ROBOT NAVIGATION FOR SPRING LOADED DOORS

Puru Rastogi[1] and Lekha Walajapet Mohan[2]

Abstract— **Navigating a holonomic robot through a door involves considering the constraints of the manipulator for motion planning, contacting and holding the door while passing through it and navigate around it. In our work we have created a planner to figure out planning and coordination of manipulator arm and manipulator's base in order to pass through a spring loaded 'pulling' door . Inspired by the work of [1] the planner uses a lattice based representation and is able to transition between robot-door contacts and non robot-door contacts. We have further validated our planner in simulation and presented the results.**

I. INTRODUCTION

Doors are important part of human structures, Any mobile robot working indoors has to know how to navigate around them in order to be fully autonomous.Generally doors need to be either pushed, pulled or both. Among them pulling is a more complex task especially in a constrained environment, since the robot has make space for door's motion further the robot has to go around the door in order to exit the door. The task becomes further complex when the door is spring loaded. The spring loaded doors automatically close for convenience. This creates a challenge for robots since now they have to hold the door open till the robot passes through it.

A lot of work has already been done on the problem of opening doors and passing through them in indoor environments [2], [5], [6], [3], [4]. Doors come in variety of sizes, variety of door knobs, variety of colors and variety of types of hinges. This creates another set of challenges.The overall door opening and navigation problem could be divided into following broad steps:

1) The robot would first need to detect a door, then identify the kind of door and get its properties which includes door size and information of whether it is push or pull and whether it is spring loaded or not.
2) Once the door and its properties are identified, the door knob needs to located and identified. This step could be skipped if the door is push kind.
3) After locating the door knob, it has identify the door knob, in order to know how it could be unlocked.
4) The robot then needs to grab the door knob and pull/push the door open. Grabbing the door knob brings in another set of problems, for example the shape of the door knob and the gripping strength needed to be applied. This could be simplified by using a library of door knobs.
5) Robot will then navigate to the other side the room. While navigating, depending on the kind of door, the robot might need to hold it open. During this and previous step, it has to be ensured that the robot is able to provide enough torque to do the door manipulation.
6) In the end, the robot needs to close the door. If the door is spring loaded then robot has just got out of door's sweeping area and let it go. Otherwise, the robot has to pull/push the door to close it.

As can be seen the overall door opening and navigation problem is huge, and contains a lot of steps, which can have their own dedicated research. In our work we focus on step 5, i.e. assuming the manipulator is already holding the door, the robot has to push/pull it and navigate through it. For this part of the problem [1] is state of the art. They have proposed a very general planner which could open push/pull non/-spring loaded doors. They went forward and tested their planner on a PR2, which is a mobile robot with two robotic arms. We wanted to test their approach on single arm mobile robots since they are decently common and if the planner can work efficiently on them then it can work efficiently on multi-arm robots too. Further we also wanted to test the performance of the planner on different parameters of the environment for example the door size and work-space size of the manipulator. Our work builds upon the work of [1].We present a simplified version of the same planner for pulling open and navigating through spring loaded doors. We chose pulling spring loaded doors, since among all the configurations they involve most amount of motion of the base, and transitions of door-robot contact i.e. they are the most complex door opening and navigation problem. Further the planner was simplified for the ease of implementation.

The paper proceeds to literature review of approaches to execute different steps of overall door-opening and navigation problem, and after this it leads to description of our simplified planner under the section 'Motion Planning', then to implementation of the planner and finally to results of simulation, conclusion and future work.

II. LITERATURE REVIEW

A. Manipulation

Briefly, approaches a decade ago do not plan and do not count for collision avoidance with the door or the environment around it[2], talks about one of the earliest works done in coordination of arm and base planning for door manipulation.Coordinated motion of the base and the arm requires collision free path generation for the navigation of the base through for the door. The constraint here is that the base motion is constrained by the arm grasping the door handle. Thus, the authors came up with a constrained planner

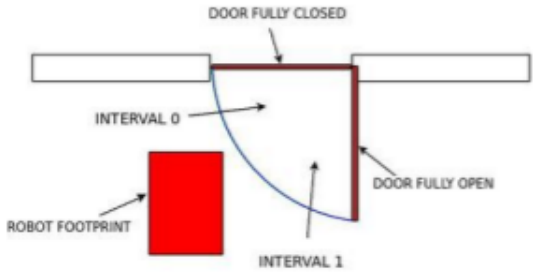

Fig. 1: The door interval. Permission requested [2]

Fig. 2: Depicting compliant motion. Contacting the surface and moving lateral while pushing it

that coordinates the motion of the base with the action of the arm, generating a collision free path for navigation through the door. Constraints acting on the base is that the base cannot collide with environment and the motion of the base must allow the end effector to grasp the handle for facilitating pull and push by the arm.

A general idea of state representation is present in the following. A graph-based approach was followed that has the following representations:

1) Lattice Based Representation :
State Variables:

- Three dimensional search space is used for positioning the base, where is the heading of the base. Note that the goal for the task, i.e. opening the door cannot be represented using this representation since it does not contain the door angle.
- A simplified representation of the door angle explaining the door interval was added to the representation. All valid positions for the door belong to one of two intervals. One interval indicates that the interval is fully closed, while another interval indicates that the door is fully open.
- State space can now be represented as (x, y, , d) where d = 0, 1 represents the interval as shown in the Figure 1.

Transitions:
Lattice-based representation is a discretization of the configuration space into a set of states, and connections between these states, where every connection represents a (short-term) feasible path defined as a sequence of poses. Actions are translational and rotational motions produced by the search, and it's successors are the resulting end configurations of these actions.

A transition is valid between any two states s and s' whose door intervals are the same exists only if it corresponds to the lattice. The consecutive poses of the manipulator should overlap with the (door angle) intervals.Another important take away of this approach is its applicability to other robotic systems as well owing to its compact based planner representation.

One of the drawbacks of this approach is that, it doesn't account for dynamic obstacles. The robot has limit sensing coverage, for which it plans. The geometry of of the manipulator is a important consideration here. The door is also assumed to be static here. Dynamic of the door might be that it is swinging.We have addressed this consideration as a part of future work in sections to follow below

2) Equilibrium Point Control:
The above paper's limitation on pulling the door has been answered in [6]. Here the authors implement of impedance control inspired by the Equilibrium Point Hypothesis[27]. Also known as Equilibrium Point Control(EPC)([25],[26]), we can infer their kinematics without detailed prior models.This extends to the motion of the robots omni-directional base both before and during pulling. Using location and orientation of the handle the robot is capable of autonomously opening the door.EPC can coordinate the movement of the robots omni-directional base and robot's compliant arm while performing the pull operation, which leads to an efficient operation, as shown in Figure 2. This method demonstrates improved robustness of the system.

Inspired from bio mechanical models, as per definition, "equilibrium point is the configuration at which the mechanical system would settle in when there are no externally applied forces apart from gravity". By adjusting the Cartesian Space Equilibrium Point(CEP), one can compute where the robot's end-effector would end in the absence of external forces apart from the gravity. The CEP controller uses CEP to perform push and pull operations, computes Inverse Kinematics to compute Joint Space Equilibrium Point which is given as the input to the real-time controllers.

The robot has a camera under the mobile base for computation that detects and tracks features as it navigates on the floor. Only limitation that the author mentions is about the performance of wheel encoders suffer from the slippage of the floor. We can infer from the test results that the errors were due to visual odometry, which is a global problem to any robotic application that tries to use images for processing. Another major part of failure was the location of the handle. Given these limitations, EPC enables the robot an autonomous door opening capability and coordinate motion between arm and the arms. This, like most of the other planners, can not work for spring loaded doors or doors that require huge force application. Like previously discussed in other applications, this fails in dynamic environment.

A review of above system clearly conveys that the robot

is expected to make contact with components while looking to grasp handle, but is not designed to accommodate large forces. The impedance control factor used here is lower than other impedance controlled arms by a factor of 5. An extension of this approach would be to account for spring loaded doors(large forces) and improving the visual subsystem for better identification. We can try to adapt 2D sliding window detector used in [5], in which case a floor facing camera would not be eligible. Our project is not in the scope of this paper, but our extension would be experimenting with above mentioned methods.

3) Behavior Based Mobile Manipulation:
Actuator commands, also known as behaviors of the robot interact with the environment and perform action that is suitable to that specific environment([8], [28]). Behaviors are individual set of actuator commands[17], there is a central coordinator that interacts with all the behaviors and acknowledges the behaviors if they are used for the final control command.

'Pull' behavior that actuates the pulling action and 'FollowForce' behavior that regulates the gripper force. These behaviors interact together and are weighted. As and when, pulling action is required, this combined action is executed. This type of paradigm involves human interaction. 'FollowForce' is one such behavior involving human guidance in leading the robot to the gripper.

For reference, look at Figure 4. One another behavior that executes local intelligence is the 'Move' behavior, where three behaviors interact with each other to move forward, avoid obstacles in the left and avoid obstacles in the right.

Behavior theory is very debatable approach as control-based theory seems to outperform behavior-based theory of planning. Our argument is that, there is a very wide scope of improvement. Given that machine learning's underlying theory is it's observation based on behaviors[29]. If machine learning can be asserted as robust by current algorithms, so can these behavior-based control algorithms.

4) Manipulation of Documented Objects:
Paper [9] talks about a very interesting approach - prior knowledge of object manipulation and collision avoid at reduced dimensionality of the robot. A review of motion planning algorithms, their comaparison of performance can be referred to at [32], [33] and [34] planner's input is the subgoal of a task. Subgoals are constraints that are shown in Figure 5, these constraints interact with each other or "projected" into the configuration space.This gives a prior knowledge on valid transition between the sub-goals.

Configurations are generated by random sampling[30], but, validity of the transitions are checked before sending the commands to the controllers and the checks are made by interpolating the configuration spaces. A graph of transitions is built to generate valid path from random sampling of configurations. One such valid transition is shown in Figure 3, where a humanoid robot is trying to move through a sliding door using two of its arms.

As this is specific to a humanoid robot, the gait generation [18],[31] is given the highest priority considering the dynamics of the robot. A generalized inverse kinematics solver is used to generate the joint angles of the humanoid manipulator. The door parameters and Degree of Freedom Values that cause that specific constraint in the simplified robot model are parameters on which the kinematic solution depends on.

We adapted the idea proposed by this paper. We simplified our robot model to our robot workspace and we check for the robot's contact on the door knob for the transition from one state to another to be valid. We do not use the documentation part mentioned by the authors as we have made assumption about the location and type of the door handle. The biggest limitation of this approach will be that, the collision checker works only for the simplified model of the robot. In real-time, collision might happen with the robot with 12 DOF's or any Degree of Freedom for that matter. This assumption of simplified model makes it unusable for robust situations. The future work will also be in regard to improving its global collision checking.

5) Generating Low Cost Motion Trajectories: Heuristic search for high dimensional planners are not effective enough owing to its computation cost and expansion/repair of states as the dimensions increase. In [11], authors have proposed a heuristic based planner that satisfies three conditions. Firstly, it finds a low dimensional planner which will serve as a heuristic search that are informative and will guide the search of the planner. Next condition being, the majority of the plan will be broken into sub-plans or chunks for motion primitives, based upon which we build our graph to further perform the heuristic informative search.Thirdly, using real time heuristic search like Anytime Real A* planner which has bounds on the sub-optimality.

The authors have proposed an algorithm is based on motion primitive graph generation. Heuristic function is framed by simplified low dimensional search. The key features are captured in the simplified search, here the heuristic h is admissible and consistent. One of the heuristic used here is to remove the dimensionality, another heuristic function they have used is the difference between the orientaiton of the end effector. The search used for ARA* generates a suboptimal solution, but is very quick compared to the other standard algorithms as shown in Figure 6.

Path constraints are constrained by manipulator joint limits or joint position. Given multiple input to the planner, the output is the least cost paths of all Figure7, which is the primary essential feature of the motion planner., The author has validated its result in valid grasp poses. Additional smoothing feature will try to connect waypoints further from each other to avoid collision free path [19]. Smoothing also enables the motion based planner to come up with a least

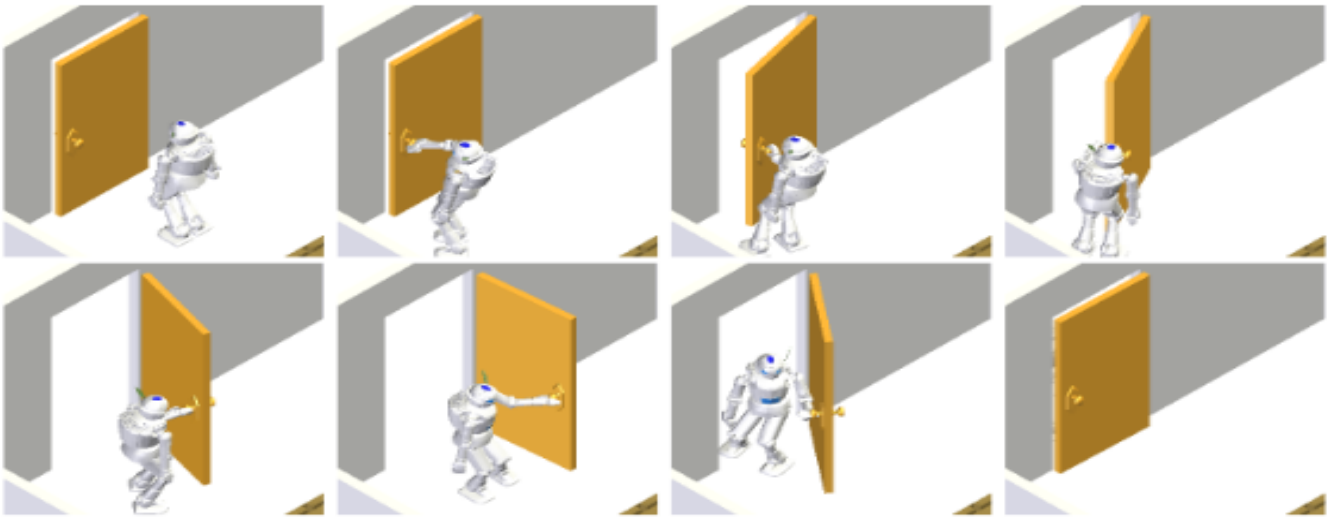

Fig. 3: Opening and Closing of the sliding Door. Permission requested [9]

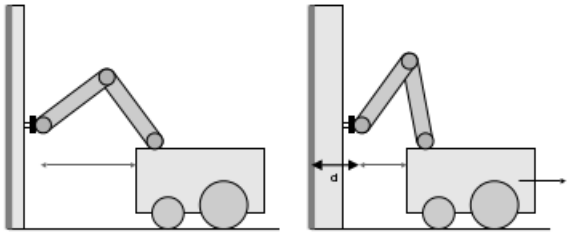

Fig. 4: Base and arm joints reacts when the base moves towards the door. Permission requested[8]

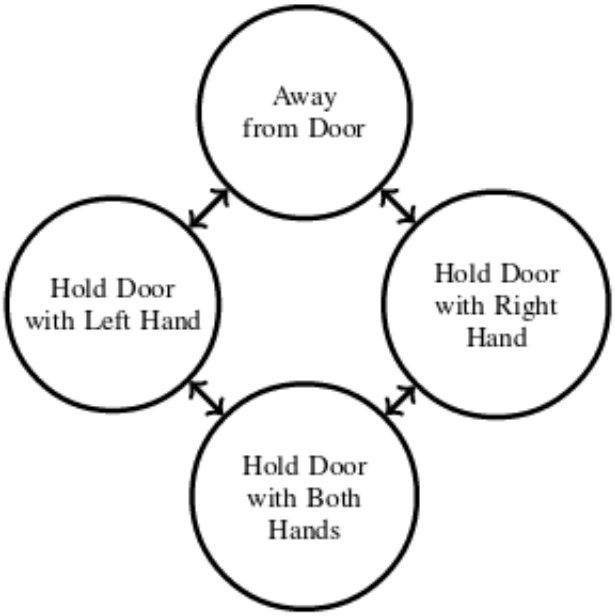

Fig. 5: Graph representing Constraints. Permission requested[9]

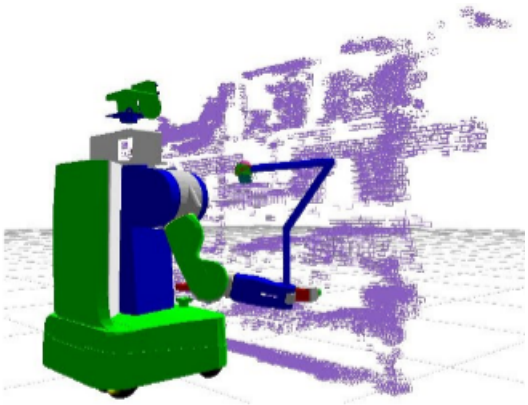

Fig. 6: Heuristic function is represented by the blue pipe. Permission requested[[11]]

heuristic	expansions	planning time(sec)	solution cost
dijkstra	88,858	13.65	46000
euclidean	432,561	60	-

Fig. 7: Planning statistics comparison between Djikstra and Euclidean. Permission requested[[11]]

cost[21] path given enough time. Smoothing[22] tends to remove uneccessary motion primitives.

B. Feature Detection of Door Handles

Door handles and doors vary in shapes, sizes and appearances that makes it difficult for the robots to open a novel door. To avoid the problem of occlusion while detecting the door handles, the authors in [5] have proposed an algorithm to detect local features in the door handle using vision and machine learning.

1) Waypoint Trajectory Estimation:
Three concrete points p_1,v_1,p_2 belonging to the 3D space are defined. p_1 indicates surface location of the handle at the axis of rotation, v_1 is the axis of rotation of the door handle, normal to the plane of the door. p_2 is the location on the door handle to push it down. These 3D points define waypoints(pose) of the robot end-effector that is passed on to position-controlled end-effector, which performs pulling and pushing actions.

2D sliding window detector algorithm decides if each rectangular region roughly bounds a door handle as depicted in Figure 8.It then outputs the detection with confidence values. Vision-based classifier, along with a 2d laser scan of the door plane is used to determine the axis of rotation of the door handle, the location to apply force, and the door plane normal

In [5], Using waypoints, The 3D features geometrically computes the path of rotation, rotation of the robot wrist, in order to pull the door handle down. The results were very impressive with localization error of 2 cm and recognition rate of 94.5%. This method works really well for pushing the door. The force feed back is unaccounted here, which is very essential for a robust door navigation planner. This definitely a part of our future work, where we intend to integrate the kinematics of the door, to build a generalizable planner irrespective of the door sizes and shapes. This will also, implicity extend to the operation of pull operation.

The planner has effectively proven to work well in the high dimensionality problem, as we can infer from the results above. The planner returns the sub optimal plan and is inefficient under the suboptimal bounds. Future work would be to expand this work to improve the efficiency or optimal cost returned by the planner.

2) Door Identification:
In [7], The aim is to select the required parts of the image, given with a confidence rate and comprising of key features to identify the door handle parts. The authors here obtain three steps for door identification, namely region detection,colour segmentation and statistical validation

Circle Hough Transform(CHT) [16] is used for detecting circular shaped objects in the images. Images of the Door handle patches are used for detecting the circles and eventually the radius of the handles because the images taken by the robot are from a specific distance. Non-circular handles are taken care of by approximating or using blob detection algorithms. To confirm the door handle patch detection, a supervised machine algorithm is run for door handle detection. Like most of the algorithms, the varying conditions of lighting intensity does affect the segmentation output. Statistical validation comprises of comparing the perception output results from the first two methods and validating the resultant output as shown in Figure 9. This step is to avoid the effect of noise, misdetection due to the lighting conditions.

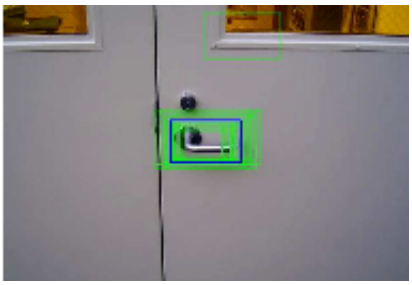

Fig. 8: Door handle classifier

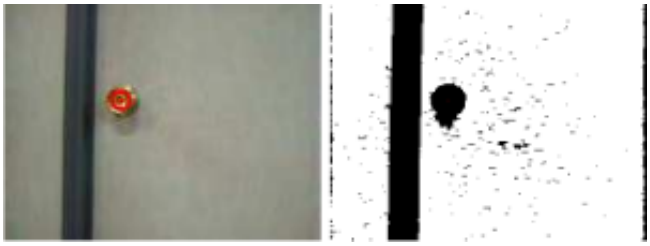

Fig. 9: Door handle segmentation output. Permission requested [7]

Local image descriptors can also be used for identifying the objects, here door handles from the images. Scale Invariant Feature Transform [15] is one such feature descriptor that is invariant to image transformations, scale and rotation as the invariant operators using Potential of Gaussian function, shown in Figure 10.

The perception subsystem is out of the scope of this project. Above discussed methods are essentially proven methods for improving the efficiency of the door navigation of manipulators. We have discussed further about visual features computation in our future work.

III. MOTION PLANNING

We solve the planning problem for pulling a spring loaded door, moving on the other side of the door and finally exiting the door, using only on robotic manipulator. The approach also works in constrained environment when the door cannot be fully-opened without moving the base of the robot.

The planning task can be broken into following sub-tasks:

1) Pull the door, while holding it from door knob
2) Move on the other side of the door (while holding the door)
3) Transfer the hold of door from arm to the base of the robot
4) Hold the door from the other side through arm
5) Exit the door

Further, the planning solution has to satisfy the following constraints:

1) The robot shouldn't collide with the walls or the door
2) The door's handle should always be in the work-space of the robotic arm when it is holding the door.
3) The robot should always be holding the door either through robotic arm or the base.

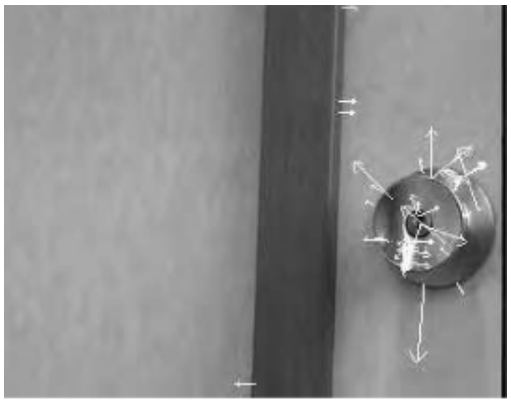

Fig. 10: SIFT feature detector. Permission requested [15]

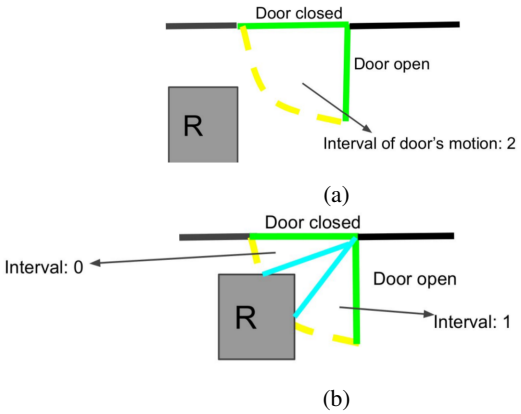

Fig. 11: The figure (a) and (b) show three different door intervals. The green and blue lines represent different orientations of the door, and yellow dashed line represent the arc made by the edge of the door. The grey square represents the robot. a) When the robot is outside the sweeping area ($d = 2$), b) When the robot is inside the sweeping area ($d = 0, 1$)

Since the robotic arm is holding the door knob most of the time, hence we have restricted the motion of its end-effector to the door knob's plane of motion. This simplifies the problem, since the planner needs to check the position of door knob in 2D work-space of arm instead of 3D. The proposed planning algorithm (henceforth referred as the planner) first constructs the graph based on predefined motion primitives [12], then it performs a graph search over the constructed graph from the start state (based on the position of the robot) to the state satisfying the goal condition. It has to be noted that in every start state, the arm would be holding the door knob, while the door is closed, moreover, there could be multiple states satisfying the goal conditions.

A. State Representation

The planner uses lattice-based representation for constructing the graph, where a lattice is discretised form of configuration space made up of set of states. Hence, the state needs to contain all the necessary information for planning i.e. the position of the robot, door angle, the how the door is being held. We use (x, y) to represent the position of the robot's base. Since the robot could be non-holonomic, hence the orientation of its base will affect its direction of motion. Therefore, for simplification we make the robot to always face forward, and define our motion primitives based on it. Further, it also lets us fix the work-space of the arm with respect to the position of the robot which otherwise is dependent on the orientation of the base.

In order to fix the position of arm's end-effector the planner needs to know the door angle. We use the 'd' for representing the state of the door. A typical door moves from 0 to $\pi/2$ rad. based on very compact representation of door angles given by [1] ,we represent door angle using the angle-interval the door can freely move in. This leads to three possibilities as shown in Figure 11:

1) $d = 0$, when the robot is inside the sweeping area of the door such that the door can fully close but not fully open.
2) $d = 1$, when the robot is inside the sweeping area, and the door is able to fully open but not fully close.
3) $d = 2$, when the robot is outside the sweeping area of the door, letting the door fully close and fully open.

The door can be held in multiple ways. It could be held using robotic arm or base of the robot, further it could be held from different sides of the door. We represent the information of how the door is being held using two variables. The variable h tells the side of door from where it is being held. it takes the value 1 when it is being held from the side facing the goal position of the robot, else it will take the value 0 when the door is being held from the other side.

To store information of whether it is the robotic arm holding the door or robot's base, we use variable v. v is 0 when it is the arm of the robot holding the door and it is 1 when the robot is using its base to hold the door.

Using the information about position of the robot, door angle and contact with the door, the state is represented as:

$$s = (x, y, d, h, v) \tag{1}$$

The planner uses a set of transitions defined using lattice-based planning representation [13], [14], in order to transition between states. We call these transitions motion primitives. A motion primitive is a finite-length feasible path between two neighboring states. Since our states are composed of two parts $((x, y), (d, h, v))$, a motion primitive could be composed of transitions between (d, h, v) or a discretized path of (x, y) or combination of both.

Every time a state is expanded, the successors are checked for validity. Validity is checked on the following grounds:

1) The door interval d cannot jump from 0 to 1 and vice versa without going through 2. It is direct implication of the fact the robot cannot pass through the door.
2) The door knob should always be in work-space of the robotic arm.
3) The place of door contact h cannot jump from 0 to 1 while the door is being held by arm $v = 0$, it has to

first hold the door by base $v = 1$ and then move the arm and hold the door from the other side.

B. Cost Function and Heuristic

The transition cost in the graph used by the planner depends on the difficulty of the transition, and the state where the transition is made. The transition cost is defined as:

$$c(s, s') = c_{motion} \times c_{map} \tag{2}$$

c_{motion} is the part of the cost dependent on the kind of transition. For example if the robot has track based locomotion, with tracks aligned along Y-axis, then the transition along X-axis would have higher cost since it will take more time and longer route. c_{map} depends on the position of robot. For example c_{map} would be higher near the walls and lower at the points away from the wall.

The heuristics for the planner should efficiently drive the robot behind the door and eventually on the other side of the wall, which is the goal state. Since the robot would be in the other side of the wall, in goal state the door could be fully opened and closed. Therefore heuristic could be defined based on the position of the robot and door interval. The heuristic is given as:

$$H(s) = max(0, |X_{robot} - X_{hinge}| - r_{len}) + |\alpha(s) - \alpha_{open}| \tag{3}$$

In equation 3, r_{len} is the radius of the door, whereas X_{robot} and X_{hinge} are the positions of the robot and the door's hinge respectively. α is the range of the door angle. The first part of the heuristic equation estimates the cost to reach the door, it becomes zeros when the robot is in the sweeping area of the door. The second part estimates the remaining door angle, it is zeros when the robot is outside the sweeping area of the door. Both of these parts are admissible and consistent, and affect the heuristic value at different positions of the robot. Therefore there sum gives an admissible and consistent heuristic.

C. Search

To generate a path on the graph of states defined in the previous section, we need to use an efficient graph based search algorithm. One of the option is to use A*. It is good while running the planner in simulation, but in real time there could be a lot of changes in the environment during the execution of the plan. The standard A* could take a lot of time depending on the size of robot and arm, to get an optimal path, moreover an optimal path is not always required. Therefore, using anytime variant of A*, Anytime Repairing A* (ARA*) [20]would be a better choice. ARA* works by getting a path using a inflated heuristic and repeating the search with decreased amount of inflation. This way, the planner give a solution in very small amount time, and will improve the solution with time. Further, ARA* uses information from previous searches, hence can account for changes in environment without doing the search from scratch again.

IV. IMPLEMENTATION

As mentioned in previous sections the door opening tasks consists of detecting and identifying the door, gripping and holding the door knob, pulling open the door, moving around it and exiting the door. Detecting and identifying the door is a work of perception and learning, and it is outside the scope of this work. Further gripping and holding door knob can itself be a separate challenge to solve. Therefore taking in account the time constraints we focused on implementing the planner, and provided it with information regarding the door, as well how to grip the door. The planner doesn't take it into account the configuration of manipulator on the robot. The manipulator could be 2 link or 4-link, the planner only requires to know the work-space of the manipulator to give a solution. Therefore the planner assumes, that at every point of plan the manipulator would be able to apply enough torque to execute the plan.

For the purpose of testing a simulator was built on matlab. The simulator has a mobile robot with a robotic arm. The arm is a planar manipulator with 2 links. IN the simulator the length of links, size of robot, discretization of its motion, door size can be manipulated. The planner generates a text file containing information about the problem and the planning solution which is an ordered list of states. The simulator reads this text file, and simulates it. The planner itself is implemented on python, without the use of any external planning library. Since the planner assumes that the robot has already gripped the door handle, therefore the initial state of all the problems will have the robot holding the door in fully closed position. Further, since robot has to always be in contact with the door, therefore the farthest position of the robot is limited by the work-space of its arm and the size of the door. This causes the the planning solution to be independent of the size of the room after a certain point, hence the size of the grid (of possible positions) comes out to be only dependent on the discretization factor.

V. RESULTS

In order to check the efficiency of the planner we looked into the number of expansions and the execution time of the planner, the lesser they are the better is the performance of the planner. We did three kinds of tests. In the first kind we analyzed the effect of discretization value i.e. the size of a grid cell (for position for the robot) on the number of state expansions and time execution of the planner. The results of this test are shown in Table I. In each of these tests the size of the door was 1 m, and length of each of the two links of manipulator being 0.5 m. It can be seen the results, that as the grid cell size decreases, number of number of cells increases and hence number of expansion increases too. Further with the increase in number of expansions, the planning time also increases. The planning time increases with as now more cells need to be checked (validated) and stored.

In the second kind we tested the effect of the door size on number of state expansions and planner's execution time. We varied the size of door from first, six times the width of the

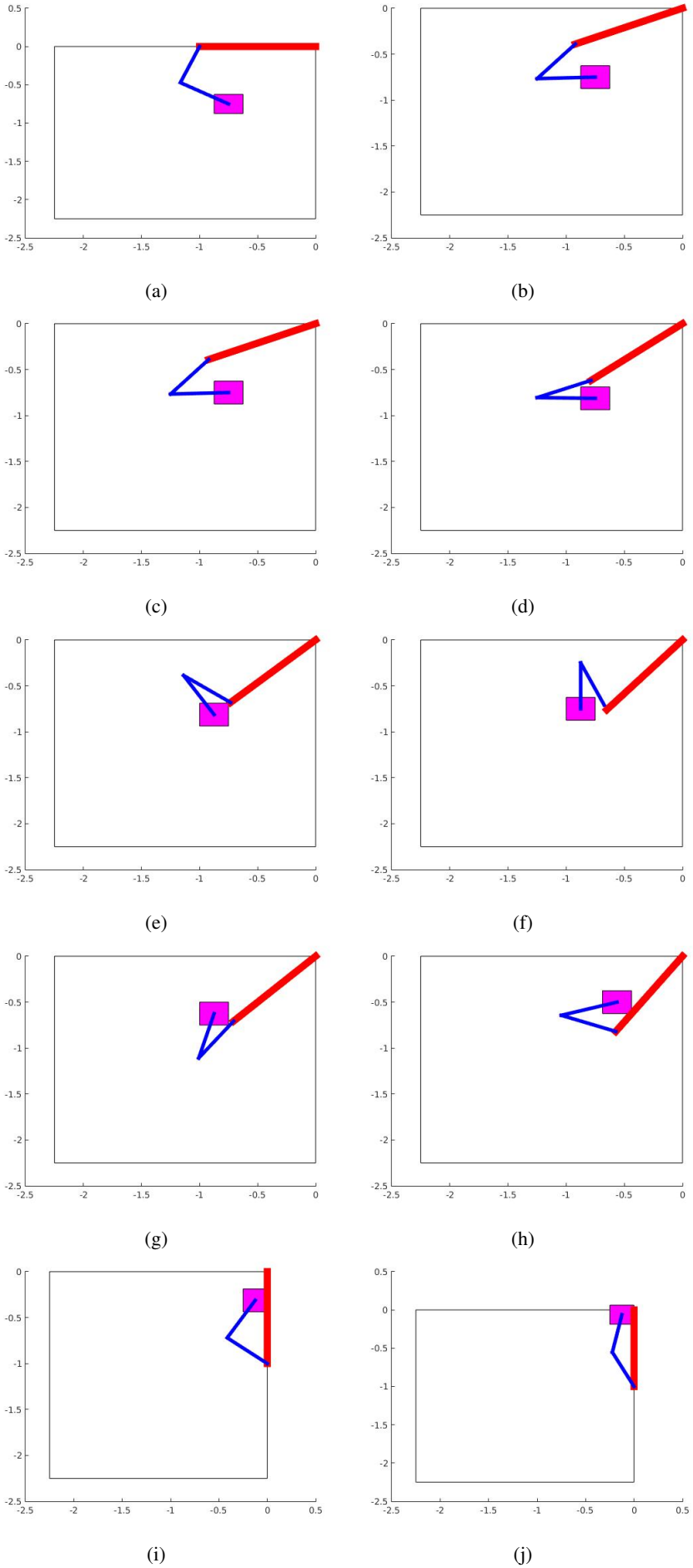

Fig. 12: The sequence of images showing the result of simulation. The robot starts from (a) and goes in order to end up at (j). The magenta square represents the robot, the red line represents the door, and the blue lines represent the links of the arm.

Grid Cell Size (m)	Grid Size	Number of Expansions	Planning Time (s)
0.0625	36X36	1014	0.1
0.125	16X24	162	0.02
0.25	8X12	28	0.002

TABLE I: Effects of varying grid cell size (discretization) on number of expansions and planning time

Door size / robot's width	Number of Expansions	Planning Time (s)
6	114	0.009
4	116	0.009
2	104	0.007

TABLE II: Effects of varying door size with respect to size of robot on number of expansions and planning time.

Workspace Radius / door size	Number of Expansions	Planning Time (s)
2	91	0.014
1.5	116	0.015
1	116	0.009

TABLE III: Effects of varying the work-space radius of the robotic arm with respect to the door size, on number of expansions and planning time

Parameters	Values
Robot's Width	0.25 m
Robot's Length	0.25 m
Door Size	1 m
Length of Link 1	0.5 m
Length of Link 2	0.5 m
Grid Cell Size	0.0625 m
Grid Size	36X36

TABLE IV: Values of different parameters used in the simulation.

robot then the four times the width of the robot and finally double to the width of the robot. The results are tabulated in Table II. During each of these run, the initial position of the robot was next to the door knob. As can be seen from the data in Table II, the bigger the door is, more are the number of expansions, and hence more is the execution time. One possible reason for this could be, since initial position of the robot is right next to the door knob, hence in order to open the door, the robot has to move more when the radius of sweeping circle of the door is larger.

In the third we vary the size of work-space of the robotic arm with respect to the door size, and noted the number of expansions and execution time of the planner in each run. The results of these runs are given Table III. In all the runs for this test, the size of the door was four times the width of the robot. It can be seen from the results that the number of expansions is lower when the work-space radius is large, which is quite intuitive since the large work-space will allow more number of states to be valid hence less rejection of expanded states. This further results in decrease in planning time.

Figure 12 shows the sequence of images from the simulator. The parameters for simulator are given in Table IV .

VI. CONCLUSION

We started with the review different approaches to solve the different steps of the overall door opening and navigation problem. We later presented our own simplified version of state of the art door-navigation planner, for pulling open and navigating through spring-loaded doors. We implemented the planner and tested it in various scenarios: different grid cell size, different door sizes, and different sizes of manipulator's work-space. Finally we presented the result. along with the sequence of images from simulator, simulating one of the scenario. The results clearly showed variation in the performance of the planner based on number of state expansions and execution time.

VII. FUTURE WORK

One of the immediate future work would be to improve the environmental awareness of the robot or sensor capabilities of the robot to avoid collisions in dynamic environment. Current methods cited in [1] make a lot of geometric assumptions of the environment. Our intention is to make a robust planner that plans combining informative heuristics at low dimension and arm-based coordination.

In [6], the future work would be to enable robots to handle large forces like a spring loaded or a revolving door.These dynamic door movements will create collision environment and hence it is required that our framework built for handling large forced doors are able to handle collision avoidance. Our future work in [8] would be to implement machine learning algorithms to observe their behaviors. We have a concept of experience graphs, where we try to memorize the trajectory and reuse the trajectory for the next planning query. Our idea is on similar bounds, of trying to observe behaviors. But, the question of it's feasibility in dynamic environments will bee unanswerable at the moment.

In [5] and [7], we can see the performance of many vision algorithms in detecting features of the door handle

or properties of doors. We can extend it further using depth data and multiple supervised and unsupervised algorithms. We are focusing on depth data, as it is invariant to lighting conditions, noise, illumination etc., Depth cloud doesn't work well for reflective surfaces which is huge setback. But, we see that, it can help detect the properties of door and obstacles that are in the configuration space of the robot. Algorithms like Image registration and Iterative Closet Point can give a good pose estimate of the door handles apart from object detection and modelling the environment for the robot to navigate. This also answers the "sensor coverage" that we were previously discussing about.

In [35], Multi-resolution lattice involves planning for the start configuration and end configuration at high dimensions and the rest of the intermediate planning in the low level. We feel planning in such lower dimensional manifolds for intermediate planning, that captures only the key mechanisms/properties of the environment is an god scope of futurework. Dominant states are the states where irrelevant or not-so-significant states are pruned and only the relevant states are retained and given as the input to the planner. Such planning will not compromise a lot on optimality but are very ideal for planning in high dimensional spaces - which is more close to the real world scenario. All the above methods have a great scope in future for research and extension.

VIII. Acknowledgements

We would like to thank Andrew Dornbush from Search Based Planning Laboratory for his technical support during the research.

References

[1] Gray, Steven, et al. "A single planner for a composite task of approaching, opening and navigating through non-spring and spring-loaded doors." Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013.

[2] Chitta, Sachin, Benjamin Cohen, and Maxim Likhachev. "Planning for autonomous door opening with a mobile manipulator." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[3] Nagatani, Keiji, and S. I. Yuta. "An experiment on opening-door-behavior by an autonomous mobile robot with a manipulator." Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. Vol. 2. IEEE, 1995.

[4] Niemeyer, Gnter, and J-JE Slotine. "A simple strategy for opening an unknown door." Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on. Vol. 2. IEEE, 1997.

[5] Klingbeil, Ellen, Ashutosh Saxena, and Andrew Y. Ng. "Learning to open new doors." Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, 2010.

[6] Jain, Advait, and Charles C. Kemp. "Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[7] Jauregi, E., B. Sierra, and E. Lazkano. Approaches to door identification for robot navigation. INTECH Open Access Publisher, 2010.

[8] Waarsing, B. J. W., Marnix Nuttin, and Hendrik Van Brussel. "Behaviour-based mobile manipulation: the opening of a door." Proc. of the Int. Workshop on Advances in Service Robotics. 2003.

[9] Dalibard, Sbastien, et al. "Manipulation of documented objects by a walking humanoid robot." 2010 10th IEEE-RAS International Conference on Humanoid Robots. IEEE, 2010.

[10] Cohen, Benjamin J., Sachin Chitta, and Maxim Likhachev. "Search-based planning for manipulation with motion primitives." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010

[11] Cohen, Benjamin J., Sachin Chitta, and Maxim Likhachev. "Search-based planning for manipulation with motion primitives." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[12] Cohen, Benjamin J., Sachin Chitta, and Maxim Likhachev. "Search-based planning for manipulation with motion primitives." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[13] Pivtoraiko, Mihail, and Alonzo Kelly. "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces." 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2005.

[14] Likhachev, Maxim, and Dave Ferguson. "Planning long dynamically feasible maneuvers for autonomous vehicles." The International Journal of Robotics Research 28.8 (2009): 933-945.

[15] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[16] Ye, Huashan, et al. "A new method based on hough transform for quick line and circle detection." 2015 8th International Conference on Biomedical Engineering and Informatics (BMEI). IEEE, 2015.

[17] Waarsing, B. J. W., Marnix Nuttin, and Hendrik Van Brussel. "A software framework for control of multi-sensor, multi-actuator systems." 11th Int. Conf. on Advanced Robotics. 2003.

[18] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi,and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In IEEE International Conference on Robotics and Automation, volume 2, pages 16201626. Citeseer, 2003

[19] Howard Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. In Principles of Robot Motion, pages 203215. Cambridge, MA: MIT Press,2005

[20] M. Likhachev, G. Gordon, and S.Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In Advances in Neural Information Processing Systems (NIPS) 16. Cambridge, MA: MIT Press, 200

[21] Nazareth Bedrossian Jeff M. Phillips and Lydia E. Kavraki. Guided expansive spaces trees: A search strategy for motion- and cost- constrained state spaces. In Proceedings of the International Conference on Robotics and Automation (ICRA), 2004.

[22] Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In IEEE International Conference on Robotics and Automation, 2008

[23] D. Anguelov, D. Koller, E. Parker, and S. Thrun, Detecting and modeling doors with mobile robots, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) , 2004

[24] C. Ott, B. Baeuml, C. Borst, and G. Hirzinger, Autonomous Opening of a Door with a Mobile Manipulator: A Case Study, in 6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV), 2007.

[25] A. Jain and C. C. Kemp, Pulling Open Novel Doors and Drawers with Equilibrium Point Control, in Humanoids , 2009

[26] A. Jain and C. C. Kemp, Behavior-based door opening with equilibrium point control, in RSS Workshop: Mobile Manipulation in Human Environments , 2009

[27] R. Shadmehr, Equilibrium point hypothesis. In (Ed.), M. AA, editor, Handbook of brain theory and neural networks, 20

[28] B. J. W. Waarsing, M. Nuttin, and H. Van Brus- sel, Behaviour-based mobile manipulation in- spired by the human example, Accepted for the International Conference on Robotics and Au- tomation, 2003.

[29] R. A. Brooks, Cambrian Intelligence, the early history of the new AI, The MIT Press, Cam- bridge Massachusetts, U.S.A, 1999.

[30] LE Kavraki, P. Svestka, J.C. Latombe, and MH Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Robotics and Automation, IEEE Transactions on , 12(4):566 580, 1996.

[31] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In IEEE International Conference on Robotics and Automation , volume 2, pages 16201626. Citeseer, 2003.

[32] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. Principles of Robot Motion: theory, algorithms, and implementation . MIT Press, 2005.
[33] J.C. Latombe. Robot Motion Planning . Kluwer Academic Publishers, 1991
[34] S. M. LaValle. Planning Algorithms . Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/
[35] Brock, Oliver, Jeff Trinkle, and Fabio Ramos. "Planning Long Dynamically-Feasible Maneuvers for Autonomous Vehicles." 214-221.