Spatial Trajectory Planning based on Visibility Clustering Analysis in Urban Environments

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. Visibility motion primitives are set by using Spatial Visibility Clustering (SVC) analysis. Our Spatial Visibility Clustering (SVC) method estimates the number of clusters (groups) k, based on 3D visible volumes analysis in urban environments. Our SVC method proposes fast and exact 3D visible volumes analysis in urban scenes based on an analytic solution. We test and analyze the SVC method by using real records of pedestrians' mobility datasets from the city of Melbourne and by setting control points for efficient monitoring and control using a K-means clustering algorithm.**

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 [1]. The motion and trajectory planning fields have been extensively studied over the last two decades [2][4][8][12][14][16][38] [39][40][41][54]. 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 [22][50][55].

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 [9].

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 [15][35][47][60]. These planners are too slow, do not guarantee safety and neglect spatial aspects.

Recently, iterative planners [5][16][17][30][48][59] have been developed that compute several steps at a time, subject to the available computation time. The trajectory is generated incrementally by exploring a search-tree and choosing the best branch.

Efficient solutions for an approximated problem were investigated by LaValle and Kuffner, addressing non-holonomic constraints by using the Rapidly Random Trees (RRT) method [39][41]. Over the years, many other semi-randomized methods were proposed, using evolutionary programming [7][43][52].

The randomized sampling algorithms planner, such as RRT, explores the action space stochastically. The RRT algorithm is probabilistically complete, but not asymptotically optimal [32]. The RRT* planner [33] 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 [22][55].

'Clustering methods' refers to the division of data sets into groups, each containing similar objects. Data modeling is extensively studied in statistics, mathematics and machine learning [19]. Most of the common clustering methods can be divided into hierarchical and partitioning methods.

Partitioning algorithms determine the clusters directly, such as the well-known K-Means method [27][28], where by a hierarchical mechanism, builds the clusters gradually [24].

Clustering methods of 2D spatial data (such as GIS database) were also studied, defining data proximity by using, inter alia, a Delaunay diagram. These methods focused on performances and low complexity, by keeping K-nearest neighbors using a connectivity graph where clusters become connected components [13][26].

Many clustering methods are based on a significant user's input parameter, the number of clusters k. Over the years, several criteria were introduced to find the optimal k. In the case of k-means clustering method, F-statistic (also known as the F-test) generates the optimal k. Another popular choice of separation measure is a Silhouette coefficient [34].

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][29], simulation model [51] and agent-based modeling [25][31][37][57].

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 [53].

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 [10]-[11]. Lately, fast and accurate visibility analysis computation in 3D environments has been presented [18][20][21].

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.

The SVC is a unified method for estimating the number of clusters using 3D visible volumes analysis, called Spatial Visibility Clustering (SVC). Based on our previous work, we use a fast and efficient analytic solution, setting visibility boundaries of visible surfaces from the viewpoint. We extend our solution to 3D volumes, computing 3D visible volumes. By using F-criteria, we set the optimal number of clusters from the visibility aspect.

We demonstrate SVC method using real datasets from the city of Melbourne's 24-hours pedestrians monitoring system, localizing control points at each hour during the day, using a K-means algorithm with SVC output, i.e., number of clusters *k*. We analyze pedestrians' mobility behavior and suggest dividing the day into four time zones, based on our datasets and setting optimal control points during these time zones.

In the following sections, we first introduce the RRT planner and our extension for a spatial analysis case, such as 3D visibility. Later, we present the SVC method, and the extended visible volumes analysis and SVC simulation using the city of Melbourne's datasets. We demonstrate the SVC method by dividing daylight hours into four time zones and setting optimal control points. Later on, we present the STP planner, using RRT and SVC capabilities.

II. SPATIAL RAPID RANDOM TREES

In this section, the RRT path planning technique is briefly introduced with spatial extension. RRT was first introduced in [39][41], dealing 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 [39] 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.

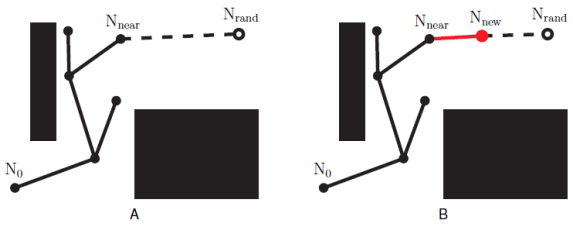


Figure 1. The RRT algorithm: (A) Sampling and node selection steps; (B) Expansion step.

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.

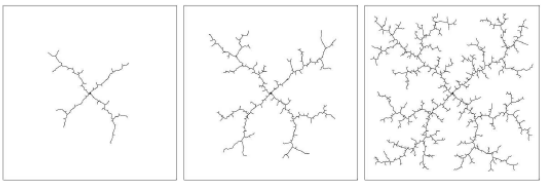


Figure 2. Example for growing RRT algorithm (source [39]).

B. Spatial RRT Formulation

We formulate the RRT planner and revise the basic RRT planner [39] 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 VISIBILITY CLUSTERING (SVC) METHOD

We present, for the first time as far as we know, a unified spatial analysis defining the number of clusters in a data set based on analytic visibility analysis, called Spatial Visibility Clustering (SVC). The output of our method can be efficiently used by common clustering methods (e.g., K-means or hierarchical). The number of clusters in dense environments can be used for civil and security applications in urban environments, based on 3D visibility analysis from points of view.

For the last twenty years, many methods were proposed in order to estimate the number of clusters in data sets [6][23][34][36][46][59]. As previously mentioned by [23], the approaches can be divided into global and local methods.

First, we introduce the main steps of our method and formulate the problem of estimating the number of clusters and the proposed volumes visibility analysis in 3D. Later, we present the analysis of the number of clusters using the SVC method, based on real pedestrians' mobility data sets. Finally, we examine a unique division of a twenty four-hour day into four different time zones in Melbourne [44], for control points based on pedestrians' mobility datasets in a number of points of interest, presented in Figure 3.

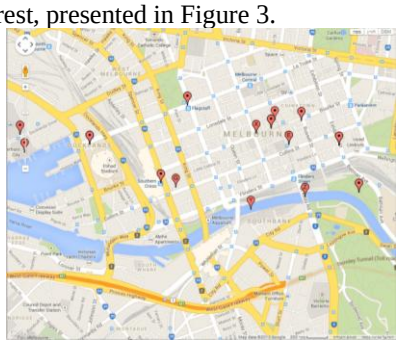


Figure 3. Melbourne Sensors Location for Monitoring Pedestrians' Mobility Data

A. Spatial Visibility Clustering - Main Stages

Our data set $\{X_{ij}\}$, *i=1,2,..,n, j=1,2,..,p,* consists of *p* features measured on n independent viewpoints, marked with blue circles are illustrated in Figure 4. We clustered the data into k clusters, $C_1, C_2, .. C_k$. For cluster r, denote as C_r with n_r viewpoints:

$$V_r = \sum_{i \in C_r} \sum_{j \in C_r} \left\| V(x_i) - V(x_j) \right\| \quad (1)$$

$$V_r = \sum_{i \in C_r} \left\| V(x_i) - V(\bar{x}) \right\|$$

$$T_k = \sum_{r=1}^{k} \frac{1}{S} V_r$$

Where V(x) denotes the visible volumes from a viewpoint *x* bounded inside the total volume *S*, V_r is the sum of the absolute visibility differences of all viewpoints from their cluster visibility mean, and the normalized visible volumes T_k for all clusters *r=1..k*, called ***dispersion***.

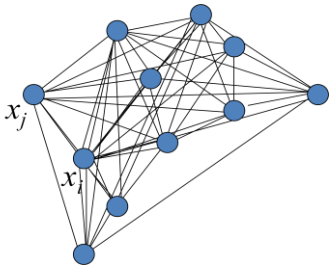


Figure 4. Pedestrians' location architecture based on monitoring datasets, viewpoints marked with blue circles

Similarly to many other methods estimating the number of clusters [23], we define reference data sets distributed uniformly inside bounding volume *S*. We define our reference data sets with the same size of the original data set *X*, and calculate the dispersion of these datasets, T_k^*.

Based on *F* statistic, datasets are analyzed, where adding another cluster does not give a better modeling of the data, also known as *F-test* criteria. By setting a group's visibility variance, the number of clusters can be estimated efficiently:

$$SVC_n(k) = T_k^* - T_k \quad (2)$$

Fast and efficient visibility volume computation from a specific viewpoint, bounded in volume *S*, is presented in the next subsection.

We can summarize SVC steps as follows:

1. Calculate the sum of absolute visibility differences of all points from their cluster visibility mean. Normalize this sum for all possible clusters T_k, also called dispersion.
2. Generate a set of reference datasets, simulated by a uniform distribution model inside bounding volume *S*.
3. Calculate the dispersion of each of these reference datasets, and calculate their mean visibility values.
4. Define SVC for each possible number of clusters as: Expected dispersion of reference datasets - Dispersion of original dataset.

Originally, *F* statistic was used to test the significance of the reduction in the sum of squares as we increase the number of clusters [27]. In general, when the number of clusters increases, the in-cluster decay first declines rapidly. From a certain *k*, dividing a dataset into *k+1* clusters decreases the value of *F-test* function which depends on *k*.

Approximated *F-test* function: Assuming that T_k is the partition of n instances into *k* clusters, and T_{k+1} is obtained from T_k splitting one of the clusters, then the overall mean ratio can be approximated as:

$$F_k = \frac{SVC_n}{SVC_{n+1}} \quad (3)$$

We adapted aspects of previous F statistic theory for visibility analysis. More detailed F statistic analysis can be found in [27].

The spatial meaning of this mathematical clustering formulation can be simplified as a group of viewpoints with minimal difference to the average visible volume in the same bounding box.

B. Analytic 3D Visible Volumes Analysis

In this section, we present fast 3D visible volumes analysis in urban environments, based on an analytic solution which plays a major role in our proposed method of estimating the number of clusters. We extend our previous work [18] for surfaces visibility analysis, and present an efficient solution for visible volumes analysis in 3D.

We analyze each building, computing visible surfaces and defining visible pyramids using analytic computation for visibility boundaries [18][21]. For each object we define Visible Boundary Points and Visible Pyramid.

Visible Boundary Points (VBP) - we define VBP of the object *i* as a set of boundary points *j=1..N_{bound}* of the visible surfaces of the object, from viewpoint *V(x_0, y_0, z_0)*.

$$VBP_{i=1}^{j=1..N_{bound}}(x_0, y_0, z_0) = \begin{bmatrix} x_1, y_1, z_1 \\ x_2, y_2, z_2 \\ .. \\ x_{N_{bound}}, y_{N_{bound}}, z_{N_{bound}} \end{bmatrix} \quad (4)$$

Roof Visibility – The analytic solution for visibility boundaries does not treat the roof visibility of a building [18]. We simply check if viewpoint height *V(z_0)* is lower or higher than the building height $h_{\max_{C_i}}$ and use this to decide if the roof is visible or not:

$$V_{z_0} \geq Z = h_{\max_{C_i}} \quad (5)$$

If the roof is visible, roof surface boundary points are added to VBP. Roof visibility is an integral part of VBP computation for each building.

A simple case demonstrating analytic solution from a visibility point to a building including visible roofs can be seen in Figure 5. The visibility point is marked in black, the

visible parts colored in red, and the invisible parts colored in blue.

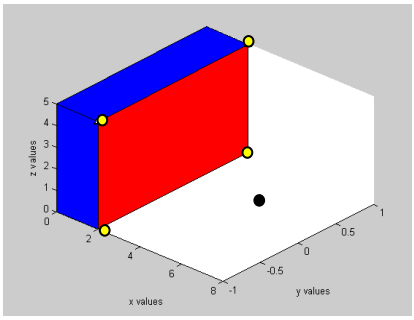


Figure 5. Visibility Volume Computed with the Analytic Solution. Viewpoint is marked in Black, Visible Parts Colored in red, and Invisible Parts Colored in Blue; VBP marked with Yellow Circles

In the previous part, we treated a single building case, without considering hidden surfaces between buildings, i.e., building surfaces (or parts of surfaces) occluded by other buildings, which directly affect the visibility volumes solution. In this section, we introduce our concept for visible volumes inside bounding volume by decreasing visible pyramids and projected pyramids to the bounding volume boundary. First, we define the relevant pyramids and volumes.

The Visible Pyramid (VP): we define $VP_i^{j=1..Nsurf}(x_0, y_0, z_0)$ of the object *i* as a 3D pyramid generated by connecting VBP of specific surface *j* to a viewpoint $V(x_0, y_0, z_0)$.

In the case of a box, the maximum number of N_{surf} for a single object is three. VP boundary, colored with green arrows, can be seen in Figure 6.

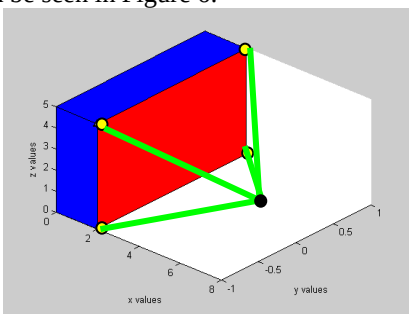


Figure 6. A Visible Pyramid from a Viewpoint (marked as a Black Dot) to VBP of a Specific Surface

For each VP, we calculate Projected Visible Pyramid (PVP), projecting VBP to the boundaries of the bounding volume S.

Projected Visible Pyramid (PVP) - we define $PVP_i^{j..N_{surf}}(x_0, y_0, z_0)$ of the object *i* as 3D projected points to the bounding volume *S*, VBP of specific surface *j* trough viewpoint $V(x_0, y_0, z_0)$. VVP boundary, colored with purple arrows, can be seen in Figure 7.

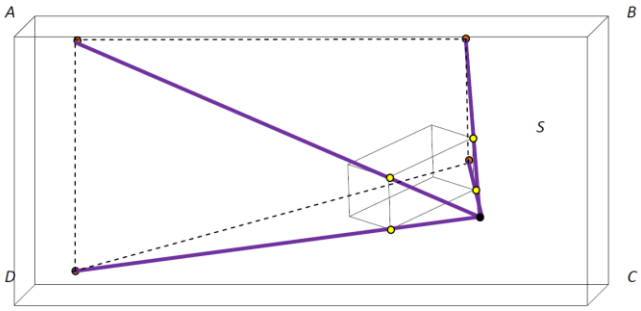


Figure 7. Invisible Projected Visible Pyramid Boundaries colored with purple arrows from a Viewpoint (marked as a Black Dot) to the boundary surface ABCD of Bounding Volume *S*

The 3D Visible Volumes inside bounding volume *S*, VV_S, computed as the total bounding volume *S*, V_S, minus the Invisible Volumes IV_S. In a case of no overlap between buildings, IV_S is computed by decreasing the visible volume from the projected visible volume, $\sum_{i=1}^{N_{obj}} \sum_{j=1}^{N_{surf}} (V(PVP_i^j) - V(VP_i^j))$.

$$VV_S = V_S - \sum_{i=1}^{N_{obj}} \sum_{j=1}^{N_{surf}} IV_{S_i}^j \quad (6)$$

$$VV_S = V_S - \sum_{i=1}^{N_{obj}} \sum_{j=1}^{N_{surf}} (V(PVP_i^j) - V(VP_i^j))$$

By decreasing the invisible volumes from the total bounding volume, only the visible volumes are computed, as seen in Figure 8. Volumes of VPV and VP can be simply computed based on a simple pyramid volume geometric formula.

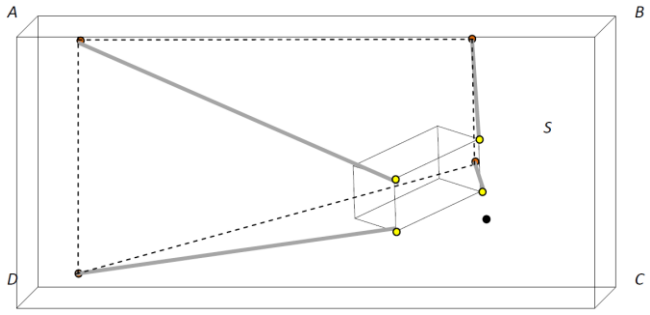


Figure 8. Invisible Volume $V(PVP_i^j) - V(VP_i^j)$ Colored in Gray Arrows. Decreasing Projected Visible Pyramid boundary surface ABCD of Bounding Volume S from Visible Pyramid

In a case of two buildings without overlapping, IV_S computed for each building, as presented above, as can be seen in Figure 9.

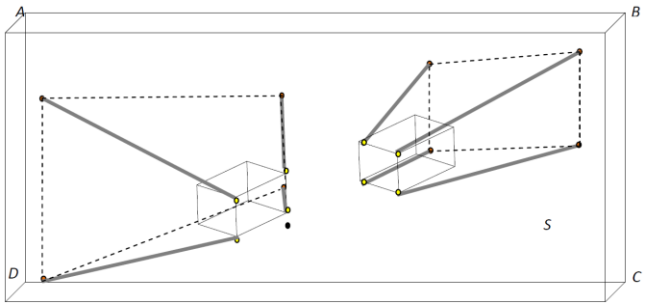


Figure 9. Invisible Volume $V(PVP_i^j) - V(VP_i^j)$ Colored in Gray Arrows. Decreasing Projected Visible Pyramid boundary surface ABCD of Bounding Volume S from Visible Pyramid

Considering two buildings with overlap between object's Visible Pyramids, as seen in Figure 10(a). In Figure 10(b), VP_1^1 boundary is colored by green lines, VP_2^1 boundary is colored by purple lines and the hidden and Invisible Surface between visible pyramids $IS_{VP_1^i}^{VP_2^i}$ is colored in white.

Invisible Hidden Volume (IHV) - We define Invisible Hidden Volume (*IHV*), as the *Invisible Surface (IS)* between visible pyramids projected to bounding box *S*.

For example, IHV in Figure 10(c) is the projection of the invisible surface between visible pyramids colored in white, projected to the boundary plane of bounding box *S*.

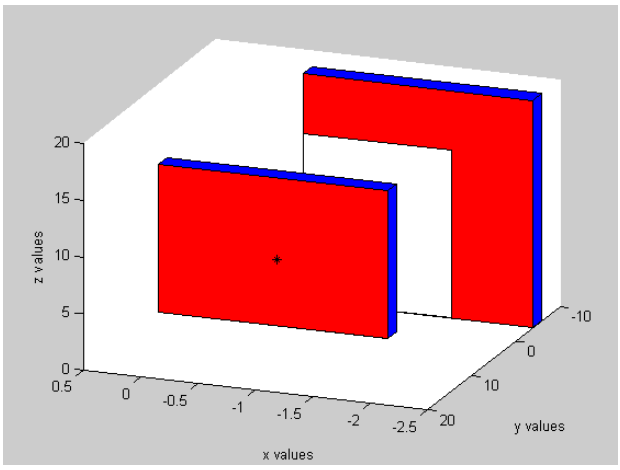
In the case of overlapping buildings, by computing invisible volumes IV_S, we decrease IHV twice between the overlapped objects, as can be seen in Figure 10(c), *IHV* boundary points denoted as $\{A_{11},..,A_{18}\}$. The same scene is presented in Figure 11, where Invisible Volume $V(PVP_i^j) - V(VP_i^j)$ is colored in purple and green arrows for each building.

The *PVP* of the object close to the viewpoint is marked in black, colored with pink circles denoted as boundary set points $\{B_{11},..,B_{18}\}$ and the far object's *PVP* is colored with orange circles, denoted as boundary set points $\{C_{11},..,C_{18}\}$. It can be seen that *IHV* is included in each of these invisible volumes, where $\{A_{11},..,A_{18}\} \in \{B_{11},..,B_{18}\}$ and $\{A_{11},..,A_{18}\} \in \{C_{11},..,C_{18}\}$.

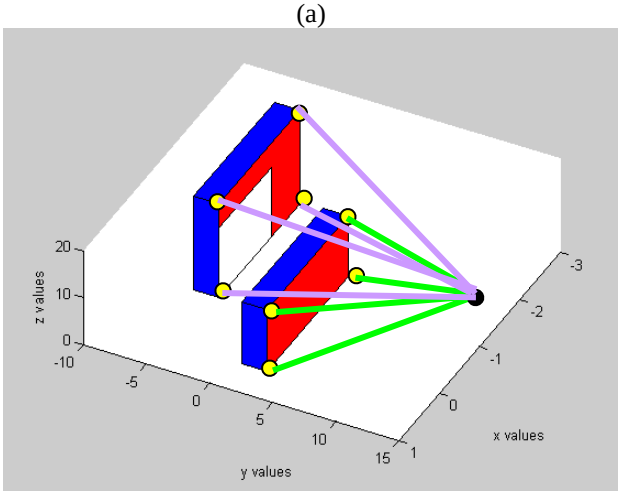
Therefore, we add *IHV* between each overlapping pair of objects to the total visible volume. In the case of overlapping between objects' visible pyramids, 3D visible volume is formulated as:

$$VV_S = V_S - \sum_{i=1}^{N_{obj}} \sum_{j=1}^{N_{surf}} (V(PVP_i^j) - V(VP_i^j) + IHV_i^j) \qquad (7)$$

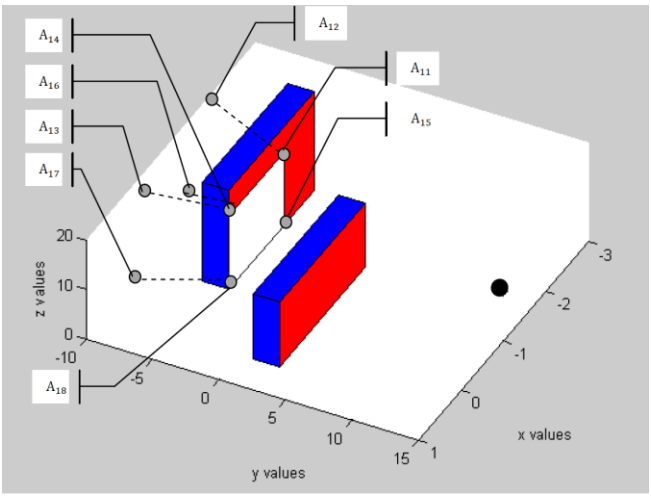
The same analysis holds true for multiple overlapping objects, adding the IHV between each two consecutive objects.



(a)



(b)



(c)

Figure 10. (a) Computing Hidden Surfaces between Buildings , VP_2^1 Base Plane, $IS_{VP_1^i}^{VP_2^i}$ (b) The Two Buildings - VP_1^1 in green and VP_2^1 in Purple (from the Viewpoint) and $IS_{VP_1^i}^{VP_2^i}$ in White (c) IHV boundary points colored with gray circles denoted

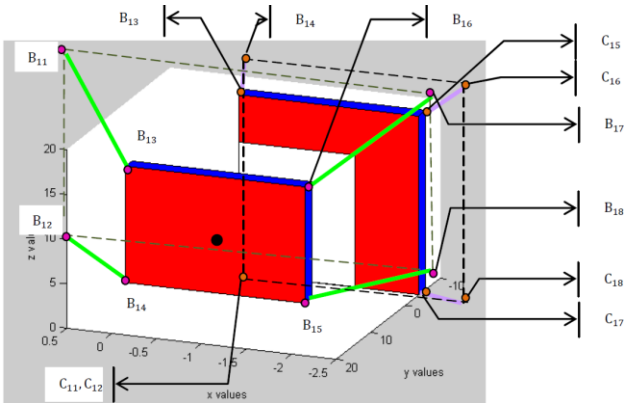


Figure 11. Invisible Volume $V(PVP_i^j) - V(VP_i^j)$ colored in purple and green arrows for each building. PVP of the object close to viewpoint colored in black, colored with pink circles and the far object PVP colored with orange circle

In Figure 12, we demonstrate the case of three buildings with overlapping. The invisible surfaces are bounded with dotted lines, while the projected visible surfaces to the overlapped building are colored in gray. In order to calculate the visible volumes from a viewpoint, *IHV* between each two buildings must be added as a visible volume, since it is already omitted at the previous step as an invisible volume.

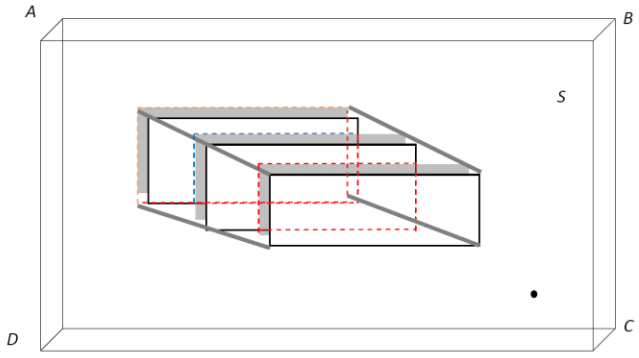


Figure 12. Three overlapping buildings. Invisible surfaces bounded with dotted lines, projected visible surfaces of the overlap building colored in gray

C. Simulations

In this section, we demonstrate the SVC method of estimating the number of clusters based on pedestrians' mobility datasets. For each pedestrian's location datasets, we analyze the 3D visible volumes inside bounding volume *S*, defined as a 3D box.

Our datasets are based on the city of Melbourne's 24-hour pedestrian monitoring system (24PM). This system measures pedestrian activity at several Points of Interests (POI) with counting sensors. Pedestrian mobility datasets are available online with interactive maps, as seen in Figure 13, and can be downloaded for a specific date.

Figure 13. City of Melbourne's 24-hour pedestrian monitoring system (24PM) – Online Visualization Map

Our datasets include the number of pedestrians in each hour during the 2nd of July 2013, at different seventeen points of interest in Melbourne where counting sensors are located and defined as viewpoints. Based on these datasets, we approximated the pedestrians' location using the well-known and common kinematic model for pedestrians presented by Hoogendoorn et al. [29]. Based on this model, pedestrian 2D location can be estimated as:

$$x(t + \Delta t) = x(t) + V(t)\Delta t + w \quad (8)$$

where *w* is a white noise of a standard Wiener Process which reflects the uncertainty in the expected traffic condition, described as Gaussian distribution.

Pedestrian speed *V* can be divided into three major groups:

1. Fast: 1.8 meters per second
2. Standard: 1.3 meters per second
3. Slow: 0.8 meters per second

$$V(t) \sim N(\mu = 1.3, \sigma^2 = 0.5)$$
$$w \sim \sqrt{\Delta t} N(0,1) \quad (9)$$

The kinematic model of a pedestrian is only a part of the estimation and prediction of his movement in an urban environment. For simplicity, we use only a kinematic model for a pedestrian's future location, since decision-making in this field is very complicated.

At time step t, pedestrian location $x(t)$, is taken from a specific POI from our dataset, and the estimated pedestrian location $x(t + \Delta t)$ can be computed. In our simulations we set Δt for five minutes. For example, pedestrians' 2D location in UTM coordination, using the Hoogendoorn etc. model [29], between 6-7 a.m., can be seen in Figure 14.

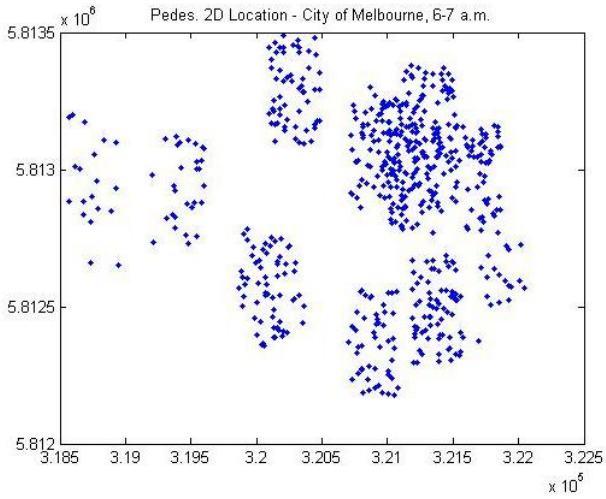


Figure 14. Pedestrians' 2D estimated location using the Hoogendoorn etc. model [29] between 6-7 a.m.

Each of pedestrian locations is processed as a viewpoint for estimating the number of clusters from spatial visibility aspects. The 3D visible volumes computation presented in the previous section are applied for computing T_k, as described in Section III.

At each POI, we set the reference dataset of the pedestrian location distributed uniformly around the POI location, where the reference dataset size is the same one as the original dataset for the same POI, computing T_k^*.

We set the possible number of clusters from one to ten, demonstrating the SVC method. The number of clusters based on visible volumes analysis per day hour is presented in Figure 15.

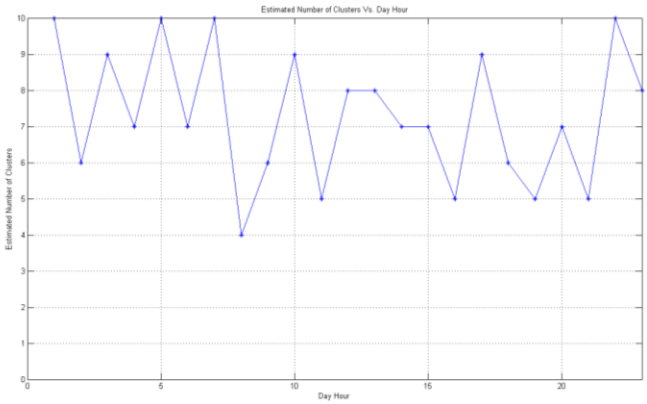


Figure 15. Number of Clusters for each Hour of 2/7/2013 Using SVC

As we can see in Figure 15, there is a correlation between the number of clusters and the pedestrians' mobility behavior. The number of clusters is close to the maximum (ten clusters in our case) during 6-9 AM, as can be predicted due to pedestrians' mobility while going to work. The number of clusters drops to a figure between eight to four clusters during the midday hours, and climbs again during nigh hours. More incentives analyzing pedestrians' mobility patters are presented in the next section.

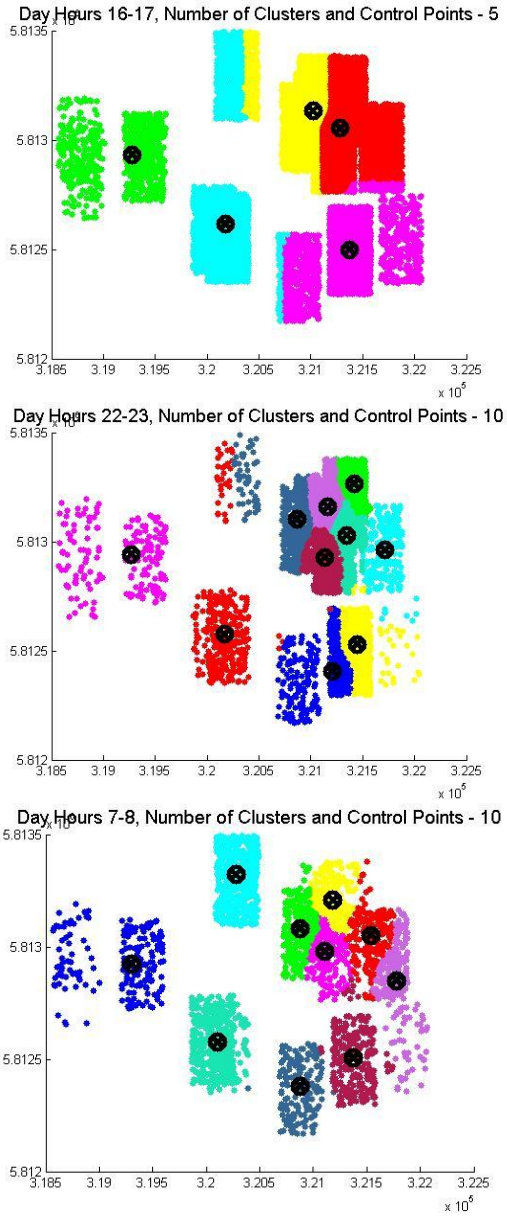


Figure 16. Control Points Location and Clusters Presentation during Each Hour in a Day. Control points are marked with black circles. Pedestrians' mobility Clustered in different colors

IV. ANALYZING PEDESTRIANS' MOBILITY DATASETS

A. Control Points

In this section, we analyze pedestrians' mobility datasets during one day, estimating the number of clusters by using the SVC outcome, which is based on visibility analysis. Upon that, we use the K-means clustering method.

K-means clustering intends to partition n objects into k clusters, where each object belongs to the cluster with the nearest mean. The centroid of all objects in each cluster

is set as control point. This method produces exactly k different clusters, where k is predefined from the SVC method. The objective of K-means clustering is to minimize total intra-cluster variance, or the squared error function. K-means algorithm stages can be described as:

1. Cluster the data into *k* groups, where *k* is predefined from the SVC method.
2. Select *k* points at random as cluster centers.
3. Assign objects to their closest cluster center using Euclidean distance function.
4. Calculate the centroid all objects in each cluster.
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster.

By using K-means and SVC method, control points location can be seen in Figure 16.

It can be noticed in Figure 16 that in some cases the geometric location of the sensor location is separated into two different clusters. Our maximal number of clusters is set to ten, whereas there are seventeen sensors. We set the maximal number of clusters to be smaller than the number of sensors on the scene. One of the major contributions of our work, related to the adaptive clustering capability, is separating datasets into a different clustering and setting the control points from a visibility aspect. Moreover, control point location should cover more than one area, as can be seen in Figure 14, and also depends on pedestrians' mobility during this hour, as can be seen in the next sub-section.

Video simulations showing control points locations using K-means clustering and SVC methods are available in [56].

B. *Time Zones*

In this section, we concentrate on learning pedestrians' patterns for setting *Optimal Control Points (OCP)*, i.e., control points for each time zone. We divide the day into four time zones for efficient pedestrian monitoring:

1. Morning hours (movement to work) – 6 – 9 AM.
2. Mid-Day Hours (between morning and afternoon) – 10 AM to 16 PM.
3. Afternoon hours (back from work and activity hours) – 17- 20 PM.
4. Night hours 20 – 23 PM.

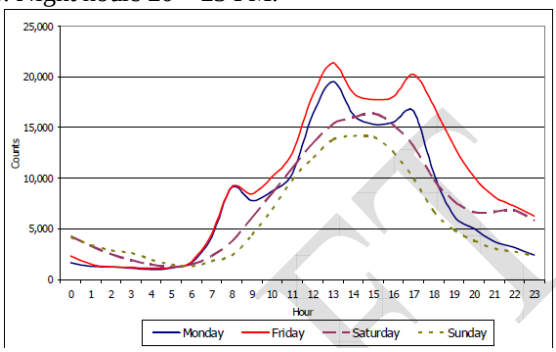


Figure 17. Pedestrian Activity Analysis [45]

The suggested division of time zones partition can also be seen clearly in an official pedestrian monitoring report of the city of Melbourne [45] (see Figure 17). The number of pedestrians counted by the monitoring system rises at the suggested time zones.

In order to get reliable and comprehensive results regarding pedestrian mobility patterns, we tested a full month's (July 2013) dataset, analyzing each day for twenty-four hours.

Based on the average estimated number of clusters using SVC on these datasets, we found out that the number of optimal control points during these time zones is:

- Morning hours – Nine control points
- Mid-Day Hours – Six control points
- Afternoon hours – Seven control points
- Night hours – Eight control points

The localization of the optimal control points and the number of clusters for each time zone can be seen in Figure 18.

It can be seen that in the different time zones, three optimal control points and their cluster division are almost identically marked with arrows and numbers in Figure 18.

Four optimal control points with similar clustering can be seen in three time zones in Figure 18. These results can also be applicable for personal-security and homeland security application in urban environments, localizing forces and sensors for optimal monitoring and trajectory planning during a daylight hours.

V. 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.

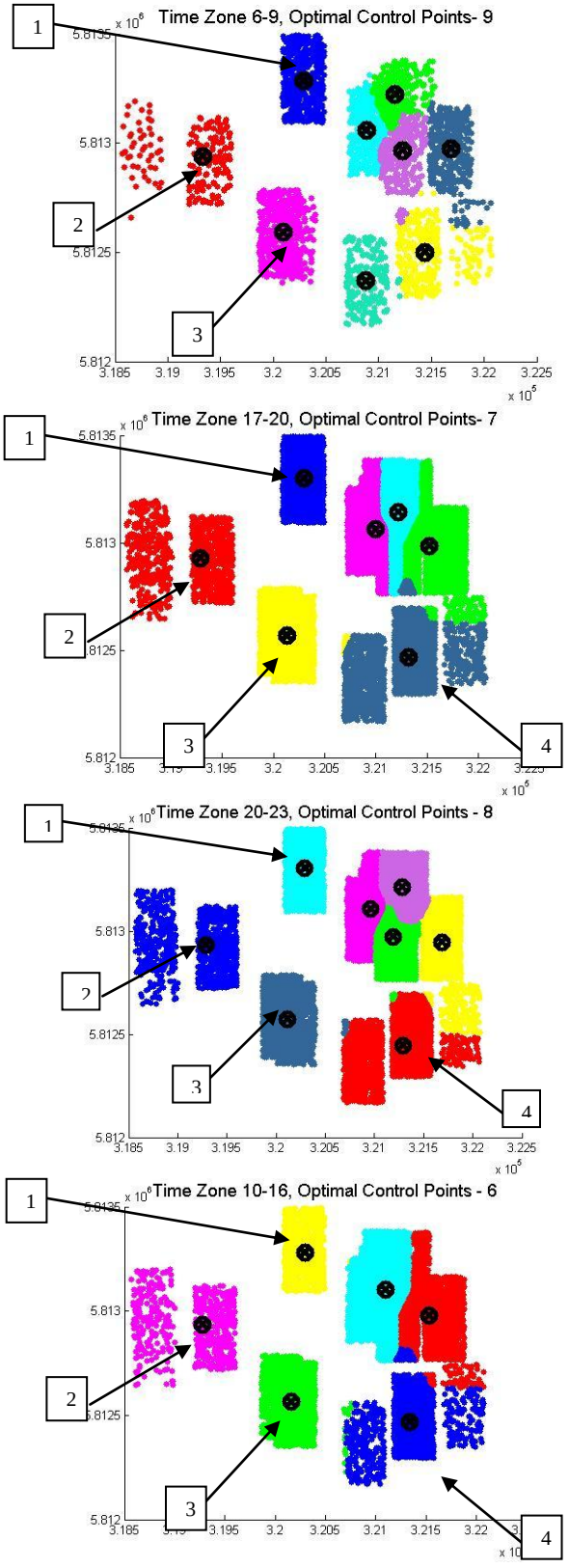


Figure 18. Optimal Control Points Location in Four Time Zones. Optimal Control points marked with black circles. Pedestrians' mobility Clustered in different colors

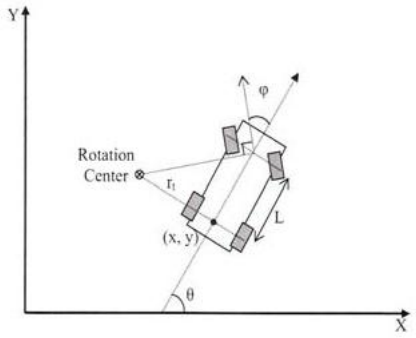


Figure 19. Four-Wheeled Car Model with Front-Wheel Steering [Lewis]

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 [42]. 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.

Thus, UGV dynamic model can be described as:

$$\dot{x} = f(x,u) = \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{Bmatrix} = \begin{Bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(\emptyset) \end{Bmatrix} \quad (10)$$

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.

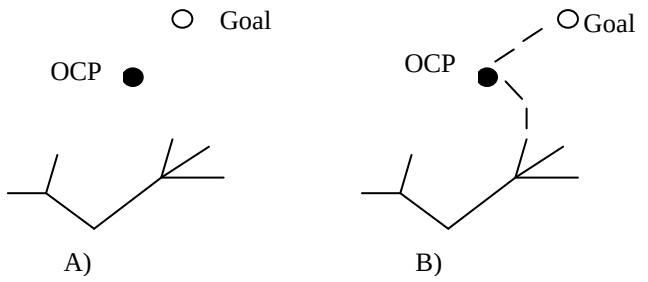


Figure 20. STP Search Method: (A) Start and Goal Points; (B) Explored Space to the Goal Through OCP

C. STP Planner Pseudo-Code

We present our STP planner pseudo code described in Table II, for spatial case generating trajectory T with 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 to 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.

VI. 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.

The SVC method is based on fast and efficient 3D visible volumes computation. Estimating the number of clusters is based on minimum normalized visible volumes to reference datasets distributed uniformly inside bounding volume S. We demonstrated the SVC by using datasets from the city of Melbourne's 24-hour pedestrian monitoring system (24PM).

In the second part of this research, based on the SVC-estimated number of clusters, we analyzed pedestrians' mobility behavior, setting control points during daylight hours and dividing a daylight hours into four time zones. We found a correlation of several optimal control points in different time zones.

Based on similar spatial analysis in other urban scenes, one can set optimal control points for various applications, such as entertainment events that can be efficiently visible at such points, or monitoring crowds' movements from these control points in emergencies, planning medical assistance.

The STP concept includes probabilistically complete properties which changes dynamically during daylight hours. The planner allows us to generate trajectory for various applications such as personal security and homeland security applications in urban environments, localizing police forces and sensors for optimal monitoring at different hours of a day.

Future work will focus on simulation in real data records using the STP planner, generating trajectories in 2D and 3D urban environments using an Unmanned Aerial Vehicle (UAV) model. Future research will also include performances and algorithm complexity analysis for STP and SVC methods.

VII. 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, 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, pp. 364–368, 2000.

[5] O. Brock and O. Khatib, "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, 2000.

[6] R. B. Calinski and J. Harabasz, "A Dendrite Method for Cluster Analysis," Communications in Statistics, vol. 3, pp. 1–27, 1974.

[7] 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, 2001.

[8] 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.

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

[10] 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.

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

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

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

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

[15] 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.

[16] T. Fraichard, "Trajectory planning in a dynamic workspace: A 'state-time space' approach," Advanced Robotics, vol. 13, pp. 75–94, 1999.

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

[18] 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 [accessed February 2014].

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

[20] 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.

[21] O. Gal and Y. Doytsher, "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, vo 4, April 2013, ISSN 2251-6379, [accessed February 2014].

[22] O. Gal and Y. Doytsher, "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 DOI: http://dx.doi.org/10.1017/S0263574713000787, [accessed February 2014].

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

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

[25] 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.

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

[27] J. Hartigan, "Clustering Algorithms". John Wiley & Sons, New York, NY, 1975.

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

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

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

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

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

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

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

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

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

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

[38] J. C, Latombe, "Robot Motion Planning,", Kluwer Academic Press, 1990.

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

[40] S. M. LaValle, "Planning Algorithms," Cambridge,U.K.:Cambridge Univ. Press, 2006.

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

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

[43] 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, August 2006.

[44] Melbourne City: http://www.pedestrian.melbourne.vic.gov.au/#date=31-08-2013&time=9 [accessed February 2014].

[45] Melbourne Report: https://docs.google.com/file/d/0B380gpj_lbU-dnRaRTFXdlh5Znc/edit [accessed February 2014].

[46] G.W. Milligan and M. C. Cooper, "An Examination of Procedures for Determining the Number of Clusters in a Data set," Psychometrika, vol. 50, pp. 159–179, 1985.

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

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

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

[50] K. J. Obermeyer, "Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain," in Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago, 2009.

[51] 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.

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

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

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

[55] V. Shaferman and T. Shima, "Co-evolution genetic algorithm for UAV distributed tracking in urban environments," in ASME Conference on Engi- neering Systems Design and Analysis, July 2008.

[56] https://sites.google.com/site/orenusv/home/svc [accessed February 2014].

[57] T. Schelhorn, D. Sullivan, and M. Haklay, "STREETS: An agent-based pedestrian model," http://www.casa.ucl.ac.uk/streets.pdf., 1999, [accessed February 2014].

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

[59] R. Tibshirani, G. Walther, and T. Hastie, "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, 2001.

[60] 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.