Multi-Agents Spatial Visibility Trajectory Planning and Patrolling 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 generating spatial visibility trajectory planning. Additionally, we study the visible trajectories planning for patrolling application using heterogeneous multi agents in 3D urban environments. Our concept is based on spatial clustering method using visibility analysis of the 3D visibility problem from a viewpoints in 3D urban environments, defined as locations. We consider two kinds of agents, with different kinematic and perception capabilities. Using simplified version of Traveling Salesman Problem (TSP), we formulate the problem as patrolling strategy one, with upper bound optimal performances. We present a combination of relative deadline UniPartition approaches based on visibility clusters. These key features allow new planning optimal patrolling strategy for heterogeneous agents in urban environment. We demonstrate our patrolling strategy method in simulations using Autonomous Navigation and Virtual Environment Laboratory (ANVEL) test bed environment.**

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

I. INTRODUCTION AND RELATED WORK

Spatial clustering in urban environments is a new spatial field from trajectory planning aspects (Gal and Doytsher 2014). The motion and trajectory planning fields have been extensively studied over the last two decades (Bellingham et al. 2002; Bortoff 2000; Chitsaz and LaValle 2007; Erdmann and Lozano-Perez 1987; Fiorini and Shiller 1998; Fraichard 1999; Latombe 1990; LaValle 1998; LaValle 2006; LaValle and Kuffner 1999; Sasiadek and Duleba 2000).

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 (Gal and Doytsher 2013; Obermeyer 2009; Shaferman and Shima 2008).

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 (Donald et al. 1993).

Efficient solutions for an approximated problem were investigated (LaValle and Kuffner 1999), addressing non-holonomic constraints by using the Rapidly Random Trees (RRT) method (LaValle 1998). Over the years, many other semi-randomized methods were proposed, using evolutionary programming (Capozzi and J. Vagners 2001; Lum et al. 2006; Pongpunwattana and Rysdyk 2004).

The randomized sampling algorithms planner, such as RRT, explores the action space stochastically. The RRT algorithm is probabilistically complete, but not asymptotically optimal (Karaman and Frazzoli 2011). The RRT* planner Karaman et al. 2011) 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 (Gal and Doytsher 2013; Shaferman and Shima 2008).

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.

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 (Plantinga and Dyer 1990).

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 DEM models, 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 (Doytsher and Shmutter 1994; Durand 1999). Lately, fast and accurate visibility analysis computation in 3D environments has been presented (Gal and Doytsher 2012; Gal and Doytsher 2013).

Multi-agents decision making and control methods can be divided into two major disciplines, centralized and decentralized approaches. The basic idea of centralized approach is to make all the decisions in one place. All tasks are concentrated by a single entity, named 'Central Task Planner and Scheduler' (CTPS).

The CTPS translates the tasks into smaller tasks (sub-tasks), which will later be sent to the appropriate agents, according to their capabilities, their assignment and their workload. Theoretically, the centralized approach appears to do the trick. It allows knowing in advance all the tasks to be done and the connections among them, allows choosing the most fitting disassembling of the problem to sub-tasks. Indeed, this is a significant advantage, as there is no disassembling which will be ideal for all missions.

However, this approach does not fit a dynamic environment, in which unpredictable events may occur. Multi-agents in marine environment usually not in a constant contact with CTPS nor with each other, even though the CTPS requires a continuous stream of data about the forthcoming events in order to provide an effective response. Solutions to this problem (such as placing multiple sensors in the environment) are expensive and hard to apply.

On the other hand, at the decentralized approach, each agent is responsible for a group of tasks, and there is no need using entity such as CTPS. A predetermined disassembling is applied on the problem, and the agents can try to contact each other, in order to improve it. As mentioned above, this solution is problematic, as there is no disassembling which will be ideal for all problems.

Despite this fact, the lack of the CTPS allows every agent to process the data it collects by itself, and, for example, plan its own trajectory using local sensors data and decide what the next action is. The benefit of this approach is, of course, the speed of reaction and the independence of the agents. Moreover, it allows real time reaction to dynamic changes in the environment. As said, this is a problematic matter in the centralized approach.

In this paper, we present, for the first time as far as know, a unique conceptual Spatial Trajectory Planning (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, we first introduce the RRT planner and our extension for a spatial analysis case, such as 3D visibility. Later on, we present the STP planner, using RRT and SVC capabilities. In the last part 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.

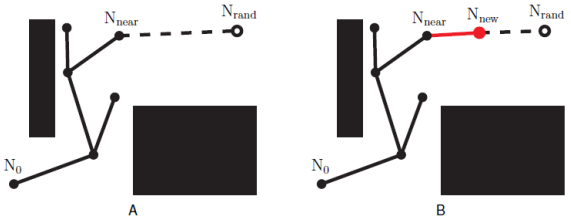


Figure 1. The RRT algorithm: (A) Sampling and node selection steps; (B) Expansion step.

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.

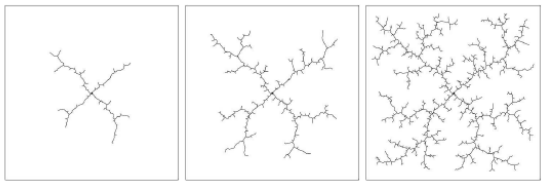


Figure 2. Example for growing RRT algorithm.

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 .

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 set 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)

In this section, 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.

A. Dynamic Model

In this section, we suggest an Unmanned Ground Vehicle (UGV) dynamic model based on the four-wheeled car system (UGV) with rear-wheel drive and front-wheel steering (Lewis 2006). 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 19, where L is the length of the car between the front and rear axles. r_t is the instantaneous turning radius.

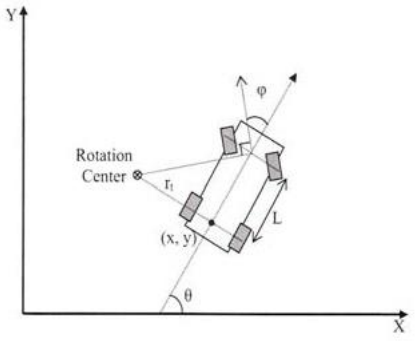


Figure 3. Four-Wheeled Car Model with Front-Wheel Steering (Lewis 2006)

Thus, UGV dynamic model can be described as:

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

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 20.

3.1.1 STP Planner Pseudo-Code

We present our STP planner pseudo code described in Table II, for spatial case generating trajectory T with search space method presented in the Section V.B. The search space is based on POCPi and Prand. We apply K steps from initial state xinit. 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 to goal states.

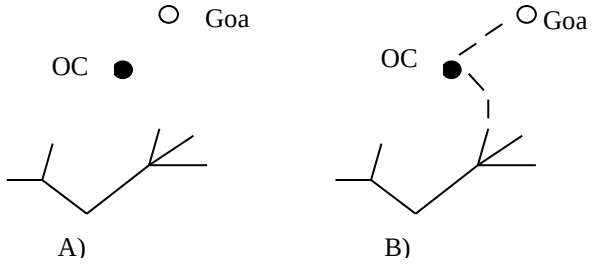


Figure 4. STP Search Method: (A) Start and Goal Points; (B) Explored Space to the Goal Through OCP

TABLE II. STP PLANNER PSEUDO CODE

STP Planner (x_{init}; x_{Goal} ;K; Δt; OCP)
T.init (x_{init});
$x_{rand} \leftarrow$ *random.state*();
$x_{near} \leftarrow$ *nearest.neighbor*(x_{rand}; *T*);
$u \leftarrow$ *select.input*(x_{rand}; x_{near});
$x_{new} \leftarrow$ *new.state.OCP* (OCP_1; *u*; Δ*t*; *f*);
While $x_{new} \neq x_{Goal}$ do
    $x_{rand} \leftarrow$ *random.state*();
    $x_{near} \leftarrow$ *nearest.neighbor*(x_{rand}; *T*);
    $u \leftarrow$ *select.input*(x_{rand}; x_{near});
    $x_{new} \leftarrow$ *new.state.OCP* (OCP_i; *u*; Δ*t*; *f*);
    T.add.vertex(x_{new});
    T.add.edge(x_{near}; x_{new}; *u*);
end
return *T;*

Function new.state.OCP (OCP_i;*u*; Δ*t*; *f*)
Set P_{OCPi} , *Set* P_{rand}
p←*uniform_rand[0..1]*
    if $0 < p < P_{OCPi}$
    return $x_{new} = f(OCP_i,u,\Delta t)$;
else
    if $P_{OCPi} < p < P_{rand} + P_{OCPi}$
then
    return RandomState();
    end.

C. 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 represent a viewpoint. The Q-value function is based on greedy search velocity, with greedy local search method. Based on that, TD and SARSA methods for RL can be used, generating visible trajectory in 3D urban environment.

A. Markov Decision Processes (MDP)

The standard Reinforcement Learning set-up can be described as a 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

Temporal-difference learning (or TD) interpolates ideas from Dynamic Programming (DP) and Monte Carlo methods. TD algorithms are able to learn directly from raw experiences without any particular model of the environment.

Whether 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). Whereas 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 Inverse Reinforcement Learning (IRL) approach based on the proposed Spatial RRT planning method. It considers that the value function *f* related to each point *x*. The Spatial RRT planner seeks to obtain the trajectory *T** that 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 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 $c(x) = \sum c_j f_j(x)$. 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 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 (Abbeel and Ng 2014; Kuderer et al. 2015), 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 (Ziebart et al. 2008) 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:

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

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

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

As mentioned in (Kuderer et al. 2015), this gradient can be intuitively explained. If the value of one of the features for the trajectories returned by the planner are higher from 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, setting the relative *w* values for our planner characters.

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 ← *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 ←*UpdatedWeigths (*∇L)
Until convergence
Return w

V. PATROLLING PLANNING USING STP

In this section, we study the visible trajectories planning for patrolling application using heterogeneous multi agents in 3D urban environments. Our concept is based spatial clustering method using visibility analysis of the 3D visibility problem from a viewpoints in 3D urban environments, defined as locations. We consider two kinds of agents, with different kinematic and perception capabilities. Using simplified version of Traveling Salesman Problem (TSP), we formulate the problem as patrolling strategy one, with upper bound optimal performances. We present combination of relative deadline UniPartition approaches based on visibility clusters. These key features allow new planning optimal patrolling strategy for heterogeneous agents in urban environment. We demonstrate our patrolling strategy method in simulations using Autonomous Navigation and Virtual Environment Laboratory (ANVEL) test bed environment.

VI. PROBLEM FORMULATION

The definition of the problem commonly set according to a predefined strategy planning. Patrolling strategy deals with different situations and hence yields different formulation. We demonstrate these differences using the following division:

1. Force Planning - with a given set of locations that should be protected, we have to determine what is the minimal number of agents (K) with patrolling strategy which meets all constraints. That becomes the common case, when we do not have any available agents compatible to the patrolling mission and we need to decide how many agents should be used in order to meet mission constraints.
2. Force Division - The locations are situated in different areas in urban environment. We need to decide how to allocate the agents to all the areas in such way that it will be possible to find a patrolling strategy that meets all constraints in all the areas.

In our case, we deal with force division problem combining relative deadline and visibility clustering. Given a set of N locations and K different types of agents (which available for patrol in a given moment), we are focusing on finding patrolling strategy, where each route for an agent passes through a number of locations. Patrolling strategy aims to minimize cost function which based on 3D visible volumes and meets the relative deadline constraints.

A. Locations

We set on our urban environment number of assets that should be protected, named as our Locations. Each location is characterized by:

1. Coordinates - its actual location in the environment generated from Visibility Clustering based on 3D visible volumes analysis.
2. Required Protection - different locations have different characteristics and therefore may differ in their protection needs. As will be discussed later, we refer to the type of agent required for protection.
3. Relative Deadline - which is the maximum time that may pass between consecutive visits for optimal

patrolling (for example, reduce the probability for being attacked beneath a certain threshold).

B. Agents

Our Agents are modeled as Unmanned Ground Vehicles (UGVs). We will generalized and differentiate between two types of these kind of agents:

1. Large - Agent with long range patrolling capabilities, sensing and target engaging abilities.
2. Small – Agent with limited ranges in the aspects described above.

Each of the agent's type has different average speed, dynamic and kinemtic constraints with hourly operational cost of its own and perception capabilities related to visibility analysis. There are many other factors to be considered but it is also beyond the scope of this paper.

VII. PATROLLING STRATEGY

We define collection of routes for all agents (one route per agent) which cover all the locations, by one agent or another, as Patrol Strategy. The total patrol cost defined as sum of all the routes cost, where route cost includes 3D visible volumes aspect integrated into visibility clusters.

A. Main Assumptions

We have assumed the following assumptions:

1. Agent Type - Each location requires different type of protection, and each type of agent has unique and different capabilities. Therefore, we will assume that each location can be protected by only one type of agent: large or small one. These assumptions split our problem into two independent problems, each with its relevant locations and relative deadlines.
2. Disjoint routs for each agent - In order to simplify our problem, we assume that each agent is allocated to patrol on a specific subset of locations. The subsets are disjoint sets which create disjoin routes for each agent. The union of those subsets is the initial set of locations.
3. Single visit to locations - Another simplification of the problem is the assumption that along the route each location can be visited only once, and the problem can be described as TSP private case. Due to this assumption, when we have locations with a relatively short deadlines, there might be cases where no solution will be found. This will be addressed in the review of the solution technique.
4. Patrolling along a cyclic path - We assume that each patrol route is a cyclic path. The definition of cyclic path is a path which start and end at the same location.
5. Traveling time between locations - Shortest traveling time - The given traveling times are the shortest possible times between given two locations, based on agent's dynamic model.

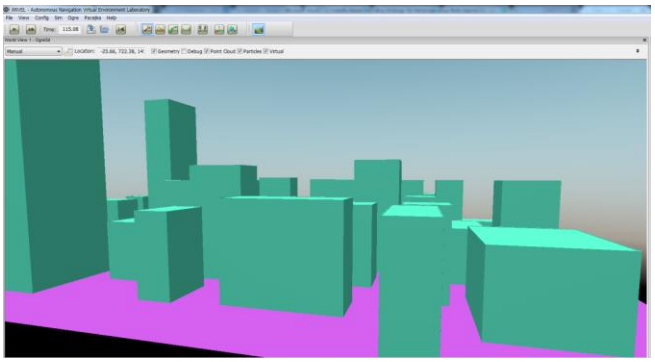
B. Program Inputs

The inputs of the program are as follows:

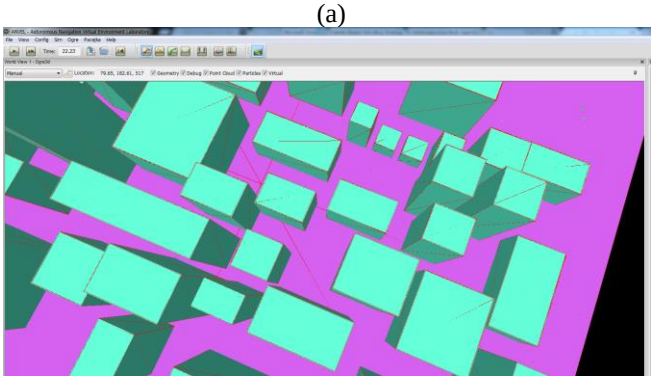
- N- number of locations
- K- number of agents
- Size of patrolling area
- Agent's average speed and physical dimensions
- C- The cost per one operational hour
- F- The cost per one hour of deviation from the deadlines (the fine cost)

We assumed 10 locations in total (N=10) and 3 agents in total (K=3), C=100$, F=1000$ and average speed of the agent is 65 km/h. We randomly generated the relative deadline of each location (9h in average) and also the coordinates of the N locations in the patrolling area (500 km^2).

The problem has the following dimensions: ten locations for small agents, six locations for large agents, and three small agents patrol. Our testbed urban environment can be seen in Figure 5. We set two large agents on patrol and one small agent in a given time. Agents in simulation environment can be seen in Figure 6. Perception capabilities of each agent can be seen in Figure 7 and Figure 8. Locations are changed according to the initial position of the agents as each simulation. The locations are set as the outcome of our visibility clustering analysis as can be seen in Figure 9 and Figure 11.



(a)



(b)

Figure 5. Our test bed urban environment model in ANVEL, (a) Sideview; (b) Topview.

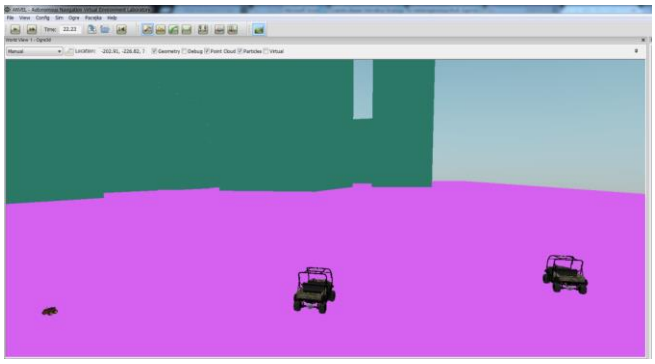


Figure 6. Three agents in our simulation, two large agents on the right side and a small agent to the left.

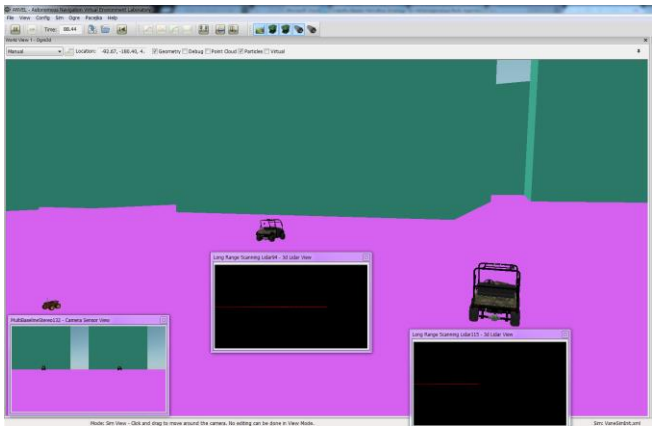


Figure 7. Perception sensor; LIDAR sensor for large agents and camera sensor for a small agent.

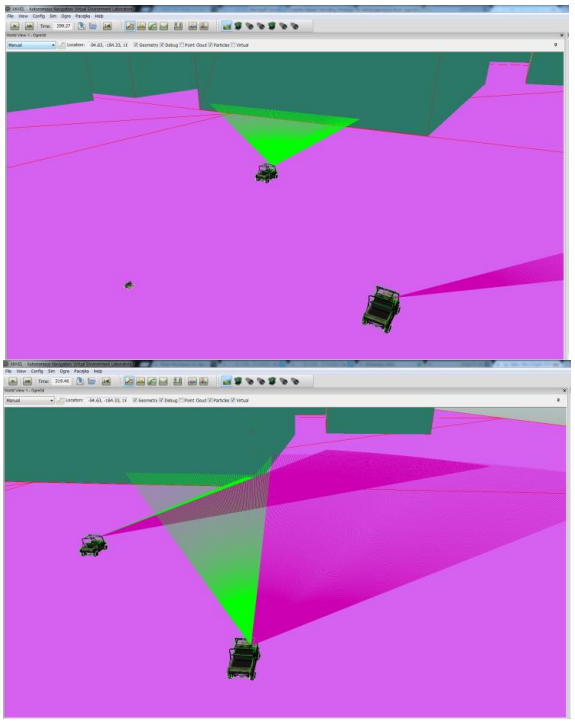


Figure 8. LIDAR perception capabilities during time

C. Simulations Results

We will preset several examples of the program results, based on spatial clustering and patrolling strategy.

In the first case, coordinates of the 10 locations are presented in Figure 9.

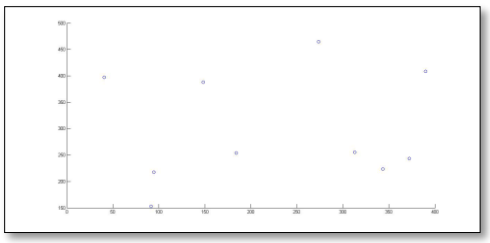


Figure 9. Location Coordinates based on Visibility Clustering

The patrolling graph can be seen in Figure 10.

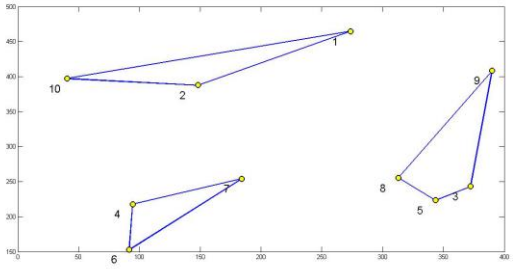


Figure 10. Patrolling Strategy Trajectories According to Problem Constraints

Patrol #1 [10, 1, 2, 10], Patrol #2 is [6, 4, 7, 6] and Patrol #3 [8, 5, 3, 9, 8]. Total time of the solution is (h) 48.885, and the total deviation from deadlines is (h) 0. Total cost of the solution is 4888.5. The total time of the solution is the sum of the total traveling time of all three patrols. For example, the total traveling time of patrol #3 is the time to arrive and return to location 8 plus the traveling time between the locations in the cluster, meaning the traveling time from location 8 to 5, 5 to 3, 3 to 9 and 9 to 8.

Our second simulation demonstrate patrol strategies that are non-feasible. Coordinates of the 10 locations are presented in Figure 11.

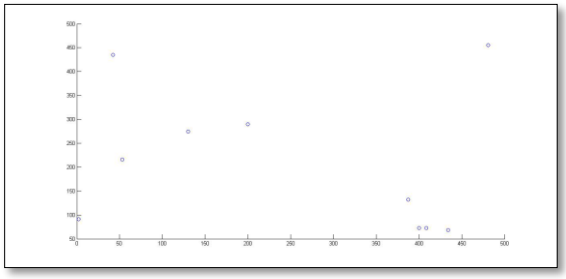


Figure 11. Location Coordinates based on Visibility Clustering in Second Simulation

The patrolling graph can be seen in Figure 12.

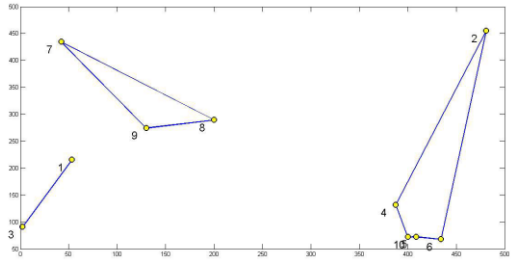


Figure 12. Patrolling Strategy Trajectories According to Problem Constraints

The program prints the following results: Patrol #1 [3, 1, 3], Patrol #2 [10, 4, 2, 6, 5, 10], Patrol #3 [9, 7, 8, 9], as can be seen in Figure 12. Total time of the solution is (h) 48.65, total deviation from deadlines is (h) 4.53. The total deviation from deadlines is the sum of the deviations of the patrol strategies that are non-feasible (it is not necessary that all the 3 patrol strategies have a deviation).

Other interesting example of a case in which one of the patrol strategy containing only one location. The reason for that is the coordinates of this location which are far away from all the other locations.

In the first case, coordinates of the 10 locations are presented in Figure 13.

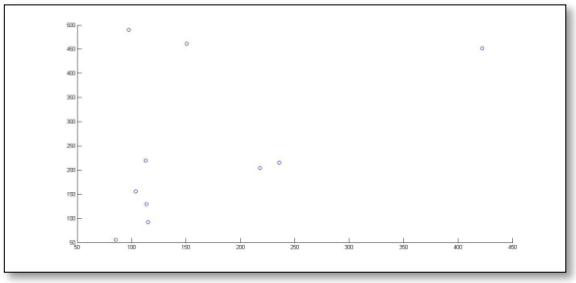


Figure 13. Location Coordinates based on Visibility Clustering in Third Simulation

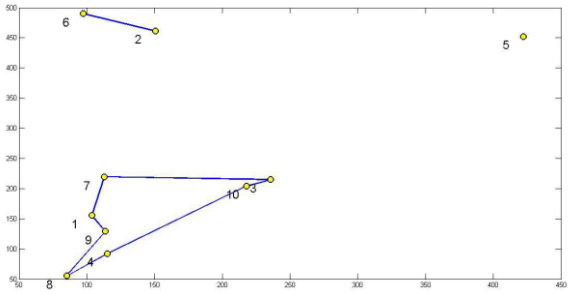


Figure 14. Patrolling Strategy Trajectories According to Problem Constraints

The program prints the following results: Patrol #1 [8, 9, 1, 7, 3, 10, 4, 8], Patrol #2 [2, 6, 2], Patrol #3 [5], as can be seen in Figure 14. Total time of the solution is (h) 46.88, total deviation from deadlines is (h) 0.

D. Discussion

Our research tackled simplified version (given our assumptions) to a problem as described in the introduction. In this section we would like to discuss other complicated cases which are received by eliminating one or more of our assumptions and cases which are received by different problem definition. We suggest a general solution approach for those cases and summarize with recommendation for possible research extensions.

1. **Finding a patrol strategy when repeated visits are allowed** - as we described in the second step of our solution technique (detailed in section (4)), the patrol strategy in each cluster will be found by using TSP technique which does not allow repeated visits apart from the first location in the path. Naturally, there might be a path with repeated visits which is a feasible solution (all the relative deadlines are met), but the TSP technique will "miss" this solution. Using the TSP technique is one of the alleviations that we have made, and we are aware to the fact that we might miss feasible solutions. In order to deal with the case in which there are no feasible solution at all, we added the fine calculation, but here we would like to shortly present a heuristic approach for finding a patrol strategy with repeated visits.

This approach is based on general idea as follows: For each cluster, the patrol strategy receives a feasible solution. Given a starting location (which is also the end location), the method uses the following algorithms:

- Forward Checking - given the last location in the patrol forming strategy this algorithm finds all the possible locations for that will satisfy the second and third constraints. All those locations are added to a location vector.

- Recursive call - given the last location in the patrol forming strategy and the index of the following step, it checks if the patrol have not ended with all constraints satisfied. If that is not the case, it calls Forward checking and run itself recursively on every location in and the index until a patrol strategy is achieved. Meaning, we have a route which is a closed path and every location was visited at least once. When the algorithm backtracks till a different location can be chosen and continue that route.

2. **Non disjoint routes** - one of the alleviations that we made is to assume that the route of each agent is disjoint from all other routes. The reason for that was to avoid collisions between the agents and to simplify our problem. Obviously, in reality, the routes does not have to be disjoint. When eliminating this assumption the complexity of the problem rises due to the fact that there are a great deal more cluster partition options. We suggest a heuristic approach to try and find a solution to the problem. First preform the

same partition of the location into clusters (or similar partition method based on geographical proximity) and then check if there is a solution with disjoint routes. If there is no such solution, find the locations that for them the relative deadline constraint is not met. Then, try and add each of those locations to another cluster based on a proximity criteria, such as the shortest average distance to all locations in the cluster and so forth. Now these locations are visited by more then one agent and therefore the time between consecutive visits is reduced to a point that may yield a feasible solution.

3. **Joint routes for a number of agents while the goal is to minimize the maximal idle time of all locations** - This is the approach mentioned in the introduction (see section (1)). Well known heuristic algorithms solve the following problem:

- Create an Outer-planner Graph from all given locations. This is done by connecting all the locations with arcs that does not cross one another. Allocate all agents to this graph and calculate the maximal idle time.

- By removing two arcs at a time from the graph we are creating two separated graphs. Remove all possible pairs of arcs and for each graph pair created find the agent allocation (meaning the number of agents allocated to each graph) that minimizes the maximal idle of both graphs.

- Find the division that yields the minimal maximal idle time and for each graph preform the same process described in the previous stage.

- Continue until the maximal idle time can not be reduced or when the only possible allocation is single agent to a graph.

4. **Locations that require protection from both type of agents** - in our solution, we assumed that each agent can be protected by only one type of agent. The reason for that was to separate the initial problem into two disjoin problem. That way, we can solve each problem separately. Obviously, in reality, there exist locations that require protection by both types of agents (we will refer to them as hybrid locations). While there are different relative deadlines for each type of agent then we can continue referring to the problem as two separated problems. The complication begins when the relative deadline is unified meaning the maximal time that may pass between consecutive visits of any type of agent. A possible way of approaching this case without unifying the problems (which may yield a large dimension problem that is harder to solve) is as follows:

- Solve the separated problems.

- If no feasible solution exists for one of the problems, and that is due only to a problem in meeting one or more hybrid locations relative deadline constraints. Continue to the next stage

- Combine the solution with the total minimal deviation from the relative deadlines of the hybrid locations with feasible solutions (or such with minimal deviation as well) for the other type of agents and update the time between consecutive visits to hybrid locations accordingly. Check if now the relative deadlines are met.

- If no feasible solution can be found that way, the fine method we introduced in this paper may be used.

VIII. CONCLUSIONS

In this paper, we have presented a unique planner concept, STP, generating trajectory in 3D urban environments based on 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 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 demonstrated trajectories.

We also presented visible trajectories planning for patrolling application using heterogeneous multi agents in 3D urban environments. Using ANVEL simulation environment, we demonstrated spatial clustering method using visibility analysis of the 3D visibility problem from a viewpoints in 3D urban environments. As part of our simulations we modeled two kinds of agents, with different kinematic and perception capabilities. Patrolling strategy formulated as Traveling Salesman Problem (TSP), with relative deadline UniPartition approaches based on visibility clusters.

We showed implementation of differences cases, using large and small agents in urban environment scenarios where sometime trajectory can no be found. Some other problem formulation discussed with suggested solution for further research, such as: finding a patrol strategy when repeated visits are allowed; non disjoint routes as part of our patrol graph; joint routes for a number of agents while the goal is to minimize the maximal idle time of all locations and other cases of locations that require protection from both type of agents.

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

IX. REFERENCES

O. Gal and Y. Doytsher, (2014) "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.

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

A. Borgers and H. Timmermans, (1996) "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.

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

O. Brock and O. Khatib, (2000) "Real time replanning in high-dimensional configuration spaces using sets of homotopic paths," In Proc. of the IEEE International Conference on Robotics and Automation, San Francisco, CA, pp. 550-555.

R. B. Calinski and J. Harabasz, (1974) “A Dendrite Method for Cluster Analysis,” Communications in Statistics, vol. 3, pp. 1–27.

B. J. Capozzi and J. Vagners, (2001) “Navigating Annoying Environments Through Evolution,” Proceedings of the 40th IEEE Conference on Decision and Control, University of Washington, Orlando, FL.

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

B. Donald, P. Xavier, J. Canny, and J. Reif, (1993) “Kinodynamic Motion Planning,” Journal of the Association for Computing Machinery, pp. 1048–1066.

Y. Doytsher and B. Shmutter, (1994) "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.

F. Durand, (1999) "3D Visibility: Analytical Study and Applications," PhD thesis, Universite Joseph Fourier, Grenoble, France.

M. Erdmann and T. Lozano-Perez, (1987) "On multiple moving objects," Algorithmica, Vol. 2, pp. 477–521.

V. Estivill-Castro and I. Lee, (2000) "AMOEBA: Hierarchical Clustering Based on Spatial Proximity Using Delaunay Diagram," In Proceedings of the 9th International Symposium on Spatial Data Handling, Beijing, China.

P. Fiorini and Z. Shiller, (1998) "Motion planning in dynamic environments using velocity obstacles," Int. J. Robot. Res. vol. 17, pp. 760–772.

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

T. Fraichard, (1999) "Trajectory planning in a dynamic workspace: A ’state-time space’ approach," Advanced Robotics, vol. 13, pp. 75–94.

E. Frazzoli, M.A. Daleh, and E. Feron, (2002), "Real time motion planning for agile autonomous vehicles," AIAA Journal of Guidance Control and Dynamics, vol. 25, pp. 116–129.

O. Gal and Y. Doytsher, (2012) "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.

P. Arabie and L. J. Hubert, (1996) "An Overview of Combinatorial Data Analysis," in Arabie, P., Hubert, L.J., and Soete, G.D. (Eds.) Clustering and Classification, pp. 5-63.

O. Gal and Y. Doytsher, "Fast Visibility Analysis in 3D Procedural Modeling Environments," in Proc. of the, 3rd International Conference on Computing for Geospatial Research and Applications, Washington DC, USA, 2012.

O. Gal and Y. Doytsher, (2013) "Fast Visibility in 3D Mass Modeling Environments and Approximated Visibility Analysis Concept Using Point Clouds Data," Int. Journal of Advanced Computer Science, IJASci, vol. 3, no. 4, April 2013, ISSN 2251-6379.

O. Gal and Y. Doytsher, (2013) "Fast and Efficient Visible Trajectories Planning for Dubins UAV model in 3D Built-up Environments," Robotica, FirstView, Article pp. 1-21 Cambridge University Press 2013.

A. Gordon, (1999) Classification (2nd ed.), London: Chapman and Hall/CRC Press.

S. Guha, R. Rastogi, and K. Shim, (1998) "CURE: An efficient clustering algorithm for large databases," In Proceedings of the ACM SIGMOD Conference, Seattle, WA, pp. 73-84.

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

D. Harel and Y. Koren, (2001) "Clustering spatial data using random walks," In Proceedings of the 7th ACM SIGKDD, San Francisco, CA, pp. 281-286.

J. Hartigan, (1975) "Clustering Algorithms". John Wiley & Sons, New York, NY.

J. Hartigan and M. Wong, (1979) "Algorithm AS136: A k-means clustering algorithm," Applied Statistics, vol. 28, pp. 100-108.

S. P. Hoogendoorn and P. H. L. Bovy, (2001) "Microscopic pedestrian way finding and dynamics modelling," In Schreckenberg, M., Sharma, S.D. (eds.) Pedestrian and Evacuation Dynamics. Springer Verlag: Berlin, pp. 123-154.

D. Hsu, R. Kindel, J-C. Latombe, and S. Rock, (2000) "Randomized kinodynamic motion planning with moving obstacles," Algorithmics and Computational Robotics, vol. 4, pp. 247–264.

B. Jiang, (1999) "SimPed: Simulating pedestrian flows in a virtual urban environment," Journal of Geographic Information and Decision Analysis, vol. 3, no. 1, pp. 21-30.

S. Karaman and E. Frazzoli, (2011) “Sampling-based algorithms for optimal motion planning,” Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894.

S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, (2011) “Anytime motion planning using the RRT*,” in Proc. IEEE Int. Conf. Robot. Autom., Shanghai, pp. 1478–1483, May.

L. Kaufman and P. Rousseeuw, (1990) "Finding Groups in Data: An Introduction to Cluster Analysis," John Wiley and Sons, New York, NY.

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

W. J. Krzanowski and Y. T. Lai, (1985) “A Criterion for Determining the Number of Groups in a Data Set Using Sum of Squares Clustering,” Biometrics, vol. 44, pp. 23–34.

M.P. Kwan, (2000) "Analysis of human spatial behavior in a GIS environment: recent developments and future prospects," Journal of Geographical System, no. 2, pp. 85-90, 2000.

J. C, Latombe, (1990) "Robot Motion Planning,", Kluwer Academic Press.

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

S. M. LaValle, (2006) "Planning Algorithms," Cambridge, U.K.: Cambridge Univ. Press.

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

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

C. W. Lum, R. T. Rysdyk, and A. Pongpunwattana, (2006) “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, August.

G.W. Milligan and M. C. Cooper, (1985) “An Examination of Procedures for Determining the Number of Clusters in a Data set,” Psychometrika, vol. 50, pp. 159–179.

J. Minguez and L. Montano, (2000) "Nearest diagram navigation. a new realtime collision avoidance approach," In International Conference on Intelligence Robots and Systems.

J. Minguez, N. Montano, L. Simeon, and R. Alami, (2002) "Global nearest diagram navigation," In Proc. of the IEEE International Conference on Robotics and Automation.

B. Moulin, W. Chaker, and J. Perron, (2003) "MAGS project: Multi-Agent GeoSimulation and Crowd Simulation," Kuhn, W., Worboys, M.F. and Timpf, S. (Eds.): LNCS 2825, pp. 151–168.

K. J. Obermeyer, (2009) “Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain,” in Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago.

S. Okazaki and S. Matsushita, (1993) "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.

A.Pongpunwattana and R.T. Rysdyk, (2004) “Real-Time Planning for Multiple Autonomous Vehicles in Dynamic Uncertain Environments,” AIAA Journal of Aerospace Computing, Information, and Communication, pp. 580–604.

H. Plantinga and R. Dyer, (1990) "Visibility, Occlusion, and Aspect Graph," The International Journal of Computer Vision, vol. 5, pp. 137-160.

J. Sasiadek and I. Duleba, (2000) "3d local trajectory planner for uav," Journal of Intelligent and Robotic Systems, vol. 29, pp. 191–210.

V. Shaferman and T. Shima, (2008) “Co-evolution genetic algorithm for UAV distributed tracking in urban environments,” in ASME Conference on Engineering Systems Design and Analysis.

T. Schelhorn, D. Sullivan, and M. Haklay, (1999) "STREETS: An agent-based pedestrian model,".

C. Stachniss and W. Burgard, (2002) "An integrated approach to goal directed obstacles avoidance under dynamic constrains for dynamic environment," In International Conference on Intelligence Robots and Systems.

R. Tibshirani, G. Walther, and T. Hastie, (2001) “Estimating the Number of Clusters in a Dataset via the Gap Statistic,” Journal of the Royal Statistical Society, Ser. B, vol. 32, pp. 411–423.

L. Ulrich and J. Borenstien, "Vfh+: Reliable obstacle avoidance for fast mobile robots," In Proc. of the IEEE International Conference on Robotics and Automation, 1998.

Abbeel, P., Ng, A.Y., (2004) "Apprenticeship learning via inverse reinforcement learning" In: Proceedings of the twenty-first international conference on Machine learning, ICML ’04, ACM, New York, NY, USA.

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

Ziebart, B., Maas, A., Bagnell, J., Dey, A., (2008) "Maximum entropy inverse reinforcement learning", In: Proc. of the National Conference on Artificial Intelligence (AAAI).