the following three steps: *untucking* the arms, relaxing the arms, and pushing the button. In a tucked-arm position, the arms of the robot are tightly contained within the volume bounded by the torso; *i.e.*, no part of the arm extends beyond the minimal 3-D volume containing the robot base and upper body. Untucking the arms is necessary to avoid subsequent self-collision between the arms. The robot arms enter the relaxed position to avoid environmental collisions; since untucked arms extend beyond the footprint of the robot. Relaxing also helps prepare the arms for pushing the elevator button.

For pressing the button, arm joint trajectories are not known in advance; only the Cartesian position of the button is available as input. To solve this problem, we use the inverse kinematics (IK) routine built into the PR2 software core to solve for joint angles given the final Cartesian position of the gripper. Solving for the orientation of the gripper is accomplished by using the vector normal to the wall-plane discovered via the process described in Sec. III-B.2. Once the joint locations are known, the robot is able to achieve arm motion using a joint trajectory action. Finally, the button pushing motion is split into three phases:

1) Move gripper to point slightly in front of the goal (in direction perpendicular to the wall),
2) Move gripper to point slightly behind goal (in direction perpendicular to the wall - this causes button to be pushed) and
3) Return to relaxed position.

Once the button location is known, the IK engine is capable of producing plans to achieve each of these three tasks of the arm motion action.

D. Systems Integration

To integrate the individual components into one coherent autonomous task-oriented behavior, we use the State Machine library (SMach). SMach is a python library used for building hierarchical state machines. Each state performs one or more actions, and returns an outcome indicating success or failure of the actions embodied by that state. For each outcome (of each state), the user specifies which state the state-machine should transition to next. A SMach state can be either (1) a generic state class, (2) a callback function, (3) an "action-state" that calls a predefined, built-in action, (4) a "service state" that calls a ROS service, or (5) a nested state machine. Our approach implements most components as simple action servers so that the state machine can be realized using simple action-states. The simple action state implementation has three standard outcomes - succeeded, preempted, and aborted. The high-level diagram of our state machine can be found in Fig. 7.

The top level state machine is split into the following four nested state machines:

1) `NavigateToElevator`,
2) `PressButton`,
3) `TakeElevator` and
4) `NavigateToLab`.

The purpose of the state machines are self-explanatory, and the transitions between state machines are depicted in Fig. 7. Further information on SMach can be found here [18].

IV. Evaluation

To validate our approach and obtain quantitative performance measurements, we conduct a set of experiments. The overall systems test has been performed live with the PR2,

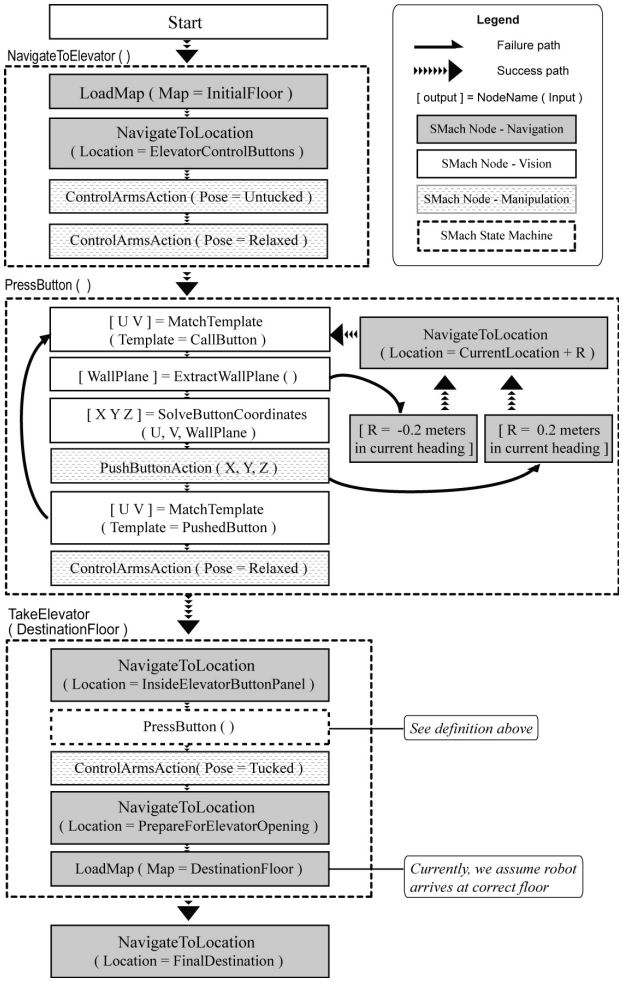


Fig. 7: Outline of the SMach diagram for the complete task sequence.

on the second and basement floors of the ICICS X-Wing building at the University of British Columbia. We isolate and test each component in controlled experiments, both in simulation and in the real operating environment in front of and inside an elevator. The on-board tests provide a wider range of quantitative performance data, particularly for the vision and manipulation subsystems. Additionally, we measure timing data for the subsystems separately, and as a whole during the performance run of the robot.

To measure quantitative performance of the vision system for button detection and manipulation for button pressing, we run a total of 34 independent trials on the real robot. The trials are evenly split between operations inside and outside of the elevator. The distinction is made to highlight the differences in perceptual and manipulation complexities, and also to demonstrate the difficulties posed by the seemingly simple task (from a human perspective) of elevator operation.

A. Time requirements

Figure 8a shows the distribution of required time for the vision system to detect the elevator buttons on the outside of the elevator, once the robot has localized at the correct place, in the correct orientation. On average, it takes 4.6 seconds to detect the button, and 9.7 seconds for the robot to push the button with the gripper.

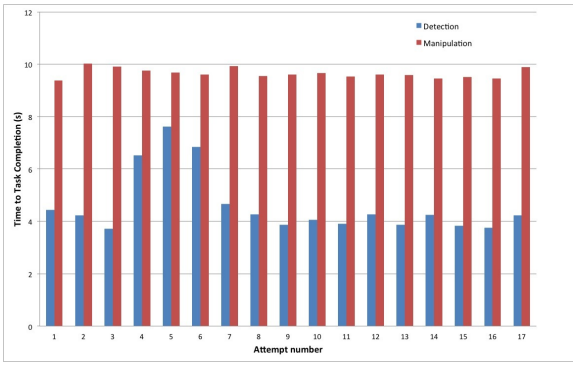
Figure 8b shows the distribution of required time for the vision system to detect the elevator buttons on the inside, once the robot has localized at the correct place and in the correct orientation. On average, it takes 4.3 seconds to detect the button, and 9.9 seconds for the robot to push the button with the gripper.

As can be seen, over a number of trials, the button detection and press routine time requirements are quite consistent for each run. However there are a few exceptions, particularly affecting the vision and localization systems. In Fig. 8a, for example, attempts $4, 5$ and 6 require more time than the other attempts. Upon further investigation, the problem was found to be within the template matching algorithm, which had difficulty to detect the button area under changes of lighting and viewing angle. Once the template matcher successfully returned a position, however, the button-pressing task accomplished its goal without any difficulty.

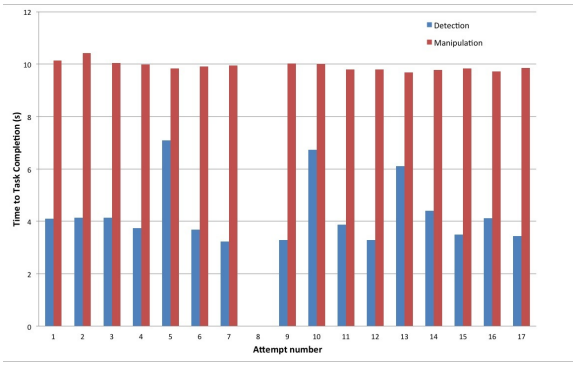
B. Accuracy

In order to evaluate the success rates of the task, we measure the ratio of successful attempts to the number of total attempts taken by the robot. Since our main contribution lies in the vision and manipulation subsystems, we evaluate these in both the outside- and inside-elevator scenarios. For the outside-elevator scenario, we achieve a 85% success rate with both button detection and button press, requiring 20 attempts to successfully perform 17 runs of the task. For inside-elevator scenarios, the vision system performs worse, as only a 50% success rate with button detection is achieved, requiring 34 attempts to find buttons in 17 trials. However, manipulation is near-perfect, as 18 attempts are needed to perform 17 button presses. This reinforces our speculation from the previous paragraph and shows that manipulation does work well once the buttons have been found.

Looking into these results, there are a number of factors affecting the performance of the robot, particularly acute during localization, button detection and button pressing. This is evident during the operation inside the elevator. The reflective surfaces of the elevator interior cause multi-path reflection of the LIDAR beams, resulting in the robot either completely failing to detect walls, or detecting walls in the wrong places. Also, lighting reflected off the walls cause the button appearance to often change drastically at certain viewing angles, which is problematic for the template matching system. In an attempt to quantify these effects, we modify the operating environments for inside-elevator operations in two ways. Firstly, we line letter-sized sheets of paper along the interior walls of the elevator, centered at 30cm above the ground - the height of the LIDAR sensor at the base of the robot, which is used for localization. This

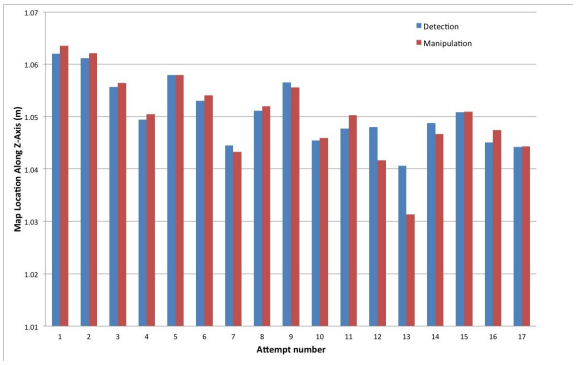


(a) Note slight variability in attempt 4, 5 and 6 due to visual noise in the scene.

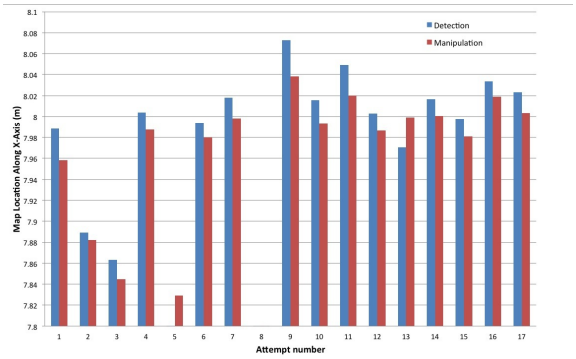


(b) The detector failed in Attempt 8, thus the button-push step is not triggered.

Fig. 8: Timing requirements of the button detector and button press subsystems, both outside (Fig. 8a) and inside (Fig. 8b the elevator. Button detection times are in blue, button pressing times are in red.



(a) Variability in measurements are due to slight errors in robot localization.



(b) Large variability in first half of measurements due to errors in localization. Variability is reduced from attempt 10 onwards due to environmental augmentation, *e*.g. placing paper over reflective surface of the elevator walls,improving localization.

Fig. 9: Accuracy of the button detector and button press subsystems, both outside (Fig. 9a) and inside (Fig. 9b the elevator. Button detection accuracies are in blue, button pressing accuracies are in red.

results in much better localization accuracy compared to the unaltered elevator interior. In turn, the button panel detection performs better inside the altered elevator, with success rates improving from 45% to 63%.

Secondly, the appearance of the buttons on the panel inside the elevator is augmented by attaching a square-shaped piece of paper, and attempts are made to use the template matcher to match that particular appearance. While the average detection time remains unchanged at approximately 4.2 seconds, the variance reduces from 2.7 seconds to 1.1 seconds. This demonstrates more consistent button detection, as the appearance changes due to lighting and shadows were well handled by the template matcher. The resulting improvement in button detection shows a positive rise from 47% to 63%.

V. Discussions

In light of the experimental results and robot validations, it can be said that our attempt towards creating an autonomous, multi-floor navigation system for robots performing delivery tasks has shown considerable success. However, a number of issues can still be improved upon, and the capabilities of the robot can be enhanced in certain aspects as well. We briefly summarize our experiences in this section.

A key lesson learned is that of the value of multimodal sensing. Charlie is equipped with a number of powerful sensors, but each is found to have shortcomings under particular environmental conditions. While this is typical of artificial sensory perception, our efforts highlights the fact that relying on a single sensor, however powerful, may not yield successful results in changeable, real-world conditions. The elevator button detection task demonstrates this principle. Both vision and LIDAR sensing together yield acceptable results, whereas individually, the system could not perform the desired task. Limitations of the LIDAR, for example, to detect glass walls is another example where using laser sensing alone would result in complete task failure.

Our results also demonstrate the benefit of designing an operating environment that is robot-friendly. Brushed aluminum elevator walls may be aesthetically appealing to human users, but pose significant problems to a robot's LIDAR-based navigation system.

For systems integration, the SMach library has been our chosen method for this task. However, SMach is not without limitations, as it can yield complex state transition designs. Also, it may be too complicated a task to embody smaller sub-problems into a SMach system.

There are a number of issues we are unable to address, mostly due to the design and capacity of our platform, some of which have been discussed in Sec. III-A. Currently, for the robot to successfully navigate through doors, we have to ensure that the doors remain open. Once the elevator arrives and the doors open, the time taken by the robot to move into the elevator is sometimes too long, and the elevator door may end up closing before the robot has a chance to enter. In this case, human intervention is still required.

VI. Conclusions

This paper presents the design and implementation results of a robotic system capable of multi-floor autonomous navigation, by operating an elevator to move between different floors. Details of the vision, navigation and manipulation systems, along with experimental evaluation of our system are discussed as well. Overall, the system performs as expected, although owing to issues inherent in the robot design, certain operating criteria need to be maintained.

We see this research as a step towards a robotic home-assistant, capable of a variety of tasks to assist in daily living of humans. This multi-floor navigation system is a prerequisite of a number of semantically complex tasks, such as find-and-fetch or delivery. Longer term goals for this work is to have a rich, robust interface for human-interaction, so that the robot can not only communicate directly with a human user, but also ask directed questions to find alternate methods of task execution or generally reduce ambiguity. Currently, our work is investigating methods of using Charlie as a kitchen assistant, which involves creating powerful object recognition, manipulation and interaction tools. This work is ongoing, and will almost certainly require enhanced methods for multimodal semantic scene understanding.

References

[1] N. Bellotto and H. Hu, "Multisensor-based human detection and tracking for mobile service robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 1, pp. 167 –181, February 2009.

[2] J. Forlizzi, "Service robots in the domestic environment: A study of the roomba vacuum in the home," in *ACM/IEEE International Conference on Human Robot Interaction*, 2006, pp. 258–265.

[3] N. Roy, G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Margaritis, M. Montemerlo, J. Pineau, J. Schulte, and S. Thrun, "Towards personal service robots for the elderly," in *Workshop on Interactive Robots and Entertainment (WIRE 2000)*, Pittsburgh, PA, May 2000.

[4] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009. [Online]. Available: http://pub1.willowgarage.com/ konolige/cs225B/docs/quigley-icra2009-ros.pdf

[5] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty." *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[6] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots." *Autonomous Robots*, vol. 5, pp. 253–271, 1998.

[7] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, February 2007.

[8] K. Lynch, "The mechanics of fine manipulation by pushing," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, May 1992, pp. 2269–2276.

[9] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008. [Online]. Available: http://ijr.sagepub.com/content/27/2/157.abstract

[10] K. Wyrobek, E. Berger, H. Van der Loos, and J. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in *IEEE International Conference on Robotics and Automation (ICRA2008)*, May 2008, pp. 2165–2170.

[11] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz, "Laser-based Perception for Door and Handle Identification," in *International Conference on Advanced Robotics (ICAR2009)*, Munich, Germany, June 22-26 2009, best paper award. [Online]. Available: http://files.rbrusu.com/publications/Rusu09ICAR.pdf

[12] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *IEEE International Conference on Robotics and Automation (ICRA2012)*, St. Paul, MN, USA, May 2012.

[13] L. Kunze, M. Beetz, M. Saito, H. Azuma, K. Okada, and M. Inaba, "Searching objects in large-scale indoor environments: A decision-theoretic approach," in *IEEE International Conference on Robotics and Automation (ICRA2012)*, May 2012, pp. 4385–4390.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints." *International Journal of Computer Vision(IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[15] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *European Conference on Computer Vision ECCV 2006*, pp. 404–417, 2006.

[16] T. Georgiou, "Fast Match Template." [Online]. Available: http://opencv.willowgarage.com/wiki/FastMatchTemplate

[17] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[18] "SMach – State Machines Library." [Online]. Available: http://www.ros.org/wiki/smach