VISIBLE ROUTES IN 3D DENSE CITY USING REINFORCEMENT LEARNING

O. Gal*[1], Y. Doytsher[1]

[1] Mapping and Geo-information Engineering, Technion, Israel - orengal@technion.ac.il

KEY WORDS: 3D GIS, Visibility, Routes, Reinforcement learning

ABSTRACT:

In the last few years, the 3D GIS domain has developed rapidly, and has become increasingly accessible to different disciplines. 3D Spatial analysis of Built-up areas seems to be one of the most challenging topics in the communities currently dealing with spatial data. One of the most basic problems in spatial analysis is related to visibility computation in such an environment. Visibility calculation methods aim to identify the parts visible from a single point, or multiple points, of objects in the environment.
In this work, we present a unique method combining visibility analysis in 3D environments with dynamic motion planning algorithm, named Visibility Velocity Obstacles (VVO) with Markov process defined as spatial visibility analysis for routes in 3D dense city environment.
Based on our VVO analysis, we use Reinforcement Learning (RL) method in order to find an optimal action policy in dense 3D city environment described as Markov decision process, navigating in the most visible routes.
As far as we know, we present for the first time a Reinforcement Learning (RL) solution to the visibility analysis in 3D dense environment problem, generating a sequence of viewpoints that allows an optimal visibility in different routes in urban city. Our analysis is based on fast and unique solution for visibility boundaries, formulating the problem with RL methods.

1. INTRODUCTION

Trajectory planning has developed alongside the increasing numbers of Unmanned Aerial Vehicles (UAVs) all over the world, with a wide range of applications such as surveillance, information gathering, suppression of enemy defenses, air to air combat, mapping buildings and facilities, etc.
Most of these applications are involved in very complicated environments (e.g. urban), with complex terrain for civil and military domains (Office Report, 2002). With these growing needs, several basic capabilities must be achieved. One of these capabilities is the need to avoid obstacles such as buildings or other moving objects, while autonomously navigating in 3D urban environments.
Path planning problems have been extensively studied in the robotics community. These problems include finding a collision-free path in static or dynamic environments, i.e., environments having moving or static obstacles. Over the past twenty years, many kinds of path planning methods have been proposed, such as starting roadmap, cell decomposition, and potential field (Latombe, 1990).
In this paper, we present a unique method combining visibility analysis in 3D environments with dynamic motion planning algorithm, named Visibility Velocity Obstacles (VVO) with Markov process defined as spatial visibility analysis for routes in 3D dense city environment.
Our method is based on two major steps. The first step is based on analytic visibility boundaries calculation in 3D environments, taking into account sensors' capabilities including probabilistic consideration. In the second step, we generate VVO transferring visibility boundaries from the position space to the velocity space, for each object. Each VVO represents velocity's set of possible future collision and visibility boundaries.
Based on our VVO analysis, we use Reinforcement Learning (RL) method in order to find an optimal action policy in dense 3D city environment described as Markov decision process, navigating in the most visible routes.
As far as we know, we present for the first time a Reinforcement Learning (RL) solution to the visibility analysis in 3D dense environment problem, generating a sequence of viewpoints that allows an optimal visibility in different routes in urban city. Our analysis is based on fast and unique solution for visibility boundaries, formulating the problem with RL methods.

2. RELATED WORK

Path planning becomes trajectory planning when a time dimension is added for dynamic obstacles (Erdmann and Lozano-Perez, 1987). Later on, a vehicle's dynamic and kinematic constraints have been taken into account, in a process called kinodynamic planning (LaValle and Kuffner, 1999). All of these methods focus solely on obstacle avoidance.
Trajectory planning for air traffic control and ground vehicles has been well studied (Mao et al. 2001), based on short path algorithms using 2D polygons, 3D surfaces (Bellingham et al. 2002). UAVs navigation has also been explored with vision-based methods (Sinopoli et al. 2001), with local planning or a predefined global path (Sasiadek and Duleba, 2000).
UAV path planning is different from simple robot path planning, due to the fact that a UAV cannot stop, and must maintain its velocity above the minimum, as well as not being able to make sharp turns.
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 (Elber et al. 2005).. 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

* Corresponding author

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). Other fast algorithms are based on the conservative Potentially Visible Set (PVS) (Durand, 1999). These methods are not always completely accurate, as they may render hidden objects' parts as visible due to various simplifications and heuristics.

3. VISIBILITY BOUNDARIES ANALYSIS

We extend our previous work (Gal and Doytsher, 2012) developed for a fast and efficient visibility analysis for buildings in urban environments, and consider also a basic structure of cylinders, which allows us to model pedestrians and trees. Based on our probabilistic visibility computation of dynamic objects, we test the effect of these by using data gathered from web-oriented GIS sources to update our estimation and prediction on these entities.

3.1 Dynamic Objects - Modeling and Probabilistic Visibility

Dynamic objects such as moving cars and pedestrians, directly affect visibility in urban environments.
Due to modeling limitations, these entities are usually neglected in spatial analysis aspects. We focus on three major dynamic objects in an urban case: moving cars and pedestrians. Each object is modeled with 3D boxes or 3D cylinders, which allow us to extend the use of our previous visibility analysis in urban environments presented for static objects (Gal and Doytsher, 2012).

3.2 Moving Car

3.2.1 3D Modeling: As we mentioned earlier, web-cameras in urban environments can record the moving cars at any specific time. Image sources such as web cameras, like other similar sensors sources, demand an additional stage of Automatic Target Detection (ATD) algorithms to extract these objects from the image (Song, 2010). In this research we do not focus on ATD, which must be implemented when shifting from the research described in the paper toward an applicable system.
The common car structure can be easily modeled by two 3D boxes, as can be seen in Figure 1, which is similar to the original car structure presented in Figure 1.
We define the Car Boundary Points (CBP) as the set of visible surfaces' boundary points of 3D boxes modeling the car presented in Figure 1. Each box is modeled as 3D cubic $C_{car}(x, y, z)$ as presented extensively in (Gal and Doytsher, 2012) for a building model case.

3.2.2 Car Boundary Points (CBP) - we define CBP of the object i as a set of boundary points $j = 1..N_{CBP\_bound}$ of the visible surfaces of the car object, from viewpoint $V(x_0, y_0, z_0)$, where the maximum surface's number is six and each surface defined by four points, $N_{CBP\_bound} \leq 24$, described in (1).
In Figure 2, car is modelled by using two 3D boxes. Visible surfaces colored in red, CBP marked with yellow points.

$$CBP_{i=1..N_{CBP\_bound}}(x_0, y_0, z_0) = \begin{bmatrix} x_1, y_1, z_1 \\ x_2, y_2, z_2 \\ .. \\ x_{N_{CBP\_bound}}, y_{N_{CBP\_bound}}, z_{N_{CBP\_bound}} \end{bmatrix} \qquad (1)$$

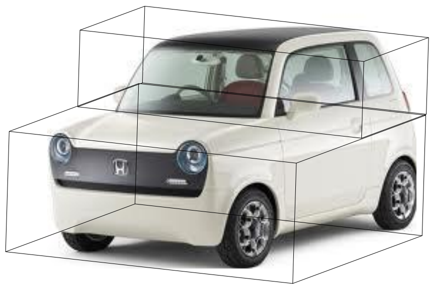
Figure 1. Car Modeling Using 3D Boxes

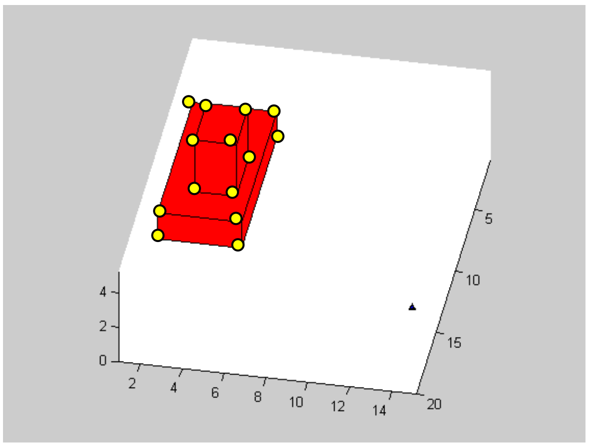

Figure 2. Modeling Car Using 3D Boxes (CBP Marked with Yellow Points)

3.2.3 Probabilistic Visibility Analysis: Visibility has been treated as Boolean values. Due to incomplete information and the uncertainties of predicting the car's location at future times, visibility becomes much more complicated. As it is well known from basic kinematics, CBP can be estimated in future time $t + \Delta t$ as shown in (2):

$$CBP_i(t + \Delta t) = CBP_i(t) + V(t)\Delta t + \frac{A(t)\Delta t^2}{2} \qquad (2)$$

Where $V(t)$ is the car velocity vector $V(t) = (v_x v_y)^T$, and the acceleration vector $A(t) = (a_x a_y)^T$. Estimation of a car's location in the future based on a web camera is not a simple task. Driver behaviour generates multi-decision modelling, such as car-following behaviour, gap acceptance behaviour, or lane-change cases including traffic flow, speed etc. (Archer, 2010).
Our probabilistic car model is based on microscopic simulation models that were properly calibrated and validated using VISSIM simulation. The average speed in urban environments is about 45 [km/hr], from a minimum of 40 [km/hr] up to a maximum of 50 [km/hr]. In the situation of a free driving case, which is the common mode in urban environments (Wiedemann, 1992), the acceleration of family car can change between $1 \text{ to } 3.5 \left[{}^{m}/_{sec^2}\right]$, and on average $2.5 \left[{}^{m}/_{sec^2}\right]$.
As can be seen from several validations of car and driver estimation, velocity and acceleration are distributed as normal ones, and lead to normal location distribution in (3):

$$\begin{gathered} V(t) \sim N(\mu = 45, \sigma^2 = 10) \\ A(t) \sim N(\mu = 2.5, \sigma^2 = 1) \\ CBP(t + \Delta t) \sim \sum N \end{gathered} \tag{3}$$

In time step t, where the car's location is taken from a web-camera, visibility analysis from $CBP(t)$ is an exact one, based on our previous visibility analysis (Gal and Doytsher, 2012), as seen in Figure 2. Visibility analysis becomes probabilistic for future time $t + \Delta t$, applying the same visibility analysis for $CBP(t + \Delta t)$ presented in Figure 3.

In Figure 3, the car's location from a web-camera appears in the bottom left side. For $\Delta t = 2[sec]$, the car's location is marked by two 3D boxes, where CBP for each of them is the boundary of visible surfaces marked in red. The probability that the visible surfaces, which are bounded by CBP, will be visible in future time is based on the last update taken from the web application (depicted with arrows in Figure 3), computed by using two different random normal PDF values for V and A.

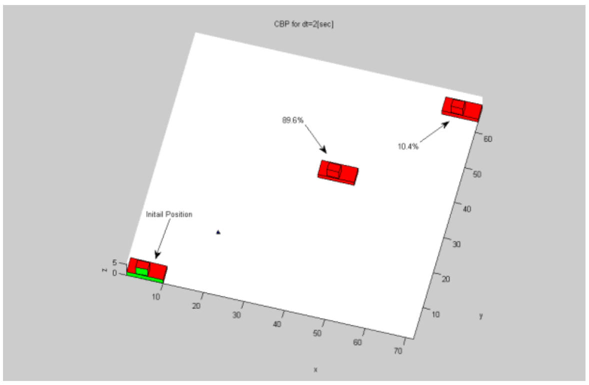

Figure 3. Probabilistic Visibility Analysis for CBP

3.3 Pedestrians

3.3.1 3D Modeling: Pedestrian modeling can be done in high resolution, but due to ATD algorithms capabilities, pedestrians are usually bounded by a 3D cylinder and not as an exact detailed model (Song, 2010). For this reason, we model pedestrians as 3D cylinders, which is somewhat conservative but still applicable.
Pedestrian can be easily modeled by 3D cylinder, as seen in Figure 4 (marked in red), which is similar to the output from ATD methods tested on a web-camera output recognizing walkers in urban environments.
We extend our previous visibility analysis concept (Gal and Doytsher, 2012) and include new objects modeled as cylinders as continuous curves parameterization, $C_{Peds}(x, y, z)$in (4). Cylinder parameterization can be described as:

$$\begin{gathered} C_{Peds}(x,y,z) = \begin{pmatrix} r\sin(\theta) \\ r\cos(\theta) \\ c \end{pmatrix} \\ 0 \le \theta \le 2\pi \\ c = c + 1 \\ 0 \le c \le h_{peds\_max} \end{gathered} \tag{4}$$

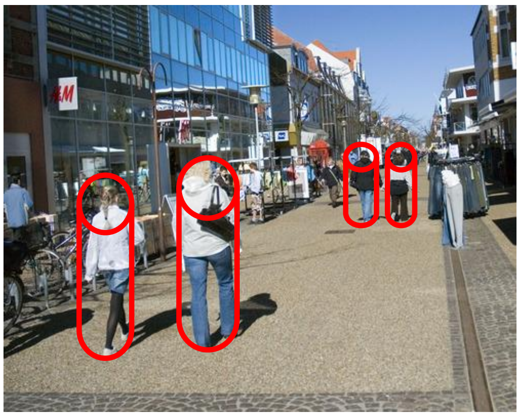

Figure 4. Modeling Pedestrians in Urban Scene Using Cylinders (Colored in Red)

We define the visibility problem in a 3D environment for more complex objects in (5):

$$C'(x,y)_{z_{const}} \times \left(C(x,y)_{z_{const}} - V(x_0, y_0, z_0)\right) = 0 \tag{5}$$

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 cylinder case. 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.
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 in (6). 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.

$$\theta = \arctan\left(-\frac{-r - \frac{\left(-vy\, r + \sqrt{vx^4 - vx^2 r^2 + vy^2 vx^2}\right) vy}{vx^2 + vy^2}}{vx}, -\frac{-vy\, r + \sqrt{vx^4 - vx^2 r^2 + vy^2 vx^2}}{vx^2 + vy^2} \right) \tag{6}$$

We define the solution presented above as x-y-z coordinates values for the cylinder case as **Pedestrian Boundary Points** (PBP). PBP are the set of visible silhouette points for a 3D cylinder modeling the pedestrian in (7):

$$PBP_{i=1..N_{PBP\_bound}=2}(x_0, y_0, z_0) = \begin{bmatrix} x_1, y_1, z_1 \\ x_{N_{PBP\_bound}}, y_{N_{PBP\_bound}}, z_{N_{PBP\_bound}} \end{bmatrix} \tag{7}$$

4. VISIBILITY VELOCITY OBSTACLES (VVO)

The visibility velocity obstacle represents the set of all velocities from a viewpoint, occluded with other objects in the environment. It essentially maps static and moving objects into the robot’s velocity space considering visibility boundaries.
The VVO of an object with circular visibility boundary points such as the pedestrians case, PBP, that is moving at a constant velocity v_b, is a cone in the velocity space at point A. In Figure 5, the position space and velocity space of A are overlaid to illustrate the relationship between the two spaces. The VVO is generated by first constructing the Relative Velocity Cone (RVC) from A to the boundaries of the object, i.e., PBP, then translating RVC by v_b.
Each point in VVO represents a velocity vector that originates at A. Any velocity of A that penetrates VVO is a occluded velocity that based on the current situation, would result in a occlusion between A and the pedestrian at some future time. Figure 5 shows two velocities of A: one that penetrates VVO, hence a occluded velocity, and one that does not. All velocities of A that are outside of VVO are visible from the current robot's position as the obstacle denotes as B, stays on its current course. The visibility velocity obstacle thus allows determining if a given velocity is occluded, and suggesting possible changes to this velocity for better visibility. If PBP is known to move along a curved trajectory or at varying speeds, it would be best represented by the nonlinear visibility velocity obstacle case discussed next.

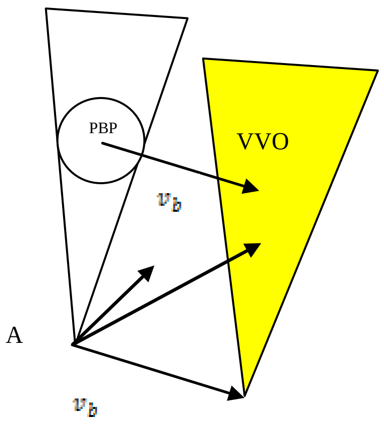

Figure 5. Visibility Velocity Obstacles

The VVO consists of all velocities of A at t_0 predicting visibility's boundaries related to obstacles at the environment at any time t>t_0. Selecting a single velocity, v_a, at time t = t_0 outside the VVO, guarantees visibility to this specific obstacle at time t. It is constructed as a union of its temporal elements, VVO(t), which is the set of all absolute velocities of A, v_a, that would allow visibility at a specific time t.
Referring to Figure 6, v_a that would result in occlusion with point p in B at time t > t_0, expressed in a frame centered at A(t_0), is simply in (8):

$$v_a = \frac{VBP_1}{t - t_0} \tag{8}$$

where r is the vector to point p in the blocker’s fixed frame, and visibility boundaries denoted as Visibility Boundary Points (VBP). The set, VVO(t) of all absolute velocities of A that would result in occlusion with any point in B at time t > t_0 is thus in (9):

$$VVO(t) = \frac{VBP_1(t)}{t - t_0} \tag{9}$$

Clearly, VVO(t) is a scaled B for two dimensional case with circular object, located at a distance from A that is inversely proportional to time t. The entire VVO is the union of its temporal subsets from t_0, the current time, to some set future time horizon t_h in (10):

$$VVO(t) = \bigcup_{t=t_0}^{t_h} \frac{VBP_1(t)}{t - t_0} \tag{10}$$

The presented VVO generate a warped cone in a case of 2D circular object. If VBP(t) is bounded over t = (t_0, ∞), then the apex of this cone is at A(t_0).We extend our analysis to 3D general case, where the objects can be cubes, cylinders and circles. The mathematical analysis with visibility boundaries is based on VBP presented in the previous part for different kind of objects such as buildings, cars and pedestrians.
We transform the visibility's boundaries into the velocity space, by moving the VBP to the velocity space, in the same analysis presented for 2D circle boundary's.
Following that, we present 3D extension for VBP case, transformed to the velocity space.
Given two objects, VBP_1, VBP_2 will create a VVO representing VBP_2 (and vice-versa) such that VBP_1 wishes to choose a guaranteed collision-free velocity for the time interval τ, and visibility boundary in velocity space.
In case of cars, buildings and pedestrians where visibility boundaries can be expressed by geometric operations of 3D boxes, analyzed in the same concept and formulation presented so far, as can be seen in Figure 6.

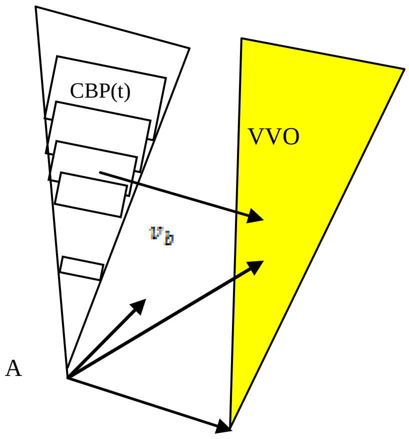

Figure 6. Visibility Velocity Obstacle for visibility boundaries consist of 3D boxes

5. REINFORCEMNET LEARNING USING VVO

In most 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. We use well-known simple and efficient greedy exploration method maximizing Q-value**.**

In our case, 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, where the best next step is not colliding with any VVO in the environment. Based on that, TD and SARSA methods for RL can be used, generating visible trajectory in 3D urban environment.

5.1 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.

5.2 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.

5.2.1 Q-Learning

A quintessential TD off-policy control algorithm is the one-step Q-learning. Q-learning is considered an off-policy algorithm since we approximate the optimal Q-value function, Q∗, independently of the current policy. The update step is as follows:

$$Q : S \times A \rightarrow R \tag{11}$$

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma\, max_a\, Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)) \tag{12}$$

The α is the learning rate (0 < α < 1) and determines the rate at which new information will override the old Q-value. γ is the discount factor 0 < α < 1) which decreases the estimated Q-values for future states. The TD-learning algorithm is presented in Figure 7.

```
procedure Q-LEARNING
    Initialize $Q(s,a) = 0$ for all $a \in A$ and $s \in S$
    Repeat until the end of the episode:
    $s_t \leftarrow InitialState$
    for each episode step do
        Select $a_t$, based on a exploration strategy from $s_t$
        Take action $a_t$, observe $r_{t+1}$, $s_{t+1}$
        $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$
        $s_t \leftarrow s_{t+1}$
        if then $s == terminal$ then
            $Q(s_{t+1},a_{t+1}) = 0$
        end if
    end for
```

Figure 7. Q-learning Algorithm

5.2.2 SARSA

SARSA is a On-Policy temporal-difference reinforcement learning method to estimate a quality function. It heritages the name from: (State, Action, Reward, Next State, Next Action). The update step is as follows:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)) \tag{13}$$

The Q value is updated through interactions with the environment, thus updating the policy depends on the action taken. The Q-value for a state-action pair is not directly updated, but gradually adjusted with a learning rate α. As with the Q-learning algorithm, SARSA will keep on improving its policy towards a better solution as long as all state-action pairs continue to be updated. The SARSA-learning algorithm is presented at algorithm 8.

```
procedure On-Policy
    Initialize $Q(s,a)$ arbitrarily for all $a \in A$ and $s \in S$
    Repeat until the end of the episode:
    $s_t \leftarrow InitialState$
    Select $a_t$ based on a exploration strategy from $s_t$
    for each episode step do
        Take action $a_t$, observe $r_{t+1}$, $s_{t+1}$
        Select $a_{t+1}$ based on a exploration strategy from $s_{t+1}$
        $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$
        $s_t \leftarrow s_{t+1}$
        $a_t \leftarrow a_{t+1}$
        if then $s == terminal$ then
            $Q(s_{t+1},a_{t+1}) = 0$
        end if
    end for
```

Figure 8. SARSA-learning Algorithm

CONCLUSIONS

In this paper, we presented a unique method combining visibility analysis in 3D environments with dynamic motion planning algorithm, named Visibility Velocity Obstacles (VVO) with Markov process defined as spatial visibility analysis for routes in 3D dense city environment. Our method is based on two steps. The first step is based on analytic visibility boundaries calculation in 3D environments, taking into account sensors' capabilities including probabilistic consideration. In the second step, we generate VVO transferring visibility boundaries from the position space to the velocity space, for each object. Each VVO represents velocity's set of possible future collision and visibility boundaries.
Based on our VVO analysis, we presented Reinforcement Learning (RL) formulation in order to find an optimal action policy in dense 3D city environment described as Markov decision process, navigating in the most visible routes.

REFERENCES

G. Elber, R. Sayegh, G. Barequet, and R. Martin, "Two-Dimensional Visibility Charts for Continuous Curves," in Proc. Shape Modeling, MIT, Boston, USA, pp. 206-215, (2005)

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)

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)

P. Fiorini, and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," Int. J. Robot. Res.17, pp. 760–772, (1998)

Office of the Secretary of Defense, Unmanned Aerial Vehicles Roadmap, Tech. rep., December (2002)

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

M. Erdmann, and T. Lozano-Perez, "On Multiple Moving Objects," Algorithmica, 2, 477–521, (1987)

T. Fraichard, "Trajectory Planning in a Dynamic Workspace: A 'State-Time Space' Approach," Advanced Robotics, 13:75–94. (1999)

S.M. LaValle, and J. Kuffner, Randomized Kinodynamic Planning. In Proc. IEEE Int. Conf. on Robotics and Automation, Detroit, MI , USA, pp: 473–479, (1999)

Z.H. Mao, E. Feron, and K. Bilimoria, "Stability and Performance of Intersecting Aircraft Flows Under Decentralized Conflict Avoidance Rules," IEEE Transactions on Intelligent Transportation Systems, 2: 101–109,(2001)

J. Bellingham, A. Richards, and J. How, "Receding Horizon Control of Autonomous Aerial Vehicles," in Proceedings of the IEEE American Control Conference, Anchorage, AK, USA, pp. 3741–3746, (2002)
B. Sinopoli, M. Micheli, G. Donata, and T. Koo, "Vision Based Navigation for an Unmanned Aerial Vehicle," in Proc. IEEE Int'l Conf. on Robotics and Automation, (2001)

J. Sasiadek, and I. Duleba, "3D Local Trajectory Planner for UAV," Journal of Intelligent and Robotic Systems, 29: 191–210, (2000)

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

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

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)

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

Y. Song: "The research of a new Auto Target Recognition directed Image compression," in 3th Int. Congress on Image and Signal Processing (CISP), 16-18 Oct, 2010, China.

J. Archer: "Methods for the Assessment and Prediction of Traffic Safety at Urban Intersections and their Application in Micro-simulation Modeling," Centre for Traffic Simulation Research, CTR, Sweden. Technical Report, (2010).

R. Wiedemann, and U. Reiter, "Microscopic Traffic Simulation: The Simulation System MISSION, Background and Actual State," Project ICARUS (V1052), Final Report, Brussels CEC.2: Appendix A, (1992).

O. Gal, Y.Doytsher. "Patrolling Strategy Using Heterogeneous Multi Agents in Urban Environments Using Visibility Clustering," Journal of Unmanned System Technology, ISSN 2287-7320, 2016.