Mail-bot

Hephaestus' Creation

Yanling He, Dennis Huang, Skyler Peterson, Sam Wilson, Weiming Zhang

1. INTRODUCTION

Our team used the PR2 by Willow Garage along with the ROS software package to create a completely automated mail delivery robot. This robot will scan for and pick up mail at a pre-determined post-office. Outgoing mail is then read and taken to it's desired mailbox. The robot will then drop the mail off and go back to the post office to get more mail.

Mail delivery is simple, necessary, and tedious. Training a robot to do this can free up human workers to do other tasks which makes our lives easier. Some companies may be able to save money by cutting back on employment of menial labor and re-distribute these savings into hiring workers related to the job the company specializes in.

Some of the challenges that arise include manipulation, perception, navigation, infrastructure, and HCI for controlling the robot. The robot must be able to pick up and distribute envelopes using it's grippers which have a limited dexterity. The robot must be able to perceive the mail it is delivering and the world around it in order to stay safe. The robot must navigate it's surroundings and plan efficient routes. In order to do these tasks props and objects must be constructed so the robot can easily interact with the world. Finally, a GUI must be simple enough to read so non-engineers can run it.

2. SYSTEM DESCRIPTION

Figure 1 shows the general finite state machine of the automated mail delivery robot. When the user starts the robot state machine on the GUI, figure 4, the robot will enter an unknown positions state. Here the robot will unfold and fold it's arms and navigate itself to the post office. It is important that the user checked to make sure the robot is properly mapping it's surroundings, before doing this. The "Go to Post Office State" is entered at this time, while the robot navigates to the post office. Next the robot will wait at the post office and enter the "Look for AR Code" state, where it will wait at the post office looking for AR codes continuously. When an AR code is found, the robot reads it to make sure it has a valid address. If the address is valid the robot will move to "Pick Up Envelope" state. This state uses the AR positions and IK solutions to pick up the envelope and tuck it away along with it's arms. Next the robot moves into the "Go to Desired Mailbox" state and navigates to the mailbox indicated by the envelopes AR marker. When the robot is there, it will again enter the "Look for AR Code" state; however, this time the robot will look for the AR marker on the mailbox. Once this is found the robot moves to the "Drop Off Envelope" state and uses IK solutions again to drop off the mail into the box. Finally, the robot goes back to the "Go to Post Office" state and continues this cycle.

In the following paragraphs we describe the details of the robots primary tasks. Our general architecture runs all GUI elements on the local machine, but uses messages and services to run all of the robot movements directly onto the PR2 computer. This creates a much smoother and more reliable experience as well as the robot's movements.

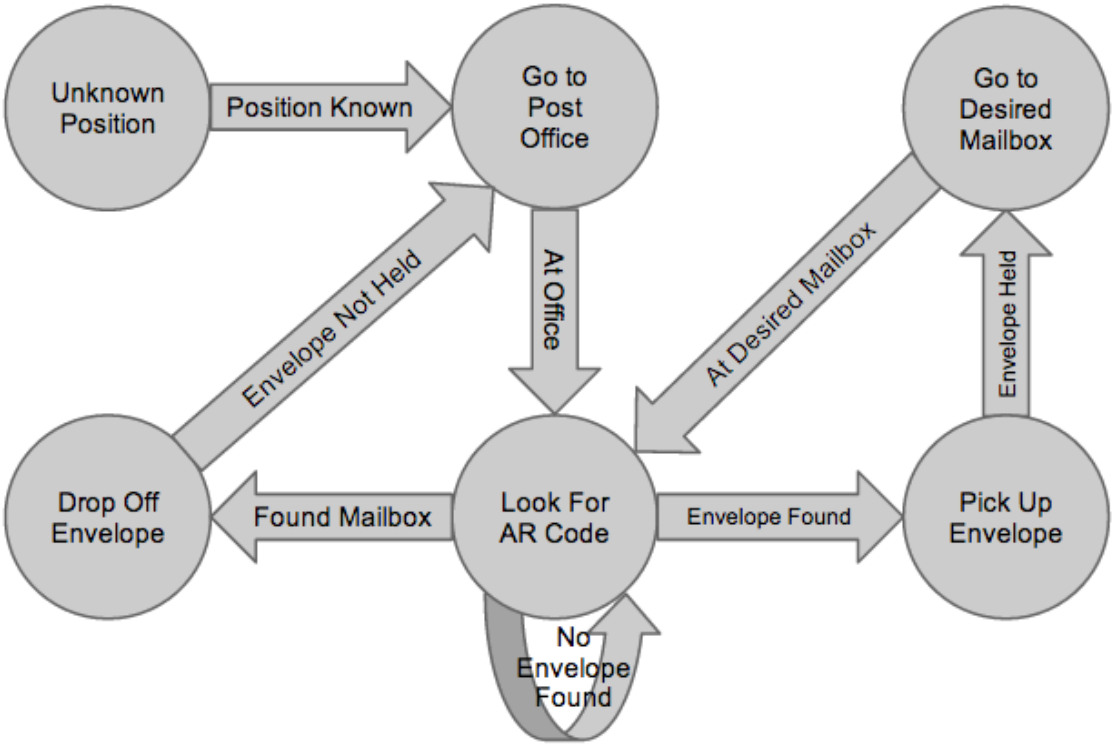

Figure 1: This is the finite state machine (FSM) for the automated mode of the mail delivery robot. Pressing start on the mail-bot GUI (Figure 4) will enter the ”unknown Position” state of this FSM.

a. Manipulation

During the “Pick Up Envelope” and “Drop Off Envelope” states, the arms and grippers are used to interact with the envelopes. To do this we use positional information from the AR markers to determine an inverse kinematic solution and then drive the arms to various offsets until the desired task is accomplished. In order to find IK solutions the ROS packages “pr2_[l,r]_arm_kinematics_simple/get_ik_solver_info” and “pr2_[l,r]_arm_kinematics_simple/get_ik” are used. These packages take 3D defined pose positions of a valid robot end effector (Wrist, gripper) and returns a series of joint angles which may be used by the ROS joint trajectory package “/[l,r]_arm_motion_request/joint_trajectory_action.” The joint trajectory package is used to directly move the arms to desired joint positions. This is used when actually moving the arms to their desired IK positions and folding the arms. The grippers are moved using the ROS gripper action, “/[l,r]_gripper_controller/gripper_action.”

b. Perception

The robot must be able to perceive the world around it in order to navigate and to know the locations of the envelopes/mailboxes. In order to navigate, the PR2 uses laser scanners to search for obstacles to avoid. This is done during it’s route planning and is automatically taken care of in the pr2_2dnav package. For tracking AR markers in the “Look For AR Code” state, we use the ROS package ar_pose_marker. This package uses the kinect camera to find AR markers and get their positions in three space. We can then add offsets and transformations by hand and send these positions to the Inverse Kinematic solution generator to get the joint angles as described in manipulation. To generate these markers we used ROS’s “ar_track_alvar” createMarker software. We then use IDs to encode the mailbox locations so that the robot knows which mailbox the envelope belongs to.

c. Navigation

Mail-bot uses navigation for picking the mails from post office and delivering to the corresponding mailbox based on the marker on the envelop. Based on our state machine the mail-bot starts at a random place on the map and then navigate to post office to pick up the mail. After it picking up the mail it navigate to the mailbox.

To use the navigation we first supply the system a map. To make the map the mail-bot need to go around the environment and collecting data using the sensors to know where is the path and where is the blocking places. We are using the data bag called basement_laserdata.bag which includes the data of CSE 014. Then using gmapping package to create map. Aftering collecting data and building the mail-bot runs the map server and localize the robot on the static map. For navigation we use pr2_navigation_global and pr2_2dnav packages. We set up 4 default locations which are post office label as a pink star and 3 mailboxes label as 51, 52 and 53. The locations are shown below in Figure 2.

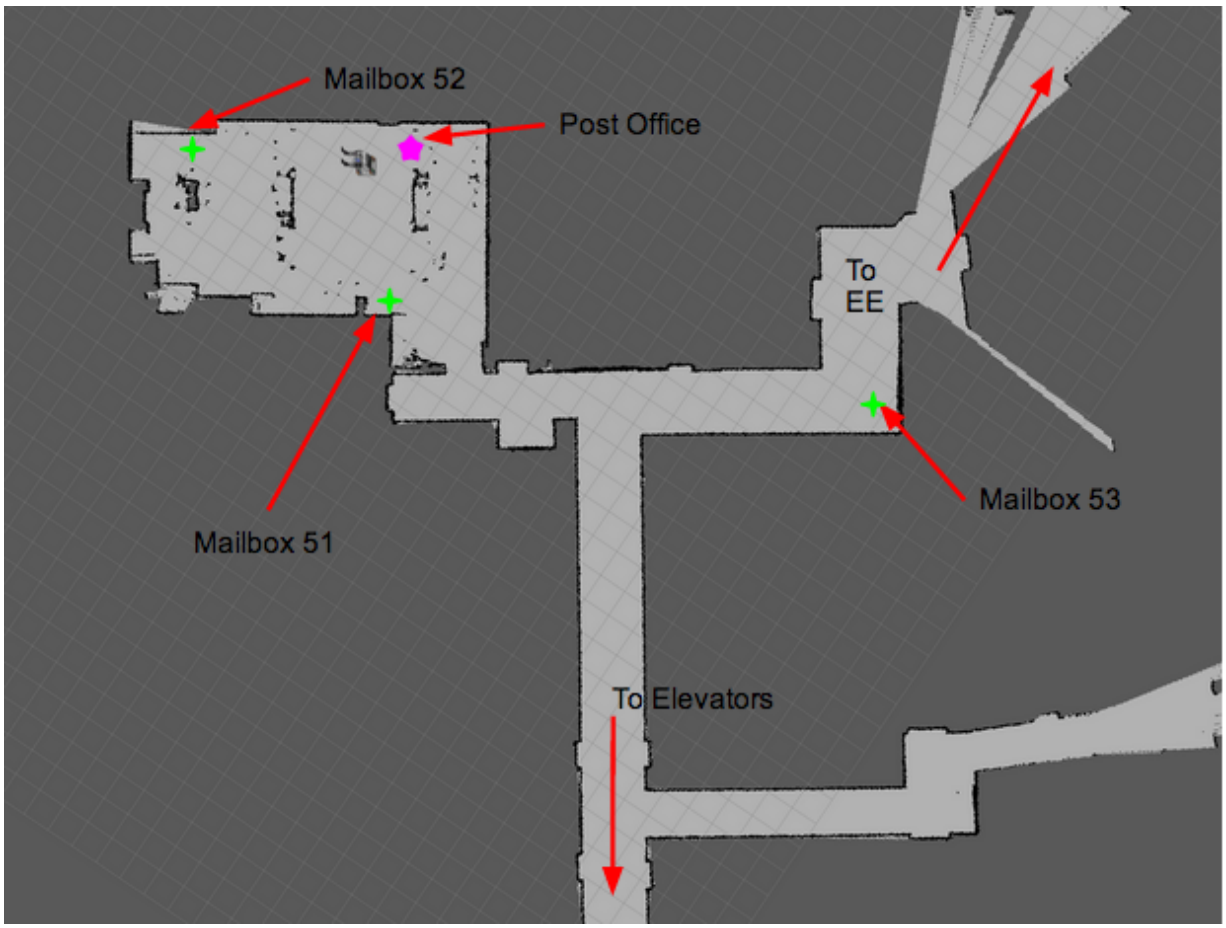

Figure 2: This is a screenshot of the map the robot uses to deliver mail to the bottom floor of the University of Washington's CSE building. The green stars represent mailboxes and the pink star is the post office in this configuration.

The robot will navigate to post office first and picking up a mail with a AR code on it, for example it picks up mail with label 51. It detects 51 and sets the mailbox with label 51 as the navigation goal, so it starts to navigate to the mailbox with label 51. After it delivered the mail, it set its new navigation goal as post office then navigate back. During the navigation, the mail-bot does the collision detection and dynamically changing the path based on the situation is faced and the map loaded.

d. Props

In order for the robot to efficiently work in it's environment using limited dexterity, we created props by hand that allow the robot to more easily do it's job. The post office, figure 3 left, is made from Gorilla tape and epoxy. The shelves are placed 3 inches apart and allow the envelopes to stick out roughly 3 inches as well. This makes it much easier for the robot to grip the envelopes using the two pronged grippers. The envelopes. figure 3 left, are standard post office page sized envelopes. They contain AR marker address stamps so the robot can easily read the address. In addition, they have paper fold in blockers to prevent the robot from seeing the wrong AR markers. The robot will grab envelopes from the top down. Finally, the mailboxes, figure 3 right, are made from cut-up cardboard boxes with AR markers to allow the robot to verify it's location and get it's arm into the box to drop the envelope off. The envelope is dropped in upside down to eliminate AR noise for tracking the box after multiple envelopes are dropped in.

Figure 3: Left is a picture of the post office and the envelopes. Right is a picture of the mailboxes.

e. User interaction

We are using PyQt for GUI design to create widgets and buttons for us to control the robot. When each button clicked the system will call the corresponding function. Our GUI has 9 parts which are labelled in figure 4 shown below:

Part 1 controls head movements with the four buttons and speech function with the central button. Input for the speech function can be set in Part 8.

Part 2 controls the arm movements each button control one joint point of the robot.

Part 3 controls the base movements.

Part 6 controls the in tensities including gripper intensity, motion intensity and arm intensity. Intensity increases the distance of the movement for each click increases.

Part 7 shows the state of the robot to let user know the current state of the robot.
Part 8 is an input window that user can type the sentence that want the robot to say.
Part 9 is used for creating a sequence of arm motions that users can save different poses to make a sequence of motion.
Part 5 is mainly used for the mail bot. It includes 4 states of the state machine and two motions of folding and unfolding arm. So user can manually control the robot to picking or delivering the mail by clicking those buttons. There is a start button in part 5 which is a automation running cutton of the mail bot. When the user click the “start” button the robot will start at the first state and run the state machine without terminate.

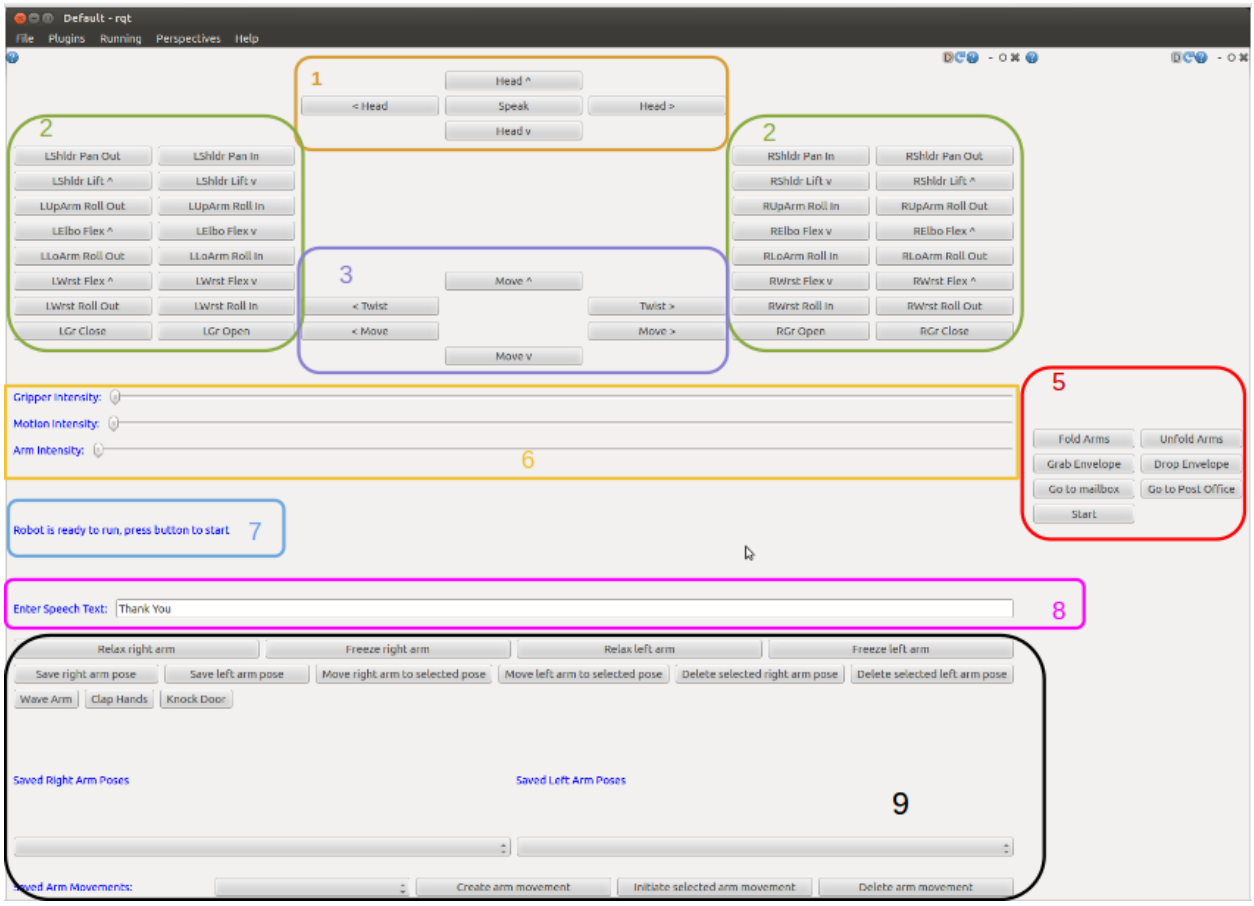

Figure 4: This is the GUI for the mail delivery robot. The left set of buttons and sliders, simple GUI represent total control of the robots motors and can be used to do fine adjustments to the robot. The right (orange circle 5), mail-bot GUI,contains 7 buttons including 6 buttons to manually drive the state machine along with a start button that automatically drives the state machine.

3. USER MANUAL

a. Installation

1. Download the final_project.tar.bz2 from this website:
https://sites.google.com/site/cse481a/projects/team-5
2. Unzip the file and put it inside /home/<user_name>/catkin_ws/src/
3. Open a terminal
4. Type [cd /home/<user_name>/catkin_ws/]
5. Type [catkin_make]

6. Place a copy of final_project.tar.bz2 on the robot in the ~/catkin_ws/src/ directory. (Use SFTP or SCP)
7. Type [cd ~/catkin_ws/] on the robot.
8. Type [catkin_make]

b. Running

1. Open a terminal
2. Type [ssh c1]
3. Enter the password
4. Type [robot claim] then type [y]es if asked
5. Type [robot start]
6. Type [roslaunch mailbot robot.launch]
7. Open a new terminal
8. Type [realrobot]
9. Type [roslaunch mailbot desktop.launch]
10. The GUI will pops up and you are ready to go!

4. EVALUATION PROPOSAL

a. Reliability

The ability of the robot to work in dynamic environments should be possible as long as the basic interface of the mailbot is implemented. This means as long as there is a singular place to put mail and there are places for the mail to be delivered, all encoded in proper AR tags, the robot should work.

In terms of navigation, we would test the ability of the robot to handle static objects (trash placed in a path taken previously) and dynamic objects (people moving in and out of the robot's sensors). Questions we would need to answer include does the robot recalculate paths which allow the robot to move to it's destination? If the robot was not able to calculate it's path, how does it recover in a way that the robot may continue to traverse the state machine. Can the robot deal with environments which involve doors, elevators, or multi floored destinations?

In terms of object manipulation, we would evaluate the robots handling of the envelope: from grabbing the envelope to delivering it into a mailbox. We would consider properties such as the physics of the envelope including; height, width, thickness, weight, shape, smoothness, envelope material. We would also test how well the robot can deal with envelope's of different combinations of these parameters. For mailboxes' we would check different combinations of height, width and opening space, and marker offset's. We would evaluate if the robot can reliably grab envelopes from the post office and reliably stick the envelopes into a mailbox. If any of these expected actions fail, how well does the robot recover (e.g if robot fails to pick up envelope, the robot should not go to mailbox).

In terms of perception, we would test the reliability of our AR code usage on envelopes and mailboxes. We would examine if the robot could detect AR codes and grab correct information at different offsets on the envelopes and mailboxes. If the robot fails to locate a mailbox, does the robot attempt to search for the mailbox and how does it recover if the mailbox is not found. If an envelope is not found, does the robot correctly stay in a waiting state. In general, we would see what sizes of AR codes the kinect can detect and how does the robot deal with identifying and ignoring AR codes which are not part of the mailbot process and taking action on AR codes which do.

In terms of security, we would examine the effectiveness of our security (not implemented) against different attempts of manipulations of props or brute force attempts to grab envelopes enroute from different directions, orientations, and at different transition states the robot goes through. When handing envelope's to a person, does the mailbot only hand the envelope to the correct person or ensure there are no bugs in

the process to identify the correct person.

For the system as a whole, we would evaluate reliability based on stressing the variables in the system. We essentially test to ensure the robot does not break in the state machine as mailbot objects are changed. For example, if the robot still is able to correctly grab an envelope when the post office is full of envelopes.

b. Effectiveness

We would test the performance of the robot on transitioning from each state in the state machine. We would measure performance by time of success or time of failure recovery and accuracy in transitioning between states. In general we would measure algorithm efficiency based on time. For example, we would examine the function of time taken for mailbot to identify envelope's destination and calculate a route. We would examine throughput (envelopes delivered per hour) and latencies with respect to destination distance.

c. Usability

In our task, usability is made to be as minor as possible. We want the robot to be able to do the task and not have too much human interaction. This isolates usability to being able to start up the robot and press the start button which begins the state machine. We could test this by giving non-engineers a copy of the user manual and see if they can effectively and efficiently start up the state machine.

In future work, we would like the robot to be able to hand the envelope to a person if they happen to be at the mailbox. If we were to implement this, we could test it by seeing how well the robot interacts with people at random.

5. LIMITATIONS AND FUTURE WORK

Reliability

Since the robot is running on battery, the main goal for improving reliability would be increase uptime of the robot. Currently the robot can run for about 40 minutes after each recharge. This is a relative low amount of time for the mail delivery tasks. We should increase the uptime by using a battery of larger capacity or setting up charging locations in the building where the robot can be recharged when at rest.

Efficiency

With one pick and delivery each time, the efficiency is the major limitation of the mailbot. Also, AR markers are required on the envelops and mailboxes in order to perform the success delivery. This requires a setup before the mail delivery. In the future, we should increase the number of mails delivered each time which is likely to be done by having the robot carrying a mail bag and deliver multiple mails from after each pickup. Other than that, we need to have an alternative way of finding destinations of delivery and identify mailboxes instead of using the AR markers. We could be reading the names/addresses from the envelops to locate the mailboxes.

Another improvement we can make to the robot is speeding up the robot's movement and enabling the capability in climbing up stairs, using elevators, etc. This will make the robot more robust in handling delivery tasks inside buildings. However, adding these features might require a big modification in the motion system of the robot.

Security

The robot would be frequently used in deliver high confidential mails or packages within the building. Thus, the security features should not be neglected. Currently the robot will hold the envelops in hands while travelling in the building. We could put the mails in a safer mail bag which mentioned in "Efficiency" part.

When handing the mails or documents to human, we need face recognition to ensure we give to the correct person. Besides that, the robot should also be able to trigger an alarm when being compromised by malicious parties.