2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Dynamic Diffusion Maps-based Path Planning for Real-time Collision Avoidance of Mobile Robots

Sanghyun Hong*†, Jianbo Lu†, and Dimitar P. Filev†

***Abstract*—Given a route to a destination, a mobile robot still needs to locally plan a path to avoid collisions in continuously changing environment, e.g., a hall with pedestrians and moving obstacles. In this paper, diffusion maps are applied to find a local path for reaching a goal and avoiding collisions simultaneously. The proposed path planning algorithm plans a local path by utilizing a receding horizon approach, and therefore the algorithm repeats planning at every sample time. With this approach, mobile robots do not have to carry a prior map all the time because updated environment information is used for planning at every sample time. Extensive simulation is performed in different scenarios and demonstrates a good performance in collision avoidance.**

I. Introduction

Mobile robots have been developed for a wide range of purposes, e.g., transportation, searching, and surveillance. In order to ensure safety, mobile robots should plan a path adaptively to continuously changing environment, e.g., streets with newly constructed buildings and moving obstacles. Particularly, importance of safe path planning is increasingly stressed as the era of autonomous vehicles is upcoming.

A variety of studies on path planning have been proposed in literature. For example, in [1], Hart et al. described the A^* algorithm for finding minimum cost paths with a fixed graph. To account for changes in environment, Stentz [2] proposed the D^* path planning algorithm which incrementally repairs previous paths. Furthermore, D^*-Lite was presented by Koenig et al. in [3] as a simple but efficient algorithm comparable to D^*. Sharma et al. [4] and Likhachev et al. [5] proposed the Anytime A^* and Anytime Reparing A^* algorithms, respectively, to improve computation cost of A^* algorithm by finding a suboptimal solution quickly and then refining it. Chen et al. [6] utilized motion primitives based on a path planned with diffusion maps, and they also avoided collisions with policies generated by a deep reinforcement learning in [7].

In this paper, we will focus on planning a path to avoid collisions in dynamic environment while trying to go to a destination. In order to plan a path avoiding collisions, finding an underlying geometry of a grid map is essential. Therefore, we will apply diffusion maps for path planning since it involves transition probabilities between grid points and, therefore, naturally finds a feasible path. In addition, a receding horizon approach will be used, in which diffusion maps are computed at every sample time, whereas an existing algorithm [6] pre-computes diffusion maps offline. With the proposed algorithm, we do not need to store a prior map in mobile robots for computing diffusion maps and use motion primitives. This algorithm is extensively simulated in different scenarios and demonstrates a good performance in collision avoidance.

∗ Corresponding author

† S. Hong, J. Lu, and D. P. Filev are with Robotics & AI, Research & Advanced Engineering, Ford Motor Company, Dearborn, MI, USA (e-mail: {shong7; jlu10; dfilev}@ford.com).

II. Diffusion Maps for Path Planning

In order to generate a path, given map grid points, we need to find out geometry or underlying manifold of those map grid points. This paper utilizes diffusion maps to discover the geometry of the given map grid points. In this section, overview of the diffusion maps are presented (see [8][9] for details).

For the diffusion maps, each map grid point is regarded as a state and transition probabilities between grid points are defined, and hence it is called a diffusion process. The transition probabilities are calculated based on a diffusion kernel which provides local similarity measures. This paper uses the Gaussian diffusion kernel:

$$k(m_i, m_j) = \exp\left(-\frac{||m_i - m_j||^2}{\sigma}\right), \tag{1}$$

where m_i and m_j are the position vectors representing the i^{th} and j^{th} grid points, respectively, and σ is the variance parameter. Then, the kernel matrix K is defined as

$$K = \begin{bmatrix} k(m_1, m_1) & \cdots & k(m_1, m_n) \\ \vdots & \ddots & \vdots \\ k(m_n, m_1) & \cdots & k(m_n, m_n) \end{bmatrix}, \tag{2}$$

where n represents the number of map grid points.

The probability p_{ij} of transition from m_i to m_j is calculated as

$$p_{ij} = \frac{k(m_i, m_j)}{\sum_{j=1}^{n} k(m_i, m_j)}. \tag{3}$$

With the transition probabilities, we can construct the diffusion matrix P of size $n \times n$. By denoting a diagonal matrix, whose diagonal elements are $1/\sum_{j=1}^{n} k(m_i, m_j)$, as D^{-1}, the diffusion matrix is expressed as a Markov matrix:

$$P = D^{-1}K, \tag{4}$$

which is not a symmetric matrix.

If the diffusion process runs forward t steps, the t-step diffusion matrix is given as follows:

$$P^t = P \times \ldots \times P = \begin{bmatrix} p^t_{11} & \cdots & p^t_{1n} \\ \vdots & \ddots & \vdots \\ p^t_{n1} & \cdots & p^t_{nn} \end{bmatrix}. \quad (5)$$

Note that an element p^t_{ij} represents the sum of probabilities of all paths that we can reach from m_i to m_j after t step transitions.

The diffusion distance between m_i and m_j is defined with the t-step diffusion matrix:

$$d^t_{ij} = \sum_{k=1}^{n} (p^t_{ik} - p^t_{jk})^2. \quad (6)$$

Note that the diffusion distance becomes small when p^t_{ik} and p^t_{jk} are roughly equal, and this happens when m_i and m_j are well connected via m_k [8]. Therefore, the diffusion distance implies the distance measured along the underlying geometry of map grid points.

We approximate the diffusion distance through dimensionality reduction which transforms n-dimensional space into $r(< n)$-dimensional space, called a diffusion space. With a symmetric matrix $P^{'} = D^{1/2}PD^{-1/2} = D^{-1/2}KD^{-1/2}$ defined, there exists the orthonormal eigenvectors of $P^{'}$ such that

$$P^{'} = S\Lambda S^{\top}, \quad (7)$$

where S represents the orthogonal eigenvector matrix and $\Lambda = diag(\{\lambda_k\}^n_{k=1})$ represents the eigenvalue matrix, respectively. Then, the diffusion matrix P is given as

$$P = V\Lambda V^{-1}, \quad (8)$$

where $V = D^{-1/2}S$ is the right eigenvector matrix of P, while $V^{-1} = S^{\top}D^{1/2}$ is the left eigenvector matrix. By selecting the r dominant eigenvalues among $\lambda_1 > \ldots > \lambda_r > \ldots > \lambda_n$, a matrix H is defined as

$$H := V_r\Lambda_r = \begin{bmatrix} \lambda_1 v_{11} & \cdots & \lambda_r v_{1r} \\ \vdots & \ddots & \vdots \\ \lambda_1 v_{n1} & \cdots & \lambda_r v_{nr} \end{bmatrix}, \quad (9)$$

where $\Lambda_r = diag(\{\lambda_k\}^r_{k=1})$ has the dominant r eigenvalues and V_r only contains the first r column vectors of V. Note that the row space of H is called the diffusion maps. The diffusion maps preserve distance ordering of the original space, i.e., the diffusion distance between points m_i and m_j can be represented by the distance in the r-dimensional space:

$$d^t_{ij} \equiv ||row_i(H) - row_j(H)||^2, \quad (10)$$

where $row_i(H)$ and $row_j(H)$ represent the i^{th} and j^{th} row vectors of H (see [8][9] for further details and proof).

In path planning, we will find the map grid point m_j, which is closet to m_i in regard to the diffusion distance, i.e., the smallest d^t_{ij} in (10), recursively until we reach a goal point.

III. Collision Avoidance Path Planning with Diffusion Maps on Receding Horizon

A diffusion maps-based path planning algorithm has been proposed in [6]. The algorithm proposed in [6] first computes the diffusion maps with a prior map data offline and keeps the pre-computed diffusion maps at all times. In addition, in order to avoid collisions with moving obstacles, it utilizes the motion primitives to deviate from the previously planned path, and then it generates a new path with the pre-computed diffusion maps.

This paper aims at avoiding collisions with obstacles by solely using a diffusion maps-based approach with neither prior map data nor motion primitives, given a pre-planned route to a destination, i.e., a global path. Note that the proposed algorithm introduces receding horizon to the diffusion maps to avoid collisions with moving obstacles, which is the main contribution and called *Dynamic Diffusion Maps* in this paper.

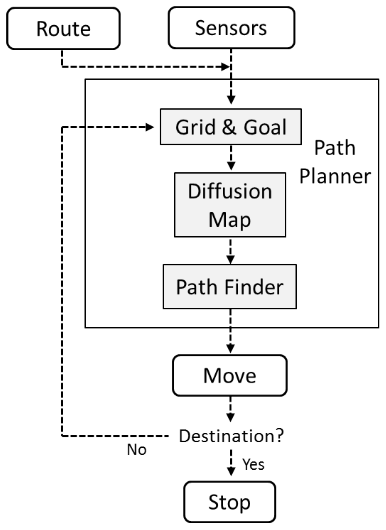


Fig. 1: Collision Avoidance based on Dynamic Diffusion Maps

Fig. 1 shows a block diagram for the proposed algorithm using the diffusion maps on receding horizon. The *Route* block in Fig. 1 represents a global path planner which creates a route from a start point to a destination. The planned global path will be followed by the algorithm proposed in this paper. The *Sensors* block represents perception sensors, e.g., cameras, lidars, and radars, to account for the dynamic environment. The sensor signals from this block will be used for constructing a local grid map and determining a local goal point, as will be discussed in the next subsection.

A. Grid & Goal

The *Grid & Goal* block in Fig. 1 represents a module that constructs a local grid map based on sensed data, e.g., distance to obstacles and angles of laser beams, and also sets up a local goal grid point based on the local grid map and the global path. In Fig. 2, the cyan dots represent the local grid map constructed based on the sensor signals, and the green dot at around $(55, 35)$ is the local goal point determined. Note that the local grid map does not include the obstacle at around $(45, 27)$, depicted as purple dots, since it is only constructed based on the sensed signals. The gray dots represent prior map grid points, the blue dot represents a robot, the orange line represents the global path to a destination, and the purple dots represent obstacles. The goal is the grid point toward the destination that is within the perception range, i.e., cyan dots, and has the smallest distance to the global path but the farthest distance from the mobile robot.

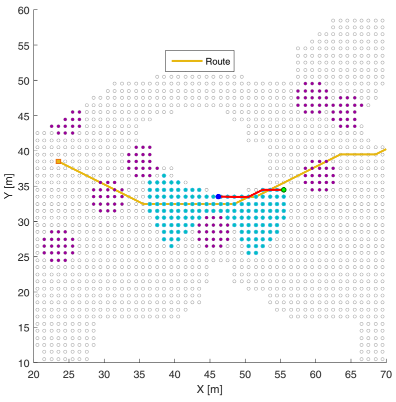


Fig. 2: Construction of grid points and a goal

B. Diffusion Maps

The *Diffusion Maps* block in Fig. 1 computes the diffusion maps corresponding to the grid map points, cyan dots in Fig. 2, constructed in the previous subsection. This step results in the diffusion maps matrix, H in (9), whose row space has the reduced dimensionality of r. The Euclidean distance in the row space of H in (10) is equivalent to the diffusion distance (6) which implies the underlying geometric connectivity between map grid points. Therefore, in order to plan a path, we will find the grid point with the smallest value of (10) with respect to the current grid point in the next subsection.

C. Path Finder

Suppose the g^{th} row of the diffusion maps matrix H corresponds to the goal point, depicted as the green dot in Fig. 2, and the s^{th} row corresponds to the start point, depicted as the blue dot in Fig. 2. We define neighbors of a grid point if they stay within a circle with a radius R centered at the grid point. With the neighboring grid points of the start point, we compute the diffusion distances in (10) to the goal point, using the row vectors of H corresponding to the neighbors and the g^{th} row vector of H corresponding to the goal point. Among the neighboring grid points, the point with the smallest diffusion distance is selected as the next grid point which the mobile robot moves toward. Likewise, these processes are performed again once the mobile robot reaches the next grid point. We repeat these processes until we reach the goal, and the following table summarizes the steps to find a local path

TABLE I: Path Finding

```
Suppose $n$ grid points,
1  Initialize $i \leftarrow s$, $Path[0] = s$
2  While $i$ is not equal to $g$,
3      For $nb_j$ in Neighbors($i^{th}$ grid point),
4          $D[j] = ||row_g(H) - row_j(H)||^2$
5      End For
6      $i \leftarrow arg\min(D)$,
7      $Path \leftarrow Path$.append($i$),
8  End while
```

from the start point to the local goal point. The red line in Fig. 2 represents the planned local path to the local goal. Note that we naturally obtained a local path for collision avoidance because the constructed grid map has already excluded the area occupied by obstacles, as shown in Fig. 2.

Once the local path is planned, the mobile robot moves along the local path for a sampling time unit T, as described by *Move* block in Fig. 1. In other words, the heading vector is calculated with the first and second grid points of the planned local path, and the mobile robot moves in the heading direction for time T at a speed of V. After moving by the distance VT in the heading direction, we repeat the whole process at the new position by going back to the *Grid & Goal* block, as shown in Fig. 1. Note that the local path could be thought of as the predicted path for a receding horizon to be followed by the robot from the current position. Therefore, the fundamental concept of the proposed algorithm is similar to the model predictive control which predicts a sequence of states and inputs at the current states [10].

IV. Simulation

A map is utilized for simulation of the proposed algorithm, and all the map grid points are depicted as gray dots in Fig. 3 with resolution of 1 m. The blue dot represents a robot whose speed is assumed to be 1 m/s, the orange square at $(23.5, 38.5)$ represents a start point, and the orange diamond at $(27.5, 103.5)$ represents a destination. A global path, i.e., a route to the destination, is represented as the orange line in Fig. 3, and it was planned with the A* algorithm which generates a minimum cost path (see [1] for details of the A* algorithm). This paper skips details of the A* algorithm because it is beyond the scope of this paper. The area sensed by the robot sensors, e.g., lidars and radars, is depicted as the cyan dots around the robot, and the perception range is assumed to be 10 m. With these data, the proposed algorithm determines a local goal point, the green dot in Fig. 3, and plans a local path, the red line in Fig. 3. In computation of the diffusion maps, we used the half of total grid points for the transition steps, i.e., $t = n/2$ in (5), and the dimensionality was reduced to 10% of the original dimensionality, i.e., $r = 0.1n$ in (9). In

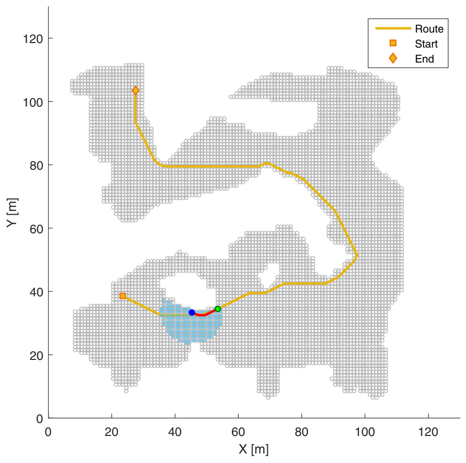


Fig. 3: Route to a destination

addition, for determining the neighbors of each grid point in *Path Finder* block in Fig. 1, we used a circle of the radius 2 m.

Fig. 4 shows obstacles positioned in the above simulation map, and they are depicted as the purple dots. We assumed that the obstacles move with a speed of 1 m/s and their radius is 2.5 m. Note that some of those obstacles keep moving in random directions. The blue dot represents a robot and the cyan dots represent the sensed area. Fig. 4 presents a local path, red line, planned by the proposed algorithm to go to the local goal point, green dot. The goal is determined based on the global path and perception range of the robot. Fig. 4 clearly illustrates the planned local path avoiding obstacles. The obstacles are excluded in the local grid map in Fig. 4, and therefore the local path for collision avoidance is naturally planned by simply applying the diffusion maps.

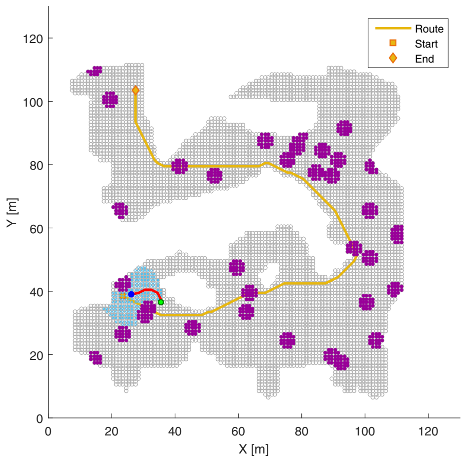


Fig. 4: Obstacles

Fig. 5 presents local paths planned to avoid obstacles at 9 time units. For example, the robot tries to follow the global path, depicted as the orange line, at the time unit $T = 30$ by setting up a point on the global path as the goal point since there is no obstacle ahead of the robot. But, the robot encounters an obstacle at the time unit $T = 37$, and therefore it deviates from the global path to avoid the obstacle even if the goal point still stays on the global path. Once the robot passes by the obstacle, it tries to come back and follow the global path, as shown at the time unit $T = 42$. At the time unit $T = 93$, a moving obstacle appears in front of the mobile robot. Therefore, at the time unit $T = 97$, the local path is planned to avoid collision with the obstacle in the same way as the local path at the time unit $T = 37$. Again, after passing by the obstacle, the mobile robot tries to follow the global path, as shown at the time unit $T = 109$. In addition, multiple obstacles are sensed by the robot at the time units $T = 123$ and 127, and the robot plans local paths to pass through between those obstacles to prevent collisions with them. After passing through between those obstacles, the robot tries to come back to the global path, as shown at the time unit $T = 139$.

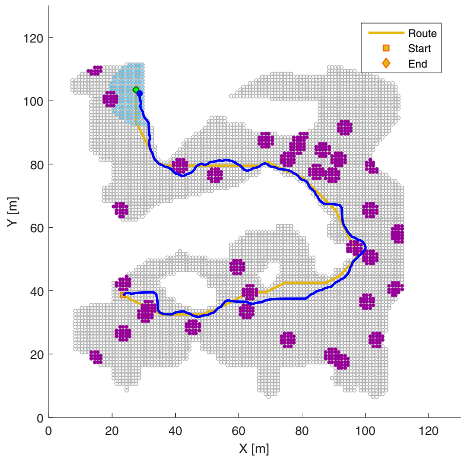
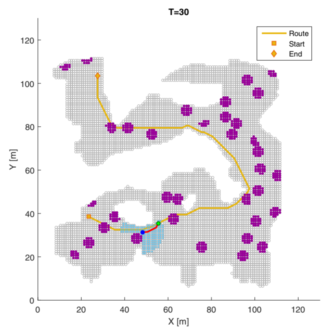


Fig. 6: Trajectory of Collision Avoidance - Static Obstacles

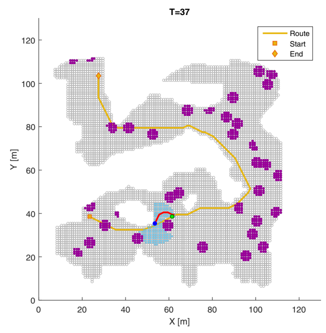
If we track the positions that the robot passed while moving from the start point to the destination, the trajectories of the robot are depicted as the blue lines in Fig. 6 and 7. The obstacles in Fig. 6 were all static, whereas some of the obstacles in Fig. 7 were dynamic, moving continuously in random directions. In the proposed algorithm, we do not have a hard constraint that the robot should follow the global path exactly. This is why there are discrepancies between the global path and the robot's trajectory in Fig. 6 even if the obstacles are static. In Fig. 7, the mobile robot avoids collisions with moving obstacles, and, therefore, it deviates more from the global path and the trajectory is less straight at around $(80.0, 40.0)$ than in Fig. 6.

V. Future Work

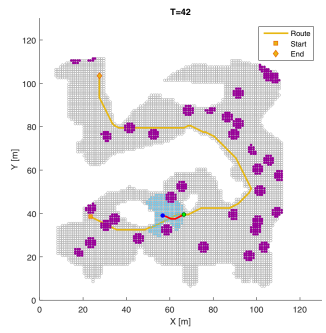
In this paper, performance analysis has not been presented. We will analyze performance of the proposed approach, e.g.,



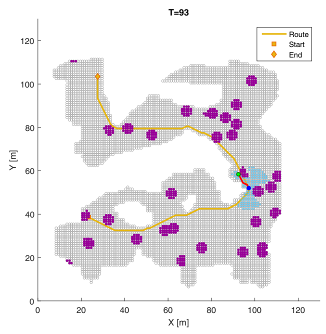
(a) T=30



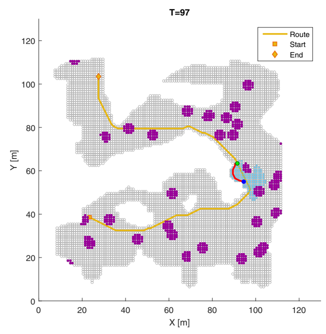
(b) T=37



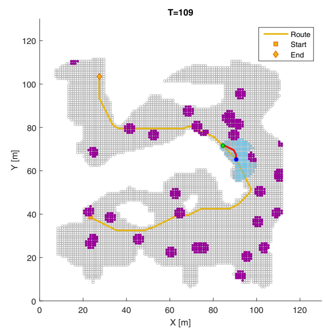
(c) T=42



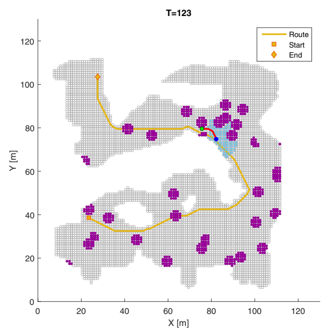
(d) T=93



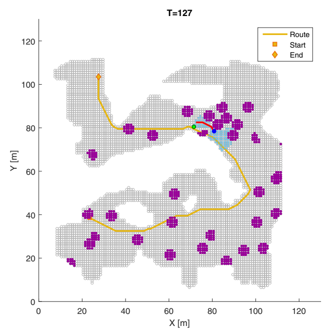
(e) T=97



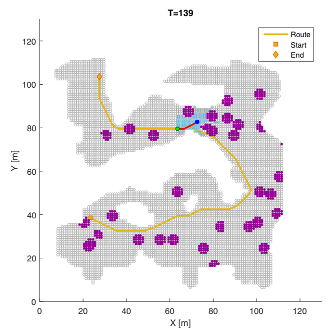
(f) T=109



(g) T=123



(h) T=127



(i) T=139

Fig. 5: Collision Avoidance - Moving Obstacles

comparing computational time with other path planning algorithms, and extend simulation scenarios to different environments, e.g., different map, different number of obstacles, and different motion of obstacles.

VI. Conclusion

In this paper, we planned local paths to avoid collisions with obstacles and follow the given global path simultaneously. With the sensed data, the local grid map is constructed, excluding the areas occupied by obstacles. The diffusion maps are computed with the local grid map points, and then we planned local paths by using the diffusion distances from the neighbors of current point to the local goal point. The receding horizon approach was applied by repeating these processes at every sample time. This enabled planning a local path to avoid collisions with obstacles without pre-determined information, e.g., prior map data and motion primitives. We simulated the proposed algorithm in different scenarios, in which a global path generated by the A* algorithm was given and some of obstacles were dynamic, moving in random directions. In the simulation, the proposed algorithm demonstrated a good performance in avoiding collisions with both static and dynamic obstacles.

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

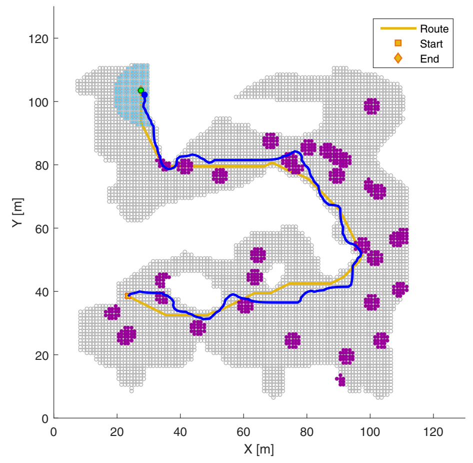


Fig. 7: Trajectory of Collision Avoidance - Moving Obstacles

[2] A. Stentz, "The focussed d* algorithm for real-time replanning," in *IJCAI'95 Proceedings of the 14th international joint conference on Artificial intelligence*, ser. IJCAI'95, 1995, pp. 1652–1659.

[3] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[4] D. Sharma and S. K. Dubey, "Anytime A* algorithm: An extension to A* algorithm," *International Journal of Scientific & Engineering Research*, vol. 4, no. 1, 2013.

[5] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[6] Y. F. Chen, S. Y. Liu, M. Liu, J. Miller, and J. P. How, "Motion planning with diffusion maps," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1423–1430.

[7] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.

[8] J. D. L. Porte, B. M. Herbst, W. Hereman, and S. J. V. D. Walt, "An introduction to diffusion maps," in *In The 19th Symposium of the Pattern Recognition Association of South Africa*, 2008.

[9] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[10] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.