HUMAN SUPERVISsed SEMI-AUTONOMOUS APPROACH FOR THE DARPA ROBOTICS CHALLENGE DOOR TASK

by

Nandan Banerjee

A Thesis Submitted to the Faculty of the WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the Degree of Master of Science in Robotics Engineering by

MAY 2015

APPROVED:

Professor Taskin Padir

Professor Michael Gennert

Professor Gregory Fischer

4/29/2015
Abstract

As the field of autonomous robots continue to advance, there is still a tremendous benefit to research human-supervised robot systems for fielding them in practical applications. The DRC inspired by the Fukushima nuclear power plant disaster has been a major research and development program for the past three years, to advance the field of human supervised control of robots for responding to natural and man-made disasters. The overall goal of the research presented in this thesis is to realise a new approach for semi-autonomous control of the Atlas humanoid robot under discrete commands from the human operator. A combination of autonomous and semi-autonomous perception and manipulation techniques to accomplish the task of detecting, opening and walking through a door are presented. The methods are validated in various different scenarios relevant to DRC door task.
Acknowledgements

The work presented here has been sponsored by the Defense Advanced Research Project Agency, DARPA Robotics Challenge Program under Contract No. HR0011-14-C-0011. I would like to thank Professor Taşkin Padır, Professor Michael Gennert and Professor Gregory Fischer for giving me the opportunity to work on the DARPA Robotics Challenge and for their invaluable support throughout the duration of this work. I would also like to thank all the members of the WPI-CMU DRC team - Mathew DeDonato, Felipe Polido, Xianchao Long, Benzun Babu, Siyuan Feng, Ben X, Lening Li, Peng He, Xiongyi Cui, Ruixiang Du, Kevin Knoedler, Perry Franklin, Christopher Bove, Aaron Jaeger and Josh Graff for their enormous help and support with the door task and for helping me make it a success. I would also like to thank my parents, Dr. Paresh Banerjee and Dr. Nandini Bhattacharrya, my little sister, Nandita Banerjee, and other treasured members of my family for their motivation and support.
Contents

List of Figures vi
List of Tables viii

1 Introduction 1
  1.1 The DARPA Robotics Challenge 1
  1.2 Boston Dynamics' Atlas 3
  1.3 DRC Tasks 3
  1.4 Overview 4
  1.5 Contributions 5

2 Atlas Perception Capabilities 6
  2.1 Handling Perception Data 7
  2.2 Model based detection 8
  2.3 Visual Servoing 9
    2.3.1 Fiducial based servoing 9
    2.3.2 Model-fitting based servoing 11
    2.3.3 Global contrast based eye in hand servoing - A study 12
    2.3.4 Time of Flight camera servoing 12
  2.4 Door detection 13
    2.4.1 Autonomous detection 13
    2.4.2 Manual detection 18

3 Atlas Manipulation Capabilities 20
  3.1 Trajectory optimizer 22
  3.2 Step Planner 23
    3.2.1 Constrained Step Planner 24

4 Door Task 27
  4.1 Event driven FSM 28
  4.2 Door detection 30
  4.3 Walking to the door 31
  4.4 Opening the door 33
  4.5 Walking through the door 34
5 Experiments and Results

5.1 Visual Servoing results ........................................... 35
5.2 Door detection results ........................................... 36
5.3 Door task results ........................................... 37

6 Discussion .................................................. 38

Bibliography .................................................. 40
List of Figures

1.1 (a) The upgraded Atlas Humanoid robot built by Boston Dynamics. (b) Atlas opening door on practice day of DRC Trials. ........................................... 2
1.2 The driving and terrain tasks from the DRC Trials. ........................................... 3
1.3 The valve and debris tasks from the DRC Trials. ........................................... 4

2.1 Multisense - Carnegie Robotics. ................................................................. 6
2.2 Assembled Point Cloud data from the Multisense LIDAR. .......................... 7
2.3 Registration (fitting) of (a) door and (b) door frame to a LIDAR point cloud using the proposed model based detection algorithm. ......................... 8
2.4 Robotiq hand with fiducials (square elements in pink). .............................. 10
2.5 Fiducial detection on a side of the Robotiq hand. ........................................ 10
2.6 Model fitting of the Robotiq hand. .............................................................. 11
2.7 Visual servoing to the valve center. Red line shows the normal to the valve. 12
2.8 Door detection algorithm flow diagram. ....................................................... 14
2.9 Edge image after applying the Canny edge filter. ....................................... 15
2.10 Vertical lines after applying Hough transform and filtering non-vertical lines. 16
2.11 Multiple door candidates. ........................................................................ 16
2.12 Detected door. ......................................................................................... 17
2.13 Detected handle. ....................................................................................... 17
2.14 Door detection algorithm flow visualization. ............................................. 18
2.15 Door scribble (green color) in the GUI near the cursor. ............................ 19
2.16 Door normal obtained in the laser cloud from the scribble seed. ............... 19

3.1 Atlas opening a door. .................................................................................. 20
3.2 Atlas Inverse Kinematics. ........................................................................... 21
3.3 Planning using TrajOpt. ............................................................................. 23
3.4 Constrained manipulation planning - Before stepping through the door. .. 25
3.5 Constrained manipulation planning - Stepping through the door. .............. 25
3.6 Constrained manipulation planning - After having stepped through the door 25

4.1 Door at the test event in South Carolina .................................................. 27
4.2 Strategy for the door task (pull door). ......................................................... 28
4.3 FSM of the door task approach. ............................................................... 29
List of Tables

5.1 Analysis of Approaches for DRC Trials and Finals . . . . . . . . . . . . . . 37
Chapter 1

Introduction

After the Fukushima Daiichi nuclear power plant disaster in 2011, disaster robots came into the limelight. Nuclear leaks and nuclear radiation contamination made it extremely hazardous and difficult for human beings to do damage control. Instead robots were sent into the disaster site for to perform repair work and decontamination of the plant environment. Most of the robots suffered from mobility challenges over the post disaster terrain and also dexterity challenges to perform certain tasks. All these factors led to the conception of a global competition to pool the cutting edge robotics technology and engineer the best disaster response robot. This competition, the DARPA Robotics Challenge incorporates various tasks that are relevant to a disaster scenario that needs to be completed within a short time frame. This chapter will introduce the DARPA Robotics Challenge, the Atlas robot (see Fig. 1.1) sported by the WPI-CMU team in the DRC, the various tasks, and give an overview of the work presented here.

1.1 The DARPA Robotics Challenge

The DARPA Robotics Challenge (DRC) is aimed at advancing robotics technologies to assist humans in responding to natural and man-made disasters [7]. The DRC was conceived in the wake of the Fukushima nuclear disaster. Unsuitable environments for human beings coupled with the fact that most robots have mobility challenges for extremely challenging environments have been the primary driving factors behind promoting humanoids in this
Figure 1.1: (a) The upgraded Atlas Humanoid robot built by Boston Dynamics. (b) Atlas opening door on practice day of DRC Trials.

challenge. The DRC has attracted teams from all over the world since the Virtual Robotics Challenge (VRC), followed by the DRC Trials and will finally end with the DRC Finals. The VRC occurred in June 2013 and tested software teams ability to effectively guide a simulated robot through three sample tasks in a virtual environment. WPI ranked 2\textsuperscript{nd} in the VRC as a Track C team, and received an Atlas Humanoid robot built by Boston Dynamics to participate in the DRC Trials in collaboration with researchers from CMU. The DRC Trials occurred December 20-21, 2013 at the Homestead-Miami Speedway, where teams guided their robots through eight individual, physical tasks that tested mobility, manipulation, dexterity, perception, and operator control mechanisms. WPI ranked 7\textsuperscript{th} in the DRC trials and won a grant of 1.5 million $ to develop the Atlas’ capabilities for the DRC finals. The DRC finals will occur June 5-6, 2015 at Fairplex in Pomona, California. The event will require robots to attempt a circuit of consecutive physical tasks, with degraded communications between the robots and their operators. The WPI-CMU team is working towards meeting the requirements of the DRC Finals which will be held in June, 2015 (see Qualification Guidelines\textsuperscript{1}).

\textsuperscript{1}http://www.theroboticschallenge.org/team-resources
1.2 Boston Dynamics’ Atlas

Atlas is a bipedal humanoid robot developed by Boston Dynamics with funding from the DARPA. The Atlas is 6 feet tall, weighs roughly around 150 kg and has 28 hydraulically actuated degrees of freedom (DOF): 6 in each arm, 6 in each leg, 3 at the torso, and 1 in the neck. It has load cells for force sensing at the hands and feet and a fiber-optic inertial measurement unit (IMU) at the pelvis for estimating the robot pose. Each actuator on the arms has a linear potentiometer for position measurement and two pressure sensors to determine the joint forces based on differential pressure measurements. The end effectors on the robot are two three fingered Robotiq hands by Qinetiq. It is equipped with two vision systems - a laser rangefinder (LIDAR) and stereo cameras. It also includes two cameras positioned around the robot to allow for a side view of the robot’s environment. Three on board computers provide the computational power for the controllers, perception and other algorithms running on board. A battery pack in the back provides about one hour of uninterrupted robot life. Wireless communication connects the operator control unit to the robot’s on board computers.

1.3 DRC Tasks

Figure 1.2: The driving and terrain tasks from the DRC Trials.
The DRC presents various challenges based off of possible scenarios at a disaster site. Driving a car (Fig. 1.2(a)) along the course of a road avoiding traffic cones is the first task in the challenge. Then comes the Egress task, where in the robot has to get down from the car on to the ground. Third is the Door task, where the robot has to open and walk through a standard push/pull door. The door task is very important as the robot cannot decide to skip this task and without completing this task, the next tasks cannot be performed. Right after crossing the door comes the Debris task where the robot needs to navigate itself through a pile of debris (Fig. 1.3(b)) by removing some and avoiding the rest. The robot then needs to pick up a drill, cut a big hole in a board, stick its hand in and rotate a valve (Fig. 1.3(a)) (Drill and Valve tasks). Then comes the Terrain task, where the robot has to walk over very uneven terrain (Fig. 1.2(b)). And finally, the Climbing/Ladder Task where the robot will climb a flight of stairs. All of these tasks except the driving and egress tasks have to be combined together within a strict time limit of 1 hour (see DRC Rules).

![Figure 1.3: The valve and debris tasks from the DRC Trials.](image)

1.4 Overview

This thesis describes the perception and manipulation capabilities of the Atlas robot with an inclination towards the door task. Chapter 2 contains perception capabilities of the Atlas robot relevant to the door task along with various algorithms that have been

\[\text{http://www.theroboticschallenge.org/team-resources}\]
implemented and tested on the robot. Chapter 3 discusses the manipulation capabilities of
the Atlas relevant to the door task, in particular the trajectory optimizer and how costs
and constraints are set to achieve desired results. Chapter 4 describes the entire DRC
door task architecture, the state machine design, the coupling between manipulation and
perception, the door manipulation costs and constraints, walking, opening of the door,
and door detection and tracking. Chapter 5 presents the results from the capabilities and
algorithms described in Chapters 2, 3, and 4. Chapter 6 concludes with a discussion on the
efficacy and the reliability of the presented algorithms relevant to the door task.

1.5 Contributions

The following points highlight the main contributions to the WPI-CMU DARPA Robotics
Challenge team and towards the successful completion of the DRC door task -

- Design and implementation of vision based and cloud based visual servoing algorithms
  (Perception)
- Design and implementation of a door and door handle detection algorithm (Percep-
tion)
- Implementation of a model based detection algorithm (Perception)
- Design and implementation of a constrained step planner for the Atlas (Manipulation)
- Design of the door task state machine architecture and implementation of a finite
  state machine (FSM) (Software)
- Planning and execution of the door task strategy in simulation and on the real robot
  (Task Lead)
Chapter 2

Atlas Perception Capabilities

Perception, in the context of the Atlas is the ability of the robot to perceive its surroundings. This is predominantly achieved through visual and laser data from the Carnegie Robotics’ Multisense Head (see Fig. 2.1). The Multisense has a spinning LIDAR, stereo cameras, and LED arrays. The Multisense is mounted on the neck joint for its swift up and down movement. In the ideal task case - the robot looks at a scene, recognizes and segments out task specific objects, and then performs manipulation tasks over those objects. This chapter discusses the Perception capabilities of the Atlas robot pertaining to visual servoing and door detection.

Figure 2.1: Multisense - Carnegie Robotics.
2.1 Handling Perception Data

Although the robot perceives through the force/torque sensor on the wrists and IMUs on the Atlas, the visual and LIDAR data is what is meant by perception over here. The Multisense head is the major source of perception data. It sends out images, disparity information, and laser scans to the onboard computers. A lot of time has been spent in the development of an efficient and reliable framework which would mediate data to the software running on the robot computers from the Multisense without adding communication or compatibility overheads. This framework, “perception_common” takes in all the raw data whenever requested by a high level program (user), does necessary modifications (like laser scan assembly (see Fig. 2.2), unifying cloud, time synchronization) and sends them out to the user. Whenever firmware and/or driver changes are made, only perception_common is changed, thereby effectively abstracting the user from the underlying communication and raw data scenario.

Figure 2.2: Assembled Point Cloud data from the Multisense LIDAR.
2.2 Model based detection

In almost every robotic task, pose of an object is more important than just detecting it in a scene. Model based detection or tracking is a technique where an object model is fit to a scene containing that object to determine its pose in the scene. In this work, a model is a mesh model and the scene is embedded in a laser or stereo point cloud. In 1992, [3] was a seminal paper describing the Iterative Closest Point (ICP) algorithm to register 3-D shapes to a scene composed of points. ICP essentially iterates over the point cloud transforming the source cloud at every iteration trying to reduce the mean squared Euclidean error between the points. Lots of ICP variants [19] exist. [16] explored Normal Distribution Transform (NDT) for point cloud registration which provides better results while mapping mines by autonomous robots.

Figure 2.3: Registration (fitting) of (a) door and (b) door frame to a LIDAR point cloud using the proposed model based detection algorithm.

Model based detection has been done using the ICP algorithm. The input to the model fitting ICP algorithm is a mesh model of the object, a target cloud of the scene, and an initial guess of where the object is in the scene. The algorithm then spits out a transformation which when applied to the source cloud will give the object in the scene. ICP works best when given a proper seed, i.e. the source cloud should be fairly close to the target cloud when ICP starts fitting them.

The mesh model of the object is first sampled using random sampling to give a point cloud of the entire object. Other sampling methods like Grid, Poisson Disc, Jitter and
Rotated Grid can also be used. Now, based on the initial guess of where the object is in the scene, rays are shot out from the viewpoint to the object and the points that are in the shadow region are discarded. This gives a point cloud of the object which better resembles the object cloud in the scene. The ICP algorithm is then fed this filtered cloud, the scene cloud, and the initial guess to compute the pose of the object in the scene. The ICP does a fairly good job of registering the two point clouds. Based on the initial seed, i.e. the knowledge of where the object is in the point cloud scene and an accurate knowledge of the model of the object, ICP gives very good results. This model based detection algorithm was used on the door (see Fig. 2.3) and the valve. This approach was finally discarded in favour of the autonomous/manual door detection methods and other valve detection methods because of the time taken to provide the initial seed using an interactive marker.

2.3 Visual Servoing

Control based on feedback of visual measurements is termed visual servoing. It is essentially a broad class of control schemes \[11\] which aims at reducing the error between the goal pose of a robotic manipulator and its actual pose by means of feedback via a vision/perception system. There are two broad sub categories of visual servoing - “eye in hand” and “eye to hand”. In the eye in hand approach, the camera is attached to the robot (generally the end effector) whereas in the eye to hand approach, there is a static transformation between the camera and the robot.

2.3.1 Fiducial based servoing

Fiducials (markers/tags) are placed on the hand such that they cover most of the visible surfaces. Once a fiducial is detected and the pose of that fiducial computed, the location of the hand frame wrt the camera frame is found out by applying the static transform from the hand frame to the fiducial frame. Because of kinematic model irregularities, this approach to getting the hand position instead of traversing the kinematic chain proves to be quite superior if the pose can be determined with a high accuracy from the fiducials. The error vector is then the difference between the pose of the hand frame and the pose of the object
which is then reduced using a control law.

Figure 2.4: Robotiq hand with fiducials (square elements in pink).

Figure 2.5: Fiducial detection on a side of the Robotiq hand.

Fiducials, specifically April tags [17] have been used in this method. Seven tags are placed on different parts (see Fig. 2.4) of the Robotiq hand (Atlas’ end effector). The camera is the Multisense head (eye to hand approach). Pose information about the tag’s placement is obtained using the PnP (pose and perspective) equation [21] after the tag is detected. Tag tracking (see Fig. 2.5) using an Optical Flow algorithm has also been implemented.
Hand pose coordinates are found out by applying the static transform corresponding to the detected tag.

The target object is detected using an object detection algorithm, and the pose of the hand is found out using Fiducial detection/tracking. The error vector is given by the difference in the poses of the target object and the hand. This error vector is then minimised using a proportional controller to achieve movement of the hand to the target object.

2.3.2 Model-fitting based servoing

This approach differs from the previous approach in the sense that instead of using April Tags to determine the location of the hand, a mesh model of the hand is used. The algorithm described in Section 2.2 is used to register the mesh model to a point cloud of the object in it. The hand model is registered in the laser point cloud (which has a huge sweep angle) containing the robot hand by using the kinematics estimate as the seed for the hand position (see Fig. 2.6). The green hand represents the kinematics estimate of the hand, the red dots correspond to the points from the laser point cloud and the white hand is the hand after registering a model to the point cloud using the kinematics seed.

![Figure 2.6: Model fitting of the Robotiq hand.](image)

As before, the target object is detected using an object detection algorithm and a simple proportional controller is used to reduce the error vector, i.e., the difference in the poses of the target object and the hand.
2.3.3 Global contrast based eye in hand servoing - A study

In the past five years, Cheng et al have made significant progress in the area of global contrast based salient region detection [4] [5] [6]. The algorithm developed by them uses the global contrast in the Lab space to detect salient regions in an image. For certain tasks in the DRC like the door task, when the robot hand is near the handle of the door, a camera mounted on the hand would see for most part the handle and it will be in significant contrast to the door. This relationship is exploited here. A camera is mounted on top of the robot hand. When the robot’s hand is close to the door handle and the final visual servoing stage needs to be performed to align the hand correctly enclosing the handle, the global contrast based salient region detection segments out the contrasting handle from the background door and then the hand tries to align itself to the centroid of the salient region. A few tests were performed but because this algorithm depends quite a bit on the lighting conditions, it did not give consistent results.

2.3.4 Time of Flight camera servoing

Figure 2.7: Visual servoing to the valve center. Red line shows the normal to the valve.

In this approach, instead of an RGB camera, a time of flight camera (Sentis M100) is fixed to the end effector. The TOF camera gives depth images based on the range limit specified using the integration time of the camera. Using point cloud algorithms, objects of
interest like the valve (see Fig. 2.7) and the door handle are detected and the end effector is then servoed to the object with proper alignment. This algorithm depends heavily on the segmentation algorithms used to segment the desired objects of interest.

2.4 Door detection

The door detection algorithms described here deal with single frame doors with a standard width of about 35 inches. According to the American Disabilities Act, a door’s width has to be at least 32 inches (35 inches is the standard). This is an important feature that is used to segment out door candidates from an image. A color based handle segmentation algorithm is used to detect the handle inside the door. Two methods for door detection, one autonomous and the other manual is described in the following subsections.

The door detection problem in the context of the DRC can be defined as the detection of a standard door not exceeding the dimensions of 36” by 82” with a combination lever handle standing vertically on the ground.

2.4.1 Autonomous detection

Work has been done on door detection using various kinds of sensors. [20] explored a way in which doors can be detected in a point cloud of a robot neighbourhood using plane detection techniques. [23] used monocular vision to detect doors after segmenting corridors. [24] investigated door detection based on the inherent low level geometric features and its close relationship with corridors using a hypothesis generation and verification using a feedback method. A probabilistic framework modelling shape, color and the motion properties of a door and its surrounding objects fed through an EM algorithm was extensively studied and implemented in a real world scenario at Stanford [1]. As mentioned earlier, two assumptions are made -

- Since the door “always” stands vertical on the ground, and because the camera is mounted on the robot, the vertical edges will almost always stay vertical in the image if the robot is standing straight. This is the pose of the robot before door detection and hence, this is a reasonable assumption to make.
• Since color based segmentation is used for handle detection, the door handle color should be significantly different from the door color which is usually the case for most doors in the world, and hence a reasonable assumption to make.

Figure 2.8: Door detection algorithm flow diagram.

Let \( P \in R_P \) be a vector containing two vectors \( P_{2D} \) and \( P_{3D} \) where \( R_P \) is the set of all \( P \)s for a particular image. Let \( P_{2D} = \begin{bmatrix} u \\ v \end{bmatrix}^T \) be the pixel coordinates in the 2D image. Let \( P_{3D} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}^T \) be the coordinates in the 3D task space with respect to the camera frame for the corresponding 2D image pixel denoted by \( P_{2D} \).

\[
P = \begin{bmatrix} P_{2D} \\ P_{3D} \end{bmatrix}^T
\]  \hfill (2.1)

Let \( P_1, P_2 \in R_P \) and \( S_P \) be a set of \( P \in R_P \) where \( P_{2D} \) lie in the line joining \( P_{1/2D} \) and \( P_{2/2D} \). Let \( D \) represent a door candidate containing \( P_1, P_2, S_P \). \( D \) is given by -

\[
D = \begin{bmatrix} P_1 \\ P_2 \\ S_P \end{bmatrix}^T
\]  \hfill (2.2)

For \( D \) to belong to a class of doors as specified above,

- \( \forall P, \) where \( P \in S_P, \) and \( a, b \in \mathbb{R}^3, \) \( (a - P_{3D}) \cdot b = 0 \)

- \( 80 \text{ cm} \leq || P_{1/3D} - P_{2/3D} ||_2 \leq 100 \text{ cm} \)
The general idea as described above behind the autonomous door detection algorithm (see Fig. 2.8) is to use the geometrical features of the door, i.e. to segment out parallel vertical lines, find the perpendicular distance between them and then accept those lines which have a distance of the door width between them. The 2D image from the Multisense is first filtered. Histogram equalisation is performed to increase the contrast of the image which helps in better detection of edges. A bilateral filter is then applied to blur (low pass filter) the image. The second stage is detecting edges (see Fig. 2.9) using the Canny edge detector. The Probabilistic Hough Transform is then applied over the edge image to find lines. By checking the line slopes, only the vertical lines which have a pixel length greater than a preset threshold are kept (see Fig. 2.10).

![Figure 2.9: Edge image after applying the Canny edge filter.](image)

The vertical lines are then grouped into pairs to form probable door candidates (see Fig. 2.11). Line pairs that have a pixel distance of less than a hard pixel threshold are eliminated which reduces the search space but compromises on the detection distance.

The entire image is then reprojected in 3D. The line equations for the grouped line pairs are then found using \[10\]. Pairs that have a perpendicular distance of \(90 \pm 10\) cm are kept and the rest are discarded. To reduce door pair redundancies, door pairs that have the same line equations are merged together. The door is finally validated by checking that a solid plane exists in between the door lines. The validation is done by taking diagonals of the quadrilateral formed by the two lines, sampling points on them in the image and checking to see if their corresponding 3D points are on the same plane. If they are not, then the line
Handle detection is done based on the fact that the handle is of a significantly different color than the door (assumption 2). It is done in the following way -

- The mean ($\mu$) and standard deviation ($\sigma$) of the pixels in the image space inside the door region detected earlier are found.

- Connected Component Analysis (CCA) is used to grow the region which corresponds
Figure 2.12: Detected door.

to the pixel value within the range $\mu - \sigma$ and $\mu + \sigma$.

- Based on the knowledge that the handle is on the left or the right hand side, a region is selected correspondingly to exclude the central part of the door.

- In that selected region, the region which is not white starting from the bottom and exceeds a certain contour area threshold ("hard") is selected as the door handle (see Fig. 2.13).
Finally, the entire algorithm is repeated over a window of 10 frames. Detections from every frame is recorded. At the end of ten iterations, the detection with the maximum occurrence is selected as the door (Fig. 2.12). In Fig. 2.14, a complete visualization flow is shown.

2.4.2 Manual detection

Unlike the previous method where a scene is given to the detection algorithm to look for a door, this method asks the user (hence manual) to scribble (see Fig. 2.15) on the door handle in the scene. This scribble serves as an initial seed for the algorithm which then tries to find the door within a fixed neighbourhood of the handle.

The user scribbles on the door handle in a 2D image on the GUI. The centroid of the scribbled points is a relatively good estimate for the center of the door handle. After reprojecting the image into 3D, the 3D point corresponding to the 2D center point’s reprojection is found out. From the laser point cloud, nearest neighbours to this 3D center point of the handle are found out. A plane is then fitted to these neighbours. Finally, the algorithm’s output variables contain the door handle’s center and normal (see Fig. 2.16).
Figure 2.15: Door scribble (green color) in the GUI near the cursor.

Figure 2.16: Door normal obtained in the laser cloud from the scribble seed.
Chapter 3

Atlas Manipulation Capabilities

Manipulation is defined as the robot’s ability to generate trajectory plans for different manipulation tasks like moving and grasping (see Fig. 3.1) while avoiding obstacles in the environment and executing the generated trajectories.

Figure 3.1: Atlas opening a door.

There are sampling based motion planners like the Rapidly Exploring Random Trees (RRT) [15] and its successors [14] [13], which are probabilistically complete and can find a feasible path in the search space. However, searching high DOF (30) configuration spaces
is still a very time consuming task. Most of the RRT generated trajectories needs post processing. The other category deals with optimization based algorithms. CHOMP [18], STOMP [12] and TrajOpt [22] can generate high quality paths (smooth and robot executable) from an initial seed trajectory that could be in-collision or dynamically infeasible. The solutions are obtained pretty fast even with high DOF problems. One problem with these classes of algorithms is that they may get stuck in local optima. Since low computation time and deterministic solutions are a high priority for the DRC, optimization based planners were chosen over sampling based planners.

The low level full body controller [8] used in the DRC computes the desired joint torques, velocity and position and sends it to the joint servo controllers. The high level goals are set (like desired Cartesian motions for specific locations on the robot like the hand, foot, center of mass) in the high level controller which are taken as inputs by the full body controller and it computes the physical quantities (torque, velocity, position) for each individual joint. These outputs are then referenced in the joint level servos. Joint position and velocity are computed using the Inverse Kinematics (see Fig. 3.2) and the joint torque is computed by the Inverse Dynamics. Both the IK and ID are formulated as Quadratic Programming
problems. In this chapter, the trajectory optimizer is presented in detail and finally the step planners for generating steps are discussed, mainly the constrained step planner.

### 3.1 Trajectory optimizer

As mentioned earlier, due to the low computation time and deterministic solutions requirement of the DRC, TrajOpt is chosen as the motion planning algorithm. TrajOpt has an advantage of being able to incorporate collision avoidance as a constraint in the optimization problem instead of spending a lot of time doing collision checking. TrajOpt moulds the problem as a sequential convex programming problem and solves it (see Fig. 3.3) using a convex optimization solver (Gurobi).

The desired Cartesian motions for the robot are specified, such as, feet, hands, and COM positions. The optimizer formulates these motions as its costs and constraints and computes a trajectory represented by the joint states at a set of waypoints. The general formulation of the optimization problem where \( f, g_i \) and \( h_j \) are scalar functions -

\[
\begin{align*}
\min_x & \quad f(x) \\
\text{s.t.} & \quad g_i(x) \geq 0, \quad i = 1, 2, \ldots, n_{\text{ineq}} \\
& \quad h_j(x) = 0, \quad j = 1, 2, \ldots, n_{\text{eq}} 
\end{align*}
\]

(3.1)

For the Atlas, only kinematics is considered and the trajectory is represented as a sequence of \( T \) waypoints. The optimization variable \( x \) is of the form \( x = q_{1:T} \), where \( q_t \in \mathbb{R}^K \) describes the joint configuration at the \( t \)-th time step for a system with \( K \) DOF. The cost function \( f(x) \) is written as:

\[
f(q_{1:T}) = \sum_{t=1}^{T} ((q_{t+1} - q_t)^T Q_1 (q_{t+1} - q_t) + (q_t - q_{\text{norm}})^T Q_2 (q_t - q_{\text{norm}}) + d^T Q_3 d)
\]

(3.2)

where \( Q1, Q2, Q3 \geq 0, q_{\text{norm}} \) represents a nominal posture. These quadratic cost terms are penalization of the weighted sum on the joint displacements between the waypoints, posture error in joint space and posture error in Cartesian space. The posture error in
Figure 3.3: Planning using TrajOpt.

Cartesian space can be obtained by calculating the distance $d_\Delta$ from a given configuration to a task space region introduced in [2].

### 3.2 Step Planner

Step planners are high level planners that generate steps for the robot to execute. The step planner generates steps and the steps are then sent to a low level walking controller [9]. The Atlas robot’s step planners can be divided into two broad categories -

- Flat ground planner
- Terrain planner

The Flat ground planner has two implementations. One uses linear interpolation to generate the steps from the initial to the goal state and the other uses the A* algorithm to
generate steps avoiding obstacles. The terrain planner is used to generate steps for walking on uneven terrain.

### 3.2.1 Constrained Step Planner

The planners described above assume that the only constraints on the robot are its center of mass and the position and orientation of its feet. But for the door task with a pull door, the robot needs to walk through the door while keeping the door open with one hand. One of the challenges faced while doing this is that the contact forces on the robot’s end effector changes while sliding the hand along the door to keep it open. These changes give rise to unnatural behaviour in the robot’s motion which sometimes leads to a fall. This is avoided by using a constrained step planner which at every step tries to maintain the hand (which is holding the door open) position at the exact same location while moving thus minimizing the chances of fall.

Footstep planning is done in a three dimensional space, the ground plane and orientation. A* is used to plan for the footsteps. The A* neighbouring states are determined using a pre-determined set of feasible steps given a swing foot and a fixed foot. The constraints (center of mass, feet and hand poses) are embedded in a Jacobian matrix and is used to solve for the Inverse Kinematics at every iteration of the A* algorithm. The A* cost function includes weighted Euclidean metrics for each dimension and also accounts for the step length by adding a penalty for step length which biases the robot towards big steps if they are feasible.

Where $q = (x, y, \theta)$ denotes a state comprised of $(x, y)$ position and $\theta$ orientation in a 2D flat ground plane, the cost function for A* is -

\[
g(q_i) = \lambda_1 ||x_i - x_s||^2 + \lambda_2 ||y_i - y_s||^2 + \lambda_3 ||\theta_i - \theta_s||^2 \\
h(q_i) = \alpha_1 ||x_g - x_i||^2 + \alpha_2 ||y_g - y_i||^2 + \alpha_3 ||\theta_g - \theta_i||^2 \\
f(q_i) = g(q_i) + h(q_i) + (\beta \times \text{steplen}_{\text{length}}) \tag{3.3}
\]
Figure 3.4: Constrained manipulation planning - Before stepping through the door.

Figure 3.5: Constrained manipulation planning - Stepping through the door.

Figure 3.6: Constrained manipulation planning - After having stepped through the door.
In Fig. 3.4, the robot starts at a location on the left of the door with its hand holding the door in place. The goal position is to the right of the image after crossing the door. In Fig. 3.5, the robot is halfway through the door with the hand at the same position as when it had started. In Fig. 3.6, the robot has crossed the door and is on the other side. The robot hand can safely slide down then.
Chapter 4

Door Task

Figure 4.1: Door at the test event in South Carolina.

The door task is the third task that needs to be performed by the Atlas. As mentioned earlier, this task is very important as the robot cannot decide to skip this task meaning that without completing this task, the next tasks cannot be performed. According to the new rules\footnote{http://www.theroboticschallenge.org/team-resources}, the door (see Fig. 4.1) with a lever handle will open inward away from the robot and this task cannot be skipped as it will provide access to the indoor manipulation tasks. The door will not include a threshold. And, once fully opened, the door is designed to stay open. Work on both the push and pull door have been done and both of them are
presented. This chapter focuses on the approach that has been taken for completing the door task. The perception and manipulation capabilities described in earlier chapters, the door task completion strategy and the software framework is discussed in detail.

Figure 4.2: Strategy for the door task (pull door).

4.1 Event driven FSM

The current approach for a pull door is shown in Fig. 4.2. The robot starts at an initial position. It plans steps to the door, and gets to a position at an offset from the door. Then it opens the door with one hand, blocks it with the other and aligns itself sideways perpendicular to the frame of the door. Then it starts walking through the door. A very similar approach is employed for a push door. The robot walks up to the door, stands at an offset, opens it with one hand, aligns itself and walks straight through. An event driven finite state machine (FSM) with the sub-tasks as the states is used to control the execution of the strategy with human validation at critical junctions (see Fig. 4.3). The first state is door detection. This is performed using the detection methods discussed in Section 2.4. The next state is walking to the door based on the pose of the door. The third state is opening the handle which is divided into four sub states, moving to the handle, grasping the handle, turning the handle and then pulling the door back. The final state is walking through the door.
The DoorMotion GUI (see Fig. 4.4) is a widget in the WPI-CMU team’s Atlas GUI framework (drc_gui). When there is no error in any of the states mentioned above, i.e., when the execution of the task follows the course of the fsm flow, then the user only uses the NEXT button to switch to the next state and the YES button to validate states that require validation of outputs. The REPEAT button is there in case the action performed in the current state needs to be repeated. In case there is some error and the task deviates from the fsm flow, the Manual FSM Control checkbox allows the user to access the controllable Events to switch to states that do not correspond to the fsm flow.
4.2 Door detection

Door detection is the first state in the door task. When the FSM is in this state, the Door Detection object is instantiated. The door detection algorithm subscribes to the image feed from the Multisense camera on the head and tries to detect the door using the autonomous door detection algorithm described in Section 2.4.1. The algorithm runs for 10 seconds. The user then validates the detection by looking at the normal from the door handle that is generated by the algorithm (see Fig. 4.5).

Figure 4.5: Door detection on the robot using autonomous door detection.

Figure 4.6: Validation step for a push door after manual door detection.
If the detection fails, then the user changes the flow of the FSM by manually initiating a switch to the manual detection algorithm as described in Section 2.4.2. After manual detection is done, the user validates (see Fig. 4.6) the detection like before. Once the validation is performed, the FSM switches to the IDLE state.

4.3 Walking to the door

Figure 4.7: Atlas walking to the door.

The next state (GUI view in Fig. 4.8) is walking to the door (see Fig. 4.7). Since it is difficult to establish a direct correlation between the distance travelled depending on the number of steps when the distance required to traverse is more than about one and a half metres due to drift, the walk to the door is done in two stages. The door handle position is projected onto the ground plane and based off of the normal to the door, the robot is given an initial target midway between the door and the current position of the robot. This target is passed on to the FLATGROUNDASTAR planner which finds a step plan (see Fig. 4.9) using the A* algorithm and passes on the steps to the CMUWalk low level controller for performing the actual walk on the robot. After the robot reaches the first target, the door handle’s position is updated based on the robot’s state estimator which is based on the kinematics.
of the robot. A nearest neighbour search is performed and a plane fitting is done at the updated handle position in the laser point cloud which yields the updated normal to the door. Now, this updated position of the door handle with an offset (to position the robot accordingly in front of the door) is passed on to the A* step planner which then generates the steps for the second stage of the walk. The offset is dependent on the type of door (push/pull) and also the location of the handle (left/right side).

Figure 4.8: GUI view while walking to the door action is in progress.

Figure 4.9: Planned steps using the step generator to walk to the door.
4.4 Opening the door

Figure 4.10: Generated handle turning and door pulling motions. (left: robot’s current pose; middle: handle turning trajectory; right: door pulling trajectory)

Figure 4.11: Atlas opening a door (left) and its next motion plan to block the door from closing (right).

After the robot is positioned at an offset in front of the door, door opening is the next state. Door opening is also dependent on the type of the door and the location of the handle. All the motion planning is done using the trajectory optimizer, TrajOpt.

Door opening for a pull door with its handle on the left is discussed first. The first task is moving the right hand to the handle and then visual servoing the hand to make sure that
the hand surrounds the handle. Then, the handle is turned clockwise by 90 degrees (see Fig. 4.10). Once the door opens, the robot hand is pulled a little bit by nudging and the handle is turned back 90 degrees anticlockwise. This is to ensure that the movement of the robot hand is in its normal less restrictive pose. Then, the door is pulled back with the trajectory of the hand following a circular path about the door hinge. Next is blocking the door with the left hand (see Fig. 4.11) to prevent it from closing. For a push door with its handle on the left, the robot instead of using a hand to block the door, it pushes it forward while walking through it.

4.5 Walking through the door

The last state in the entire door task flow is walking through the door. As mentioned in the door task strategy earlier, because the robot barely fits through the door frame when it is staring directly at the door, the robot is oriented sideways and then it is walked through. At the beginning of this state, The door is already open and steps are generated so that the robot first aligns itself to the center of the door frame. After aligning, the robot puts its arms close to itself and starts walking sideways (see Fig. 4.12) until it is through the door. The steps here are generated manually instead of using the Constrained step planner because of planning time limitations and because it is very difficult to check for collisions with the environment as the robot gets extremely close to the door frame in this maneuver.

Figure 4.12: Atlas oriented sideways and walking through the door.
Chapter 5

Experiments and Results

This chapter presents the recorded experimental data to evaluate the algorithms, methods and strategies proposed in the earlier chapters. The experimental data is mostly related to the success/failure rates over numerous trials/runs and also the computation time taken by the algorithms. This chapter describes experiments related to the various visual servoing algorithms described earlier, then the door detection algorithm, and finally the door task.

5.1 Visual Servoing results

Robotiq hand tracking using April tags gave a detection and tracking ratio of 90% and the pose estimation started drifting based on the distance of the hand from the camera (Multisense head). This is partly due to the inaccuracies in modeling the lens parameters of the camera because of which the PnP solutions were not as good as expected. The model fitting did not perform well on the hand because sampling the mesh model of the Robotiq hand did not give an accurate representation of the point cloud that was seen by the LIDAR. Both these approaches had to be discarded because of constant modifications to the Robotiq hands to attach hand cameras or other materials to aid in the completion of various DRC tasks. Global contrast based salient object detection methods were tested very briefly but the results were not deterministic as the salient object detection depended heavily on the ambient light. The TOF camera worked 75% of the time for visual servoing to the valve and is being actively used in the valve task.
5.2 Door detection results

For multiple runs of the autonomous door detection algorithm, the algorithm performed very well for doors situated within a proximity of 2.5m to the robot. The detection results from various perspectives in a scene with moderate ambient light are shown in Fig. 5.1.

Figure 5.1: Door detection experiments from various viewpoints.
5.3 Door task results

![Pie chart](image)

**Figure 5.2:** Pie diagram of the door task results. Time is in seconds.

After all bug fixes and heavy testing, the door task has a current success rate of 100% for the push door on the real robot. The success rate is 80% when a the normal flow is observed and there are no issues in executing any of the tasks. The pie charts in Fig. 5.2 describe execution times for the various door task stages in simulation and on the real robot. Table 5.1 describes an analysis of the approaches used during the DRC Trials and the DRC Finals based on computation time.

<table>
<thead>
<tr>
<th></th>
<th>Door Detection</th>
<th>Walking to the Door</th>
<th>Opening the Door</th>
<th>Walking through the Door</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DRC Trials</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Record</td>
<td>108s</td>
<td>79s</td>
<td>406s</td>
<td>286s</td>
<td>14min39s</td>
</tr>
<tr>
<td>Success Rate</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>90%</td>
<td>90%</td>
</tr>
<tr>
<td>Time</td>
<td>20.5s</td>
<td>Step planning: 41.6s</td>
<td>one initial guess(4 times): 40.4s</td>
<td>124.2s</td>
<td>7min40s</td>
</tr>
<tr>
<td><strong>Simulation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success Rate</td>
<td>87.5%</td>
<td>100%</td>
<td>80%</td>
<td>100</td>
<td>80%</td>
</tr>
<tr>
<td>Time</td>
<td>15.4s</td>
<td>80s</td>
<td>170</td>
<td>205</td>
<td>14min39s</td>
</tr>
<tr>
<td><strong>Real World</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success Rate</td>
<td>87.5%</td>
<td>100%</td>
<td>80%</td>
<td>100</td>
<td>80%</td>
</tr>
<tr>
<td>Time</td>
<td>15.4s</td>
<td>80s</td>
<td>170</td>
<td>205</td>
<td>14min39s</td>
</tr>
</tbody>
</table>
Chapter 6

Discussion

The approach presented in this thesis demonstrated a human supervised way to traversing doors using the Atlas humanoid robot. As of now, we have not yet reached a stage where a robot can be granted complete autonomy in an unknown environment. This thesis dealt with enabling robots with powerful perception and manipulation capabilities but with a user at the helm to validate state completions and helping recover from unknown field errors. Instead of having a user teleoperate everything or giving a robot complete autonomy, in our current robotics scenario it makes more sense to do a bit of both and solve important problems in the real world like the Fukushima nuclear disaster.

Several different methods for visual servoing have been studied. In reference to the DRC, depending on the task and the object being visual servoed to, different methods work better. In early 2014, because of issues related to the transform tree not mapping the transforms correctly because of kinematic modeling errors, the Fiducial based servoing and Model fitting based servoing methods were investigated. But with the latest version of the robot model from Boston Dynamics, it is not required to try and estimate the hand position using vision, as the kinematics transform gives very good results. For the valve task, valve segmentation with the normal to the valve using point cloud algorithms from the time of flight camera’s point cloud works very reliably and hence is used for the valve task. But the time of flight camera cannot detect the points on the door handle as it is very shiny. So, based on the object being segmented, the choice of visual servoing technique varies.
The autonomous door detection algorithm detects correctly when the robot head is oriented straight in line with the door (first assumption). In a complex scene with sufficient ambient lighting, the door and handle detection works till a distance of about 2.5m. After that, it starts failing to a considerable extent. If ambient lighting is insufficient, the door detection still works fairly well but the handle detection fails as it is segmented based on the second assumption, i.e., the color of the handle is different from the door color. Overall, it performs very well on the real robot when it is trying to detect a door according to the DRC specifications.

As mentioned in the results, the door task has a success rate of 80% when a normal state flow is observed. If something goes wrong during one of the states, the operator switches to manual state control and creates a circumstantial deviation in the state flow. For the 20% failures in the normal state flow, the operator took control and made the robot walk through the door 100% of the time. The main ideology that has been followed here is that the robot should be allowed to do as much autonomous work as possible but a human should be there to intervene and validate at critical junctures, where being critical is defined by environment uncertainties with possible potential state failures. Detection, walking to, opening, and walking through the door parts coupled with human supervision works very well for the door task. In conclusion, a 100% success rate of this task with human supervision on the real robot presents a strong case for a human supervised semi-autonomous approach to robots performing activities in uncertain and unfamiliar environments.
Bibliography


