Motion planning under uncertainty and sensing limitations using exploration versus exploitation

Saumya Saxena[1], Matthew Travers[1] and Howie Choset[1]

Abstract— **We address the problem of planning under motion and sensing uncertainty using sensors that have inherent limitations such as limited effective range. Traditional optimization-based planning methods use local sensing information to guide the system toward regions of reliable measurements in order to gain information while planning to ensure high-level tasks are completed. In the case of sensors with limited sensing capabilities, this local sensing information is not readily available. Thus, we need methods that can efficiently explore the environment and find these regions of reliable measurements. We present a novel sampling-based planner (Particle Filter based Affine Quadratic Tree — PF-AQT) that explores the state space of the robot, composed of effective and ineffective sensing domains, and plans to reach a goal with minimal uncertainty. We then use the output trajectory from PF-AQT to initialize an optimization-based planner that finds a locally optimal trajectory that minimizes control effort and uncertainty. In doing so we reap the exploration benefits of sampling-based methods and exploitation benefits of optimization-based methods for dealing with uncertainty and limited sensing capabilities of the robot. Though generalizable, this work focuses on the problem of planning under uncertainty for robots with range and binary contact sensors. We demonstrate our results using two dynamical systems: double integrator model and a non-holonomic car-like robot.**

I. Introduction

Sensing modalities such as cameras, proximity sensors, LIDAR, contact sensors, etc., are popular localization tools, but have their inherent limitations such as having a limited range of effective use. To ensure successful completion of high-level tasks, robots should take actions to explore regions where reliable sensor measurements can be obtained for better localization. Traditional optimization based methods use local sensing information to drive the system to regions with reliable measurements. Sensors with limited effective range exhibit steep transitions from reliable to unreliable sensing domains, thus when a sensor is outside its range of effective use, the capability of such methods to estimate a direction of motion that can lead to reliable measurements is inhibited. We propose a novel method that overcomes this challenge by using a combination of sampling-based and optimization-based techniques, thus giving us the advantage of both exploration and exploitation. The proposed method is composed of two parts:

Exploration: We present the Particle Filter based Affine Quadratic Tree (PF-AQT) — a novel sampling-based planner that explores the state space of a robot, by effectively sampling in regions of reliable and unreliable measurements. Based on these samples, the planner grows a search tree giving emphasis to growth toward states sampled in regions of reliable measurement. Though generalizable, we focus on a robotic system having proximity sensors and binary contact sensors. Our method uses uncertainty as a measure to govern when the robot needs to come in contact to better localize itself. Thus, PF-AQT outputs a *feasible* trajectory that uses reliable measurements for localization while ensuring successful completion of high-level tasks.

Exploitation: Optimization-based motion planning methods start with an initial trajectory and iteratively find a locally optimal trajectory that minimizes the uncertainty growth and control effort along the path to a goal. This local optimality is obtained in the neighborhood of the initial trajectory thus making these methods very sensitive to the choice of trajectories for initialization. We need to initialize such methods with trajectories that explore our regions of interest. Thus, for systems with limited sensing range, we need to initialize these methods with trajectories that pass through regions of reliable measurements. The output trajectory from PF-AQT serves as a good initialization as it already incorporates the benefit of exploring the state space to reduce uncertainty. In this work, we leverage the optimization-based motion planning under uncertainty algorithm developed by [1] and modify it to handle contacts.

In the remainder of this paper, Section II presents the prior work that we leverage for the development of our algorithm, Section III presents the details of the PF-AQT planner, Section IV presents the use of an optimal belief space planner which we modify to handle contacts and in Section V we demonstrate the performance of our algorithm on two systems: (1) the double integrator (2) nonholonomic car-like robot.

II. Prior work

This work builds on ideas in planning under uncertainty using sampling-based and optimization-based motion planning approaches. Some of the relevant prior work has been discussed to provide context to the work presented in the further sections.

Sampling-based planning methods, like Rapidly exploring Random Tree (RRT), use incremental sampling to grow a tree in the state space and compute motion plans that take the robot from a start to a goal state while avoiding obstacles. Typically, for kinematic systems, the metric defining the distance between two nodes of a RRT is usually assumed as the Euclidean distance [2], [3], but for dynamic systems finding this metric is non-trivial. Several algorithms have

[1]Carnegie Mellon University, Pittsburgh, PA, USA (saumyas, mtravers, choset)@andrew.cmu.edu

been proposed [4], [5], [6], [7], [8] that use optimal control methods to derive this distance metric. Our PF-AQT planner leverages the AQR-based heuristic [5] because of its capability to drive a dynamical system towards a non-fixed point but uses it to plan in the belief space.

A planner that plans in the space of probability distributions over the state of the robot is called a belief space planner (BSP). Sampling-based belief space planning methods have been developed [9], [10], [11] that find trajectories stabilized with linear estimators and controllers, and search over the space of these trajectories to find a low uncertainty path to the goal. Sampling-based methods that plan for contacts to reduce uncertainty [12], [13] have also been developed but they only deal with kinematic systems and use randomized selection of actions. Our PF-AQT planner, however, plans in the continuous state and action space of kinodynamic systems and uses uncertainty as a measure to govern contact.

Optimal belief space planning (BSP) methods [14], [15], [1] deal with motion and sensing noise by using a combination of optimal estimators, like EKF [16], and optimal controllers, like linear quadratic regulator, and find a trajectory that minimizes a given cost function (a trade-off between control effort and uncertainty). These algorithms use gradient information from measurement models to drive the system near regions of low uncertainty measurements. In the absence of these gradients as in the case of limited range sensors, these algorithms fail to find a trajectory that takes advantage of reliable meaurements. Methods to deal with this problem have been proposed [17], [18] that smooth out sensing discontinuities and sequentially dilute this assumption over subsequent iterations. These assumptions are not valid for sensing modalities such as binary contact sensors. Our algorithm on the other hand is robust to all sensing modalities including binary contact sensing. Additionally, dealing with contacts in optimal BSP scenarios requires us to include deterministic contact dynamics [19], [20] and perfect zero-variance pseudo-measurement [21] upon contact in the belief dynamics of the robot.

III. EXPLORATION: SAMPLING-BASED MOTION PLANNING

We introduce the sampling-based planner PF-AQT that explores the state space of the robot by growing a search tree and uses a particle filter to propagate belief across free and contact state motion.

A. Problem definition

The problem is to find a trajectory in a known environment that begins at a start state and reaches the goal state, while avoiding obstacles, satisfying the system dynamics and minimizing a given cost function $J = \mathbf{g}(\mathbf{b}_t, \mathbf{u}_t)$. The details of the cost function are discussed later sections.

Our system of interest is nonlinear with motion and sensor uncertainty and is partially observable. The stochastic nonlinear dynamics and observation model of such systems can be written in discrete time as:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t)$$
$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{n}_t)$$

where $\mathbf{x}_t$ is the state vector, $\mathbf{u}_t$ is the input vector, $\mathbf{m}_t$ is the motion noise and $\mathbf{n}_t$ the measurement noise. The motion and measurement noise can be state or control dependent. The contact dynamics are assumed inelastic, i.e. the system comes to rest on contact with an obstacle. The initial *belief* of the robot is specified as a Gaussian distribution: $\mathbf{b}_0 = \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$.

B. Belief Propagation

Belief evolves using a particle filter. Belief at every time step t is represented by a set of random samples taken from a distribution $\mathbf{b}_t$, which can be written as, $S_t = \{\mathbf{x}_t^{[1]}, \mathbf{x}_t^{[2]}, \ldots, \mathbf{x}_t^{[N]}\}$. These random states are also called particles, thus the belief is represented by a set of particles. Each particle evolves using a separate control strategy specified by the AQR controller [5].

We model a robot navigating an environment with onboard proximity sensor and binary contact sensor. A proximity sensor gets measurements only when the robot is within a finite distance to an obstacle. On contact, the binary contact sensor, based on the robot's current belief, gives with high certainty the location of the robot along the contact normal of the known obstacle.

C. Particle filter based affine quadratic tree — PF-AQT

The planning algorithm is inspired by RRT* [3]. We grow a tree ($\mathcal{T}$) in belief space, each node *n* having the following set of parameters: 1) Particle set *S* representing the belief, 2) Total *Cost* of the sequence of trajectories connecting the start node to *n* (definition of this cost is given in later sections), and 3) Pointer to the *Parent* node, connecting node *n* to the rest of the tree.

$$n = (S, Cost, Parent)$$

The process of tree expansion for our PF-AQT algorithm can be summarized as follows. The algorithm begins by initializing the tree with the start node. The start node is obtained by taking *N* random samples from the initial belief $\mathbf{b}_0 = \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$. Samples are then generated at random from the state space and the tree is expanded. One such tree expansion step is as follows. Once a random sample $\mathbf{x}_{rand}$ is obtained, the tree is spanned to find the node nearest to this sample based on the AQR heuristic [5]. After finding the nearest node $n_{nearest}$, we use the AQR controller to reach $\mathbf{x}_{rand}$ starting from the *mean* of the particle set of the nearest node. It is important to note that we are distinguishing between a node n, the particle set associated with a node $n.S$ and a state $\mathbf{x}$. We then look for a parent node for the new state $\mathbf{x}_{new}$ in the tree ($\mathcal{T}$). The criteria for selecting the parent node is given in Section III-C.4. The parent node is then extended to $\mathbf{x}_{new}$ using a particle filter. The trajectory is checked for collision at every time step *dt* where *dt* is the time discretization along each trajectory. If a feasible

trajectory is found, the node at the end of the trajectory gets added to the tree. The various steps of the algorithm are explained in detail below and a high level description can be found in Algorithm 1.

Algorithm 1: PF-AQT

$\mathcal{T} \leftarrow InitializeTree()$
for $i \leftarrow 1$ *to* M **do**
  $\mathbf{x}_{rand} \leftarrow$ RandomSample()
  $n_{nearest} \leftarrow$ AQRNearest($\mathcal{T}, \mathbf{x}_{rand}$)
  $\mathbf{x}_{new} \leftarrow$ AQRSteer($n_{nearest}, \mathbf{x}_{rand}$)
  $n_{Parent} \leftarrow$ ChooseParent($\mathcal{T}, \mathbf{x}_{new}$)
  if $Cov(n_{Parent}.S) < CovThresh$ **then**
    $S_{new} \leftarrow PFExtend(n_{Parent}, \mathbf{x}_{new})$;
  else
    $S_{new} \leftarrow PFContact(n_{Parent})$;
  end
  if *CollisionFree*(n_{parent}, s_{new}) **then**
    $n_{new}.Cost \leftarrow n_{Parent}.Cost +$
      $AQRdist(n_{Parent}, S_{new})$
    $n_{new}.Parent \leftarrow n_{Parent}$
    $n_{new}.S \leftarrow S_{new}$;
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{n_{new}\}$
  end
end

1) RandomSample(): Samples are generated independently from a uniform distribution whose dimension is equal to the dimension of the state space.

2) AQRNearest ($\mathcal{T}$, $\mathbf{x}_{rand}$): As in the case of RRT, the planner spans the tree to find the node which is "nearest" to the random sample $\mathbf{x}_{rand}$. The performance of an RRT depends highly on the accuracy with which the chosen distance heuristic represents the cost to travel from one node to the other [2]. We choose the AQR heuristic developed in [5], wherein the authors find an optimal control policy and cost function for driving a linearized affine system to a non-fixed point. Since we sample in the full state of the robot, $\mathbf{x}_{rand}$ will have a non-zero velocity. Thus, we use the AQR cost function as the heuristic to find the nearest node to $\mathbf{x}_{rand}$ in the tree. We refer to this cost function as AQRdist($\mathbf{x}_1, \mathbf{x}_2$) in our algorithm.

We evaluate the cost to travel from the mean of the particle set of each node in the tree to the random state. We find $n_{nearest}$ as the node in the tree closest to $\mathbf{x}_{rand}$ based on the above cost function. Note that AQRdist is based on a linearization about the target location and therefore is not a true measure of distance; for nodes that are close together, this approximation is sufficient but for nodes that are far apart, which is the case early in the planning phase, this approximation may not hold. As the tree grows denser, we no longer have to worry about this.

3) AQRSteer ($n_{nearest}$,$\mathbf{x}_{rand}$): Starting from the state $\mathbf{x}_{nearest}$ (mean of the particle set $n_{nearest}.s$), we use the AQR controller to generate a trajectory from $\mathbf{x}_{nearest}$ to as close as possible to $\mathbf{x}_{rand}$. The end of the trajectory generated, is our new state $\mathbf{x}_{new}$. In this function, we propagate the mean of the particle set to get an estimate of the position of the new node. The actual node propagation is done using a particle filter as explained in further subsections.

4) ChooseParent ($\mathcal{T}$,$\mathbf{x}_{new}$): As in the case of RRT*, we look for a parent node for $\mathbf{x}_{new}$. The parent node is a node that connects $\mathbf{x}_{new}$ to the tree such that the cost to travel from the start node to $\mathbf{x}_{new}$ is minimum. We have modified this criteria to include uncertainty of the parent node in the cost. This has been done so that the planner chooses a node as a parent not only if its control cost is low but also if its uncertainty is low. This allows for nodes in the discontinuous measurement domains to grow.

The criteria for choosing a parent is as follows – let $\mathbf{x}'$ represent the mean of the particle set of a node n'. For each node n' in the tree, we check if it is in the neighborhood of $\mathbf{x}_{new}$. The neighborhood is defined as AQRdist($\mathbf{x}', \mathbf{x}_{new}$) $< min(\eta, \gamma(\frac{\log d}{d})^{1/n})$ where d is the size of the tree and η and γ are constants. For a thorough discussion on this neighbourhood bound the reader may refer to [2]. From among these *near* nodes we choose the node with the lowest cost. The cost is defined as

$$\begin{aligned}\text{Cost}_{near} = W_{dist} * [n'.Cost + AQRdist(\mathbf{x}', \mathbf{x}_{new})] \\ + W_{cov} * Cov(n'.s)\end{aligned} \quad (1)$$

Where, $Cov(n'.s)$ represents the covariance of a particle set. Here, we can adjust the weights W_{dist} and W_{cov} in the cost function (1) to find a balance between expanding a node that is closer versus expanding a node with low uncertainty. The node with the lowest cost is chosen as the parent node n_{parent}. If no parent is found the current sample is rejected and we resample.

5) ChooseAction: After having found the parent node n_{parent}, we evaluate if we should find a trajectory connecting n_{parent} and $\mathbf{x}_{new}$ or if n_{parent} should move towards an obstacle to come in contact. If the covariance of the particle set of n_{parent} is greater than a threshold value $CovThresh$ (specified by the user), it should come in contact with the nearest obstacle to reduce its uncertainty, else it is connected to $\mathbf{x}_{new}$. This function ensures that nodes with high uncertainty are not propagated forward towards the goal. Such nodes come in contact with a wall to reduce uncertainty.

6) PFExtend(n_{parent}, $\mathbf{x}_{new}$): This function finds a trajectory from n_{parent} to $\mathbf{x}_{new}$ using a particle filter for belief propagation and checks for collisions at every time step.

To drive the system from n_{parent} to $\mathbf{x}_{new}$, we linearize the system about $\mathbf{x}_{new}$ to evaluate the AQR feedback controller. As in a particle filter, each particle is a different state, representing an estimate of the current location of the robot. Thus, the AQR feedback controller is evaluated and used as the motion model for each particle independently. The motion model for each particle *m* at time step *t* is represented as [16]:

$$\mathbf{x}_t^{[m]} \sim p(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1}^{[m]})$$

Next we take a measurement at each time step and find the probability of the measurement $\mathbf{z}_t$ under each particle $\mathbf{x}_t^{[m]}$, that is, $w_t^{[m]} = p(\mathbf{z}_t|\mathbf{x}_t^{[m]})$

We use the above importance factor to resample the particle set. The term $p(\mathbf{z}_t|\mathbf{x}_t^{[m]})$ includes in it the measurement noise. The higher the uncertainty in the measurement, the larger the spread of the Gaussian, resulting in more uniformly distributed weights. Thus, the particles get resampled uniformly when the measurement noise is high which leads to lower reduction in uncertainty and vice-versa. The node is propagated forward towards $\mathbf{x}_{new}$ and a new particle set S_{new} is generated. If no undesired collision is detected along the path, S_{new} is added as a new node (n_{new}) to the tree with n_{parent} as its parent. $Cost$ of n_{new} is defined as:

$$n_{new}.Cost = n_{parent}.Cost + AQRdist(mean(n_{parent}.S), mean(S_{new})) \quad (2)$$

where AQRdist represents the AQR controller cost of travelling from the mean of the particle set of the parent node n_{parent} to the mean of the particle set S_{new}.

7) PFContact (n_{parent}): After a node is selected to come in contact, the planner searches in the environment for the nearest contact location. The node is then propagated in the direction of nearest contact using a particle filter.

Belief propagation for the node coming in contact with an obstacle takes place using a particle filter in broadly the same way as described in the previous subsection. The differences are in the controller used for the motion model and in the measurement on coming in contact. Instead of an AQR controller, we use infinite-horizon LQR controller to drive each particle to come in contact with the nearest wall. We use this controller because it is very effective in driving the system to a fixed point (zero velocity) without specification of the time horizon. As we assume that we come to rest upon collision, this controller is very effective in slowing down the system as it tries to come in contact. Upon coming in contact, we assume a perfect zero-variance pseudo-measurement [21] in the direction of contact normal, thus the node loses all uncertainty in one direction upon contact. The new node n_{new} is then added to the tree.

8) GoalReached: Every time a new node n_{new} is added to the tree, it is checked for two criteria. First, we check if n_{new} is in a given neighborhood of the goal. Second, we check if the uncertainty of n_{new} is less than a pre-specified value *GoalThresh*. Once the goal is reached, we output a feasible trajectory connecting the start and the goal state.

In the absence of binary contact sensors, the *PFContact* step can be ignored and the algorithm works well for other traditional limited field of view sensors.

IV. EXPLOITATION: OPTIMAL BELIEF SPACE PLANNING

We now use the output of our PF-AQT planner and exploit the space of trajectories in its vicinity to find a locally optimal trajectory that minimizes a given cost function. We build on the optimal BSP approach developed in [1].

A. Belief propagation

Belief evolves using an extended Kalman filter that assumes Gaussian motion and sensor noise, and is applicable to systems with nonlinear dynamics. Measurement model is same as used for PF-AQT. Starting with an initial nominal trajectory, we approximate the value function at each time step *t* in the trajectory and optimize it to find the optimal policy. We *forward integrate* this policy to find the nominal trajectory for the next iteration. We then find the value function for this new nominal trajectory and the process continues until convergence to a locally optimal trajectory. For the belief update steps and evaluation of the optimal policy the reader may refer to [1].

B. Forward integrate

The control policy is forward integrated using the deterministic belief dynamics [1]:

$$\begin{aligned} \bar{\mathbf{b}}_0^{i+1} &= \mathbf{b}_0 \\ \bar{\mathbf{u}}_t^{i+1} &= L_t^i(\bar{\mathbf{b}}_t^{i+1} - \bar{\mathbf{b}}_t^i) + \mathbf{l}_t^i + \bar{\mathbf{u}}_t^i \\ \bar{\mathbf{b}}_{t+1}^{i+1} &= \mathbf{g}[\bar{\mathbf{b}}_t^{i+1}, \bar{\mathbf{u}}_t^{i+1}] \end{aligned} \quad (3)$$

where $\mathbf{b}_0$ is the belief at the start position. The superscripts represent the iteration, thus, given a nominal trajectory at the i^{th} iteration we find the control trajectory and forward integrate it to find the nominal trajectory for the $(i+1)^{\text{th}}$ iteration. The nominal trajectory for the first iteration is the output trajectory from PF-AQT, which gives the benefit of exploration.

As evident from the belief dynamics, this formulation does not take into account contact with the environment. Thus, we explicitly incorporate linear complimentarity constraints in the forward simulation of dynamics at every time step. Let us write the mean of the belief as $\hat{\mathbf{x}}_t = (\mathbf{q}_t, \dot{\mathbf{q}}_t)$. The 2D contact dynamics can be written as a linear complimentarity problem in the following manner [19], [20]:

$$\mathbf{q}_t - \mathbf{q}_{t+1} + \dot{\mathbf{q}}_{t+1}\, dt = 0 \quad (4)$$

$$\dot{\mathbf{q}}_{t+1} - \dot{\mathbf{q}}_t - (\mathbf{u}_{t+1} + \boldsymbol{\lambda}_{t+1})\, dt = 0 \quad (5)$$

$$\phi(q_t), \lambda_{t,y}, \lambda_{t,x}^+, \lambda_{t,x}^-, \gamma_t \geq 0 \quad (6)$$

$$\mu\lambda_{t,y} - \lambda_{t,x}^+ - \lambda_{t,x}^- \geq 0 \quad (7)$$

$$\gamma_t \pm \Psi(q_t, \dot{q}_t) \geq 0 \quad (8)$$

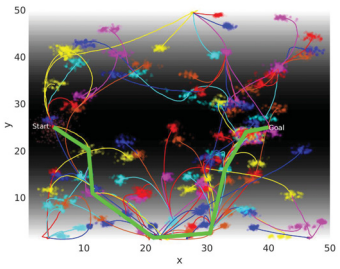
$$\phi(q_t)^T \lambda_{t,y} = 0 \quad (9)$$

$$(\mu\lambda_{t,y} - \lambda_{t,x}^+ - \lambda_{t,x}^-)\gamma_t = 0 \quad (10)$$

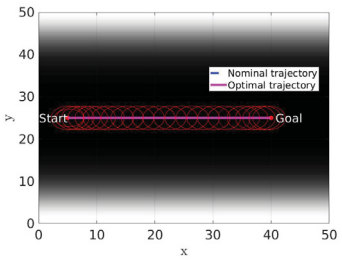
$$(\gamma_t + \Psi(q_k, \dot{q}_t))^T \lambda_{t,x}^+ = 0 \quad (11)$$

$$(\gamma_t - \Psi(q_t, \dot{q}_t))^T \lambda_{t,x}^- = 0 \quad (12)$$

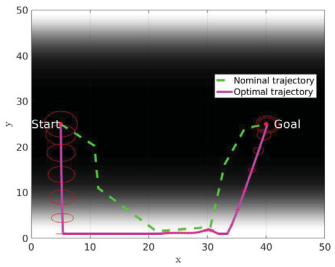
$\boldsymbol{\lambda}_t = [(\lambda_{t,x}^+ - \lambda_{t,x}^-)\,,\ \lambda_{t,y}]$ represents the contact force. The above equations are written assuming that the normal contact force $\lambda_{t,y}$ acts in the y-direction and the tangential force has been split into its +x and -x components, $\lambda_{t,x}^+$ and $\lambda_{t,x}^-$ respectively. (4) & (5) represent the discrete time dynamics for forward propagation. Slack variable γ_t represents the magnitude of relative velocity of bodies coming in contact. $\phi(q_t)$ is the distance from contact location. (7) enforces



(a) Final trajectory (green) generated using the PF-AQT planner



(b) Locally optimal solution (magenta) using iLQG BSP method. Nominal trajectory is generated using deterministic trajectory optimization (Straight line joining start and goal).



(c) Locally optimal solution (magenta) using iLQG BSP method. Nominal trajectory is the output trajectory from PF-AQT planner (dotted green).

Fig. 1: A robot with double integrator model moving in a 2D environment with walls at y = 1 and y = 50. (a) PF-AQT tree with sub-trajectories and associated beliefs represented as particle sets at the end of each sub-trajectory (shown in yellow, red, blue, cyan and magenta colors). Nominal trajectory (green) exploits reliable measurements to better localize itself before moving towards the goal. (b) Locally optimal solution (magenta) using a nominal trajectory generated using a deterministic trajectory optimization method (straight line connecting start and goal). iLQG planner converges to a locally optimal trajectory identical to this nominal trajectory. (c) Locally optimal solution (magenta) using the output trajectory from PF-AQT as the nominal trajectory (dotted green). Robot travels near the wall to get better measurements from proximity sensors and contact sensors before moving towards the goal with low uncertainty.

friction cone constraints. $\Psi(q_t, \dot{q}_t)$ is the relative tangential velocity. (9) represents positive contact force upon contact and no contact force at a distance. (10) ensures that the contact forces lie on the edge of the friction cone when the contact is sliding.

We satisfy the constraints (4) - (12) at every time step and find the control policy and state trajectory that satisfies the free space as well as contact dynamics of the system. The uncertainty update upon contact follows the procedure developed by [21]. We apply a zero-variance pseudo-measurement update at the contact location to enforce the equality constraints. The measurement variance is zero in the direction of the contact normal.

V. Results

We apply our approach in simulation to two systems: (i) double integrator with linear dynamics, (ii) nonholonomic car-like robot with nonlinear dynamics. The motion planning scenario involves motion and measurement uncertainties and contact with nearby walls.

For each dynamic system, we first present results for the PF-AQT planner. The results for the optimal belief space planner are then presented in two scenarios. First, we seed the planner with a nominal trajectory generated using direct collocation without considering uncertainty in the system. Second, we seed the planner with the nominal trajectory obtained from our PF-AQT planner. The results are compared for these two scenarios.

The cost function for the iLQG based belief space planning algorithm is taken as a cost that minimizes the control effort and uncertainty along the trajectory and penalizes distance from the goal at the final time T:

$$c_T(\mathbf{b}_T) = \hat{\mathbf{x}}_t^T Q_T \hat{\mathbf{x}}_t + trace[\sqrt{\Sigma_T}\, Q_T \sqrt{\Sigma_T}]$$
$$c_t(\mathbf{b}_t, \mathbf{u}_t) = \mathbf{u}_t^T R_t \mathbf{u}_t + trace[\sqrt{\Sigma_t}\, Q_t \sqrt{\Sigma_t}]$$

A. Double integrator model

A robot with double integrator dynamics (13) moving in a 2D environment is considered with noisy motion and measurement models. The state is represented as $\mathbf{q}$ = (x, y, $\dot{x}$, $\dot{y}$), where (x, y) is the position of the robot in 2D space and ($\dot{x}$, $\dot{y}$) are the velocities. The acceleration of the robot is directly controlled $\mathbf{u}$ = ($\ddot{x}$, $\ddot{y}$).

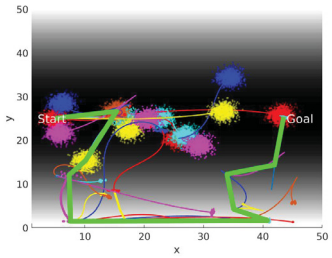
$$\mathbf{q}_{t+1} = \mathbf{q}_t + \begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} dt + M\mathbf{m} \quad , \quad \mathbf{m} \sim \mathcal{N}(0, I) \tag{13}$$

where M scales the motion noise. The state is partially observable, as only the position of the robot is sensed. The measurement model comprises of a limited field of view proximity sensor and a binary contact sensor. Thus, measurements are obtained only near the walls/obstacles, and most of the free space has no measurements.

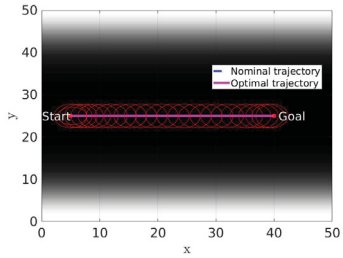
$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix} + N\mathbf{n} \quad , \quad \mathbf{n} \sim \mathcal{N}(0, I) \tag{14}$$

where N scales the measurement noise based on the robot's distance to the wall. Contact detection using binary contact sensors allows us to have perfect measurements along the contact normal when contact occurs.

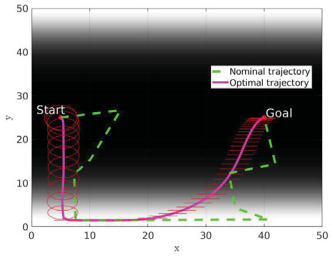
The robot navigates in a 2D environment with walls at $y = 1$ and $y = 50$, thus, enclosing the state space at the top and bottom of the Figures 1a, 1b, 1c. The initial state is at



(a) Final trajectory (green) generated using the PF-AQT planner



(b) Locally optimal solution (magenta) using iLQG BSP method. Nominal trajectory is generated using deterministic trajectory optimization (Straight line joining start and goal).



(c) Locally optimal solution (magenta) using iLQG BSP method. Nominal trajectory is the output trajectory from PF-AQT planner (dotted green).

Fig. 2: A non-holonomic car-like robot moving in a 2D environment with walls at y = 1 and y = 50. (a) PF-AQT tree with sub-trajectories and associated beliefs represented as particle sets at the end of each sub-trajectory (shown in yellow, red, blue, cyan and magenta colors). Nominal trajectory (green) exploits reliable measurements to better localize itself before moving towards the goal. (b) Locally optimal solution (magenta) using a nominal trajectory generated using a deterministic trajectory optimization method (straight line connecting start and goal). iLQG planner converges to a locally optimal trajectory identical to this nominal trajectory. (c) Locally optimal solution (magenta) using the output trajectory from PF-AQTC as the nominal trajectory (dotted green). Robot travels near the wall to get better measurements from proximity sensors and contact sensors before moving towards the goal with low uncertainty.

the left center of the figure at $\mathbf{q} = (5, 25, 0, 0)$ and the goal state is at the right center of the figure at $\mathbf{q} = (40, 25, 0, 0)$. Control cost matrix of $Q_t = I$ is used for the AQR controller. For the infinite horizon controller we define state and control cost matrices as $Q_t = I$ and $R_t = 10I$.

Figure 1a shows the evolution of the PF-AQT tree with the various branches (sub-trajectories) plotted with their associated beliefs represented as particle sets at the end of each sub-trajectory (shown in yellow, red, blue, cyan and magenta colors). The sampling-based planner leads the robot close to the wall to get measurements from the proximity sensors and come in contact, in order to better localize itself, and then move towards the goal. The robot starts with an initial covariance of 2 units at the start position and reaches the goal with a covariance of 0.26 units. The goal reaching covariance threshold *GoalThresh* was set at 0.3 units.

Figure 1b shows results for the iLQG based optimal belief space planner (which we modified to handle contacts). The planner uses the result from direct collocation, as its nominal trajectory. For a double integrator model this nominal trajectory is a straight line joining start and goal states. The nominal trajectory is shown in blue and the optimal trajectory is shown in magenta. We observe that the iLQG planner converges to a locally optimal trajectory identical to the nominal trajectory. This is because the robot is far from areas where it can get reliable measurements (the walls). The local optima is at the nominal trajectory where the control effort is lowest. No gradients are detected to give the optimizer a direction for uncertainty reduction. Thus, in the absence of a good nominal trajectory, that can drive the system to areas of discontinuous measurements, it is very difficult for locally optimal planners to explore such regions.

Figure 1c shows results for the iLQG based belief space planner that uses the output trajectory from our PF-AQT planner as the nominal trajectory. The nominal trajectory is shown in dotted green and the optimal trajectory is shown in magenta. We observe that the planner leads the robot close to the wall and slides along it, to get reliable sensing, and then moves towards the goal. Thus, it is evident that the nominal trajectory plays a big role in allowing the system to find a solution in regions where low uncertainty measurements are available. The algorithm converges to a locally optimal trajectory in 6 iterations. The expected cost of the initial nominal trajectory is 95.1 units and the cost of the locally optimal trajectory obtained upon convergence is 4.1 units.

B. Non-holonomic car-like robot

A car-like robot with nonholonomic constraints navigating through a 2D environment is considered with noisy motion and measurement models. The state is partially observable, only the position of the robot is sensed. The state is represented as $\mathbf{x} = (x,\ y,\ \theta,\ v)$, where $(x,\ y)$ is the position of the robot in 2D space, θ is the orientation and v is the speed. The control input consists of $\mathbf{u} = (a,\ \dot{\theta})$ where a is the acceleration and $\dot{\theta} = \frac{v}{L}\tan\phi$, ϕ is the steering angle. Thus, the car dynamics can be written as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \dot{\theta} \\ a \end{bmatrix} dt + M\mathbf{m} \quad , \quad \mathbf{m} \sim \mathcal{N}(0,\ I)$$

where M scales the motion noise. Measurement model is the same as in the case of double integrator model (14). Robot navigates in a 2D environment with walls at $y = 1$ and $y = 50$, thus, enclosing the state space at the top and bottom of the figure. The initial state is at the left center of the figure at $\mathbf{q} = (5, 25, 0, 0)$ and the goal state is at the right

center of the figure at $\mathbf{q} = (40, 25, 0, 0)$. Control cost matrix of $Q_t = I$ is used for the AQR controller. For the infinite horizon controller we define state and control cost matrices as $Q_t = I$ and $R_t = 10I$.

Figure 2a shows the evolution of the PF-AQT tree for the car-like robot with the various sub-trajectories plotted with associated beliefs represented as particle sets at the end of each sub-trajectory (shown in yellow, red, blue, cyan and magenta colors). The sampling-based planner leads the robot close to the wall to get measurements from the proximity sensors and comes in contact, in order to better localize itself, and then move towards the goal. The robot starts with an initial covariance of 2 units at the start position and reaches the goal with a covariance of 0.19 units. The goal reaching threshold was set at 0.3 units.

Figure 2b shows results for the iLQG based optimal belief space planner. The planner uses the result from direct collocation, as its nominal trajectory. For a car-like robot with initial pose (5, 25, 0, 0), that is, pointing right (towards the goal), this nominal trajectory is a straight line joining start and goal states. The nominal trajectory is shown in blue and the optimal trajectory is shown in magenta. We observe that the iLQG planner converges to a trajectory identical to the nominal trajectory. This is because the robot is far from areas with reliable measurements, thus the gradients in the optimization cannot drive it towards those regions and the planner converges to the local minima. In the absence of a good nominal trajectory that can drive the system to areas of reliable measurements, it is very difficult for the locally optimal planner to explore such regions.

Figure 2c shows the locally optimal trajectory (magenta) in the vicinity of the trajectory provided by the PF-AQT algorithm shown in dotted green. We observe that the planner leads the robot close to the wall and slides along it, to get reliable measurements, and then moves towards the goal. Thus, the nominal trajectory plays a great role in allowing the system to find a solution in the regions where reliable measurements are available. The algorithm converges to a locally optimal trajectory in 10 iterations. The expected cost of the initial nominal trajectory is 35.1 units and the cost of the locally optimal trajectory obtained upon convergence is 2.8 units.

VI. Conclusion

Our work has been completed in two parts. In the first part, through the development of the PF-AQT algorithm, we have provided a means for successfully "exploring" the state space and using reliable measurements from the environment for better localization along the path to goal attainment. In the second part, the output from PF-AQT was used to initialize an optimal BSP method that "exploited" the space of trajectories in the vicinity of the initial trajectory to find a locally optimal trajectory. Our results have been demonstrated using the double integrator model and non-holonomic car-like robot.

References

[1] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, p. 2, 2010.

[4] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5054–5061.

[5] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5021–5028.

[6] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7681–7687.

[7] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2429–2436.

[8] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," 2012.

[9] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[10] L. Jaillet, J. Hoffman, J. Van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg, "Eg-rrt: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2646–2652.

[11] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 723–730.

[12] A. Sieverling, C. Eppner, F. Wolff, and O. Brock, "Interleaving motion in contact and in free space for planning under uncertainty," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 4011–4073.

[13] E. Páll, A. Sieverling, and O. Brock, "Contingent contact-based motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6615–6621.

[14] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," 2010.

[15] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using differential dynamic programming in belief space," in *Robotics Research*. Springer, 2017, pp. 473–490.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[17] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, "Gaussian belief space planning with discontinuities in sensing domains," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6483–6490.

[18] K. Hausman, G. Kahn, S. Patil, J. Müller, K. Goldberg, P. Abbeel, and G. S. Sukhatme, "Cooperative occlusion-aware multi-robot target tracking using optimization," *rll. berkeley. edu*, 2015.

[19] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[20] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 527–542.

[21] S. Tully, A. Bajo, G. Kantor, H. Choset, and N. Simaan, "Constrained filtering with contact detection data for the localization and registration of continuum robots in flexible environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3388–3394.