Multi-World Motion Planning

Bobby Davis, Nicholas Sohre, and Stephen J. Guy

***Abstract*— While predictive planning approaches have had broad success in robot navigation, most implementations of them have important limitations. In particular, they plan around a single predicted outcome (plus uncertainty) or the union of several likely outcomes. Here, we introduce the Multi-World Motion Planning problem, where a robot plans around each likely outcome separately, while maintaining a safe trajectory until it can determine which of the predictions actually happened. We introduce a method both for split-detection (understanding when multiple plans can be created) and split-planning (planning path over these separate contingencies). We apply our techniques to simulated robots with both holonomic and non-holonomic dynamics in a variety of scenarios. We show our resulting approach can produce substantially safer and shorter trajectories than traditional motion planning in critical situations that commonly arise in tasks such as navigating in human environments.**

I. Introduction

Predictive planning approaches have revolutionized the ability of robots to move through complex environments with dynamic obstacles. By allowing robots to anticipate what outcomes are likely to happen in the world around them, we can have robots take more confident paths that reach their goals quickly and safely [1]. However, there are often cases where no single predicted path can successfully capture the entire range of likely outcomes an dynamic obstacle may take in the future. This is especially true in scenarios involving humans moving through constrained environments with branching paths and obstacles, such as commonly found in building interiors. For example, consider a pole in the middle of a hallway, a pedestrian must walk either left or right (but not both!) to avoid the pole. Our goal is to exploit this latent structure to find new robot trajectories not otherwise possible.

In order to capture the branching nature inherent in some prediction tasks we introduce a new formalization called Multi-World Motion Planning. Here, the goal is to compute the best robot trajectory possible given a discrete set of likely predictions for nearby obstacles, *of which only one instance will actually be encountered*. Existing techniques for motion planning typically account for this prediction uncertainty using two broad strategies. One, is by using a maximum likelihood assumption to compute the single prediction which is most representative of the likely future outcomes. The other, is to plan a conservative path around the entire range of likely outcomes. Here, we propose a third approach of **Branched Planning**, where the robot plans separately for each future eventuality, while maintaining a safe, flexible trajectory until it has enough information to determine which eventuality will actually occur.

To this end, we propose three main contributions.

- *Multi-World Motion Planning problem*: We introduce the Multi-World Motion Planning problem, providing both a formal problem description and a discussion of its applicability to various planning problems.
- *Split Detection*: We propose and analyze a new method for analyzing predicted obstacle trajectories to automatically determine when they can be safely decomposed into disjoint sets of predictions.
- *Branch Planning*: We propose a space-time planning approach which extends RRT and iLQR to account for the branched predictions which result from split detection.

II. Related Work

A wide variety of previous work has proposed various techniques for robot planning among dynamic obstacles. When obstacles paths are known, planning over configuration space-time graphs using Probabilistic Roadmaps (PRM) [2] or Rapidly-exploring Random Trees (RRT) [3] allows a robot to find collision-free trajectories that can be optimal [4] even under kinematic constraints [5]. When there is uncertainty in a robot's sensing or control, the problem can be formulated as a POMDP [6] and solved using value iteration to determine optimal policies [7], [8]. Recent work has focused on overcoming computational limitations of POMDP using local gradient-based optimization in belief space [9], [10], [11], combined with RRT or PRM to initialize a global path. Other recent work flattens the belief space by utilizing local controllers to drive the system to specific belief states [12].

Multi-world motion planning has potential application in areas where highly predictive models of obstacles exist, such as robots planning among obstacles following classical physical laws [13], multi-robot motion planning when a robot has access to potential motion models of other robots [14] in its environment, and motion planning in human environments where crowd simulations can give predictions of likely human paths [15], [16].

Human-aware robot navigation, in particular, has received significant recent attention from researchers [17]. Work in this area often focuses on human-specific optimizations such as maintaining comfortable passing distances [18], [19], [20], minimizing confusing turns [21], and analyzing the effect biomechanics has on likely human motion [22].

Most closely related to our work are planning techniques which are designed to closely integrate with (probabilistic)

This work was supported in part by the National Science Foundation, under grants #IIS-1748541 and CNS-1544887.

Authors are with the Department of CS&E, University of Minnesota, USA. {davi1510,sohre007,sjguy}@umn.edu

predictions of likely human motion. For example, Ziebart et al. [23] proposed a joint optimization function which minimized the trajectory cost over all likely future human paths. More recent work has highlighted the dangers that planning over all paths can have, as it may lead to a "robot freezing" problem where the robot stands stuck, unable to find a path which satisfies all predicted eventualities simultaneously. Trautman et al. [24], proposes to overcome this problem by jointly optimizing over the set of robot and human paths. In effect, this assumes the humans and robots will cooperate when needed, allowing the robot taking bold paths through the crowd.

At a high level, the works of Ziebart et al. and Trautman et al. illustrate the fundamental trade-offs inherent in planning under a set of likely trajectory predictions. Plans which account for all outcomes can be overly conservative, causing a robot to freeze. Planning over a single best (e.g., maximally likely) set of trajectories can be overly optimistic and require the obstacles in the environment to cooperate to produce collision-free trajectories. While this assumption can work very well in the case of attentive, cooperative humans, it can be overly optimistic in some cases (e.g,. distracted pedestrians) and cannot account for planning around inanimate obstacles which do not respond do the robot. In contrast with previous work, our work provide guaranteed collision avoidance of the entire set of likely obstacles trajectories while still working to unfreeze the robot by considering each prediction independently whenever possible.

III. Approach Overview

We will consider a robot following a Sense-Predict-Plan-Act loop (e.g., as in [25]) as is typically applied in predictive, simulation-driven navigation. As discussed above, the key feature of our approach is that the robot does not attempt to avoid all predictions at once, rather we allow a robot to plan over a variety of different predictions independently, choosing a single plan only when enough information is present to know which of the predicted outcomes is actually occurring.

When confronted with a probabilistic distribution of likely obstacle trajectories, traditional discrete-time motion planning (such as in [23]) uses an 'Avoid All' approach which can be formalized as follows: Given an initial state, $\mathbf{x}_0$, a goal state, $\mathbf{g}$, a dynamics function, $f(\mathbf{x}, \mathbf{u})$, and a time-varying function which predicts likely obstacle states, $\mathcal{O}(t)$. Our goal is to optimize the cost function $\mathcal{C}(\mathbf{x}, \mathbf{u})$:

$$\min_{T, \mathbf{u}_0 \dots \mathbf{u}_T} \sum_{t=0}^{T} \mathcal{C}(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

such that:

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t) && \forall t \\ \mathbf{x}_{T+1} &= \mathbf{g} \\ \mathbf{x}_t &\notin \mathcal{O}(t) && \forall t \end{aligned}$$

where $\mathbf{x}_t$ and $\mathbf{u}_t$ are, respectively, the state and controls of the robot at time t. That is, we find a cost minimal path from $\mathbf{x}_0$ to $\mathbf{g}$ that obeys the dynamics of the robot, and avoids any configuration in the collision state, $\mathcal{O}$.

However, it is possible to outperform this Avoid All approach by exploiting the structure of the obstacles set $\mathcal{O}$. Here, we are exploiting the observation that an given obstacle can only be in one place at a time. Given an estimate of distributions of likely obstacles locations (e.g., as a Gaussian mixture model), we identify conditions where this distribution can be represented a disjoint distribution as described in Section IV. We refer to each of these opportunities as splits, and the process of identifying them a split-detection.

Importantly, a robot is free to plan over each branch of a split separately (assuming it can reliably identify which predicted branch happens in execution). Specifically, we define a branch as the set of controls and states that all are avoiding the same split in the prediction. To define the relations between branches, we use a parentage function, denoted as $p(t, j)$ to map the state in branch j at time t to its parent state in branch k at time $t-1$. Note that if there are no splits between $t-1$ and t, the parent branch is the current branch. We use $\mathbf{x}_{(t,j)}$, to represent the state of the robot at time t on branch j (and the same for $\mathbf{u}_{(t,j)}$). We denote the number of branches at time t as b_t.

Using the split from our split-detection, we can define the formal multi-world motion planning problem. Given an initial state $\mathbf{x}_{(0,0)}$, a goal state, $\mathbf{g}$, a dynamics function, $f(\mathbf{x}, \mathbf{u})$, a time- and branch-varying obstacle function, $\mathcal{O}(t, j)$, the branch parentage function, $p(t, j)$, a branch likelihood, $P(t, j)$, and a cost function $\mathcal{C}(\mathbf{x}, \mathbf{u})$, we optimize:

$$\min_{T, \mathbf{u}_{(0,0)} \cdots \mathbf{u}_{(T,b_T)}} \sum_{t=0}^{T} \sum_{j=0}^{b_t - 1} P(t, j)\, \mathcal{C}(\mathbf{x}_{(t,j)}, \mathbf{u}_{(t,j)}) \tag{2}$$

such that:

$$\begin{aligned} \mathbf{x}_{(t+1,j)} &= f(\mathbf{x}_{(t,p(t+1,j))}, \mathbf{u}_{(t,p(t+1,j))}) && \forall t\, \forall j \\ \mathbf{x}_{(T,j)} &= \mathbf{g} && \forall j \\ \mathbf{x}_{(t,j)} &\notin \mathcal{O}(t, j) && \forall t\, \forall j \end{aligned}$$

Note that if there are no splits (i.e. $b_t = 1$ and $p(t, j) = j\ \forall t\,, j$), this is exactly the traditional motion planning optimization function. The dynamics constraint enforces that two split trajectories with the same parent state must start at that parent state at the point of the split. The branch likelihood function P serves to avoid weighing the costs more heavily when there are more branches.

Key Assumptions. We assume the robot has access to a noisy estimate of its current position (e.g., via SLAM [26], [27], [28]) and noisy estimates of the location of nearby obstacle (such as pedestrians). Many prior results have demonstrated the feasibility of fast, approximate solutions for this localization in the presence of humans using Kinect cameras [29], and have shown methods to identify and estimate pedestrian locations using RGB+Depth cameras [30], [31]. We also assume that a robot is provided, via a higher level planner, with a goal location which it desires to reach in a timely fashion.

Trajectory Prediction. Throughout this work we assume that the robot has access to some prediction of the likely future trajectories of the obstacles around it. While simple constant-velocity assumptions may make sense in some settings, typically more sophisticated predictions are preferable. In many environments these dynamic obstacles will represent humans, whose trajectories can be predicted using crowd simulation techniques like the PowerLaw model [32], or by using data-driven approaches which learn from human trajectories [23], [33], [34], [35], [36].

After completing the process of *Split Detection* (Section IV), we must find plans which can exploit this knowledge of an impending split in the predicted paths. We discuss our approach this problem of *Branched Planning* in Section V. Finally, Section VI presents comparisons between our proposed method and alternative planning methods.

IV. Split Detection

Our split detection algorithm serves two purposes within our framework. First, we need to transform a (potentially) sampled distribution of likely obstacle trajectories into a continuous form suitable for optimization-based planning. Secondly, we must detect structure in the predictions which can allow us to separate the predicted motion into easily identifiable branches. Here, we accomplish both tasks simultaneously by fitting Gaussian mixture models (GMM) to the sampled predictions at each future timestep, and analyzing the goodness-of-fit. Distributions which have sufficiently well separated components represent splits in the predicted obstacle states.

We require two conditions to be met by a given collection of Gaussians to be considered well representing the underlying distribution for the purposes of split detection. First, the Gaussians must match the true predicted distribution locally (i.e., where the Gaussians have values sufficiently larger than zero). This ensures that avoiding collisions with each Gaussian will avoid the underlying samples that Gaussian represents. Secondly, we partition the Guassians into subsets such that they are sufficiently distinct from every other subset so that, given a single random sample, it is possible to identify which subset it is from with high probability. This ensures that when a robot senses the state of the environment after a split has happened, it knows which branch of the split it is in. We can formalize these two conditions as follows:

Given a partitioning of a Gaussian mixture $C = \{S_1, ..., S_n\}$, it must satisfy the following for the subsets to be considered separable:

1. Classification confidence. We define the confidence with which we can classify a given sample z based on the chance that the sample comes from the subset of Gaussians from which it is mostly likely, that is:

$$\frac{\max_i(P(z|S_i))}{\sum_j(P(z|S_j))} \tag{3}$$

where $P(z|S_i)$ is the probability that sample z would be generated by the Gaussian mixture S_i. As $P(z|S_i) = P_{S_i}(z)/\sum_j P_{S_j}(z)$, where P_{S_i} is the pdf of a subset mixture model, we can rewrite eq. 3 in terms of the pdfs:

$$\frac{\max_i(P_{S_i}(z))}{\sum_j(P_{S_j}(z))} \tag{4}$$

When this value is high for all samples z, it implies each sample can be reliably identified as "belonging" to a single subset of the mixture, as the probability it belongs to the subset is significantly higher than the probability it belongs to any other subset.

2. Quality of approximation. In order to be separable, each partition must be able to independently represent all of the samples that belong to it without support from other partitions. That is, for all samples, z, the error in reconstructing the total probability with a single subset is equal to:

$$|P_{S_i}(z) - \sum_j P_{S_j}(z)| \tag{5}$$

We note that the confidence (eq. 4) has a maximum value of 1 (as $P_{S_i}(z) >= 0, \ \forall i$) and the error (eq. 5) has a minimum value of 0. To achieve these values, it must be the case that $P_{S_i}(z) = \sum_j P_{S_j}(z)$. As such, the confidence is maximized, and the error is minimized when the value of the total Gaussian mixture model is nearly equal to the maximum of the subset mixture models for all the samples in the distribution (see Figure 1). Therefore, we may test the accuracy of a partition's representation of the total mixture by measuring the difference between the maximum subset and total mixture value at each sample state:

$$\sum_i P_{S_i}(z) - \max_j P_{S_j}(z) \tag{6}$$

If this sum is sufficiently small for a large enough percentage of the samples, the distributions are considered separable; otherwise, the partition is rejected.

In the scenarios shown here, the number of partitions is typically two or less. However, if multiple separable partitions are found, the partition with the most subsets is chosen to allow maximum flexibility while planning. It is also important to note that this framework does not strongly rely on the branches being a single Gaussian. Branches can split between GMMs rather than single Gaussians, or different additive mixture models can be used (as long as their PDFs can be efficiently computed). However, because this increased complexity comes at the cost of additional runtime, we focus on GMMs as they can be quickly fit using an EM approach.

Once a stable split has been detected, we label all subsequent samples later in time with the same partitioning. As a result, once samples have been partitioned separately, they will not be in the same partition again. This temporal partitioning relationship is stored in the parentage function. The overall split-detection algorithm in summarized in Algorithm 1. Here, *isSeparable* is a function that takes as input a partition, the samples, and a confidence threshold. The partition is then tested using equations 4 and 5 against the threshold to determine if the partition is viable. The function

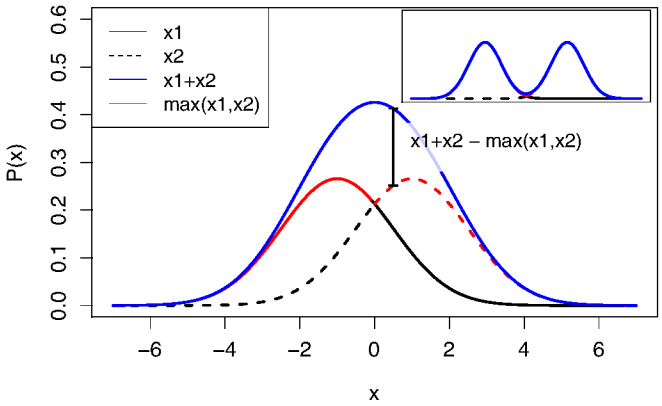


Fig. 1: **Mixture Separability Test**. *Main Figure:* A case in which a Gaussian mixture is not well approximated by their max. *inset:* When two Gaussians are well separated, the max is a very close approximation to their sum.

Algorithm 1: Split Detection

Input : $EnvironmentPrior, T_{horiz}$
Output: $splits, parentages$
$samples \leftarrow$ genSamples($EnvironmentPrior$);
$splits \leftarrow list < partition >$;
$parentages \leftarrow map < partition, partition >$;
foreach $t \in t_0 \cdots t_{T_{horiz}}$ **do**
  $samples \leftarrow$ forwardSimulate($samples$);
  $gaussMixture \leftarrow$ fitModel($samples$);
  $partitions \leftarrow$ getAllPartitions($gaussMixture$)
  foreach $p \in partitions$ **do**
    $candidates \leftarrow list < partition >$;
    if *isSeparable*$(p, samples, confidence)$ **then**
    end
  end
  $splits$.add(maximalSet($candidates$));
end
$parentages \leftarrow$ computeParentages($splits$);
return($splits, parentages$);

maximalSet selects the partition among all viable partitions having the most subsets.

V. Branched Planning

After obtaining the split behavior of the agents in the environment, we need to plan over the branching distribution of predictions. We do this in a two phase process. We first use a modified space-time RRT to computing a branching set of trajectories where each split in prediction gives rise to *simultaneous* branches in the planned trajectories. Secondly, we optimize these branched trajectories using a modified, split-aware version of iterative LQR [37].

A. Split-Aware Space-Time RRT

Our approach to global optimization is a direct extension of existing space-time RRT approaches. To extend the RRT to account for split-planning, we modify the sampling, nearest-neighbor computation, and node connection steps in an RRT as described below.

Sampling: We first sample a time, t. At time t, we independently sample b_t states $(\mathbf{x}_{(t,0)}, \ldots, \mathbf{x}_{(t,b_t)})$, one for each branch at that time.

Nearest Neighbor: As two nearby RRT nodes in the configuration space may have a different number of branches, and therefore a different number of states $b_{\hat{t}}$ (i.e. just before and after a split), the approach to computing distances between these two nodes must be adjusted. Here we compute the distance between nodes as the sum of the distance from each of the states, $\mathbf{x}_{(t,i)}$, to its ancestor in the node's states, $\mathbf{x}_{(\hat{t},j)}$.

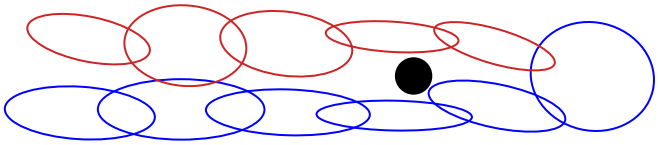
Node Connections: In order to connect two nodes with different state sizes (i.e., differing number of branches) we first create an intermediate node that lies exactly on the split point between these branches. We consider a connection to be valid only if the path from the parent node to the split point is collision-free and all of the paths from the split point to the branches are also collision-free. We mark an RRT path as colliding if it falls within the 3-sigma shell of any of the Gaussian obstacles.

B. Split-Aware iLQR

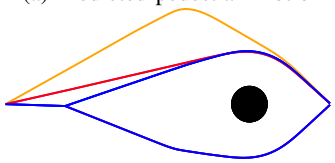
Because an RRT is a sampling-based planner the resulting trajectories can have large amount of jitter in the robot paths. We utilized a modified local optimization technique (iterative LQR [37]) to smooth the paths and ensure safer overall obstacle avoidance (as the RRT only avoids the 3-sigma shell of the Gaussian distributions). Iterative LQR is a 2nd order local optimization method that iteratively linearizes the dynamics and quadraticizes the cost function in order to compute descent directions. To allow iLQR to handle splitting trajectories, we only need to modify the value function of the node prior to a split to be the expected value across all branches after the split. For instance, if a node has 2 equally likely children, the new value function would take the form:

$$V(\mathbf{x}_{(t,j)}) = \min_{\mathbf{u}_{(t,j)}} C(\mathbf{x}_{(t,j)}, \mathbf{u}_{(t,j)}) + \frac{1}{2}(V(\mathbf{x}_{(t+1,k)}) + V(\mathbf{x}_{(t+1,l)})) \quad (7)$$

where k and l are the child branches. As both of these value functions are quadratic in $\mathbf{x}$, their sum will also be quadratic in $\mathbf{x}$, and therefore the value function of this pre-split stage will also still be quadratic (as required by the iLQR framework). As we are just extending the number of stages we have to compute without increasing the size of each stage, the cost of each LQR iteration will also at most increase linearly in the number of branches. Standard iLQR is $O(Tn^3)$ per iteration, where T is the time horizon and n is the state size. Our modified split-iLQR is $O((\sum_{t=0}^{T} b_t)n^3)$ per iteration. In addition, this can be further sped up by updating the value functions of all the branches at each time in parallel.



(a) Predicted pedestian motion



(b) Planned trajectories

Fig. 2: **Planning Method Comparison (paths).** Here, we are uncertain if the pedestrian will pass to the left or right of the pole. Maximum likelihood methods (red), are overly optimistic and result in collision if the pedestrian path does not match the prediction. Avoid All planning (orange) results in overly conservative trajectories to avoid all outcomes. Branched planning (blue) calculates trajectories for both sets of likely outcomes, and then selects which to follow depending on what is observed during execution.

Planning Method	Planned Length	Executed Length
ML	4.27	4.70
Avoid All	4.66	4.66
Branched	4.31	4.31

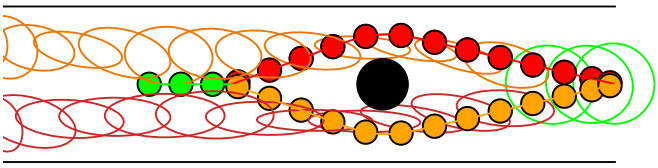
TABLE I: **Planning Method Comparison (length)**. Planned trajectory lengths versus executed trajectory lengths for the scenario in Fig. 2. Both Avoid All and Branched planners can successfully execute their plans as expected. However, with the robot making a maximum likelihood (ML) assumption the execution can be much worse than planned if the guess was wrong. On average, Branched planning has the shortest trajectory.

VI. Results

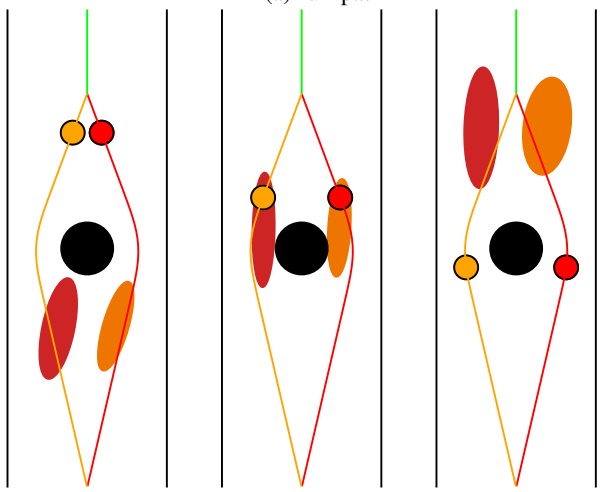
In this section, we show the results of the split-detection and split-planning algorithms detailed above. We give the results in several different scenarios, and provide some comparisons to alternative planning approaches. In all our experiments we base our cost function on three terms: chance of collision, control magnitude, and distance to the goal (for the final stage). All experiments were run in C++ on a laptop with an Intel i7 2.6GHz processor.

A. Pole In Hallway

As a motivating scenario, consider the case of a robot observing a pedestrian moving towards a pole (Fig 2). Here, we are uncertain if the pedestrian will pass to the left or right of the pole. We compare the robot paths produced by different classes of planning approaches in the face of this uncertainty. *Maximum likelihood* (ML) methods plan trajectories assuming that the person will travel along a single (most likely) instance of the predicted path. Here, it assumes the person will pass the pole on their right. This works well if the guess is correct, but leads to much worse paths,



(a) full path



(b) $t = 2.0$ (c) $t = 3.0$ (d) $t = 4.0$

Fig. 3: **Hallway Pole Simulation**. *a.* A robot (colored circle) avoids likely positions of a pedestrian moving down a hallway (ellipse shows 95% confidence intervals). The predicted pedestrian path is bifurcated by a large pole (black circle). Colors indicate time, with green being states that occur before the split, and red and orange states in the two branches after the split. *b.-d.* Time slices of the simulations.

and potentially collisions, if the guess was wrong. *Avoid All* planning is safe under both paths the pedestrian may take, but results in a much longer path than ML planning. Branched planning calculates contingencies for both paths and commits to a branch only when its clear which route the pedestrian will take. As a result, Branched planning finds trajectories that are more efficient than Avoid All planning. As compared to Maximum likelihood methods, Branched planning is slight less efficient when the ML methods make the right predication, but better on average. This is because branched planning must take a compromise trajectory that accounts for both possibilities until it observes which path the pedestrian takes. See Table I.

We note that the Maximum Likelihood approach can work very well in scenarios where the obstacles will respond to the robot. Here, even if the robot predicts incorrectly, the obstacles (e.g., people) can switch to a different path. This insight is one of the key assumptions which allows work such as [24] and [38] to unfreeze the robot in highly constrained situations. Our approach can also find collision free paths in certain constrained environments, but without assuming that obstacles will respond to the robot (though at the cost of less optimal paths when the obstacles do respond).

The freezing effect of constraints can be seen more clearly by considering the pole scenario occurring in the middle of a

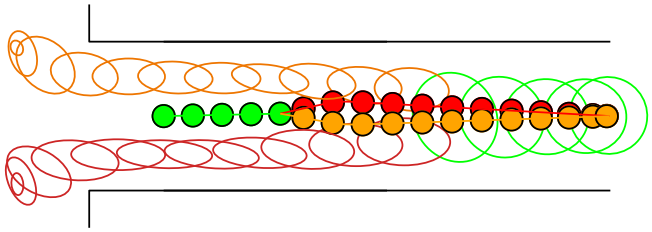


Fig. 4: **Hallway T Simulation**. A predictions of likely pedestrian paths is bifurcated by splits in possible goals at the T-junction at the end of the hallway. (C.f. Figure 3).

narrow hallway (see Figure 3). Here, there is not enough free space between the left and right distributions to fit both the robot and person. As such, Avoid All planning is not able to find any collision free path to the goal. In contrast, Branched planning is able to find an efficient path by moving forward towards the middle of the obstacles until it becomes clear which direction the human will go. It is worth emphasizing that the robot is not simply reacting to local conditions, but rather the robot is *planning to react* when the necessary information becomes available.

B. Hallway T Junction

In some scenarios, clustering the predicted paths only on positions can fail to find splits until agents are very near their goal. We can improve our split detection by clustering on other aspects of the obstacle state. Consider the case shown in Figure 4 of an person heading towards a T-junction in a hallway. Here, we use a goal estimation (based on velocities) in addition to positions when clustering predicted obstacle states. Using this additional knowledge, the robot can determine which split the agent will take prior to the predicted positional distributions physically separating. By exploiting this knowledge of this split, the robot is able to move towards the end of the hallways sooner, diverting slightly left or right to avoid the person as they head towards the T. In the traditional Avoid All planning, the robot would have to wait until the distribution physically separated before it could traverse further down the hallway.

C. Differential Drive

In this scenario, we have a differential drive robot attempting to navigate a hallway. However, there are a series of poles running down the middle, and a ball is going to roll down either the left or the right side of the hallway. We assume we are able to determine the direction the ball will roll in 3 seconds. While our RRT implementation only supports holonomic dynamics, we can utilize the power of our local optimization by running a trajectory following controller over the initial trajectory, and then optimizing the new trajectory via iLQR. In this scenario, by utilizing its knowledge that the ball will fall on either side (but not both), the robot can make progress down the hallway, and dodge either left or right through the poles to ensure our path stays safe (see

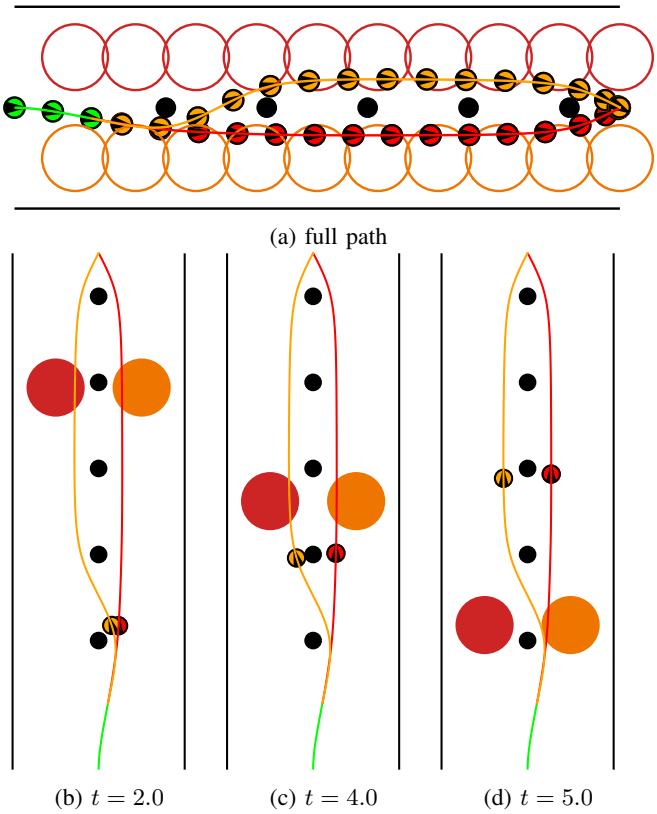


Fig. 5: **Differential Drive Simulation**. *a.* A differential drive robot (orientation indicated by arrow) travels down a hallway with a mix of static poles (black circles) and a dynamic obstacle (red/orange circles). The dynamic obstacles path is naturally constrained into one of two branched by the polls. *b.-d.* Time slices of the simulations.

Figure 5). This is in contrast to the Avoid All approach which is only able to avoid collision by moving away from the goal.

D. Performance increase over standard planning

In addition to producing safer and faster paths, our method can also be faster to compute than standard planning techniques. This is mostly due to the improved percentage of free space in constrained areas. Consider an example scenario with a pole in the middle of a hallway, and two obstacles on either side of that pole. If the center pole takes up 10% of the hallway and each side obstacle takes up 40%, the chance to find a free sample within the constrained region is 10%. However, if we allow branched planning, the chances that the sample in each branch are non-colliding is 50% for each branch, or 25% total. This higher likelihood of choosing free sample in constrained region increases the rate at which our algorithm is able to find solution paths.

This can also be seen experimentally in Figure 6. The three different lines correspond to the size of a pole in the hallway, while the x-axis shows how varying the size of the split obstacle on either side of the pole effects the relative runtime of the two planning algorithms. As the hallway becomes increasingly more occupied, the branched planning starts

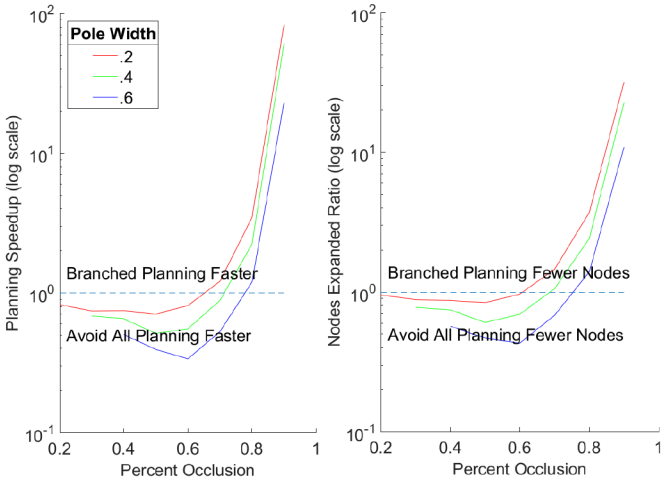


Fig. 6: **Split-Planning Speedup**. Our planning method is capable of finding paths faster when the non-split space is highly constrained. The 3 lines show our speed up over increasing obstacle size. Times are averaged over 500 runs, and show how relatively long it takes to find the first feasible path from start to goal between our method and traditional avoid all planning. The graph on the right shows the relative number of nodes added to the RRT before a goal reaching path was found.

to find paths significantly faster than the traditional Avoid All planning. Importantly, even though Avoid All planning is slighting faster when the scenario is unconstrained, both methods run at very fast speeds (5 ms for Branched Planning, and 4 ms for Avoid All). When the scenario becomes more constrained, Branched planning is still realtime (15 ms), whereas the avoid all approach takes nearly half a second to find a path. Note that, in the limit (no free space), there is no possible solution to the avoid all planning problem. In terms of nodes expanded, Figure 6 suggests a similar advantage in terms of memory footprint.

VII. Conclusion

In conclusion, we have introduced the problem of Many Worlds Motion planning, and have shown how the many worlds formulation can lead to robots finding higher quality paths with less computation costs than is possible in traditional planning methods for certain types of problem. Additionally, we have detailed an approach to solve this problem through the means of split-detection, split-aware space-time RRTs, and split-aware iLQR, and have shown an improvement in planning performance on several scenarios.

Limitations. Our current implementation of multi-world motion planning has several important limitations, relating to application to large scale planning problems. While both the split-detection and split-aware iLQR optimization can be computed in real time, our current implementation of split-aware spacetime RRTs takes several seconds to compute plans of moderate complexity, and this cost grows as the planning time horizon increases. Additionally, our planning approach currently terminates when the first path is found, as opposed to continue refining towards a (time) optimal path. Moving towards incrementally refining, anytime approaches such as SST* [39] may help with both speed and optimality,

In the future, we hope to address these limitations as we scale up the scope of where the multi-world motion planning approach can be applied. In particular, we are excited about the possibility of automatically detecting the most critical or promising space-time splits as an avenue for improving performance when encountering very complex obstacle representations (as might happen when navigating in a dense crowd).

References

[1] Z. Littlefield, A. Krontiris, A. Kimmel, A. Dobson, R. Shome, and K. E. Bekris, "An extensible software architecture for composing motion and task planners," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 327–339.

[2] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[3] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2015.

[6] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 95, 1995, pp. 1080–1087.

[7] J. M. Porta, N. Vlassis, M. T. Spaan, and P. Poupart, "Point-based value iteration for continuous pomdps," *The Journal of Machine Learning Research*, vol. 7, pp. 2329–2367, 2006.

[8] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces." in *Robotics: Science and Systems*, vol. 2008, 2008.

[9] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," *Proceedings of Robotics: Science and Systems*, 2010.

[10] 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.

[11] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, "Gaussian belief space planning with discontinuities in sensing domains," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[12] A. akbar Agha-mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[13] Y. Tian, P. Ma, P. Zherong, B. Ren, and D. Manocha, "Fluid directed rigid body control using deep reinforcement learning," *ACM Transactions on Graphics,(Proc. SIGGRAPH)*, 2018, [To Appear].

[14] J. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Moving in a crowd: Safe and efficient navigation among heterogeneous agents," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[15] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "Brvo: Predicting pedestrian trajectories using velocity-space reasoning," *The International Journal of Robotics Research*, p. 0278364914555543, 2014.

[16] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation." in *Robotics: science and systems*. Citeseer, 2012.

[17] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[18] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 874–883, 2007.

[19] R. Kirby, R. Simmons, and J. Forlizzi, "Companion: A constraint-optimizing method for person-acceptable navigation," in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE, 2009, pp. 607–612.

[20] A. Sardar, M. Joosse, A. Weiss, and V. Evers, "Don't stand so close to me: users' attitudinal and behavioral responses to personal space invasion by robots," in *Proceedings of the seventh annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2012, pp. 229–230.

[21] T. Kruse, A. Kirsch, H. Khambhaita, and R. Alami, "Evaluating directional cost models in navigation," in *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2014, pp. 350–357.

[22] V. V. Unhelkar, C. Pérez-D'Arpino, L. Stirling, and J. A. Shah, "Human-robot co-navigation using anticipatory indicators of human walking motion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6183–6190.

[23] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 3931–3936.

[24] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[25] J. Van Den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2366–2371.

[26] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 2985–2992.

[27] S. Thrun, "Robotic mapping: A survey, exploring artificial intelligence in the new millenium, chapter i," 2002.

[28] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[29] J. Biswas and M. M. Veloso, "Localization and navigation of the cobots over long-term deployments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, 2013.

[30] L. Xia, C.-C. Chen, and J. K. Aggarwal, "Human detection using depth information by kinect," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 15–22.

[31] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *Cybernetics, IEEE Transactions on*, vol. 43, no. 5, pp. 1318–1334, 2013.

[32] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Phys. Rev. Lett.*, vol. 113, p. 238701, Dec 2014.

[33] S. Choi, E. Kim, and S. Oh, "Real-time navigation in crowded dynamic environments using gaussian process motion control," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3221–3226.

[34] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 981–986.

[35] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1341–1346.

[36] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning." *IJ Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[37] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *ICINCO 2004, Proceedings of the First International Conference on Informatics in Control, Automation and Robotics, Setúbal, Portugal, August 25-28, 2004*, 2004, pp. 222–229.

[38] P. Trautman, "Sparse interacting gaussian processes: Efficiency and optimality theorems of autonomous crowd navigation," *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 327–334, 2017.

[39] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.