Spatial Visibility Trajectory Planning Using Inverse Reinforcement Learning

Oren Gal and Yerach Doytsher
Mapping and Geo-information Engineering
Technion - Israel Institute of Technology
Haifa, Israel
e-mails: {orengal, doytsher}@technion.ac.il

Abstract—**In this paper, we present a conceptual Spatial Trajectory Planning (STP) method using Rapid Random Trees (RRT) planner, generating visibility motion primitives in urban environments using Inverse Reinforcement Learning (IRL) approach. Visibility motion primitives are set by using Spatial Visibility Clustering (SVC) analysis. Based on the STP planning method, we introduce IRL formulation and analysis which learns the value function of the planner from demonstrated trajectories and generates spatial visibility trajectory planning.**

Keywords-Visibility; 3D; Spatial analysis; Motion Planning.

I. INTRODUCTION

Spatial clustering in urban environments is a new spatial field from trajectory planning aspects [1]. The motion and trajectory planning fields have been extensively studied over the last two decades [2][4][6]. The main effort has focused on finding a collision-free path in static or dynamic environments, i.e., in moving or static obstacles, using roadmap, cell decomposition, and potential field methods [11].

The path-planning problem becomes an NP-hard one, even for simple cases such as time-optimal trajectories for a system with point-mass dynamics and bounded velocity and acceleration with polyhedral obstacles [7].

Path planning algorithms can be distinguished as local and global planners. The local planner generates one, or a few, steps at every time step, whereas the global planner uses a global search to the goal over a time-spanned tree. Examples of local (reactive) planners are [9][14]. These planners are too slow, do not guarantee safety and neglect spatial aspects.

Efficient solutions for an approximated problem were investigated by LaValle and Kuffner, addressing non-holonomic constraints by using the Rapidly Random Trees (RRT) method [15][16]. Over the years, many other semi-randomized methods were proposed, using evolutionary programming [5][18].

The randomized sampling algorithms planner, such as RRT, explores the action space stochastically. The RRT algorithm is probabilistically complete, but not asymptotically optimal [13]. The RRT* planner challenges optimality by a rewiring process each time a node is added to the tree. However, in cluttered environments, RRT* may behave poorly since it spends too much time deciding whether to rewire or not.

Overall, only a few works have focused on spatial analysis characters integrated into trajectory planning methods such as visibility analysis or spatial clustering methods [11].

Our research contributes to the spatial data clustering field, where, as far as we know, visibility analysis has become a leading factor for the first time. The SVC method, while mining the real pedestrians' mobility datasets, enables by a visibility analysis to set the number of clusters.

Analyzing pedestrian's mobility from a spatial point of view mainly focused on route choice [3], simulation model [19] and agent-based modeling [12].

The efficient computation of visible surfaces and volumes in 3D environments is not a trivial task. The visibility problem has been extensively studied over the last twenty years, due to the importance of visibility in GIS and Geomatics, computer graphics and computer vision, and robotics. Accurate visibility computation in 3D environments is a very complicated task demanding a high computational effort, which could hardly have been done in a very short time using traditional well-known visibility methods.

The exact visibility methods are highly complex, and cannot be used for fast applications due to their long computation time. Previous research in visibility computation has been devoted to open environments using Digital Elevation Model (DEM), representing raster data in 2.5D (Polyhedral model), and do not address, or suggest solutions for, dense built-up areas.

Most of these works have focused on approximate visibility computation, enabling fast results using interpolations of visibility values between points, calculating point visibility with the Line of Sight (LOS) method [7]. Lately, fast and accurate visibility analysis computation in 3D environments has been presented [10].

In this paper, we present, for the first time as far as we know, a unique conceptual Spatial Trajectory Planning

 ISBN: 978-1-61208-732-0

(STP) method based on RRT planner. The generated trajectories are based on visibility motion primitives set by SVC Optimal Control Points (OCP) as part of the planned trajectory, which takes into account exact 3D visible volumes analysis clustering in urban environments.

The proposed planner includes obstacle avoidance capabilities, satisfying dynamics' and kinematics' agent model constraints in 3D environments, guaranteeing probabilistic completeness. The generated trajectories are dynamic ones and are regularly updated during daylight hours due to SVC OCP during daylight hours. STP trajectories can be used for tourism and entertainment applications or for homeland security needs.

In the following sections, in Section II, we introduce the RRT planner and our extension for a spatial analysis case, such as 3D visibility. In Section III, we present the STP planner, using RRT and SVC capabilities. In the last section of the paper, we present the Inverse Reinforcement Learning (IRL) approach and algorithm based on the proposed STP planning method, learning the value function of the planner from demonstrated trajectories.

II. SPATIAL RAPID RANDOM TREES

In this section, the RRT path planning technique is briefly introduced with spatial extension. RRT can also deal with high-dimensional spaces by taking into account dynamic and static obstacles including dynamic and non-holonomic robots' constraints.

The main idea is to explore a portion of the space using sampling points in space, by incrementally adding new randomly selected nodes to the current tree's nodes.

RRTs have an (implicit) Voronoi bias that steers them towards yet unexplored regions of the space. However, in case of kinodynamic systems, the imperfection of the underlying metric can compromise such behavior. Typically, the metric relies on the Euclidean distance between points, which does not necessarily reflect the true cost-to-go between states. Finding a good metric is known to be a difficult problem. Simple heuristics can be designed to improve the choice of the tree state to be expanded and to improve the input selection mechanism without redefining a specific metric.

A. RRT Stages

The RRT method is a randomized one, typically growing a tree search from the initial configuration to the goal, exploring the search space. These kinds of algorithms consist of three major steps:

1. **Node Selection:** An existing node on the tree is chosen as a location from which to extend a new branch. Selection of the existing node is based on probabilistic criteria such as metric distance.
2. **Node Expansion:** Local planning applied a generating feasible motion primitive from the current node to the next selected local goal node, which can be defined by a variety of characters.
3. **Evaluation:** The possible new branch is evaluated based on cost function criteria and feasible connectivity to existing branches.

These steps are iteratively repeated, commonly until the planner finds feasible trajectory from start to goal configurations, or other convergence criteria.

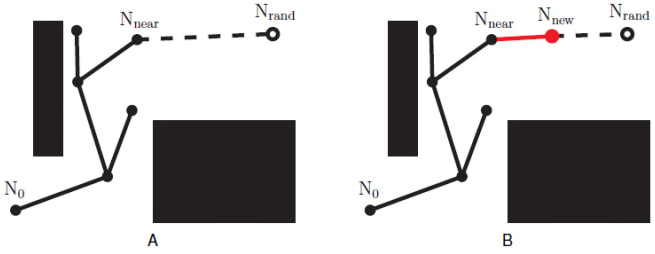

Figure 1. The RRT algorithm: (A) Sampling and node selection steps; (B) Expansion step.

A simple case demonstrating the RRT process is shown in Figure 1. The sampling step selects N_{rand}, and the node selection step chooses the closest node, N_{near}, as shown in Figure 1.A. The expansion step, creating a new branch to a new configuration, N_{new}, is shown in Figure 1.B. An example for growing RRT algorithm is shown in Figure 2.

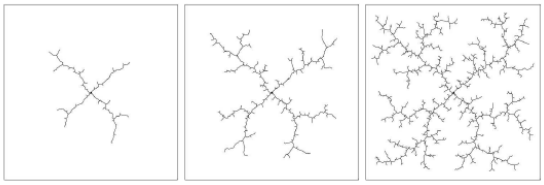

Figure 2. Example for growing RRT algorithm.

B. Spatial RRT Formulation

We formulate the RRT planner and revise the basic RRT planner for a 3D spatial analysis case for a continuous path from initial state x_{init} to goal state x_{goal}:

1. **State Space:** A topological space, X.
2. **Boundary Values:** $x_{init} \subset X$ and $x_{goal} \subset X$.
3. **Free Space**: A function $D: X \rightarrow \{true, false\}$ that determines whether $x(t) \subset X_{free}$ where X_{free} consist of the attainable states outside the obstacles in a 3D environment.
4. **Inputs:** A set, U, contains the complete set of attainable control efforts u_i, that can affect the state.
5. **Incremental Simulator:** Given a current state, $x(t)$, and input over time interval Δt, compute $x(t+\Delta t)$.
6. **3D Spatial Analysis:** A real value function, $f(x; u, OCP_i)$ which specifies the cost to the center of 3D visibility volumes cluster points (OCP) between a pair of points in X.

 ISBN: 978-1-61208-732-0

C. Spatial RRT Formulation

We present a revised RRT pseudo code described in Table I, for spatial case generating trajectory T, applying K steps from initial state x_{init}. The f function defines the dynamic model and kinematic constraints, $\dot{x} = f(x; u, OCPi)$, where u is the input and OCP_i sets the next new state and the feasibility of following the next spatial visibility clustering point.

TABLE I. SPATIAL RRT PSEUDO CODE

```
Generate Spatial RRT (x_init; K; Δt)
T.init (x_init);
For k = 1 to K do
        x_rand ← random.state();
        x_near ← nearest.neighbor (x_rand; T );
        u ← select.input (x_rand; x_near);
        x_new ← new.state (x_near; u; Δt; f);
        T.add.vertex (x_new);
        T.add.edge (x_near; x_new; u);
End
Return T
```

III. SPATIAL TRAJECTORY PLANNING (STP)

Next, we present a conceptual STP method based on RRT planner. The method generates visibility motion primitives in urban environments. The STP method is based on a RRT planner extending the stochastic search to specific *OCP*. These primitives connecting between nodes through OCP are defined as visibility primitives.

A common RRT planner is based on greedy approximation to a minimum spanning tree, without considering either path lengths from the initial state or following or getting close to specific OCP. Our STP planner consist of a tree's extension for the next time step with probability to goal and probability to waypoint, where trajectories can be set to follow adjacent points or through OCP. The planner includes obstacle avoidance capabilities, satisfying dynamics' and kinematics' agent model constraints in 3D environments. As we demonstrated in the previous section, the OCP are dynamic during daylight hours. Due to *OCP's* dynamic character, the generated trajectory is also a dynamic one during daylight hours.

We present our concept addressing the STP method formulating planner for a UGV model, integrating *OCP's* as part of the generated trajectories along with obstacle avoidance capability.

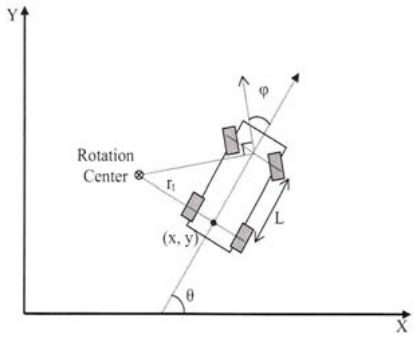

Figure 3. Four-Wheeled Car Model with Front-Wheel Steering [17]

A. Dynamic Model

In this section, we suggest an Unmanned Ground Vehicle (UGV) dynamic model based on the four-wheeled car system with rear-wheel drive and front-wheel steering [17]. This model assumes that only the front wheels are capable of turning and the back wheels must roll without slipping, and all the wheels turn around the same point (rotation center) which is co-linear with the rear axle of the car, as can be seen in Figure 3, where L is the length of the car between the front and rear axles. r_t is the instantaneous turning radius.

Thus, the UGV dynamic model can be described as:

$$=\begin{Bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(\phi) \end{Bmatrix} \qquad (1)\ \dot{x} = f(x,u) = \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{Bmatrix}$$

The state vector, x, is composed of two position variables (x, y) and an orientation variable, θ. The x-y position of the car is measured at the center point of the rear axle. The control vector, u, consists of the vehicle's velocity, v, and the angle of the front wheels, ϕ, with respect to the car's heading.

B. Search Method

Our search is guided by following spatial clustering points based on 3D visible volumes analysis in 3D urban environments, i.e., Optimal Control. The cost function for each next possible node (as the target node) consists of probability to closest *OCP*, P_{OCPi} , and probability to random point, P_{rand} .

In case of overlap between a selected node and obstacle in the environment, the selected node is discarded, and a new node is selected based on P_{OCPi} and P_{rand}. Setting the probabilities as P_{OCPi} =0.9 and P_{rand}=0.1, yield to the exploration behavior presented in Figure 4.

 ISBN: 978-1-61208-732-0

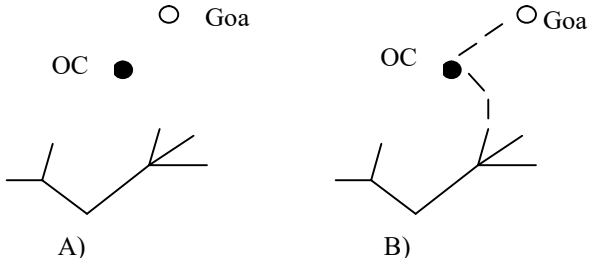

Figure 4. STP Search Method: (A) Start and Goal Points; (B) Explored Space to the Goal Through OCP

C. STP Planner Pseudo-Code

We present our STP planner pseudo code described in Table II, for spatial case generating trajectory T with the search space method presented in the Section V.B. The search space is based on P_{OCPi} and P_{rand}. We apply K steps from initial state x_{init}. The f function defines the dynamic model and kinematic constraints, $\dot{x} = f(x; u)$, where u is the input and $OCPi$ are local target points between start and goal states.

TABLE II. STP PLANNER PSEUDO CODE

```
STP Planner (x_init; x_Goal ;K; Δt; OCP)
T.init (x_init);
x_rand ← random.state();
x_near ← nearest.neighbor(x_rand; T );
u ← select.input(x_rand; x_near);
x_new ← new.state.OCP (OCP_1; u; Δt; f);
While x_new ≠ x_Goal do
        x_rand ← random.state();
        x_near ← nearest.neighbor(x_rand; T );
        u ← select.input(x_rand; x_near);
        x_new ← new.state.OCP (OCP_i; u; Δt; f);
        T.add.vertex(x_new);
        T.add.edge(x_near; x_new; u);
end
return T;
```

```
Function new.state.OCP (OCP_i;u; Δt; f)
Set P_OCPi , Set P_rand
p←uniform_rand[0..1]
    if   0 < p < P_OCPi
     return x_new = f(OCP_i,u,Δt);
else
    if P_OCPi < p < P_rand+ P_OCPi
then
     return RandomState();
     end.
```

D. Completeness

Motion-planning and search algorithms commonly describe 'complete planner' as an algorithm that always provides a path planning from start to goal in bounded time. For random sampling algorithms, 'probabilistic complete planner' is defined as: if a solution exists, the planner will eventually find it by using random sampling. In the same manner, the deterministic sampling method (for example, grid-based search) defines completeness as resolution completeness.

Sampling-based planners, such as the STP planner, do not explicitly construct search space and the space's boundaries, but exploit tests with preventing collision with obstacles and, in our case, taking spatial considerations into account. Similarly, to other common RRT planners, which share similar properties with the STP planner, our planner can be classified as a probabilistic complete one.

IV. STP-IRL ALGORITHM

In most Reinforcement Learning (RL) systems, the state is basically agent's observation of the environment. At any given state the agent chooses its action according to a policy. Hence, a policy is a road map for the agent, which determines the action to take at each state. Once the agent takes an action, the environment returns the new state and the immediate reward. Then, the agent uses this information, together with the discount factor to update its internal understanding of the environment, which, in our case, is accomplished by updating a value function. Most methods are using the use well-known simple and efficient greedy exploration method maximizing Q-value.

In case of velocity planning space as part of spatial analysis planning, each possible action is a possible velocity in the next time step, that also represents a viewpoint. The Q-value function is based on greedy search velocity, with greedy local search method. Based on that, the Temporal-Difference (TD) [10] and the State-Action-Reward-Action (SARSA) [21] methods for Reinforcement Learning (RL) can be used, generating a visible trajectory in 3D urban environment.

A. Markov Decision Processes (MDP)

The standard Reinforcement Learning set-up can be described as an MDP, consisting of:

- **A finite set of states** S, comprising all possible representations of the environment.
- **A finite set of actions** A, containing all possible actions available to the agent at any given time.
- **A reward function** $R = \psi(s_t, a_t, s_{t+1})$, determining the immediate reward of performing an action at from a state s_t, resulting in s_{t+1}.
- **A transition model** $T(s_t, a_t, s_{t+1}) = p(s_{t+1} | s_t, a_t)$, describing the probability of transition between states s_t and s_{t+1} when performing an action a_t.

B. Temporal Difference Learning

TD learning interpolates ideas from Dynamic Programming (DP) and from Monte Carlo methods. TD

algorithms are able to learn directly from raw experiences without any particular model of the environment.

While in Monte Carlo methods an episode needs to reach completion to update a value function, Temporal-Difference learning is able to learn (update) the value function within each experience (or step). The price paid for being able to regularly change the value function is the need to update estimations based on other learnt estimations (recalling DP ideas). While in DP a model of the environment's dynamic is needed, both Monte Carlo and TD approaches are more suitable for uncertain and unpredictable tasks.

Since TD learns from every transition (state, reward, action, next state, next reward) there is no need to ignore/discount some episodes as in Monte Carlo algorithms.

C. STP Using Inverse Reinforcement Learning

In this section, we present the Inverse Reinforcement Learning (IRL) approach based on the proposed Spatial RRT planning method. It considers that the value function f is related to each point x. The Spatial RRT planner seeks to obtain the trajectory T^* that is based on visibility motion primitives set by SVC Optimal Control Points (OCP) as part of the planned trajectory, which takes into account exact 3D visible volumes analysis clustering in urban environments, based on optimizing the value function f along T.

The generated trajectories are then represented by a set of discrete configuration points $T = \{x_1, x_2, \cdots, x_N\}$. Without loss of generality, we can assume that the value function for each point can be expressed as a linear combination of a set of sub-value functions, that will be called features $\mathbf{c(x) = \sum c_j f_j(x)}$. The cost of path T is then the sum of the cost for all points in the path. Particularly, in the RRT, the value is the sum of the sub-values of moving between pairs of states in the path:

$$c(\zeta) = \sum_{i=1}^{N-1} c(x_i, x_{i+1}) = \sum_{i=1}^{N-1} \frac{c(x_i) + c(x_{i+1})}{2} \|x_{i+1} - x_i\| = \omega^T \sum_{i=1}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2} \|x_{i+1} - x_i\| = \omega^T f(\zeta) \quad (2)$$

Based on the number of demonstration trajectories D, $D = \{\zeta_1, \zeta_2, \cdots, \zeta_D\}$, by using IRL, weights ω can be set for learning from demonstrations and setting similar planning behavior. As was shown by [10][21], this similarity is achieved when the expected value of the features for the trajectories generated by the planner is the same as the expected value of the features for the given demonstrated trajectories:

$$\mathbb{E}(f(\zeta)) = \frac{1}{D} \sum_{i=1}^{D} f(\zeta_i) \quad (3)$$

Applying the Maximum Entropy Principle [22] to the IRL problem leads to the following form for the probability density for the trajectories returned by the demonstrator:

$$p(\zeta|\omega) = \frac{1}{Z(\omega)} e^{-\omega^T f(\zeta)} \quad (4)$$

where Z(ω) is a normalization function that does not depend on ζ. One way to determine ω is maximizing the (log-) likelihood of the demonstrated trajectories under the previous model:

$$L(D|\omega) = -D\log(Z(\omega)) \quad + \mathbf{\sum_{i=1}^{D} (-w^T f(\zeta_i))} \quad (5)$$

The gradient of the previous log-likelihood with respect to ω is given by:

$$\nabla \mathcal{L} = \frac{\partial \mathcal{L}(\mathcal{D}|\omega)}{\partial \omega} = \mathbb{E}(f(\zeta)) - \frac{1}{D} \sum_{i=1}^{D} f(\zeta_i) \quad (6)$$

As mentioned in [22], this gradient can be intuitively explained. If the value of one of the features for the trajectories returned by the planner is higher than the value in the demonstrated trajectories, the corresponding weight should be increased to increase the value of those trajectories. The main problem with the computation of the previous gradient is that it requires to compute the expected value of the features E($f(\zeta)$) for the generative distribution (4). We suggest setting large amount of D cased, with relative w values for our planner characters, as seen in Table III.

TABLE III. STP-IRL PLANNER PSEUDO CODE

STP - IRL Planner
Setting Trajectory S Examples D, D= T.init* (x_{init});
Calculate function features Weight, w
$f_D \leftarrow$ *AverageFeatureCount(D)*;
$w \leftarrow$ *random_init()*;
Repeat
  for each T do*
    for rrt_repetitions do
      $\zeta_i \leftarrow$ *getRRTstarPath(T*,ω)*
      $f(\zeta_i) \leftarrow$ *calculeFeatureCounts*(ζ_i)
  end for
  f_{RRT} *(T*)*$\leftarrow \sum_{i=1}^{rrt\_repetitions} f(\zeta_i))/rrt\_repetitions$
 end for
 $f_{RRT} \leftarrow (\sum_{i=1}^{S} f_{RRT})/s$
 $\nabla L \leftarrow f_{RRT} - f_D$
 $w \leftarrow$ *UpdatedWeigths* (∇L)
Until convergence, Return w

V. CONCLUSIONS

In this paper, we have presented a unique planner concept, STP, generating trajectory in 3D urban environments based on the UGV model. The planner takes into account obstacle avoidance capabilities and passes through optimal control points calculated from spatial analysis. The spatial analysis defines the number of clusters in a dataset based on an analytic visibility analysis, named SVC.

Based on SVC and STP analysis, we presented an Inverse Reinforcement Learning (IRL) approach based on the proposed STP planning method, learning the value function of the planner from the demonstrated trajectories.

Future research will also include performances and algorithm complexity analysis for STP and SVC methods.

REFERENCES

[1] O. Gal and Y. Doytsher, "Spatial Visibility Clustering Analysis In Urban Environments Based on Pedestrians' Mobility Datasets," The Sixth International Conference on Advanced Geographic Information Systems, Applications, and Services, pp. 38-44, 2014.

[2] J. Bellingham, A. Richards, and J. How, "Receding Horizon Control of Autonomous Aerial Vehicles," in Proceedings ofthe IEEE American Control Conference, Anchorage, AK, USA, pp. 3741–3746, 2002.

[3] A. Borgers and H. Timmermans, "A model of pedestrian route choice and demand for retail facilities within inner-city shopping areas," Geographical Analysis, vol. 18, No. 2, pp. 115-128, 1996.

[4] S. A. Bortoff, "Path planning for UAVs," In Proc. of the American Control Conference, Chicago, IL, USA, pp. 364–368, 2000.

[5] B. J. Capozzi and J. Vagners, "Navigating Annoying Environments Through Evolution," Proceedings of the 40th IEEE Conference on Decision and Control, University of Washington, Orlando, FL, USA, 2001.

[6] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in Proc. IEEE Conf. Decision. and Control., USA, pp. 2379–2384, 2007.

[7] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic Motion Planning," Journal of the Association for Computing Machinery, pp. 1048–1066, 1993.

[8] Y. Doytsher and B. Shmutter, "Digital Elevation Model of Dead Ground," Symposium on Mapping and Geographic Information Systems (Commission IV of the International Society for Photogrammetry and Remote Sensing), Athens, Georgia, USA, 1994.

[9] W. Fox, D. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robotics and Automation Magazine, vol. 4, pp. 23–33, 1997.

[10] O. Gal and Y. Doytsher, "Fast and Accurate Visibility Computation in a 3D Urban Environment," in Proc. of the Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services, Valencia, Spain, pp. 105-110, 2012.

[11] O. Gal and Y. Doytsher, "Fast and Efficient Visible Trajectories Planning for Dubins UAV model in 3D Built-up Environments," Robotica, FirstView, pp. 1-21 Cambridge University Press 2013 DOI: http://dx.doi.org/10.1017/S0263574713000787, [accessed February 2019].

[12] M. Haklay, D. O'Sullivan, and M.T. Goodwin, "So go down town: simulating pedestrian movement in town centres," Environment and Planning B: Planning & Design, vol. 28, no. 3, pp. 343-359, 2001.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894, 2011.

[14] N.Y. Ko and R. Simmons, "The lane-curvature method for local obstacle avoidance," In International Conference on Intelligence Robots and Systems, 1998.

[15] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," TR 98-11, Computer Science Dept., Iowa State University, USA, 1998.

[16] S. M. LaValle and J. Kuffner. "Randomized kinodynamic planning," In Proc. IEEE Int. Conf. on Robotics and Automation, Detroit, MI, USA, pp. 473–479, 1999.

[17] L. R. Lewis, "Rapid Motion Planning and Autonomous Obstacle Avoidance for Unmanned Vehicles," Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, December 2006.

[18] C. W. Lum, R. T. Rysdyk, and A. Pongpunwattana, "Occupancy Based Map Searching Using Heterogeneous Teams of Autonomous Vehicles," Proceedings of the 2006 Guidance, Navigation, and Control Conference, Autonomous Flight Systems Laboratory, Keystone, CO, USA, August 2006.

[19] S. Okazaki and S. Matsushita, "A study of simulation model for pedestrian movement with evacuation and queuing," Proceedings of the International Conference on Engineering for Crowd Safety, London, UK, pp. 17-18, March 1993.

[20] P. Abbeel and P. Ng, "Apprenticeship learning via inverse reinforcement learning," in Proceedings of the twenty-first international conference on Machine learning, ICML '04, ACM, New York, NY, USA, http://doi.acm.org/10.1145/1015330.1015430, 2004.

[21] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Seattle, USA. vol. 134, 2015

[22] B. Ziebart, A. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in Proc. of the National Conference on Artificial Intelligence (AAAI), 2008.

 ISBN: 978-1-61208-732-0