Fast Visible Trajectory Spatial Analysis in 3D Urban Environments Based on Local Point Clouds Data

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 fast and efficient visible trajectory planning for unmanned vehicles in a 3D urban environment based on local point clouds data. Our trajectory planning method is based on a two-step visibility analysis in 3D urban environments using predicted visibility from point clouds data. The first step in our unique concept is to extract basic geometric shapes. We focus on three basic geometric shapes from point clouds in urban scenes: planes, cylinders and spheres, extracting these geometric shapes using efficient Random Sample Consensus (RANSAC) algorithms with a high success rate of detection. The second step is a prediction of these geometric entities in the next time step, formulated as states vectors in a dynamic system using Kalman Filter (KF). Our planner is based on the optimal time horizon concept as a leading feature ofour greedy search method, making our local planner safer. We demonstrate our visibility and trajectory planning method in simulations, showing predicted trajectory planning in 3D urban environments based on real Light Detection and Ranging (LiDAR) point clouds data.**

Keywords-Visibility; 3D; Urban environment; Spatial analysis.

I. INTRODUCTION AND RELATED WORK

In this paper, we study a fast and efficient visible trajectory planning for unmanned vehicles in a 3D urban environment, based on local point clouds data. Recently, urban scene modeling has become more and more precise, using Terrestrial/ground-based LiDAR on unmanned vehicles to generate point clouds data for modeling roads, signs, lamp posts, buildings, trees and cars. Visibility analysis in complex urban scenes is commonly treated as an approximated feature due to computational complexity.

Our trajectory planning method is based on a two-step visibility analysis in 3D urban environments using predicted visibility from point clouds data. The first step in our unique concept is to extract basic geometric shapes. We focus on three basic geometric shapes from point clouds in urban scenes: planes, cylinders and spheres, extracting these geometric shapes using efficient RANSAC algorithms with a high success rate of detection. The second step is a prediction of these geometric entities in the next time step, formulated as states vectors in a dynamic system using KF.

Visibility analysis based on this approximated scene prediction is done efficiently, based on our analytic solutions for visibility boundaries. With this capability, we present a local on-line planner generating visible trajectories, exploring the most visible and safe node in the next time step, using our predicted visibility analysis, which is based on local point clouds data from the unmanned LiDAR vehicle. Our planner is based on the optimal time horizon concept as a leading feature ofour greedy search method for making our local planner safer.

For the first time, we propose a solution for the basic limitation of the Velocity Obstacle (VO) search and planning method, i.e., when all the available dynamic velocities for the next time step are blocked in the velocity space and there is no feasible node at the next time step of the greedy search. Computation of the minimum time horizon is formulated as a minimum time problem that generates optimal trajectories in near-real time to the goal, exploring the safest and most visible node in the next time step. We demonstrate our visibility and trajectory planning method in simulations showing predicted trajectory planning in 3D urban environments using real LiDAR data from the Ford Campus Project [1].

II. VISIBILITY ANALYSIS FROM POINT CLOUDS DATA

As mentioned, visibility analysis in complex urban scenes is commonly treated as an approximated feature due to its computational complexity. Recently, urban scene modeling has become more and more exact, using Terrestrial/ground-based LiDAR generating dense point clouds data for modeling roads, signs, lamp posts, buildings, trees and cars. Automatic algorithms detecting basic shapes and their extraction have been studied extensively, and are still a very active research field [2].

In this part, we present a unique concept for predicted and approximated visibility analysis in the next attainable vehicle's state at a one-time step ahead in time, based on local point clouds data which is a partial data set.

We focus on three basic geometric shapes in urban scenes: planes, cylinders and spheres, which are very common and can be used for the majority of urban entities in modeling

 ISBN: 978-1-61208-539-5

scenarios. Based on point clouds data generated from the current vehicle's position in state k-1, we extract these geometric shapes using efficient RANSAC algorithms [3] with high success rate detection tested in real point cloud data.

After extraction of these basic geometric shapes from local point clouds data, our unified concept, and our main contribution, focus on the ability to predict and approximate urban scene modeling at the next view point V_k, i.e., at the attainable location of the vehicle in the next time step. Scene prediction is based on the geometric entities and the KF), which is commonly used in dynamic systems for tracking target systems [4],[5]. We formulate the geometric shapes as states vectors in a dynamic system and predict the scene structure the in the next time step, k.

Based on the predicted scene in the next time step, visibility analysis is carried out from the next view point model [6], which is, of course, an approximated one. As the vehicle reaches the next viewpoint V_k, point clouds data are measured and scene modeling and states vectors are updated, which is an essential procedure for reliable KF prediction.

A. Shapes Extraction

1) Geometric Shapes:

The urban scene is a very complex one in the matter of modeling applications using ground LiDAR, and the generated point clouds arevery dense. Despite these inherentcomplications, feature extraction can be made very efficient by using basic geometric shapes. We define three kinds of geometric shapes: planes, cylinders and spheres, with a minimal number of parameters for efficient time computation.

Plane: center point (x,y,z) and unit direction vector from center point.

Cylinder: center point (x,y,z), radius and unit direction vector of the cylinder axis. Cylinder height dimension will be consider later on as part of the simulation.

Sphere: center point (x,y,z), radius and unit direction vector from center point.

2) RANSAC:

The RANSAC [7] is a well-known paradigm, extracting shapes from point clouds using a minimal set of a shape's primitives generated by random drawing in a point clouds set. Minimal set is defined as the smallest number of points required to uniquely define a given type of geometric primitive.

For each of the geometric shapes, points are testedtoapproximate the primitive of the shape (also known as "score of the shape"). At the end of this iterative process, extracted shapes are generated from the current point clouds data.

Based on the RANSAC concept, the geometric shapes detailed above can be extracted from a given point clouds data set. In order to improve the extraction process and reduce the number of points validating shape detection, we compute the approximated surface normal for each point and test the relevant shapes.

Given a point-clouds $P=\{p_1..p_N\}$ with associated normals $\{n_1..n_N\}$, the output of the RANSAC algorithm is a set of primitive shapes $\{\delta_1..\delta_N\}$ and a set of remaining points $R=P\setminus\{p_{\delta_1}..p_{\delta_N}\}$.

B. Predicted Scene – Kalman Filter

In this part, we present the global KF approach for our discrete dynamic system at the estimated state, *k*, based on the defined geometric shapes formulation defined in the previous sub-section.

Generally, the Kalman Filter can be described as a filter that consists of three major stages: Predict, Measure, and Update the state vector. The state vector contains different state parameters, and provides an optimal solution for the whole dynamic system [5]. We model our system as a linear one with discrete dynamic model, as described in (1):

$$x_k = F_{k,k-1}x_{k-1} \tag{1}$$

where x is the state vector, F is the transition matrix and k is the state.

The state parameters for all of the geometric shapes are defined with shape center $\vec{s}$, and unit direction vector $\vec{d}$, of the geometric shape, from the current time step and viewpoint to the predicted one.

In each of the current states *k*, geometric shape center $\vec{s_k}$, is estimated based on the previous update of shape center location $\vec{s}_{k-1}$, and the previous updated unit direction vector $\vec{d}_{k-1}$, multiplied by small arbitrary scalar factor *c*, described in (2):

$$\vec{s}_k = \vec{s}_{k-1} + c\vec{d}_{k-1} \tag{2}$$

Direction vector $\vec{d}_k$ can be efficiently estimated by extracting the rotation matrix T, between the last two states *k, k-1*. In case of an inertial system fixed on the vehicle, a rotation matrix can be simply found from the last two states of the vehicle translations in (3):

$$\vec{d}_k = T\vec{d}_{k-1} \tag{3}$$

The 3D rotation matrix T tracks the continuous extracted plans and surfaces to the next viewpoint V_k, making it possible to predict a scene model where one or more of the geometric shapes are cut from current point clouds data in state *k-1*. The discrete dynamic system can be written as formulated in (4):

 ISBN: 978-1-61208-539-5

$$\begin{bmatrix} \vec{s}_{x_k} \\ \vec{s}_{y_k} \\ \vec{s}_{z_k} \\ \vec{d}_{x_k} \\ \vec{d}_{y_k} \\ \vec{d}_{z_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & c & 0 & 0 \\ 0 & 1 & 0 & 0 & c & 0 \\ 0 & 0 & 1 & 0 & 0 & c \\ 0 & 0 & 0 & T_{11} & T_{12} & T_{13} \\ 0 & 0 & 0 & T_{21} & T_{22} & T_{23} \\ 0 & 0 & 0 & T_{31} & T_{32} & T_{33} \end{bmatrix} \begin{bmatrix} \vec{s}_{x_{k-1}} \\ \vec{s}_{y_{k-1}} \\ \vec{s}_{z_{k-1}} \\ \vec{d}_{x_{k-1}} \\ \vec{d}_{y_{k-1}} \\ \vec{d}_{z_{k-1}} \end{bmatrix} \quad (4)$$

where the state vector x is 6×1 vector, and the transition squared matrix is $F_{k,k-1}$. The dynamic system can be extended to additional state variables representing someof the geometric shape parameters such as radius, length etc. We define the dynamic system as the basic one for generic shapes that can be simply modeled with center and direction vector. Sphere radius and cylinder Z boundaries are defined in an additional data structure of the scene entities.

III. FAST AND APPROXIMATED VISIBILITY ANALYSIS

In this section, we present an analytic analysis of the visibility boundaries of planes, cylinders and spheres for the predicted scene presented in the previous sub-section, which leads to an approximated visibility. For the plane surface, fast and efficient visibility analysis was already presented in [6].

In this part, we extend the previous visibility analysis concept [6] and include cylinders as continuous curves parameterization $C_{cln\,d}(x,y,z)$.

Cylinder parameterization can be described in (5):

$$C_{Cln\,d}(x,y,z) = \begin{pmatrix} r\sin(\theta) \\ r\cos(\theta) \\ c \end{pmatrix}_{r=const}, \quad \begin{matrix} 0 \le \theta \le 2\pi \\ c = c+1 \\ 0 \le c \le h_{peds\_\max} \end{matrix} \quad (5)$$

We define the visibility problem in a 3D environment for more complex objects as:

$$C'(x,y)_{z_{const}} \times (C(x,y)_{z_{const}} - V(x_0,y_0,z_0)) = 0 \quad (6)$$

where 3D model parameterization is $C(x,y)_{z=const}$, and the viewpoint is given as $V(x_0,y_0,z_0)$. Extending the 3D cubic parameterization, we also consider the case of the cylinder. Integrating (5) to (6) yields:

$$\begin{pmatrix} r\cos\theta \\ -r\sin\theta \\ 0 \end{pmatrix} \times \begin{pmatrix} r\sin\theta - V_x \\ r\cos\theta - V_y \\ c - V_z \end{pmatrix} = 0 \quad (7)$$

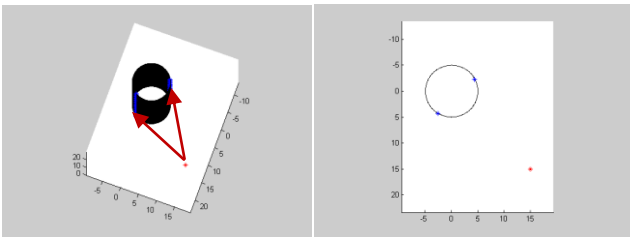
$$\theta = \arctan\left(\frac{-r - \frac{\left(-vy\,r + \sqrt{vx^4 - vx^2\,r^2 + vy^2\,vx^2}\right) vy}{vx^2 + vy^2}}{vx}, \frac{-vy\,r + \sqrt{vx^4 - vx^2\,r^2 + vy^2\,vx^2}}{vx^2 + vy^2} \right) \quad (8)$$

As can be noted, these equations are not related to Z axis, and the visibility boundary points are the same for each *x-y* cylinder profile, as seen in (7), (8).

The visibility statement leads to complex equation, which does not appear to be a simple computational task. This equation can be efficiently solved by finding where the equation changes its sign and crosses zero value; we used analytic solution to speed up computation time and to avoid numeric approximations. We generate two values of θ generating two silhouette points in a very short time computation. Based on an analytic solution to the cylinder case, a fast and exact analytic solution can be found for the visibility problem from a viewpoint.

We define the solution presented in (8) as x-y-z coordinates values for the cylinder case as Cylinder Boundary Points (CBP). CBP, defined in (9), are the set of visible silhouette points for a 3D cylinder, as presented in Figure 1:

$$CBP_{i=1..N_{PBP\_bound}=2}(x_0,y_0,z_0) = \begin{bmatrix} x_1, y_1, z_1 \\ x_{N_{PBP\_bound}}, y_{N_{PBP\_bound}}, z_{N_{PBP\_bound}} \end{bmatrix} \quad (9)$$



(a) (b)

Figure 1. Cylinder Boundary Points (CBP) using Analytic Solution marked as blue points, Viewpoint Marked in Red: (a) 3D View (Visible Boundaries Marked with Red Arrows); (b) Topside View.

In the same way, sphere parameterization can be described as formulated in (10):

$$C_{Sphere}(x,y,z) = \begin{pmatrix} r\sin\phi\cos\theta \\ r\sin\phi\sin\theta \\ r\cos\phi \end{pmatrix}_{r=const} \quad 0 \le \phi < \pi \quad 0 \le \theta < 2\pi \quad (10)$$

We define the visibility problem in a 3D environment for this object in (11):

$$C'(x,y,z) \times (C(x,y,z) - V(x_0,y_0,z_0)) = 0 \quad (11)$$

where the 3D model parameterization is $C(x,y,z)$, and the viewpoint is given as $V(x_0,y_0,z_0)$. Integrating (10) to (11) yields:

$$\theta = \arctan\left(\frac{r\sin(\phi)}{v\_y} - \frac{1}{v\_y\,(v\_y^2 + v\_x^2)} \left(v\_x \left(r\sin(\phi)\, v\_x - \sqrt{-v\_y^2 r^2 \sin(\phi)^2 + v\_y^4 + v\_x^2 v\_y^2} \right) \right), \frac{r\sin(\phi)\, v\_x - \sqrt{-v\_y^2 r^2 \sin(\phi)^2 + v\_y^4 + v\_x^2 v\_y^2}}{v\_y^2 + v\_x^2} \right) \quad (12)$$

where*r* is definedfrom sphere parameter, and $V(x_0,y_0,z_0)$ arechanges from visibility point along Z axis, as described in (12). The visibility boundary points for a sphere, together with the analytic solutions for planes and cylinders, allow us to compute fast and efficient visibility in a predicted scene from local point cloud data, which are updated in the next state.

This extended visibility analysis concept, integrated with a well-known predicted filter and extraction method, can be implemented in real time applications with point clouds data.

IV. FAST VISIBLE TRAJECTORY PLANNING

In this part, we focus on the efficiency of our analytic time horizon solution via classic VO, as demonstrated in simulations.

We use a planner similar to the one presented by [8] with the same cost function, and the Omni-directional robot model mentioned above. For one obstacle, our planner can ensure safety, but the planner is not a complete one. By using an analytic search, the planner computes near-time optimal and safe trajectory to the goal.

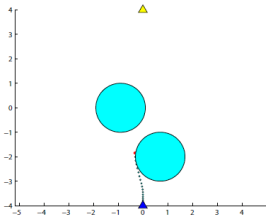


Figure 2. Avoiding Two Obstacles Using Analytic Time Horizon.

As a result, conservative trajectories are computed, althoughin some casesa safe trajectory to the goal cannot be found and collision eventually occurs. In a two-obstacles case, shown in Figure 2, the robot, represented by a point, starts near point (0,-4) at zero speed, attempting to reach the goal at point (0,4) (marked by a yellow triangle) at zero speed, while avoiding two static obstacles. The trajectory is dotted with a red dot representing the current position of the robot. The bounded velocity space, representing VO as yellow cycles and velocity vector (with green triangles), can be seen in Figure 3, relating to the state position in space as shown in Figure 2.

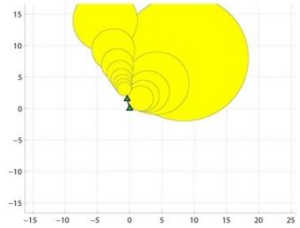


Figure 3. Blocked Velocity Space Avoiding Two Obstacles.

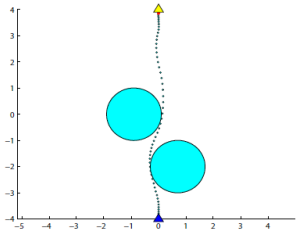


Figure 4. Final Trajectory Avoiding Two Obstacles Using Analytic Time Horizon.

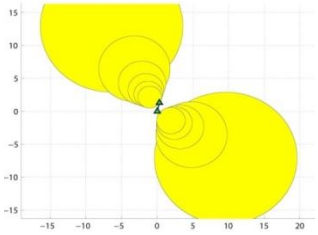


Figure 5. Escaping Blocked Velocity Space Using Analytic Time Horizon.

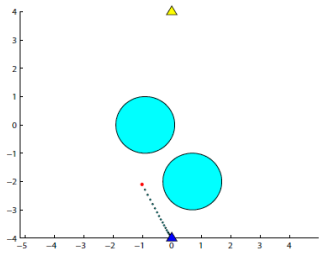


Figure 6. Conservative Solution of Avoiding Two Obstacles.

A. The Planner

As mentioned above, our planner is based on an iterative local planning method. By using RANSAC algorithm, point clouds data are extractedat each time step into three possible objects: plane, cylinder and sphere. The scene is formulated as a dynamic system using KF analysis for objects'

prediction. The objects are approximated for the next time step, and each safe attainable state that can be explored is set as candidate viewpoint. The cost for each node is set as total visible surfaces, based on the analytic visibility boundary, where the optimal and safe node is explored for the next time step.

At each time step, the planner computes the next Attainable Velocities (AV). The safe nodes not colliding with objects such as cubes, cylinders and spheres, i.e., nodes outside VO, are explored. Where all nodes are inside VO, a unified analytic solution for time horizon is presented, generating an escape option for these radical cases without affectingvisibility analysis. The planner computes the cost for these safe nodes based on predicted visibility and chooses the node with the optimal cost for the next time step. We repeat this procedure while generating the most visible trajectory.

1) Attainable Velocities

The set of maneuvers that are dynamically feasible over a time step is represented by AV. At each time step during the trajectory planning, we map the attainable velocities that the robot can choose under the effort control envelope.

Attainable Velocities, $AV(t+\Delta t)$, are integrated from the current state (x_1, x_2) by applying all admissible controls $u(t) \in U$. The geometric shape of AV depends on system dynamics. In our case,as described in (13):

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= u \end{aligned} \qquad (13)$$

where $x_1, u \in R^2$.

$$AV(t+\Delta t) = \{v | v = v(t) + \Delta t u, u \in U\}$$

The attainable velocities at time $t+\Delta t$ apply to the position $x(t+\Delta t)$. Thus, the attainable velocities, when intersected with VO that correspond to the same position, would indicate those velocities that are safe if selected at time $t+\Delta t$.

2) Cost Function

Our search is guided by minimum invisible parts from viewpoint *V* to the approximated 3D urban environment model in the next time step, $t+\Delta t$, set by KF after extracting objects from point clouds data using the RANSAC algorithm. The cost function for each node is a combination of IRV and ISV, with different weights as functions of the required task.

The cost function presented in (14) is computed for each safe node, i.e., node outside VO, considering the robot's future location at the next time step $(x_1(t+\Delta t), x_2(t+\Delta t))$ as viewpoint:

$$w\big(x(t+\Delta t)\big) = \alpha \cdot ISV(x(t+\Delta t)) + \beta \cdot IRV(x(t+\Delta t)) \qquad (14)$$

where $\propto, \beta$ are coefficients affecting the trajectory's character, as shown in (14). The cost function $w(x(t+\Delta t))$ produces the total sum of invisible parts from the viewpoint to the 3D urban environment, meaning that the velocity at the next time step with the minimum cost function value is the most visible node in our local search, based on our approximation.

We divide point invisibility value into Invisible Surfaces Value (ISV) and Invisible Roofs Value (IRV). This classification allows us to plan delicate and accurate trajectoriesupon demand. We define ISV and IRS as the total sum of the invisible roofs and surfaces (respectively). Invisible Surfaces Value (ISV) of a viewpoint is defined as the total sum of the invisible surfaces of all the objects in a 3D environment, as described in (15):

$$ISV(x_0, y_0, z_0) = \sum_{i=1}^{N_{obj}} IS_{VP_i}^{VP_i^{j=1..N_{bound}-1}} \qquad (15)$$

In the same way, we define Invisible Roofs Value (IRV) as the total sum of all the invisible roofs' surfaces, as described in (16):

$$IRV(x_0, y_0, z_0) = \sum_{i=1}^{N_{obj}} IS_{VP_i}^{VP_i^{j=N_{bound}}} \qquad (16)$$

Extended analysis of the analytic solution for visibility analysis for known 3D urban environments can be found in [6].

V. SIMULATIONS

We implemented the presented algorithm and tested some urban environments on a 1.8GHz Intel Core CPU with Matlab. We computed the visible trajectories using our planner, with real raw data records from LiDAR as part of the Ford Campus Project.

Point clouds data are generated by Velodyne HDL-64E LiDAR [9]. Velodyne HDL-64E LiDAR has two blocks of lasers, each consisting of 32 laser diodes aligned vertically, resulting in an effective 26:8 Vertical Field Of View (FOV). The entire unit can spin about its vertical axis at speeds of up to 900 rpm (15 Hz) to provide a full 360-degree azimuthal field of view. The maximum range of the sensor is 120 m and it captures about 1 million range points per second. We captured our data set with the laser spinning at 10 Hz.

Due to these huge amounts of data, we planned a limited trajectory in this urban environment for a limited distance. In Figure 7, point clouds data from the start point can be seen, also marked as start point "S" in Figure 10. Planes extracted by RANSAC can be recognized. As part of the Ford Project, these point clouds are also projected to the panoramic cameras' systems, making it easier to understand the scene, as seen in Figure 8.

As described earlier, at each time step the planner predicts the objects in the scene using KF. In Figure 9(a), objects in

the scene are presented from a point clouds data set. These point clouds are predicted using KF, and predicted to the next time step in Figure 9(b).

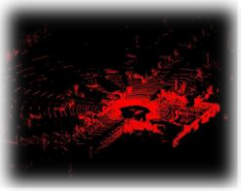


Figure 7. Point Clouds Data set at Start Point.

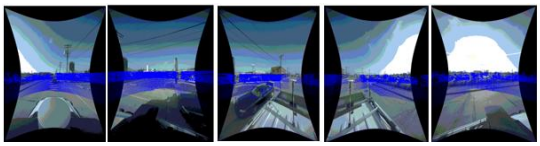
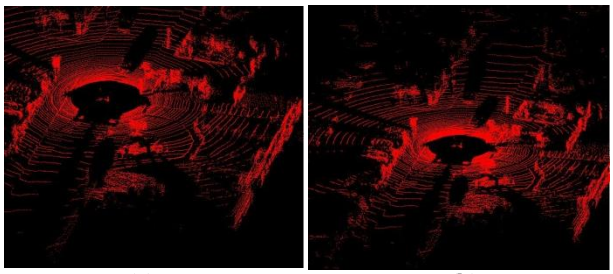


Figure 8. Point Clouds Data Projected to Panoramic Camera Set at Start Point.



(a) (b)

Figure 9. (a) Objects in point clouds data set. (b) Predicted objects using KF in the next time step.

Figure 10. Vehicle Planned Trajectory Colored in Purple.

The planned trajectory is presented in Figure 10 by a purple line. The starting point, marked as "S", is presented in Figure 10, where the cloud points in this state are presented in Figure 8. An arbitrary state during the planned trajectory, which is marked with an arrow, is also presented in Figure 10, where point clouds prediction using KF in this state are presented in Figure 9. For this trajectory, $\propto = 1, \beta = 1$, robot velocity is set to $v_a = 10 \left[\frac{km}{hr}\right]$. In this case, the robot avoided two other cars, without handling cases of analytic optimal time solution for deadlocks with bounded velocity space.

VI. Conclusion and Future Work

In this research, we have presented an efficient trajectory planning algorithm for visible trajectories in a 3D urban environment for an Omni-directional model, based on an incomplete data set from LiDAR, predicting the scene at the next time step and approximating visibility.

We extend our analytic visibility analysis method to cylinders and spheres, which allows us to efficiently set the visibility boundary of predicted objects in the next time step, generated by KF and RANSAC methods. Based on these fast computation capabilities, the on-line planner can approximate the most visible state as part of a greedy search method.

As part of our planner, we extended the classical VO method, where the velocity space is bounded and the robot velocity cannot escape from the VO in the current state. Further research will focus on advanced geometric shapes, which will allow precise urban environment modeling, facing real-time implementation with on-line data processing from LiDAR.

References

[1] G.Pandey, J.R. McBride, R.M. Eustice, "Ford campus vision and lidar data set." International Journal of Robotics Research, 30(13), pp. 1543-1552, November 2011.

[2] G. Vosselman, B. Gorte, G. Sithole, T. Rabbani. "Recognizing structure in laser scanner point clouds.", The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences (IAPRS), 2004, vol. 36, pp. 33–38.

[3] R. Schnabel, R. Wahl, R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," Computer Graphics Forum, 2007, vol. 26, no.2, pp. 214-226.

[4] R. Kalman. "A new approach to linear filtering and prediction problems.", Transactions of the ASME-Journal of Basic Engineering, 1960, vol. 82, no. 1, pp:35–45.

[5] J. Lee, M. Kim, I. Kweon. "A kalman filter based visual tracking algorithm for an object moving," In IEEE/RSJ Intelligent Robots and Systems, 1995, pp. 342–347.

[6] 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.

[7] H. Boulaassal, T. Landes, P. Grussenmeyer, F. Tarsha- Kurdi. "Automatic segmentation of building facades using terrestrial laser data", The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences (IAPRS), 2007, vol. 36, no. 3.

[8] O. Gal, Z. Shiller, E. Rimon, "Efficient and safe on-line motion planning in dynamic environment," in Proceedings of the IEEE International Conference on Robotics and Automation, 2009, pp. 88–93.

[9] Velodyne 2007: Velodyne HDL-64E: A high definition LIDAR sensor for 3D applications. Available at: http://www.velodyne.com/lidar/products/white paper. [Accessed 1/23/2017].

 ISBN: 978-1-61208-539-5