Variable Level-Of-Detail Motion Planning in Environments with Poorly Predictable Bodies

Stefan Zickler and Manuela Veloso[1]

Abstract. Motion planning in dynamic environments consists of the generation of a collision-free trajectory from an initial to a goal state. When the environment contains uncertainty, preventing a perfect predictive model of its dynamics, a robot ends up only successfully executing a short part of the plan and then requires replanning, using the latest observed state of the environment. Each such replanning step is computationally expensive. Furthermore, we note that such sophisticated planning effort is unnecessary as the resulting plans are not likely to ever be fully executed, due to an unpredictable and changing environment. In this paper, we introduce the concept of Variable Level-Of-Detail (VLOD) planning, that is able to focus its search on obtaining accurate short-term results, while considering the far-future with a different level of detail, selectively ignoring the physical interactions with poorly predictable dynamic objects (e.g., other mobile bodies that are controlled by external entities). Unlike finite-horizon planning, which limits the maximum search depth, VLOD planning deals with local minima and generates full plans to the goal, while requiring much less computation than traditional planning. We contribute VLOD planning on a rich simulated physics-based planner and show results for varying LOD thresholds and replanning intervals.

1 INTRODUCTION

Mobile robot motion control in the physical world is a challenging problem. A robot has to deal with the uncertainty that arises during the execution of its own actions (caused by e.g., drift, wheel slippage, and sensory noise). More importantly, uncertainty also makes it difficult to accurately predict the motions of any other moving obstacles in the environment, in particular when controlled by an unpredictable foreign entity, making it difficult to devise a model for valid predictions of the obstacles' motions.

Accurate motion planning in such uncertain domains can be a futile task: as soon as the robot starts executing its plan, the real world's state is likely to quickly diverge from the predictions that were made during planning. A well-known solution to this problem is replanning: the robot only executes a portion of a generated plan; it then re-observes the world's true state and plans a new solution, using the latest observations. This process is repeated as a *replanning loop* until the robot reaches its goal state.

A major problem of replanning is its computational cost. At each replanning iteration, a planner performs an intensive search for a complete solution that will bring the robot from its current state all the way to the final goal state. During this search, a motion planner ensures that the resulting plan is dynamically sound and collision-free by employing sophisticated, computationally expensive physical models to predict the robot's motions and its interactions with the predicted environment. Such an elaborate search might seem unnecessary, given the fact that the robot will only execute a small portion of the resulting plan, before discarding it and re-invoking the planner to start from scratch in the next replanning iteration. However, simply limiting the planner's search depth, an approach also known as *finite-horizon planning*, is dangerous because it can lead to a robot becoming stuck in a local search minimum as there are no guarantees that a partial plan will actually lead to the final goal state.

In this paper, we introduce Variable Level-Of-Detail (LOD) planning, a novel approach to reduce the computational overhead of replanning in physical robot environments. Unlike finite-horizon planning, VLOD planning maintains a full search to the goal state, and is therefore more robust against local minima.

We base VLOD planning on the idea that a planner should be able to speed up its search by relaxing the treatment of computationally intensive domain details that lie far in the future and that are unlikely to be accurately predicted. We present a binary LOD model that allows the planner to selectively ignore future interactions with bodies that are considered to be difficult to predict. The time threshold that defines what is to be considered "too far in the future to be accurately predicted" is a controllable parameter, and can be adjusted based on the amount of unpredictability in the domain (thus the term *Variable* LOD planning).

This paper is organized as follows: first, we cover related work. We then formally define the physics-based motion planning problem and introduce a sampling-based planning algorithm. We then present our VLOD approach, integrating it into the existing planning algorithm. We test our approach on two domains and present a detailed analysis of the results. Finally, we end with concluding remarks and ideas for future work.

2 RELATED WORK

Our work focuses on motion planning in continuous, physical domains. Sampling-based planners are an effective choice for this purpose as they can efficiently cover the continuous search space [7]. Rapidly-Exploring Random Trees (RRT) [9] is a sampling-based planning algorithm that has been shown to be usable for motion planning problems involving dynamic and kinematic constraints [10], and has been employed to solve many types of robot navigation problems [2, 11, 12].

There is a vast body of work on increasing replanning performance for robot motion planning in continuous domains. ERRT [3] is an extension to RRT that introduces the concept of a waypoint cache to

---

[1] Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, USA, email: {szickler,veloso}@cs.cmu.edu

bias the planner's search toward points from previous solutions, letting the planner more quickly discover solutions in new slightly different problems during replanning. Other approaches include DRRT [6] and Multipartite RRT [16] which use different schemes to store entire portions of previous search trees to be re-used during planning. Compared to these approaches, our work takes a fundamentally different direction: instead of trying to speed up replanning by re-using past planning solutions, we aim to directly reduce replanning complexity by selectively ignoring low-level domain details that lie too far in the future to be relevant for near term execution.

Another approach for increasing replanning performance in dynamic environments is to introduce a layered planning architecture, where a global plan is computed by a planner that uses a higher-level abstraction of the world (e.g., graph-based), which is computationally efficient, but less accurate and typically unaware of lower-level dynamics [8, 4]. The global plan is then handed to a lower-level, local planner that performs a finite-horizon search with the goal of following along the global planner's solution. One fundamental weakness of this dual-layer approach is that the global planner, due to its abstract model, cannot guarantee that its solutions are actually solvable by the local planner. VLOD planning circumvents the global planning inaccuracy problems of such layered planning approaches by relying on a single fully dynamic planner that generates global plans leading all the way to the goal state, only ignoring a certain subset of multi-body interactions that are assumed to be locally solvable.

3 PHYSICS-BASED PLANNING

We define the motion planning problem as follows: given a state space X, an initial state $x_{\text{init}} \in X$, and a set of goal states $X_{\text{goal}} \subset X$, a motion planner searches for a sequence of actions $a_1, \ldots, a_n$, which, when executed from x_{init}, ends in a goal state $x_{\text{goal}} \in X_{\text{goal}}$. Additional constraints can be imposed on all the intermediate states of the action sequence (e.g., collision avoidance).

To demonstrate our approach, we use a *Physics-Based* motion planning algorithm that aims to reflect the inherent physical properties of real world multi-body interactions. The *Rigid Body Dynamics* model [1] provides a computationally feasible approximation of basic Newtonian physics, and allows the simulation of the physical interactions between multiple mass-based bodies. Physics-based planning is an extension to *kinodynamic planning* [5], adding the simulation of rigid body interactions to traditional second order navigation planning [14, 13].

A rigid body system is composed of n rigid bodies $r_1 \ldots r_n$. A rigid body is defined by two disjoint subsets of parameters $r = \{\hat{r}, \bar{r}\}$ where $\hat{r}$ are the body's mutable state parameters (i.e., position, orientation, and velocity), and $\bar{r}$ are the body's immutable parameters (i.e., its shape, mass, and material properties). The physics-based planning state space X is defined by the mutable states of all n rigid bodies in the domain and time t. That is, a state $x \in X$ is defined as the tuple $x = \langle t, \hat{r}_1, \ldots, \hat{r}_n \rangle$.

An action a is defined as a vector of subactions $\langle \hat{a}_1, \ldots, \hat{a}_n \rangle$, where $\hat{a}_i$ represents a pair of 3D force and torque vectors applicable to a corresponding rigid body r_i.

A physics-based planning domain d is defined as the tuple $d = \langle G, \bar{r}_1 \ldots \bar{r}_n, M \rangle$ where G is the global gravity force vector, $\bar{r}_1 \ldots \bar{r}_n$ are the immutable parameters of all n rigid bodies, and M is a symmetric collision matrix. The symmetric collision matrix M is of size $n \times n$ and defines whether pairwise collisions between any two rigid bodies r_i and r_j should be resolved or ignored. A value of 1 for a matrix entry m_{ij} (and therefore also m_{ji}) implies that a colli-

sion should be resolved as if the two bodies were rigid, i.e., the bodies should not penetrate one another. A value of 0 implies that collisions should be ignored by treating the two bodies as non-rigid with respect to each other, i.e., the bodies should pass through one another. By default, M is filled with ones, except for its main-diagonal that is always zero-filled, because a body is unable to collide with itself.

3.1 Planning Algorithm

We now introduce our core planning algorithm (see Algorithm 1). The search expansion methodology of our algorithm is based on Rapidly-Exploring Random Trees (RRT) [9]. We initialize the search with a tree T containing an initial state $x_{\text{init}} \in X$. We then enter the main planning loop, which runs for a predefined domain-dependent maximum number of search iterations z, if no solution is found earlier. On each iteration, the algorithm selects a node x from the existing tree T which it will expand from by invoking the function `SelectNodeRRT`. Within `SelectNodeRRT` (see Algorithm 2), the function `SampleRandomState` uses an internal probability distribution to provide a sample y taken from the sampling space Y that is some predefined subspace of X. The function `NearestNeighbor` then finds the nearest neighbor to y, according to some predefined distance function. As with traditional RRT, it is important that the sampling space Y, the underlying probability distribution, and especially the distance function are all carefully chosen to match the domain. For our domains, we use a simple acceleration-based motion model to compute the minimal estimated time for the controlled rigid body in $x \in T$ to reach its target position and orientation in y. `SelectNodeRRT` then returns both the selected node x and the sample y.

Algorithm 1: PlanLOD

Input: Initial state: x_{init}, set of goal states: X_{goal}, RRT sampling space: Y, set of valid states: X_{valid}, timestep: $\Delta\mathsf{t}$, domain: d max iterations: z.

$\mathsf{T} \leftarrow$ `NewEmptyTree()`;
T.`AddVertex`(x_{init});
for iter $\leftarrow$ *1* **to** z **do**
   $\langle\mathsf{x},\mathsf{y}\rangle \leftarrow$ `SelectNodeRRT()`;
   $\mathsf{d.M} \leftarrow$ `SetupCollisionMatrix`($\mathsf{x},\mathsf{d.M}$);
   $\langle\mathsf{x'},\mathsf{L}\rangle \leftarrow$ `Propagate`($\mathsf{x},\mathsf{y},\Delta\mathsf{t},\mathsf{d}$);
   if `Validate`($\mathsf{x'},\mathsf{L}$) **then**
      T.`AddVertex`($\mathsf{x'}$);
      T.`AddEdge`($\mathsf{x},\mathsf{x'},\mathsf{a}$);
      if $\mathsf{x'} \in \mathsf{X}_{goal}$ **then**
         return `TraceBack`($\mathsf{x'},\mathsf{T}$);

return Failed;

Algorithm 2: SelectNodeRRT

$\mathsf{y} \leftarrow$ `SampleRandomState`(Y);
$\mathsf{x} \leftarrow$ `NearestNeighbor`(T,y);
return $\langle\mathsf{x},\mathsf{y}\rangle$;

After selection of the source node x, the algorithm then configures the domain's collision matrix M by invoking the function `SetupCollisionMatrix`. This function is a core component to Variable Level-Of-Detail planning and its purpose will be explained in the following section. For now, let us temporarily assume

Algorithm 3: Propagate

```
a ← Controller(x,y);
⟨x'.⟨r̂₁,…,r̂ₙ⟩, L⟩ ← e(x.⟨r̂₁,…,r̂ₙ⟩,a,d,Δt);
x'.t ← x.t + Δt;
return ⟨x', L⟩;
```

that `SetupCollisionMatrix` simply returns the default collision matrix, thus not changing any of the physics-engine's collision handling behavior.

Next, the planner invokes the `Propagate` function to expand the search tree by growing a branch from the source node x. To compute the successive state x', the `Propagate` function (see Algorithm 3) first computes an action for the controlled rigid body to execute. This action is computed by some controller that heuristically generates forces and torques to bring the controlled body's state in x closer to the state in the random sample y. The particular type of controller can range from a simple linear motion controller (as is assumed in this work) to very elaborate behavioral models that are aware of higher level tactical knowledge [14, 15].
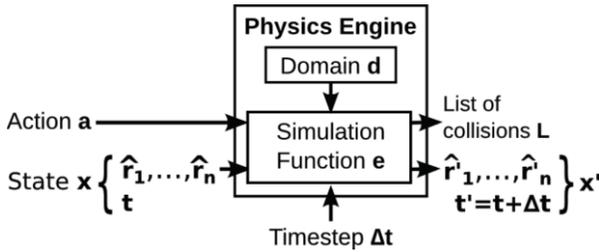

Figure 1: A Physics Engine computes state transitions.

Using the action a generated by the controller, the `Propagate` function can now invoke the physics transition function e that will forward-simulate the rigid-body dynamics and generate the new rigid-body states $\hat{r}_1, \ldots, \hat{r}_n$ that are then stored as part of the new state x'. The actual dynamics computations are performed by a rigid body simulator. There are several robust rigid body simulation frameworks freely available, such as the Open Dynamics Engine (ODE), Newton Dynamics, and NVIDIA PhysX. Frequently referred to as *physics engines*, these simulators are then used as a "black box" by the planner to simulate state transitions in the physics space (see Figure 1). Besides the new rigid body states, the transition function e also returns a list of collisions $L = \langle l_1, l_2, \ldots \rangle$ that occurred during forward simulation. Each item $l \in L$ is an unordered pair $l = \langle \lambda_1, \lambda_2 \rangle$, consisting of the indices of the two rigid bodies $r_{\lambda_1}, r_{\lambda_2}$ involved in the collision. Note, that only collisions that were enabled in the collision matrix M will be reported.

`Propagate` then returns the new state x' and the list of collisions L that occurred during the forward simulation. The algorithm's `Validate` function then checks whether the resulting state is a valid state by making sure that it did not validate any user-defined constraints (e.g., that no undesired collisions occurred in L). If accepted, the algorithm adds x' to the search tree T as a child of the chosen node x.

The complete loop is repeated until the algorithm either reaches the goal, or until it reaches the maximum allowed number of iterations z, at which point the search returns failure for this state. Once a goal state is reached, the algorithm simply traces back the chain of states and actions and returns it as a solution sequence.

4 VARIABLE LOD MOTION PLANNING

To be useful for execution in real-world domains containing uncertainty, our physics-based planning algorithm can be wrapped into a continuous, fixed-timestep replanning loop (see Algorithm 4). After observing the initial state of the world, the robot generates a plan using our physics-based planning algorithm. The robot then executes a fixed, pre-determined amount of this plan, before repeating the loop of re-observing the environment, updating the initial state, and generating a new plan. The replanning interval t_{replan} is set by the user, and generally depends on the expected domain uncertainty. Uncertain domains and robots with unreliable executions tend to require more frequent replanning as the true world state will more quickly diverge from the predicted planning solution while the robot is executing.

Algorithm 4: ExecuteAndReplan

```
while true do
    x_init ← ObserveWorldState();
    solution ← PlanLOD(x_init,…);
    if solution ≠ Failed then
        i ← 1;
        repeat
            ⟨x, a⟩ ← solution[i];
            Execute(a);
            i ← i + 1;
        until x.t > t_replan or i > length(solution);
```

In this type of replanning environment, a planner will perform many computationally intensive searches for detailed solutions, only to have them be partially executed and then scrapped for the next replanning iteration. To alleviate this situation, we can now introduce *Variable Level-Of-Detail* (LOD) planning, that is able to find global planning solutions while ignoring execution-irrelevant domain details that lie far in the future. We introduce the *LOD-horizon* t_{LOD}. This horizon acts as a threshold in the time component t of our planning space X. The purpose of t_{LOD} is to control at what point the planner should begin to ignore certain domain details during its simulated state transitions. The value of t_{LOD} is a global parameter, to be set by the user. A reasonable guideline is that t_{LOD} should be greater than the replanning interval t_{replan} because there is the inevitable assumption that the plan will be executed up to length t_{replan} and as such should be planned with maximum detail for at least that length.

VLOD planning assumes knowledge about the types of bodies present in the domain. We classify the types of bodies in the domains of the physics-based planner, using the hierarchy shown in Figure 2.
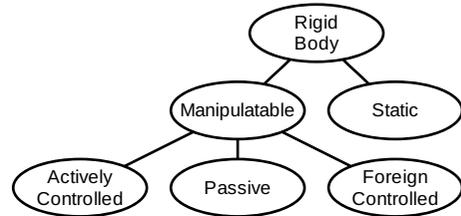

Figure 2: Rigid Body classes

Every body is by definition a *rigid body*. There are *static* rigid bodies that do not move, even when a collision occurs, which are

often used to model the ground plane and all non-movable bodies, such as walls and heavy objects. All other bodies are *manipulatable*, meaning that they react to collision forces exerted upon them. Among these, the planner can directly control the *actively controlled* bodies, i.e., the planner has available actions directly applicable to these bodies. *Passive* bodies can only be actuated by external influences and interactions, such as being carried or pushed. *Foreign controlled* bodies are actively actuated, but by external control to our planner.

Based on this classification hierarchy, we define corresponding sets of rigid body groups: , that allow us to classify each rigid body in the domain by letting it become a member in the corresponding sets: B_{All}, B_{Manip}, B_{Static}, $B_{\text{Controlled}}$, B_{Passive}, B_{Foreign}.

To apply Variable Level-Of-Detail planning, we need to clearly define what we mean by "details". We use a binary notion of detail, allowing the planner to selectively ignore particular pairwise multi-body interactions. In our physics-based domains, we regard the subset of multi-body interactions as *details* if they are solvable through a local finite horizon search without requiring a global change of plans. In our particular planning model, we treat the interactions between controlled bodies and other manipulatable bodies as *details*, whereas we treat any other interactions as *essential*. The idea is that the avoidance and/or manipulation of moving or manipulatable bodies can normally be considered a locally solvable problem, whereas global navigation, such as finding the path through a maze of static wall bodies, requires full-depth planning to successfully reach the goal state without ending up in local minima. Another line of reasoning is that foreign-controlled bodies are not accurately predictable in the long term and as such qualify as details that are only relevant for short term planning. Because foreign-controlled bodies can interact (e.g., push) any other manipulatable body, we consider all manipulatable bodies to fall into the unpredictable *detail* category.

Of course one could imagine special cases of domains where even interactions with manipulatable bodies have implications on the global topology of the plan that go beyond the LOD search horizon t_{LOD}. For such domains, it might make sense to either increase the value of t_{LOD}, or – in extreme cases – manually reduce the selection of pairwise interactions that should be considered "details".

Algorithm 5: SetupCollisionMatrix

```
// Let m_ij denote the element at the i-th
   row and j-th column of M.
for i ← 1 to n do
    for j ← i to n do
        if i=j then
            m_ij ← 0;
        else
            if (x.t > t_LOD and
            ((r_i ∈ B_Controlled and r_j ∈ B_Manip) or
            (r_j ∈ B_Controlled and r_i ∈ B_Manip))) then
                m_ij ← 0;
                m_ji ← 0;
            else
                m_ij ← 1;
                m_ji ← 1;
return M;
```

The planner (see Algorithm 1) applies the LOD-horizon to the physics model before each state transition by calling the function `SetupCollisionMatrix` (see Algorithm 5). This function configures the collision matrix M to effectively let the physics en-
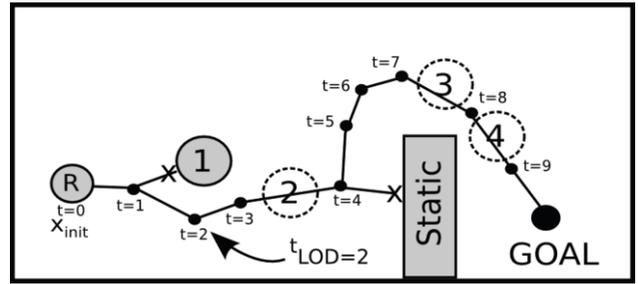

Figure 3: An illustration of VLOD planning

gine know about which rigid-body collisions should be resolved and which ones should be ignored. If the current source state x has a time index less than t_{LOD}, then all collisions are fully simulated and resolved. However, if $x.t$ lies beyond t_{LOD}, then the collision is set to be ignored if it involves a pair of bodies, with one body being a member in the set of controlled bodies $B_{\text{Controlled}}$ and the other being a member in the set of manipulatable bodies B_{Manip}. All other pairwise collisions are treated normally.

Figure 3 shows an illustrative example of VLOD planning. The controlled robot body (R) has to navigate from its current state to the goal. The domain contains four poorly predictable foreign-controlled moving bodies (labeled 1-4) and a static obstacle. Assuming a LOD horizon $t_{\text{LOD}} = 2$ seconds, the first body is treated with full detail during a predicted collision that occurred before $t = 2$, thus requiring planning a path around the body. The static obstacle is fully predictable and relevant to the global path topology, thus it is always treated as an obstacle (even when $t > t_{\text{LOD}}$). The other moving bodies (2-4) only make contact with the search tree beyond the horizon t_{LOD}, thus they are ignored in this planning iteration.

5 RESULTS

We tested our physics-based VLOD planner using a detailed simulation framework that is able to model robot execution under uncertainty. We used NVIDIA PhysX as the underlying physics engine with $\Delta t = 1/60s$.

We devised two challenging kinodynamic robot navigation domains to test the effects of VLOD planning. In each domain, the controlled robot body has to navigate from a starting state to a goal area while avoiding multiple oscillating obstacles. The planner in this case uses a simple deterministic prediction model of where it expects the obstacles to move. We actively model the domain's execution uncertainty by controlling the amount of random divergence of the obstacle's actual motion paths from the planner's prediction model.

In the "Hallway" domain, the main challenge is to execute a safe trajectory through a dense field of 12 rapidly moving and not fully predictable obstacle robots. Figure 4(a) shows a kinodynamic RRT search tree (black) as generated by our planner during the first replan iteration. The planner's linearly predicted trajectories of the moving obstacles are indicated by the vertical paths. Figure 4(b) shows what individual obstacle trajectories actually look like during execution under maximum domain uncertainty (uncertainty value of 1.0).

The "Maze" domain (see Figure 4(c)) is even more challenging from a navigation standpoint. In its layout, the domain contains several "horseshoe"-shaped walls, consequently requiring a deep search all the way to the goal state because a finite-horizon search would get the robot stuck in a local search minimum. Similar to the "Hallway" domain, the "Maze" contains several fast-moving foreign-controlled

(a) A search tree in the Hallway domain

(b) A solution trajectory in the Hallway domain, under high uncertainty

(c) A search tree in the Maze domain

(d) A solution trajectory in the Maze domain, under high uncertainty
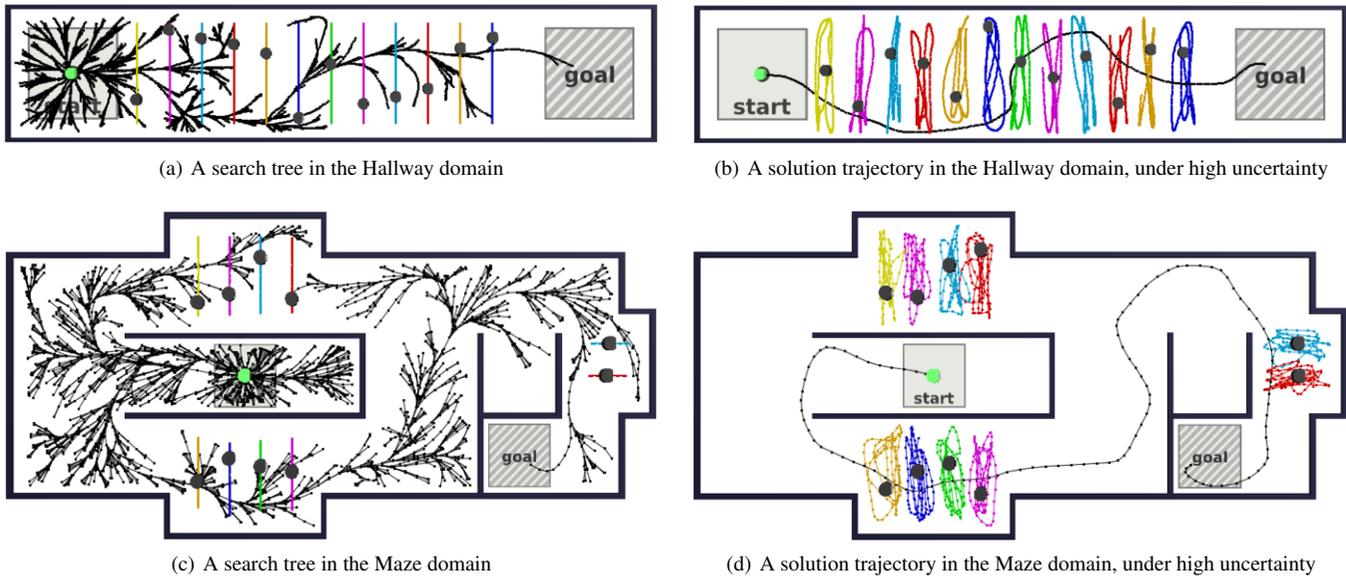
Figure 4: Experimental Domains

obstacles along the way, further increasing the difficulty of the planner's search. Again, the planner uses a deterministic prediction of the obstacles, as shown in Figure 4(c), whereas their actual motions during execution can be significantly different due to the uncertainty model (see Figure 4(d)).

5.1 Performance

The controlled variables of our experiments are the LOD time horizon t_{LOD}, the replanning interval t_{replan}, and the domain uncertainty. The most relevant performance metrics are the number of collisions that occurred during simulated execution, as well as the amount of total cumulated planning time required to get the robot from its initial state to the goal state. Combining these two variables, we can express the planner's overall performance using a single relative performance comparison metric, defined as

$$\text{performance} = (1 - \text{NormCollisions})(1 - \text{NormTime}),$$

where NormCollisions and NormTime are both values ranging from 0-1, normalized over an entire experiment. An ideal planning strategy would use a minimum amount of cumulative planning time and generate a minimum number of collisions, thus generating a maximum performance value.

In experiments, our planner was able to find valid solutions for both domains. Overall planning and execution performance trends were similar for both domains. In the following selected graphs and discussions we explain in detail how our VLOD planning approach performs under various conditions.

Figures 5(a) and 5(d) show the performance values for the Hallway and Maze domains respectively, under varying LOD-horizons and varying replanning intervals. Each data-point in the graphs was generated using 120 simulated trials, thus totaling in 6480 trials per domain. In both domains, it is clear to see that VLOD planning has a positive impact on performance for all tested replanning intervals. Generally, the VLOD planner achieves its highest performance with t_{LOD} values that are slightly greater than the corresponding replanning interval. This result makes sense, as using a t_{LOD} value lower

than the replanning interval would mean that the robot will execute partial solutions that have not been planned with the maximum level of detail, thus likely to collide with obstacles. This reasoning is verified if we look at the corresponding collision rates shown in Figures 5(b) and 5(e). Here, selecting a t_{LOD} value lower than the replanning interval results in an agressive growth of collision rates.

The true benefit of VLOD planning, however, becomes clear when looking at the total accumulated planning time in Figures 5(c) and 5(f). For example, in the Hallway domain using a replanning interval of 0.5, we see an approximately 70% decrease in planning time, when reducing t_{LOD} from its maximum of 5.7s down to 0.5s. The reason for why the VLOD approach is able to perform so much faster, lies within the fact that a lower value of t_{LOD} allows the planner's search to find simpler solutions that ignore obstacle interactions in the future. In the Maze domains, we see a similar trend of shorter planning times with smaller values of t_{LOD}, but the effect is less pronounced. The reasoning for this behavior lies in the fact that the Maze domain remains significantly difficult to solve, even when foreign-controlled bodies are ignored due to a low value of t_{LOD}. Although ignoring these dynamic interactions makes the search evidently easier, the planner still needs to find a trajectory that leads around the maze of static wall bodies without getting stuck in local minima.

Conclusively, achieving optimal planning performance depends on multiple factors. First, a replanning interval appropriate for the domain's level of uncertainty should be chosen. Given the replanning interval, maximum performance is then achieved by selecting a value for t_{LOD} that minimizes planning time and collision rate. Our overall performance metric tries to capture this trade-off, proposing a t_{LOD} value that not only reduces collisions, but that's also low on computational expense. Depending on one's particular needs and computational power available, one could weigh this metric differently to put special emphasis on either safety or computational time.

6 CONCLUSION AND FUTURE WORK

In this paper, we have introduced VLOD planning for physics-based domains with poorly predictable bodies. We have tested its perfor-
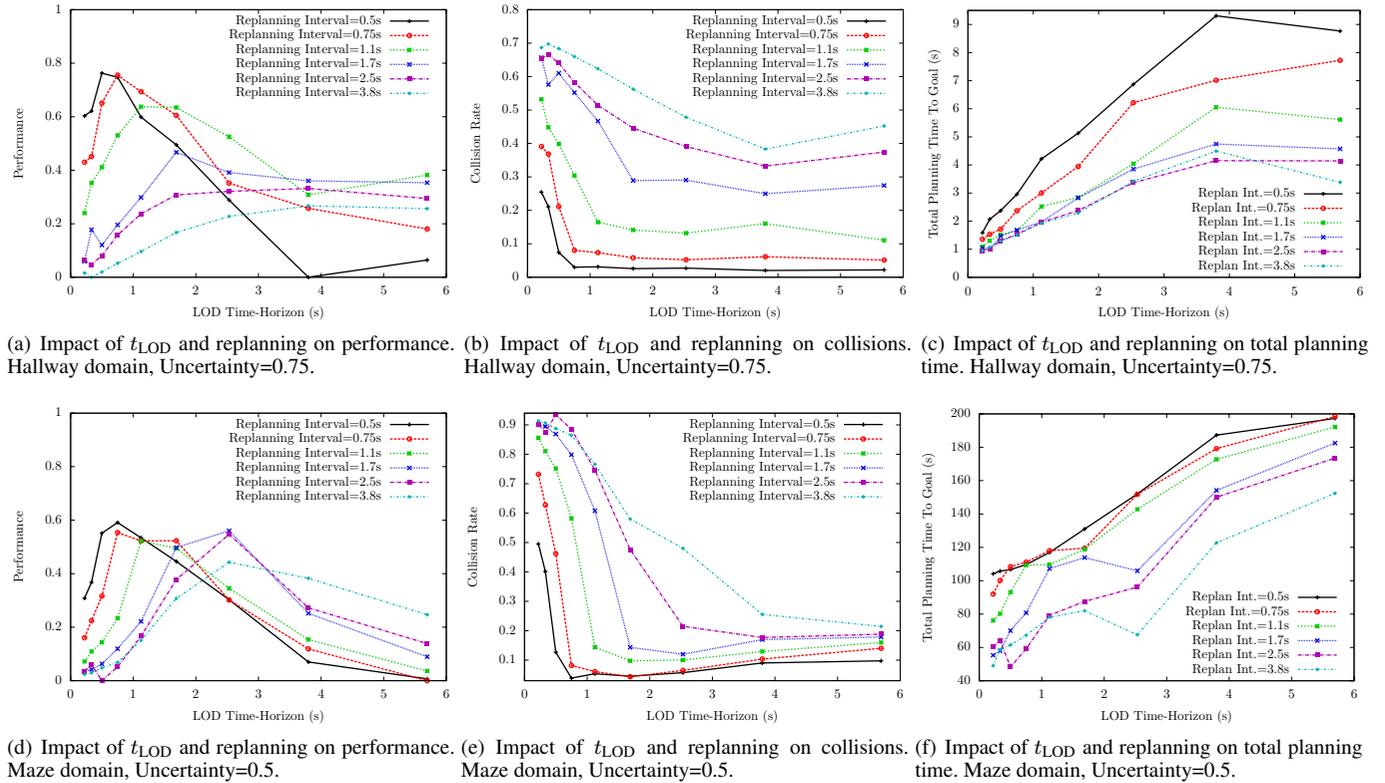
(a) Impact of t_{LOD} and replanning on performance. Hallway domain, Uncertainty=0.75.

(b) Impact of t_{LOD} and replanning on collisions. Hallway domain, Uncertainty=0.75.

(c) Impact of t_{LOD} and replanning on total planning time. Hallway domain, Uncertainty=0.75.

(d) Impact of t_{LOD} and replanning on performance. Maze domain, Uncertainty=0.5.

(e) Impact of t_{LOD} and replanning on collisions. Maze domain, Uncertainty=0.5.

(f) Impact of t_{LOD} and replanning on total planning time. Maze domain, Uncertainty=0.5.

Figure 5: VLOD performance analysis for the Hallway (a–c) and Maze (d–f) domains.

mance on multiple experiments, using a rich simulated model. We have analyzed the impact of the LOD horizon on planning performance over different replanning intervals and domain uncertainty. Our overall results clearly show that VLOD planning is able to significantly cut down computational cost at little to no expense to collision safety. We therefore conclude that VLOD planning is effective for improving planning performance in dynamic motion planning domains.

For future work, it would be interesting to extend the definition of detail beyond a time-horizon threshold only involving collisions. For example, one could modify the planner's internal planning granularity (e.g., the value of Δt, or the node selection strategy itself) as it plans further into the future to gain additional speedups without sacrificing costs. Additionally, it will be interesting to analyze how our approach could be combined with other existing replanning improvements, such as solution caching.

REFERENCES

[1] D. Baraff, 'Physically Based Modeling: Rigid Body Simulation', *SIGGRAPH Course Notes, ACM SIGGRAPH*, (2001).

[2] J. Bruce and M. Veloso, 'Safe Multi-Robot Navigation within Dynamics Constraints', *Proceedings of the IEEE, Special Issue on Multi-Robot Systems*, (2006).

[3] J. Bruce and M.M. Veloso, 'Real-Time Randomized Path Planning for Robot Navigation', *Robocup 2002: Robot Soccer World Cup VI*, (2003).

[4] S. Chakravorty and R. Saha, 'Hierarchical motion planning under uncertainty', in *Decision and Control, 2007 46th IEEE Conference on*, pp. 3667–3672, (2007).

[5] B. Donald, P. Xavier, J. Canny, and J. Reif, 'Kinodynamic motion planning', *Journal of the ACM (JACM)*, **40**(5), 1048–1066, (1993).

[6] D. Ferguson, N. Kalra, and A. Stentz, 'Replanning with RRTs', *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 1243–1248, (2006).

[7] 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).

[8] R. Knepper, S. Srinivasa, and M. Mason, 'Hierarchical planning architectures for mobile manipulation tasks in indoor environments', in *Proceedings of ICRA 2010*, (May 2010).

[9] S.M. LaValle, 'Rapidly-exploring random trees: A new tool for path planning', *Computer Science Dept, Iowa State University, Tech. Rep. TR*, 98–11, (1998).

[10] S.M. LaValle and J.J. Kuffner Jr, 'Randomized Kinodynamic Planning', *The International Journal of Robotics Research*, **20**(5), 378, (2001).

[11] N.A. Melchior, J. Kwak, and R. Simmons, 'Particle RRT for Path Planning in very rough terrain', in *NASA Science Technology Conference 2007 (NSTC 2007)*, (2007).

[12] N. Vahrenkamp, C. Scheurer, T. Asfour, J. Kuffner, and R. Dillmann, 'Adaptive motion planning for humanoid robots', in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 2127–2132, (2008).

[13] S. Zickler, *Physics-Based Robot Motion Planning in Dynamic Multi-Body Environments (Thesis Number: CMU-CS-10-115)*, Ph.D. dissertation, Carnegie Mellon University, May 2010.

[14] S. Zickler and M. Veloso, 'Efficient physics-based planning: sampling search via non-deterministic tactics and skills', in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 27–33, (2009).

[15] S. Zickler and M. Veloso, 'Tactics-Based Behavioural Planning for Goal-Driven Rigid-Body Control', *Computer Graphics Forum*, **28**(8), 2302–2314, (2009).

[16] M. Zucker, J. Kuffner, and M. Branicky, 'Multipartite rrts for rapid replanning in dynamic environments', *Proc. IEEE Int. Conf. on Robotics and Automation*, 1603–1609, (2007).