A Framework for Planning Motion in Environments with Moving Obstacles

Samuel Rodriguez, Jyh-Ming Lien, and Nancy M. Amato

***Abstract*— In this paper we present a heuristic approach to planning in an environment with moving obstacles. Our approach assumes that the robot has no knowledge of the future trajectory of the moving objects. Our framework also distinguishes between two types of moving objects in the environment: *hard* and *soft* objects. We distinguish between the two types of objects in the environment as varying application domains could allow for some collision between some types of moving objects. For example, a robot planning a path in an environment with people could have the people modeled as circular disks with a safe zone surrounding each person. Although the robot may try to stay out of each safe zone, violating that criteria would not necessarily result in planning failure. We will show the effectiveness of our planner in general dynamic environments with the soft objects having varying behaviors.**

I. Introduction

Planning a path for a robot has been widely studied. There has been quite a lot of work on planning a path for a holonomic, free-flying, robot [1]–[3], and planning a path for a nonholonomic robot with constraints on the movement of the robot [4]–[6]. One problem that has been less studied is planning a path for a robot with constraints in a realistic environment. This includes environments that can change dynamically and that include other agents in the environment. The problem we will study in this paper includes dynamically moving obstacles and agents. This has applications in robotics, graphics, virtual reality and games.

The goal of the robot is to avoid the moving obstacles in the environment as well as the other agents. Although we cannot guarantee to avoid objects in the environment (as the environment is unknown) we will present a heuristic planner that works well in many planning instances.

In our framework we distinguish between *hard* and *soft* moving objects. *Hard* objects in the environment are moving or static objects that the robot will attempt to avoid as a highest priority. Collisions with *hard* objects will result in planning failure for the robot. These types of objects could include objects such as a wall, automatic sliding door, carousel or any other barrier in the environment. The robot is allowed some "collision" with *soft* objects. *Soft* objects could be considered other agents in the environment. Examples of these types of agents could include members of a crowd in a virtual reality scenario or pursuers in a pursuit-evasion application. These are some examples where it could be acceptable to have some amount of "collision" between the robot and soft objects, as shown in Figure 2. For example, a robot could enter a safe zone surrounding a person without actually colliding with the person, Figure 2(a). A robot could also enter an area, represented as a soft object, which could increase the probability of being caught in a pursuit-evasion application, Figure 2(b). In an application where a robot can endure a certain amount of damage and certain areas or agents can cause the damage, soft objects can represent these areas or agents, as shown in Figure 2(c).

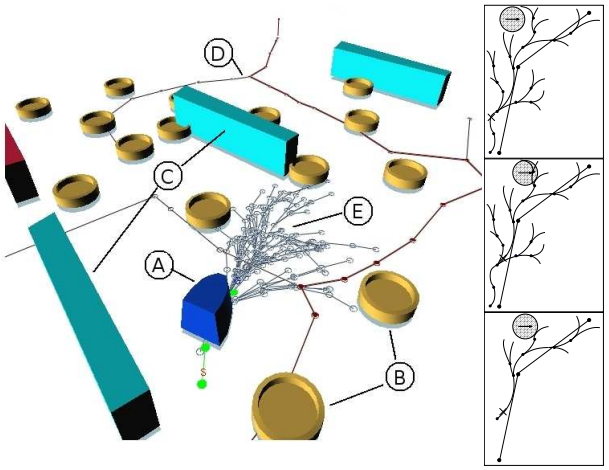


Fig. 1. A test environment showing different aspects considered in our planning framework. These aspects include (a) a robot, (b) *soft* objects (other agents), (c) *hard* objects (moving and static), (d) a global-dynamic roadmap and (e) kinodynamic local plans (also showing the update step on the right).

Planning among moving obstacles is a very difficult problem. It is made even more difficult in environments when the future trajectories of the moving objects are unknown. In this work, the only information we are assuming the robot has access to during planning is the position and orientation information of the other objects at each planning step. It is important to note that we give no guarantee of planning success. This is in large part due to the dynamic nature of the environments as well as time constraints. Whereas many other approaches have constrained the problem to make it easier to solve and allow for success we put very few constraints on the problem. Examples of constraints that have been used include restricting the robot and obstacle shape (usually to disks) [7], [8], assuming constant or known velocities [8] or knowing the dynamics of the moving obstacles a priori [9].

Our approach. We propose a two-stage approach to planning in environments with different types of moving obstacles. The first stage maintains approximate information about the dynamic global connectivity of the environment. In particular, a roadmap is constructed that considers only the hard objects and may also use a conservative approximation

S. Rodriguez and N. M. Amato with Parasol Lab, Department of Computer Science,Texas A&M University ({sor8786,amato}@cs.tamu.edu)

J-M. Lien with Department of Computer Science, George Mason University (jmlien@cs.gmu.edu)

This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-0326350, and by the DOE. Rodriguez supported in part by a National Physical Sciences Consortium Fellowship.

(a) (b) (c)

Fig. 2. Examples of soft objects: (a) a person modeled as a disk with a safe surrounding area, (b) a pursuing agent with increasing probability of capturing other agents closer to the center, (c) an area, agent or danger zone with increasing probability of causing damage to a robot depending on area of collision.

of the robot. The roadmap is updated as the hard objects move. Hence, the global roadmap encodes a conservative approximation of the connectivity of the environment when only hard objects are considered. The second stage uses a global path extracted from the dynamic roadmap to locally plan in the environment. These kinodynamic local plans take into account the soft objects, or neighboring agents, to find safe plans. The local paths are also updated as the environment changes.

Benefits of our approach over others proposed for dynamic environments include:

- Integrating global planning amidst moving objects and kinodynamic local planning.
- Kinodynamic local paths are generated incrementally to follow a global path.
- Focused computation to local regions of the environment where the robot is positioned
- Consideration of other agents in the environment and how to plan around them.

Outline. The remainder of the paper is outlined as follows: related work is described in Section II, we describe aspects of planning among moving obstacles in Section III and IV, in Section V we describe how kinodynamic planning is done among other agents and we conclude with results and conclusions.

II. Related Work

In this section we describe work that is related to our path planning technique. The general methods that we will discuss include global randomized motion planning techniques, and kinodynamic path planning in static and dynamic environments. Although we cannot give a full overview of related work, we will describe work that is closely related to our approach.

A. Motion Planning

Probabilistic Roadmap Methods (PRMs) [1] are one of the most popular types of sampling-based planners. PRMs construct a roadmap in a two-staged approach. First configurations, or placements of the robot, are randomly generated. The free configurations are then connected using local planners. After the roadmap has been constructed, a solution can be extracted by connecting both start and goal configurations to the resulting roadmap. There have been many attempts to improve the node generation portion of the PRM framework. These approaches try to generate nodes in interesting areas such as near the obstacle surface [2], [3] or near the medial axis of the configuration space [10].

A tree-based path planner was developed in [11], [12] that is useful for exploring C-space. This path planning technique is known as the Rapidly-Exploring Random Tree (RRT). A similar tree-based planner, Expansive Space Tree (EST), was developed in [13]. These tree-based methods work well for single query problems by only exploring the relevant portions of the configuration space needed to solve a given query.

B. Kinodynamic Planning

Static Environments.
In [4], [5], a randomized kinodynamic approach is presented to planning a path in a static environment. The state space is explored by applying a set of allowable control inputs in order to grow a tree. The exploring of the state space is complete when the goal configuration can be reached and a path in the tree can be extracted.

In [14] an RRT based planner is proposed which performs better over time, improving paths, over a number of runs, in a given environment. In this work, an RRT is integrated with a "way-point" cache for improved performance in static environments. It is not described how these way-points are generated or how the way-points could be updated in a dynamic environment.

In [15], a modified expansive space tree approach to planning a path for a robot with motion constraints is proposed to search the space. The goal is to find a path that has a low cost and has a relatively straight path.

Dynamic Environments.
An early approach to kinodynamic planning among moving obstacles was proposed in [7]. An EST approach is used to plan a path for a circular robot through an environment consisting of circular obstacles with restrictions on the velocity and shape of the obstacles. In this approach, when uncertainty in the environment is found, a path is completely replanned.

A PRM based approach is proposed for planning paths among moving objects in [16]. A roadmap is built and updated according to the movements of the obstacles. In this work, they attempt to update only the necessary portions of the roadmap, that are relevant to the moving obstacles. If simple repairs cannot be made in the roadmap, then an RRT is used to repair the roadmap. Complete paths for the robot are obtained for the robot from the roadmap and a path to the goal is always required.

Another PRM based approach to planning a path in a dynamic environment is proposed in [9]. The roadmap used during the planning phase is one that is constructed as a preprocessing step and does not change as the environment changes. This alone could cause many problems in environments that are constantly changing. Local trajectories are then planned along valid portions of the roadmap from the start to goal configuration. This approach also assumes the dynamics of the moving obstacles are known a priori. Although this may work in some situations, in highly dynamic environments, this could result in unnecessary computation when constantly replanning from the start to goal configuration and when the dynamics of the obstacles are not known beforehand.

In [8], the problem of planning a safe path for a robot disk is proposed among unpredictably moving obstacles. The obstacles are modeled as disks growing over time given their known maximum velocities, modeling unpredictable motion.

Although this would model unpredictable motion, given the restrictions on the shape and velocities of the robot and obstacles, many situations could occur where planning a path to the goal is unfeasible. Some of these situations include having a large number of moving obstacles or having some obstacles with large maximum velocities.

III. Planning Among Moving Obstacles

The framework proposed here for planning a path for a robot in an environment with moving obstacles is a two-staged approach. A rough outline of this approach, taken at each planning step for the robot, can be seen in Algorithm 1. The first stage will use a dynamic global roadmap when needed in order to find a conservative path avoiding hard objects. The dynamic global roadmap should be maintained as much as possible as the environment changes. Changes in the environment can change the connectivity of the free-space, which can then be reflected in the dynamic global roadmap by repairing the roadmap.

The second stage of our approach will involve planning local valid paths in the environment taking into account constraints on the robot's movement. Another thing considered in the second planning phase are soft objects or neighboring agents. The local paths obtained attempt to avoid these neighboring agents as much as possible. However, in our approach it cannot be guaranteed that these agents will be avoided.

Algorithm 1 Robot's planning stages during each time step.

```
# Stage 1
Update Global Roadmap
regular_planning = TRUE
if not Valid Global Path then
    Global Path Repair
    if Path To Goal Exists then
        Set Global Path
    else
        regular_planning = FALSE
    end if
end if
# Stage 2
if regular_planning then
    Plan Along Global Path
else
    Alternative Planning
end if
```

IV. Dynamic Global Roadmaps

We use a dynamic global roadmap to represent the connectivity of the free space in the environment at a given time step. In this stage of planning, only the hard objects in the environment are considered. In this section we describe how the roadmap is maintained. When the roadmap is needed in planning, it is important that the connectivity of the roadmap be as accurate as possible. This is so a path to the goal can be found. The roadmap is created using PRM with MAPRM [10] used for node generation. MAPRM was selected as the node generation method as this will generate configurations near the medial axis of the free space allowing for a maximal amount of space between neighboring obstacles.

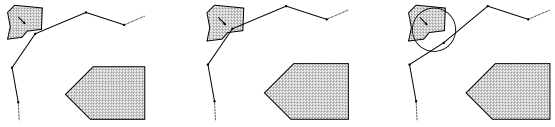


Fig. 3. In this figure (a) a global path obtained by the robot, (b) the global path becomes invalid, and (c) the global path is repaired.

A. Roadmap Validity

We use the validity of the roadmap to mean that a path extracted at a given time will not include nodes or edges in collision with hard objects. The validity of the roadmap can be checked at any point during planning. Our roadmap validity tests include removing any nodes or edges that are in collision with the hard objects. While removing the collision nodes in the roadmap, we keep track of the colliding nodes for use in next roadmap repair stage.

B. Roadmap Repair

In order to maintain the connectivity of the free space in the environment, we want to attempt to repair any disconnections that may have occurred. The first step to attempt to maintain connectivity is to approximate the previous coverage. We achieve this by generating samples in areas where colliding nodes were previously found. This can be seen as an OBPRM node sampling method sampling nodes near the surface of moving objects [2]. This is only one way to maintain coverage that has worked empirically.

We then attempt to ensure the connectivity of the roadmap by connecting smaller connected components to the largest connected component in the environment. The smaller connected components include the nodes that have been regenerated and any other unconnected components that were previously in the roadmap. The connectivity of the global roadmap is considered throughout the planning stage and should be maintained while planning.

C. Query

When a global path is needed during planning, the roadmap is queried. A path should be returned that connects the goal to the current configuration of the robot. This path will be a continuous sequence of configurations, approximating a free path in the environment. It cannot be guaranteed that the roadmap contains a valid path. This could happen, for example, when the goal position is in collision or when there is no path in the environment connecting the current start and goal positions. It could also happen if a path to the goal does not exist in the environment or if the roadmap does not appropriately reflect the free space in the environment. In the event that a global path does not exist, it is up to the robot to determine what to do, described in more detail in Section V-D.

D. Maintaining a Global Path

Although the roadmap is maintained throughout the planning process, the roadmap is only queried if the robot does not already have a valid global path. The global path is used

to approximate a potential path the robot can take to reach the goal. The robot keeps track of the global path taken as it progresses through the environment. As the hard objects in the environment move, this global path can become invalid. In order to maintain some of the local paths generated in the next planning stage, the global path is altered, if needed, in a similar way the roadmap is repaired. If a portion of the global path becomes in collision, the nodes causing the collision are regenerated within a certain distance of the collision configuration. This process can be seen in Figure 3 and is done to maintain as close of an approximate representation to the previous path as possible. If the global path cannot be repaired, the global roadmap is queried in order to obtain another global path which will be used in planning local paths.

V. Kinodynamic Local Planning

Kinodynamic planning involves the planning for a robot based on a given set of valid control inputs resulting in valid paths for the robot. We will use the approximate path to goal obtained from the global roadmap to guide local planning along the path. This planning stage not only considers hard objects but will consider interactions with soft objects and how these soft objects can be avoided. The soft objects considered at this stage are the objects within the robot's viewing range. The local paths are also updated at each stage during planning so paths can be selected when needed that avoid both hard and soft objects. Due to space constraints, we give a detailed overview of some of the aspects that should be considered, but omit some of the robot-specific details.

A. Planning Along A Global Path

The global path obtained is used to plan kinodynamic local paths. We use the approximate global path as a guide and do not require that the robot strictly follow the path. In this way, the local paths obtained follow the global path and likely remain free from collision with the hard objects, although we still need to verify this criteria during this phase.

An overview of the algorithm can be seen in Algorithm 2. For a given number of iterations K, a tree, $\mathcal{T}_l$, is expanded from a random start configuration c_{src} in the tree (line 2). Next a subgoal is found taking into account the global path (line 3). A subgoal is selected to be a configuration along or near the path, in order to plan along the global path. Subgoals biased toward the global path are selected with probability δ and biased toward the goal with probability $1 - \delta$. The distance in which a subgoal can be generated near the global path is a predefined value set by the user. One possibility, which we have not yet implemented, is to place the subgoal in such a way that it avoids the neighboring hard and soft objects. Next a control input U toward the subgoal is applied while there is no collision with any object in the environment and applying U results in a satisfactory outcome.

It is important to note that we do not attempt to locally plan an entire path to the goal (unless the goal is reachable within a given number of iterations K). We plan incrementally along the global approximate path. This avoids planning the more expensive local paths in regions potentially far from the current location of the robot, which are more likely to become invalid in highly dynamic environments. The areas further from the robot are also areas in which the robot may not have local information about such as the positions of soft objects. This is something that other PRM based planners do not consider [9], [16].

Algorithm 2 Planning Along a Global Path.

Require: tree $\mathcal{T}_l$, global path P, Iterations K, Max_i, and Min_i

1: **for** $i = 1 \ldots K$ **do**
2: c_{src} = random node in $\mathcal{T}_l$
3: subgoal = configuration given P, δ
4: U = control toward subgoal
5: iter = 0, Collision = false, $c_{new} = c_{src}$
6: **while not** Collision **and** iter $< Max_i$ **do**
7: c_{prev} = c_{new}
8: c_{new} = apply U
9: Collision = isCollision(c_{new})
10: **if** Collision **or not** satisfactory(c_{new}, c_{prev}, subgoal) **then**
11: c_{new} = c_{prev}
12: break
13: **end if**
14: iter++
15: **end while**
16: **if** iter $\geq Min_i$ **then**
17: $\mathcal{T}_l$.addNode(c_{src}, c_{new})
18: **end if**
19: **end for**
20: **return** $\mathcal{T}_l$

B. Considering Soft Objects

When trying to avoid soft objects at each planning step, it is ideal to have local plans that are clear of soft objects. We do a coarse check of this criteria. This can be considered pruning edges in the tree $\mathcal{T}_l$ or local paths that collide with the soft objects. It is also done before the local paths are updated so that valid local paths can be created from collision free paths.

The pruning of the local paths are done at a much coarser level along each edge in the local path. We do this by checking configurations along the start, midpoint and end of an edge in the local path. An edge colliding with any of the local soft objects along any of those configurations can be discarded along with the remainder of the path connected to the collision edge as the path is no longer valid. Along with invalid edges in the local paths, we also remove parts of the local paths that are no longer reachable. An example of these steps can be seen on the right side of Figure 1.

This type of tree pruning is useful when attempting to totally avoid the soft objects which is what is considered in this paper. Although not needed here, for a crowd simulation application, where the robot could be totally surrounded by soft objects, another approach may be to only prune old portions of the tree and weight local paths based on the amount of collision. This could be measured by the penetration depth with soft objects or number of neighboring soft objects along an edge.

C. Extracting Local Paths

Once the local paths have been updated, a local path is selected for the robot to follow. We select this path from the available safe local paths, that also keep the robot near the global approximate path. A local path LP_i is selected such that the end point of the path is the closest node in $\mathcal{T}_l$ to an unreached point along the global path. Although there are many possible ways to determine which local path to take at a given time step, this is an intuitive way to select the local path. We however do not consider the entire local path. Rather, only a portion of the local path is used, as configurations further along the local path are more likely become invalid later. For a crowd simulation application, the local path LP_i could also be selected based upon the weights assigned to each local path and the likelihood of the path being free of soft agents.

D. Alternative Safe Planning

There are many situations in which the robot may not have a good candidate local path. In this case, some safe local planning method can be selected and is dependent on the application domain. This could include a stopping behavior where the robot waits until a safe path can be found, although this may not work in the presence of hostile agents. Another behavior could be to still try and progress toward the goal, considering collision inevitable. This behavior is reasonable depending on the application. For example, if the success of the robot is determined by the progress made toward the goal, then the robot may want to consider continuing, even though it could result in collision. An example of this could be in a game situation, where an agent attempts to reach a goal location given a certain number of chances (as in football). The behavior assigned to the robot here, and in the experiments, is one in which the robot will select a path that minimizes the time spent in collision with the soft objects and will be studied further in Section VI.

VI. Details and Experimental Results

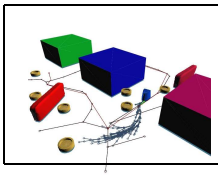
In this section, we will discuss some of the implementation details of our framework and present the performance and flexibility of our proposed approach for planning paths in highly dynamic environments under various situations.

A. Experiment Setup

As stated, the goal of the robot is to find a path through the environment avoiding, if possible, hard and soft objects. The start and goal configurations are predefined and remain the same through each run. A test run is considered a failure if collision with a hard object occurs. Collisions with soft objects in the environment are accumulated and reported; the only soft objects that need to be tested are the objects within the robot's view radius. During each planning run, the start positions of the soft objects are randomly generated. Each table will list the number of soft objects in the environment, average number of planning steps required during the planning process, percentage of successful runs, average number of soft object collisions and average time spent in the planning process.

In Sections VI-B and VI-C the soft objects have a basic flocking behavior applied to them. This means that basic flocking behavior properties are applied which are typically composed of separation, alignment and coherence with respect to their neighboring agents [17]. By emphasizing separation, we are able to have the agents cover the environment, making planning more difficult. In Sections VI-D, the soft objects have an attacking behavior (described later).

TABLE I
Corridors



# Soft Obj.	Steps	Success %	Soft Coll.	Time (sec)
0	407	100	0	0.0257
1	475	100	4	0.0272
5	569	100	18	0.0273
10	504	80	42	0.0266
20	457	70	43	0.0245

All of our experimental results are obtained using a notebook computer with a 1.7 GHz CPU and 1 Gb memory. Animations and more detailed results are available at our website[†].

B. Corridors Environment

The first environment tested is the corridors environment, shown in Table I(left). The environment consists of three static blocks, creating two distinct corridors in the environment. Two moving hard objects are present in the environment (the vertical plates in the environment) one which has only rotational motion and one with only translational motion. The number of soft objects in the environment ranges from 0 to 20. As mentioned, these agents have a basic flocking property as they move through the environment. The robot must navigate through one of the corridors in order to reach the goal from the start configuration.

In Table I, it can be seen that when planning among 0 to 5 soft objects in the environment, safe planning can be done relatively easily. Although some soft collision does occur, the amount of collision is relatively small. The amount of planning time and steps required does increase in these cases although it is only slightly.

When planning among 10 and 20 soft objects, the planner is able generate successful plans 80 and 70 percent of the time, respectively. This is very promising considering the few number of restrictions put on the the motion and shape of the hard and soft objects. The number of planning steps and planning time for 10 soft object is slightly higher than for 20, which can be contributed to the planners ability to plan safer paths in the environment, even away from the goal if necessary. Shown in Table I, is the robot traveling through one of the corridors among 10 soft objects. The potential kinodynamic local paths are also shown.

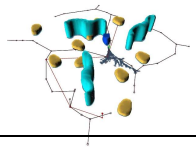
C. Rotors Environment

In the rotors environment, shown in Table II(left), the robot must navigate around the moving rotors trying to avoid soft objects in the process. The start and goal configurations are at opposite corners of the environment. The hard objects in the environment are the three large rotor-shaped objects. Two of these hard objects only have rotational motion, while the third hard object has both rotational and translational motion.

[†]`http://parasol.tamu.edu`

TABLE II

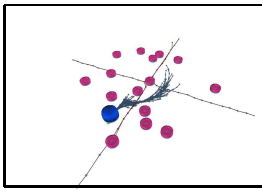
ROTORS ENVIRONMENT



# Soft Obj.	Steps	Success %	Soft Coll.	Time (sec)
0	625	100	0	0.0186
5	885	100	34	0.0170
10	1060	80	83	0.0137
20	828	60	113	0.0188

TABLE III

ATTACKING AGENTS



# Soft Obj.	Steps	Soft Coll.	Time (sec)
2	162	6	0.0330
4	162	16	0.0368
6	187	25	0.0441
8	258	37	0.0468
10	381	52	0.0517
15	226	64	0.0649

The soft objects are the tear-drop shaped objects which also have the basic flocking behavior. Shown in Table II(left) the robot is planning among 10 soft objects and shown with the current kinodynamic local paths.

In Table II a similar trend can be seen as in the previous environment. For a small number of soft objects, the robot is able to plan safe plans with relatively few soft collisions. For 10 and 20 soft objects, successful plans cannot always be found. The success rates still seem promising, given how dynamic the environment is and how little information the robot is using while planning a path. For 10 soft objects, the number of steps needed to generate a plan is generally more, as before, since the robot may spend more time planning given that safer plans can be found while avoiding the soft objects.

It is also important to note that the planning time is relatively similar regardless of the number of agents. This is in part due to the view radius of the robot, only sensing so many agents at each time step. It can also be attributed to the tree pruning done, resulting in fewer nodes to consider. Also, as the number of agents increases, invalid local paths may be found sooner when planning resulting in less computational work and potentially fewer safe paths. This can be seen in both the success rate and number of soft collisions.

D. Attacking Agents

This environment is composed of only soft objects and does not include hard or static objects. Additionally, these soft objects have an attacking behavior. If the robot is visible to these agents, then the robot will be approached, otherwise the agents have the same basic flocking behavior as before. The planner is tested with 2 to 15 soft objects in the environment. An example environment consisting of 15 soft objects can be seen in Table III(left). It is clear from the figure that planning a path in such an environment is very difficult, especially given the behavior of the soft objects in the environment. The robot must travel from one end of the environment to the other, avoiding the soft objects if possible.

As shown in Table III the trend is nearly always followed in respect to the average number of steps needed, number of soft collisions, and average planning time. While avoiding soft collisions when there are few soft objects, it gradually become more difficult as the number of soft objects increases. Although able to deal with about 10 soft objects in the environment, it becomes increasingly difficult for the robot to plan evasive actions. Rather than moving through the environment until a free path exists, the robot is swarmed and has to attempt to minimize collision with the soft objects.

VII. CONCLUSION

In this paper we have presented a preliminary approach to the problem of planning a path in environments with moving obstacles. The proposed framework takes into account different types of moving obstacles and could have applications in many application domains. While we are at the early stages of this work, the results thus far seem promising.

REFERENCES

[1] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective*. Natick, MA: A.K. Peters, 1998, pp. 155–168, proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[3] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 1999, pp. 1018–1023.

[4] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 473–479.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[6] A. M. Ladd and L. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," *Algorithmic Foundation of Robotics VI*, pp. 297–312, 2005.

[7] R. Kindel, D. Hsu, J. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 537–543.

[8] J. van den Berg and M. Overmars, "Planning the shortest safe path amidst unpredictably moving obstacles," *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pp. 885–897, 2006.

[9] J. P. van den Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot. Automat.*, pp. 885–897, 2005.

[10] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 1999, pp. 1024–1031.

[11] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 995–1001.

[12] S. M. LaValle and J. J. Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000, pp. SA45–SA59.

[13] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1997, pp. 2719–2726.

[14] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, Switzerland, 2002.

[15] J. Phillips, N. Bedrosian, and L. Kavraki, "Guided expansive spaces trees: A search strategy for motion- and cost-constrained state spaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2004, pp. 3968–3973.

[16] L. Jaillet and T. Simeon, "A PRM-based motion planner for dynamically changing environments," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2004.

[17] C. W. Reynolds, "Flocks, herds, and schools: A distributed behaviroal model," in *Computer Graphics*, 1987, pp. 25–34.