Robotica (2014) volume 32, pp. 143–163. © Cambridge University Press 2013
doi:10.1017/S0263574713000787

Fast and efficient visible trajectories planning for the Dubins UAV model in 3D built-up environments

Oren Gal* and Yerach Doytsher

Mapping and Geo-Information Engineering, Technion, Israel Institute of Technology, Haifa, Israel

(Accepted June 26, 2013. First published online: August 2, 2013)

SUMMARY

In this paper, we study the visible trajectories planning for unmanned aerial vehicles (UAVs) modeled with a Dubins airplane in 3D urban environments. Our method is based on a fast and exact spatial visibility analysis of the 3D visibility problem from a viewpoint in 3D built-up environments. We consider the 3D urban environment buildings modeled as cubes (3D boxes) and present an analytic solution to the visibility problem. Based on an analytic solution, the algorithm computes the exact visible and hidden parts from a viewpoint in the urban environment. We present a local trajectory planner generating the most visible trajectory in a known 3D urban environment model, taking into account the dynamic and kinematic UAV constraints. The planner computes, at each time step, the next UAV's attainable velocities and explores the most visible node, while avoiding buildings as static obstacles in the environments, using the velocity obstacle method. The visibility type of the trajectory can be configured beforehand as visible roofs and surfaces in the environments. We demonstrate our visibility and trajectory planning method in simulations in several 3D urban environments, showing visible trajectory planning capabilities.

KEYWORDS: Trajectory planning; 3D Urban environments; Spatial analysis; Visibility.

1. Introduction

Autonomous aircrafts and unmanned aerial vehicles (UAVs) are increasingly used for different applications such as mobile surveillance, environmental monitoring, search and rescue, and Homeland Security (HLS).

HLS has seen a rapid development since 9/11, and nowadays this is one of the major issues that are being researched. One of the relevant aspects for HLS applications concerns special forces action in urban environments, dealing with hostile activities. Planning visible trajectories for UAVs in 3D complex urban environments can be used to aid special forces in preventing terrorist actions by planning a visible or hidden trajectory in 3D urban environments. In such cases of trajectory planning, spatial analysis of a 3D urban environment should be carried out in a very short period of computation time for planning the most undetectable trajectory.

In this paper, we introduce visible trajectories planning for UAVs in 3D urban environments, based on a *fast and exact* solution to the 3D visibility problem, from a viewpoint in the urban environment. We consider 3D urban environment buildings modeled as cubes (3D boxes) and present an analytic solution to the visibility problem, a solution that does not suffer from approximations. The algorithm computes the exact visible and hidden parts from a viewpoint in the urban environment, using an analytic solution, without the expensive computational process of scanning all objects' points. The algorithm is demonstrated by a schematic structure of an urban environment, using simple topological geometric operators.

Our method uses simple geometric relations between the objects and the lines connecting the viewpoint and the objects' boundaries, extending the visibility boundary calculation from 2D to a 3D environment by bypassing these singular points.[1] The spatial relationship between different objects is

* Corresponding author. E-mail: orengal@tx.technion.ac.il

computed by using fast visible pyramid (VP) volumes from the viewpoint, projected to the occluded buildings.[2,3]

We present a local UAV trajectory planner that generates one step ahead every time step, which takes into account kinematic and dynamic constraints modeling UAV trajectories with the Dubins airplane model for time-optimal trajectories. For the first time, this planner generates fast and exact visible trajectories, based on the analytic solution described above. We assume knowledge of the 3D urban environment model, and avoid collision with buildings presented as static obstacles by using velocity obstacles (VOs).[4]

In the following, we first introduce relevant work related to trajectory planning and visibility in 3D urban environments, in Section 3. In Section 4, the UAV model is presented. We present 3D spatial analysis focused on visibility from viewpoint in 3D urban environments in Section 5. In Section 6, we present the UAV local trajectory planner. Simulations are presented in Section 7. Finally concluding comments are given in Section 8.

2. Related Work

In this section we include previous work in the area of UAV trajectory planning in 3D environments, as well as relevant work on visibility analysis in 3D urban environments.

2.1. UAV trajectory planning in 3D environments

Trajectory planning has developed alongside the increasing numbers of 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.[5]

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 extensively been studied in the robotics community, finding a collision-free path in static or dynamic environments, i.e., moving or static obstacles. Over the past 20 years, many methods have been proposed, such as starting roadmap, cell decomposition, and potential field.[6]

Path planning becomes trajectory planning when a time dimension is added for dynamic obstacles.[7,8] Later on, a vehicle's dynamic and kinematic constraints have been taken into account, in a process called kinodynamic planning.[9] All of these methods focus solely on obstacle avoidance.

Trajectory planning for air traffic control and ground vehicles has been well studied,[10] based on short-path algorithms using 2D polygons, 3D surfaces.[11] UAVs navigation has also been explored with vision-based methods,[12] with local planning or a predefined global path.[13]

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.

UAV path planning methods usually decompose the path planning into two steps: first, using some common path planning method in a polygonal environment,[6] then, considering UAV dynamic and kinematic constraints into the trajectory.[14] These methods assume decoupling which affects the trajectory, as stated by all authors.

However, most of the effort focused on UAV trajectory planning is related to obstacle avoidance with kinodynamic constraints, without taking into account visibility analysis as part of the nature of the trajectory in urban environments.

2.2. Visibility in 3D urban environments

The visibility problem has extensively been studied over the last 20 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.[15] 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

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

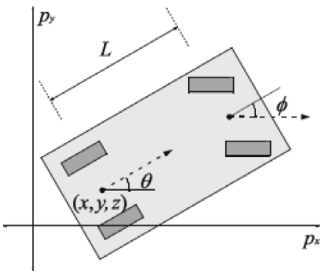


Fig. 1. Dubins airplane model in the *X–Y* plane (Source ref. [24]).

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.[16] Other fast algorithms are based on the conservative potentially visible set.[17] These methods are not always completely accurate, as they may render hidden objects' parts as visible due to various simplifications and heuristics.

A vast number of algorithms have been suggested for speeding up the process and reducing computation time. Franklin[18] evaluates and approximates visibility for each cell in a DEM model based on greedy algorithms. Wang *et al.*[19] introduced a grid-based DEM method using viewshed horizon, saving computation time based on relations between surfaces and the LOS method. Later on, an extended method for viewshed computation was presented, using reference planes rather than sightlines.[20]

One of the most efficient methods for DEM visibility computation is based on shadow-casting routine. The routine cast shadowed volumes in the DEM, like a light bubble.[21] Extensive research treated digital terrain models in open terrains, mainly triangulated irregular network and regular square grid structures. Visibility analysis in terrain was classified into point, line and region visibility, and several algorithms were introduced, based on horizon computation describing visibility boundary.[22]

Only a few works have treated visibility analysis in urban environments. A mathematical model of an urban scene, calculating probabilistic visibility for a given object from a specific viewcell in the scene, has been presented in ref. [23]. This is a very interesting concept, which extends the traditional deterministic visibility concept. Nevertheless, the buildings are modeled as cylinders, and the main challenges of spatial analysis and building model were not tackled. Other methods were developed, subject to computer graphics and vision fields, dealing with exact visibility in 3D scenes, without considering environmental constraints. Plantinga and Dyer[15] used the aspect graph – a graph with all the different views of an object. Due to their computational complexity, all of these works are not applicable to a large scene with near real-time demands, such as UAV trajectory planning.

3. The UAV Model

We introduce a UAVs model, based on the well-known Dubins airplane.[24] Nowadays, several fixed-wing UAVs fit to the Dubins airplane model,[25] such as *AscTec Hummingbird* platform,[26] and many other autopilot controllers can be designed for such trajectories.[27] The Dubins airplane[28] model extends the Dubins car model with continuous change of altitude without reverse gear, avoiding sudden altitude speed rate variation. Our UAV model includes kinematic and dynamic constraints that ignore pitch and roll rotation or wind disturbances, and allows setting time-optimal trajectory trims.[28]

3.1. Kinematic constraints

We use a simple UAV model with four dimensions, each configuration is $q = (x, y, z, \theta)$, when x, y, z are the coordinates of the origin, and θ is the orientation, in the x–y plane relative to the x-axis, as can be seen in Fig. 1 for a simple car-like model.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

The steering angle is denoted as ϕ. The distance between front and rear axles is equal to 1. The kinematic equations of a simple UAV model can be written as

$$\begin{aligned} \dot{x} &= u_s \cos\theta, \\ \dot{y} &= u_s \sin\theta, \\ \dot{z} &= u_{z,} \\ \dot{\theta} &= u_s \tan u_\phi, \end{aligned} \tag{1}$$

where $u_s$is the speed parallel to the x–y plane, climb rate (speed parallel to the z-axis) is u_z, and the control on steering angle u_ϕ. We denote the control vector as $u = (u_s, u_z, u_\phi)$. Each of the controllers is bounded, $u_\phi \in [-\phi^{\max}, \phi^{\max}]$, where $\phi^{\max} < \pi/2$, the speed $u_s \in [u_s^{\min}, u_s^{\max}]$ and climb rate $u_z \in [-u_z^{\max}, u_z^{\max}]$. $u_s^{\min} > 0$, so UAV cannot stop.

3.2. Dynamic constraints

The UAV model has to take into account the dynamic constraints, preventing instantaneous changes (increase or decrease) of the control vector $u = (u_s, u_z, u_\phi)$.

The UAV model also includes dynamic constraints,$\dot{u}_s \in [-a_s, a_s]$,$\dot{u}_z \in [-a_z, a_z]$ and $\dot{u}_\phi \in [-a_\phi, a_\phi]$.

4. 3D Spatial Analysis

In this section, we introduce a fast and efficient accumulated visibility computation from a viewpoint in 3D urban environments for UAV visible trajectory planning. Spatial analysis in a 3D environment appears 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 visible parts from a single point, or multiple points, of the objects in the environment.

The 3D visibility problem has extensively been studied over the past 20 years, due to the importance of visibility in many disciplines such as GIS, computer graphics, computer vision, and robotics. Most previous works approximate the visible parts to find a fast solution in open terrains, and do not challenge or suggest solutions for a 3D dense urban environment. The exact visibility methods are highly complex, and cannot be used for fast applications due to their long computation time.

In this section, we present an analytic solution to the visibility problem in a dense urban environment. At each viewpoint, an exact visibility value is calculated, i.e., the sum of the visible surfaces from viewpoint to buildings. Based on the calculated visible and hidden surfaces from the viewpoint, we can accumulate the viewpoint visibility value. The viewpoint visibility value can also indicate the viewpoint's hidden level, since hidden surfaces are also computed.

4.1. Spatial relations – problem statement

We consider the basic visibility problem in a 3D urban environment, consisting of 3D buildings modeled as 3D cubic (box) parameterization$\sum_{i=1}^{N} C_i(x, y, z =_{h_{\min}}^{h_{\max}})$, and viewpoint $V(x_0, y_0, z_0)$.

Given:

- A viewpoint $V(x_0, y_0, z_0)$in 3D coordinates.
- Parameterizations of N objects $\sum_{i=1}^{N} C_i(x, y, z =_{h_{\min}}^{h_{\max}})$ describing a 3D urban environment model.

Computes:

- Set of all visible points in $\sum_{i=1}^{N} C_i(x, y, z =_{h_{\min}}^{h_{\max}})$ from $V(x_0, y_0, z_0)$.

This problem seems to have been solved by conventional geometric methods, but as mentioned before, it demands a long computation time. We introduce a fast and efficient computation solution for a schematic structure of an urban environment that demonstrates our method.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

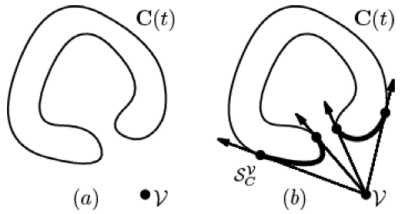


Fig. 2. Visible silhouette points S_C^v from viewpoint V to curve $C(t)$.

4.2. Solution for a single-object concept

In this section, we first introduce the visibility solution from a single point to a single 3D object. This solution is based on an analytic expression, which significantly improves time computation by generating the visibility boundary of the object without the need to scan the entire object's points.

Our analytic solution for a 3D building model is an extension of the visibility chart in 2D introduced by[1] for continuous curves. For such a curve, the silhouette points, i.e., the visibility boundary of the object, can be seen in Fig. 2.

The visibility chart solution was originally developed for dealing with the Art Gallery Problem for infinite viewpoint; it is limited to 2D continuous curves using a multivariate solver,[1] and cannot be used for online application in a 3D environment.

Based on this concept, we define the visibility problem in a 3D environment for more complex objects as

$$C'(x, y)_{z_{const}} \times (C(x, y)_{z_{const}} - V(x_0, y_0, z_0)) = 0, \tag{2}$$

where 3D model parameterization is $C(x, y)_{z_{const}}$, and the viewpoint is given as $V(x_0, y_0, z_0)$. Solutions to Eq. (2) generate a visibility boundary from the viewpoint to an object, based on basic relations between viewing directions from V to $C(x, y)_{z_{const}}$ using cross-product characters.

A three-dimensional urban environment consists mainly of rectangular buildings, which can hardly be modeled as continuous curves. Moreover, an analytic solution for a single 3D model becomes more complicated due to the higher dimension of the problem, and is not always possible. Object parameterization is therefore a critical issue, allowing us to find an analytic solution and, using that, to very quickly generate the visibility boundary.

4.3. 3D urban environmental model

In this section, we introduce a 3D urban environment model for exact and fast visibility computation at each trajectory point. Our 3D model can be easily generated from available data, such as Google Map.[3] We assume that the model is uploaded to the UAV computer offline before starting mission.

Most of the common 3D City Models are based on object-oriented topologies, such as 3D formal data structure (3D FDS), simplified spatial model (SSS), and urban data model (UDM).[29] These models are very efficient for web-oriented applications. However, the fact that a building consists of several different basic features makes it almost impossible to generate analytic representation. A three-dimensional building model should be, on the one hand, simple, enabling analytic solution, and on the other hand, as accurate as possible. We examined several building object parameterizations, and the preferred candidate was an extended n-order sphere coordinates parameterization, even though such a model is a very complex, and will necessitate a special analytic solution.

We introduce a model that can be used for analytic solution of the current problem, which is the key to generate a fast and exact solution to the hard computational demands of visibility problem in 3D environments. The basic building model can be described as

$$\begin{aligned} &x = t, \quad y = \begin{pmatrix} x^n - 1 \\ 1 - x^n \end{pmatrix}, \quad z = c, \quad -1 \le t \le 1 \\ &n = 350, \\ &c = c + 1. \end{aligned} \tag{3}$$

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

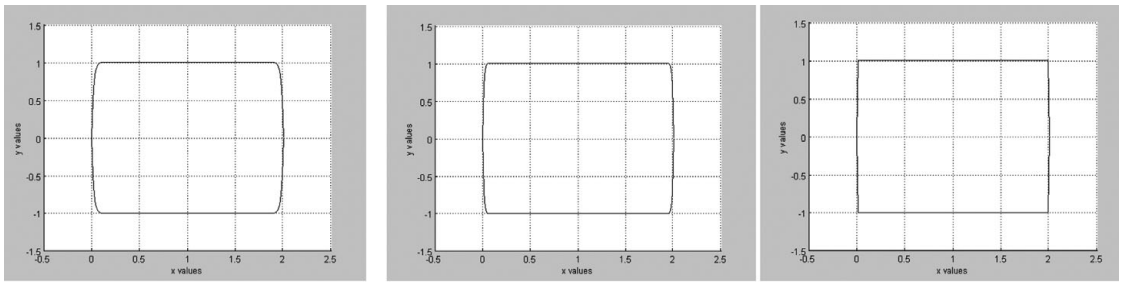


Fig. 3. Topside view of the building model using Eq. (3). (a) $n = 50$; (b) $n = 200$; (c) $n = 350$.

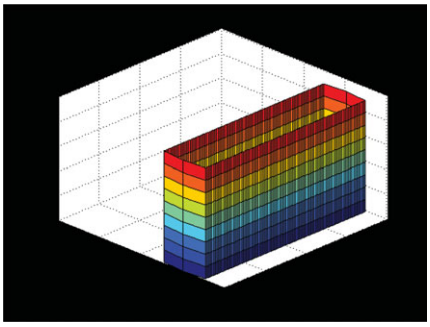


Fig. 4. (Colour online) A three-dimensional analytic building model with Eq. (3), where $z_{h_{\min}=0}^{h_{\max}=9}$.

This mathematical model approximates building corners, not as singular points, but as continuous curves. This building model is described by Eq. (3), when lower order badly approximates the building corners, as depicted in Fig. 3. Corner approximation becomes more accurate using $n = 350$ or higher. This approximation enables us to define an analytic solution to the problem.

We introduce a basic building structure that can be rotated and extracted using simple matrix operators (Fig. 4). Using a rotation matrix does not affect our visibility algorithm, and for a simple demonstration of our method, we present samples of parallel buildings.

4.4. The analytic solution

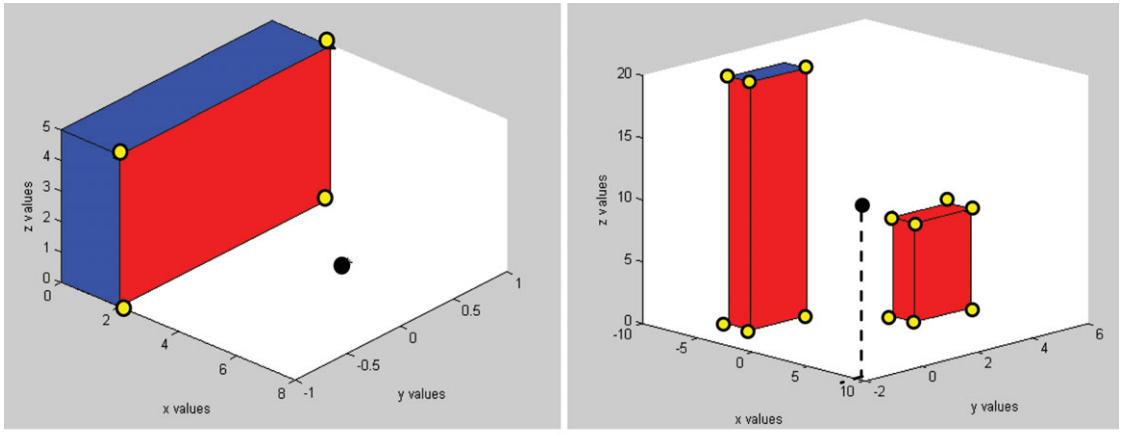
In this part, we demonstrate the analytic solution for a single 3D building model. As mentioned above, we should integrate building model parameterization to the visibility statement. After integrating Eqs. (2) and (3):

$$\begin{aligned} &C'(x, y)_{z_{const}} \times (C(x, y)_{z_{const}} - V(x_0, y_0, z_0)) = 0 \rightarrow \\ &x^n - V_{y_0} - n \cdot x^{n-1}(x - V_{x_0}) - 1 = 0, \\ &x^n + V_{y_0} - n \cdot x^{n-1}(x - V_{x_0}) - 1 = 0. \\ &n = 350 \quad -1 \leq x \leq 1 \end{aligned} \tag{4}$$

Based on the geometric relation of cross product, object parameterization, and viewpoint described in Eq. (2), visible silhouette can be generated for continuous curves, and extended for building's object in 3D in this research. Due to that, visibility boundary is the solution for these coupled equations described in Eq. (4).

As can be noticed, these equations are not related to the Z-axis, and the visibility boundary points are the same ones for each x–y surface due to the model's characteristics. Later on, we treat the relations between a building's roof and visibility height in our visibility algorithm, as part of the visibility computation.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press



(a) (b)

Fig. 5. (Colour online) Visibility volume computed with the analytic solution. (a) Single building. (b) Two nonoverlapping buildings.

The visibility statement leads to two polynomial Norder equations, which appear to be a complex computational task. The real roots of these polynomial equations are the solution to the visibility boundary. These equations can be solved efficiently by finding where the polynomial equation changes its sign and cross zero value; generating the real roots in a very short time computation (these functions are available in Matlab, Maple, and other mathematical programs languages). Based on the polynomial cross zero solution, we can compute a fast and exact analytic solution for the visibility problem from a viewpoint to a 3D building model. This solution allows us to easily define the visible boundary points.

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 $V(x_0, y_0, z_0)$.

$$VBP_{i=1}^{j=1..N_{bound}}(x_0, y_0, z_0) = \begin{bmatrix} x_1, y_1, z_1 \\ x_2, y_2, z_2 \\ .. \\ x_{N_{bound}}, y_{N_{bound}}, z_{N_{bound}} \end{bmatrix}. \tag{5}$$

Roof Visibility. The analytic solution in Eq. (4) does not treat the roof visibility of a building. 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:

$$V_{z_0} \geq Z = h_{\max_{C_i}}. \tag{6}$$

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. Currently, we assume a flat roof's surface that will be extended to more complex roof models in our future work. For simplicity, we define roof visible surface as the third one.

Two simple cases using the analytic solution from a visibility point to a building including visible roofs can be seen in Fig. 5. The visibility point is marked in black, the visible parts colored in red, and the invisible parts colored in blue. VBP are marked with yellow circles. The visible volumes are computed immediately with very low computation effort, without scanning all the model's points, as is necessary in LOS-based methods for such a case.

4.5. Visibility computation in urban environments

In the previous sections, 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 dealing

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

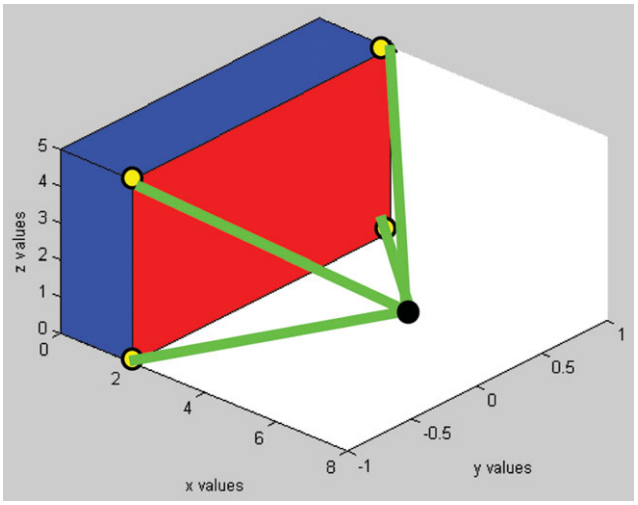


Fig. 6. (Colour online) A VP from a viewpoint to VBP of a specific surface.

with these spatial relations between buildings, based on our ability to rapidly compute visibility volume for a single building by generating VBP sets.

Hidden surfaces between buildings are simply computed by intersecting the VPs for each object. The visible volumes are defined easily using VBP, and are defined, in our case, as VPs. The invisible components of the far building are computed by intersecting the projection of the closer buildings' VP base to the far building's VP base, as described above.

4.5.1. The VP.

VP. We define $VP_i^{j=1..N_{surf}}(x_0, y_0, z_0)$ 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 viewpoint marked as black dot, as can be seen in Fig. 6.

More complex building models can be subdivided to basic box structures using procedural modeling and basic shape grammar profiles.[3]

The intersection of VPs allows us to efficiently compute the hidden surfaces in urban environments, as can be seen in the next subsection.

4.5.2. Hidden surfaces between buildings. As mentioned earlier, invisible parts of the farther buildings are computed by intersecting the projection of the closer buildings' VP to the far buildings' VP base.

Let VP_1^i and VP_2^j be VPs from a viewpoint $V(x_0, y_0, z_0)$. The projected surface $PS_{VP_1^i}^{VP_2^j}$ of the closer buildings' VP_1^i to the farther buildings' VP_2^j base plane consists of the projection of $VBP_1^{1..Nb}$ points:

$$PS_{VP_1^i}^{VP_2^j} = \begin{bmatrix} x_{p1}, & y_{p1}, & z_{p1} \\ x_{p2}, & y_{p2}, & z_{p2} \\ & .. & \\ x_{pi}, & y_{pi}, & z_{pi} \\ & .. & \\ x_{pN_{bound}}, & y_{pN_{bound}}, & z_{pN_{bound}} \end{bmatrix}. \tag{7}$$

where the normal of the $VP_2{}^j$ base plane is $(a.b.c.)$ and the plane can be written as $ax + by + cz + d = 0$. The projected point (x_{pi}, y_{pi}, z_{pi}) described in Eq. (7) is

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

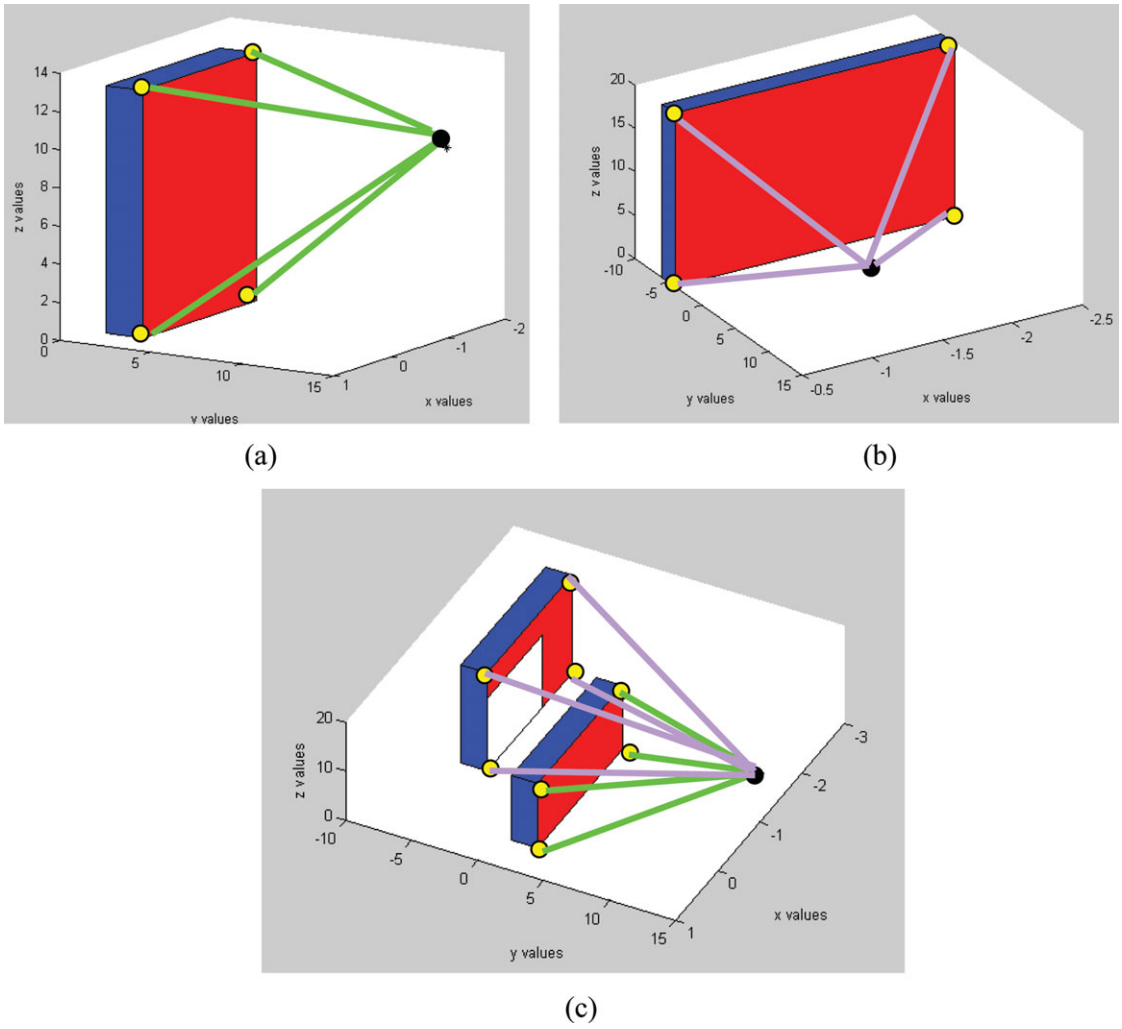


Fig. 7. (Colour online) Generating VP. (a) $VP_1{}^1$ boundary colored by green lines. (b) $VP_2{}^1$ boundary colored by purple lines. (c) The two buildings – $VP_1{}^1$ in green and $VP_2{}^1$ in purple (from the viewpoint) and $IS_{VP_1^i}^{VP_2^i}$ in white.

$$
\begin{aligned}
x_{pi} &= x_{VBP_1^i} - a\frac{ax_{VBP_1^i} + by_{VBP_1^i} + cz_{VBP_1^i} + d}{a^2 + b^2 + c^2},\\
y_{pi} &= y_{VBP_1^i} - b\frac{ax_{VBP_1^i} + by_{VBP_1^i} + cz_{VBP_1^i} + d}{a^2 + b^2 + c^2},\\
z_{pi} &= z_{VBP_1^i} - c\frac{ax_{VBP_1^i} + by_{VBP_1^i} + cz_{VBP_1^i} + d}{a^2 + b^2 + c^2}.
\end{aligned}
\tag{8}
$$

The intersected surface $IS_{VP_1^i}^{VP_2^i}$ between $PS_{VP_1^i}^{VP_2^j}$ and $VP_2{}^j$ base plane, ∂VP_2^J, can be generally described as polygons intersection:

$$
IS_{VP_1^i}^{VP_2^j} = PS_{VP_1^i}^{VP_2^j} \cap \partial VP_2^J \cup \partial PS_{VP_1^i}^{VP_2^j} \cap VP_2^J. \tag{9}
$$

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

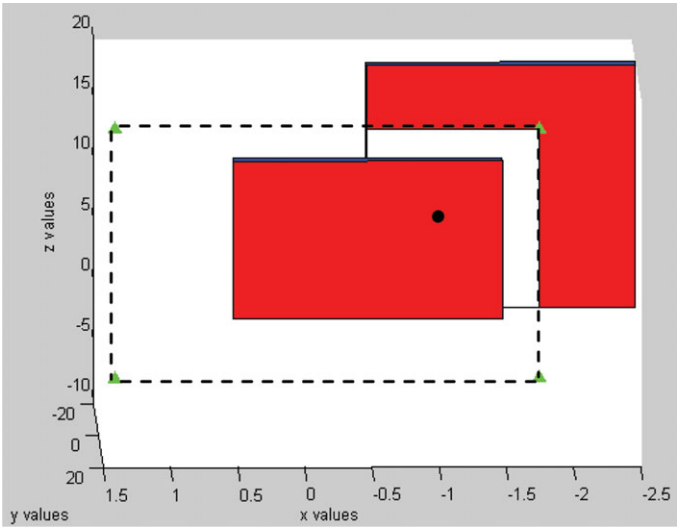


Fig. 8. (Colour online) Projection of $VP_1{}^1$ *to* $VP_2{}^1$ base plane $PS_{VP_1^i}^{VP_2^i}$ marked with dotted lines.

The intersected surface $IS_{VP_1^i}^{VP_2^j}$ is also the invisible one from a viewpoint $V(x_0, y_0, z_0)$, as can be seen in Fig. 9. For simplicity, we demonstrate the method with two buildings from a viewpoint $V(x_0, y_0, z_0)$ one (denoted as the first one) of which hides, fully or partially, the other (the second one).

As can be seen in Fig. 7, in this case, we first compute VBP for each building separately, $VBP_1^{1..4}$, $VBP_2^{1..4}$; based on these VBPs, we generate VPs for each building, VP_1^1, VP_2^1. After that, we project VP_1^1 base to VP_2^1 base plane, $PS_{VP_1^i}^{VP_2^i}$, as seen in Fig. 8, if existing. At this point, we intersect the projected surface in the VP_2^1 base plane and update $VBP_2^{1..4}$and VP_2^1 (decreasing the intersected part). The intersected part is the invisible part of the second building from viewpoint $V(x_0, y_0, z_0)$ hidden by the first building, $IS_{VP_1^i}^{VP_2^i}$, which is marked in white in Fig. 9.

In the case of a third building, in addition to the buildings introduced in Fig. 9, the projected VP will only be the visible ones, and the VBP and VP of the second building will be updated accordingly (as described in the next subsection). In cases of several buildings, the VP base would not necessarily be rectangular, due to the intersected surface profile of the previous projection. We demonstrated a simple case of an occluded building. A general algorithm for more a complex scenario contains the same actions between all the combinations of VP between the objects. Projection and intersection of 3D pyramids can be done with simple computational geometry elements, which demand a very low computation effort. Simulation results for a number of urban scenes and mass modeling complex urban environments visibility can be found in refs. [2, 3].

A pseudocode of the visibility algorithm from a viewpoint can be found in ref. [2].

4.6. Viewpoint invisibility value

Planning UAVs visible trajectory is based on the ability to accumulate the visibility value of each viewpoint explored as part of the planner algorithm. We calculate the exact invisible value of a specific viewpoint, i.e., the total sum of the invisible surfaces and roofs from viewpoint.

We divide point invisibility value into invisible surfaces value (ISV) and invisible roofs value (IRV). This classification allows us to plan delicate and accurate trajectory upon demand. We define ISV and IRS as the total sum of the invisible roofs and surfaces (respectively).

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 Eq. (10):

$$ISV(x_0, y_0, z_0) = \sum_{i=1}^{N_{obj}} IS_{VP_i^{j=1..N_{bound}-1}}^{VP_i^{j=1..N_{bound}-1}}. \tag{10}$$

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

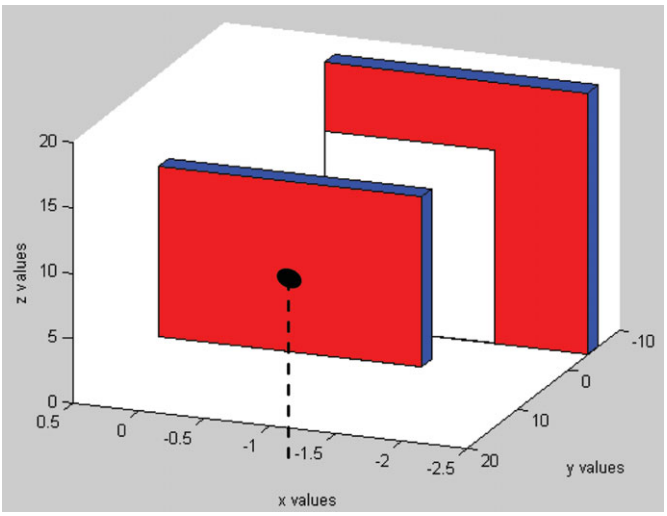


Fig. 9. (Colour online) Computing hidden surfaces between buildings, $VP_2{}^1$ base plane, $IS_{VP_1^i}^{VP_2^i}$.

In the same way, we define IRV value as the total sum of all the invisible roofs surfaces:

$$IRV(x_0, y_0, z_0) = \sum_{i=1}^{N_{obj}} IS_{VP_i^{j=N_{bound}}}^{VP_i^{j=N_{bound}}}. \tag{11}$$

5. UAV Planner

Our planner, similar to previous work,[4] is a local one, generating one step ahead every time step reaching toward the goal, which is a depth first A* search over a tree. We extend previous planners that take into account kinematic and dynamic constraints[9,14] and present a local planner for UAV with these constraints, which for the first time generates fast and exact visible trajectories based on analytic solution. The fast and efficient visibility analysis of our method, presented in Section 5, allows us to generate the most visible trajectory from a start state q_{start} to the goal state q_{goal} in 3D urban environments, and demonstrates our capability, which can be extended to real performances in the future. We assume knowledge of the 3D urban environment model as mentioned in Section 5.3, and use the well-known VO method to avoid collision with buildings presented as static obstacles, exploring safe and maximum visible node in the next time step.

At each time step, the planner computes the next eighth attainable velocities (AVs), as detailed in Section 5.2. The safe nodes not colliding with buildings, i.e., nodes outside VOs,[4] are explored. The planner computes the cost for these safe nodes and chooses the node with the lowest cost. Trajectory can be characterized by the most visible roofs only, surfaces only, or another combination of these kinds of visibility types, as will be detailed in Section 5.4. We repeat this procedure while generating the most visible trajectory.

In this planner, we consider only static obstacles as buildings. Safety can be guaranteed in static environments using the VO method, which can easily avoid moving obstacles in dynamic environments,[4] unlike other motion planning algorithms in static environments (Visibility graph, Voronoi Diagram, etc.). Considering moving obstacles in urban environments for visible trajectories planning is part of our future work.

5.1. Velocity obstacles

The VO[4] is a well-known method for obstacle avoidance in static and dynamic environments, used in our planner to prevent collision between UAV and the buildings (as static obstacles), as part of the trajectory planning method.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

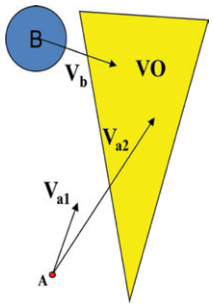


Fig. 10. (Colour online) Linear VOs.

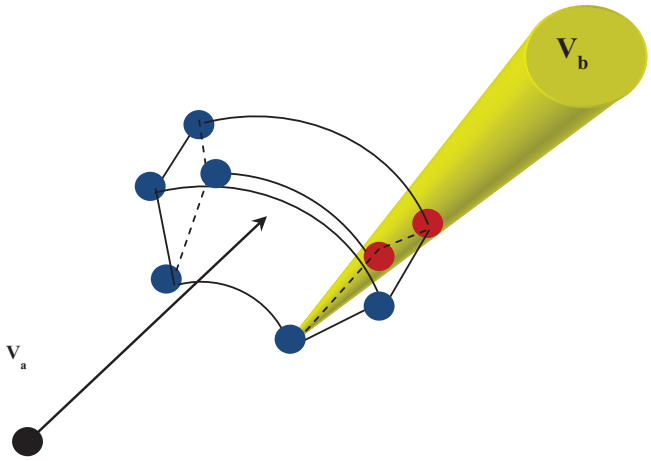


Fig. 11. (Colour online) Tree search method.

The VO represents the set of all colliding velocities of the UAV with each of the neighboring obstacles, in our case static obstacles buildings. Each building is bounded by cylinder instead of circle in 2D case[4] and mapped as static obstacle into the UAV's velocity space.

We introduce the VO of a planar circular obstacle, B, which is moving at a constant velocity v_b, as a cone in the velocity space of UAV, A, reduced to a point by correspondingly enlarging obstacle B.

Each point in VO represents a velocity vector that originates at A. Any velocity of A that penetrates VO is a colliding velocity that would result in a collision between A and B at some future time. Figure 10 shows two velocities of A: one that penetrates VO, v_{a2}, and is hence a colliding velocity, and one that does not, v_{a1}.

All velocities of A that are outside of VO are safe as long as B stays on its current course or in our case a static one. The VO thus allows us to determine if a given UAV velocity will cause a collision.

5.2. Attainable velocities

Based on the dynamic and kinematic constraints, UAVs velocities at the next time step are limited. At each time step during the trajectory planning, we map the AVs, the velocities set at the next time step $t + \tau$, which generate the optimal trajectory, as is well-known from Dubins theory.[28]

We denote the allowable controls as $u = (u_s, u_z, u_\phi)$ as U, where $V \in U$.

We denote the set of dynamic constraints bounding control's rate of change as $\dot{u} = (\dot{u}_s, \dot{u}_z, \dot{u}_\phi) \in U'$.

Considering the extremals controllers as part of the motion primitives of the trajectory cannot ensure time-optimal trajectory for the Dubins airplane model,[28] but is still a suitable heuristic based on time-optimal trajectories of the Dubins car and point mass models.

We calculate the next time step's feasible velocities $\tilde{U}(t + \tau)$, between $(t, t + \tau)$:

$$\tilde{U}(t + \tau) = U \cap \{u | u = u(t) \oplus \tau \cdot U'\}. \tag{12}$$

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

$t = t_0$. $q_{best} = q_{start}$

1. While $(q_{best} \neq q_{goal})$ do:

1.1. Calculate AV nodes from q_{best}, $AV_{i=1}^{8} = \tilde{U}_{i=1}^{8}(t+\tau)$.

1.2. For each node $q_i \in AV$ check:

if $\dot{q}_i \in \bigcup_{j=1}^{n=N_{build}} VO_j^{t_h}_{t_0}$

q_i is illegal.

Else

Calculate node visibility cost $w(q_i)$

1.3. If all nodes are illegal

STOP! No trajectory to the goal

Else

1.3.1. Find node with minimal cost $q_{\min} = \{q_i \mid \min w(q_i)\}$.

1.3.2. Update $q_{best} = q_{\min}$

1.3.3. $t = t + dt$

End

Fig. 12. UAV planner pseudocode.

Integrating $\tilde{U}(t+\tau)$with the UAV model yields the next eight possible nodes for the following combinations:

$$\tilde{U}(t+\tau) = \begin{pmatrix} \tilde{U}_s(t+\tau) \\ \tilde{U}_z(t+\tau) \\ \tilde{U}_\phi(t+\tau) \end{pmatrix} = \begin{pmatrix} u_s^{\min}, & u_s(t) + a_s\tau \\ -u_s^{\max}\tan\phi^{\max}, & u_s(t)\tan u_\phi(t) + u_s^{\max}\tan a_\phi \\ u_z^{\max}, & u_z(t) - a_z\tau \end{pmatrix}. \quad (13)$$

At each time step, we explore the next eight AV at the next time step as part of our tree search, as explained in the next subsection.

5.3. Tree search

Our planner uses a depth first A* search over a tree that expands over time to the goal. Each node $(q, \dot{q}\cdot)$, where $q = (x, y, z, \theta)$, consists of the current UAVs position and velocity at the current time

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

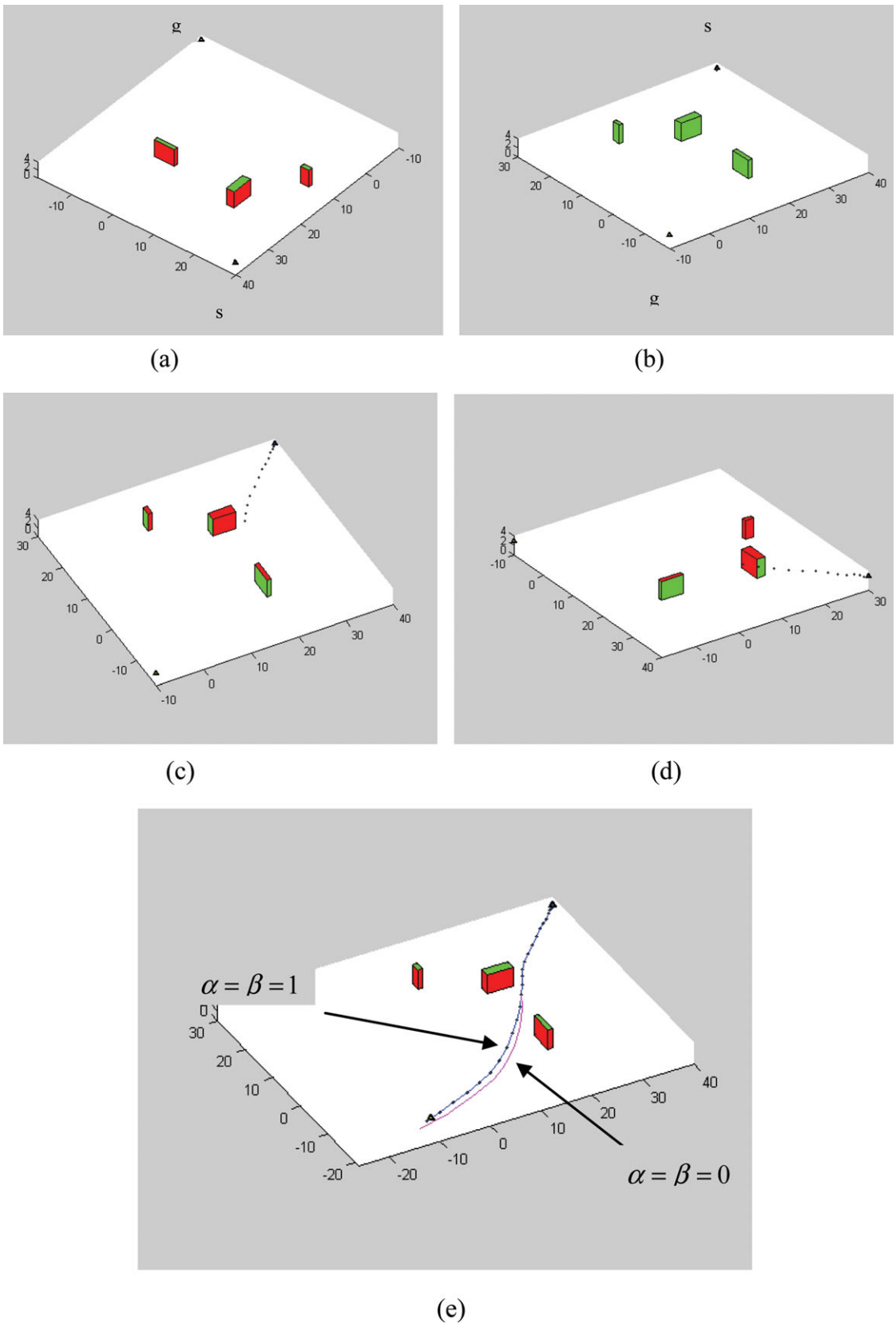


Fig. 13. (Colour online) Trajectory planning in simple urban environment with three buildings. (a, b) Start and goal points with scenario demonstration. (c, d) Visible and invisible parts in the environments at a specific point. (e) Final trajectory with two kinds of visibility characters.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

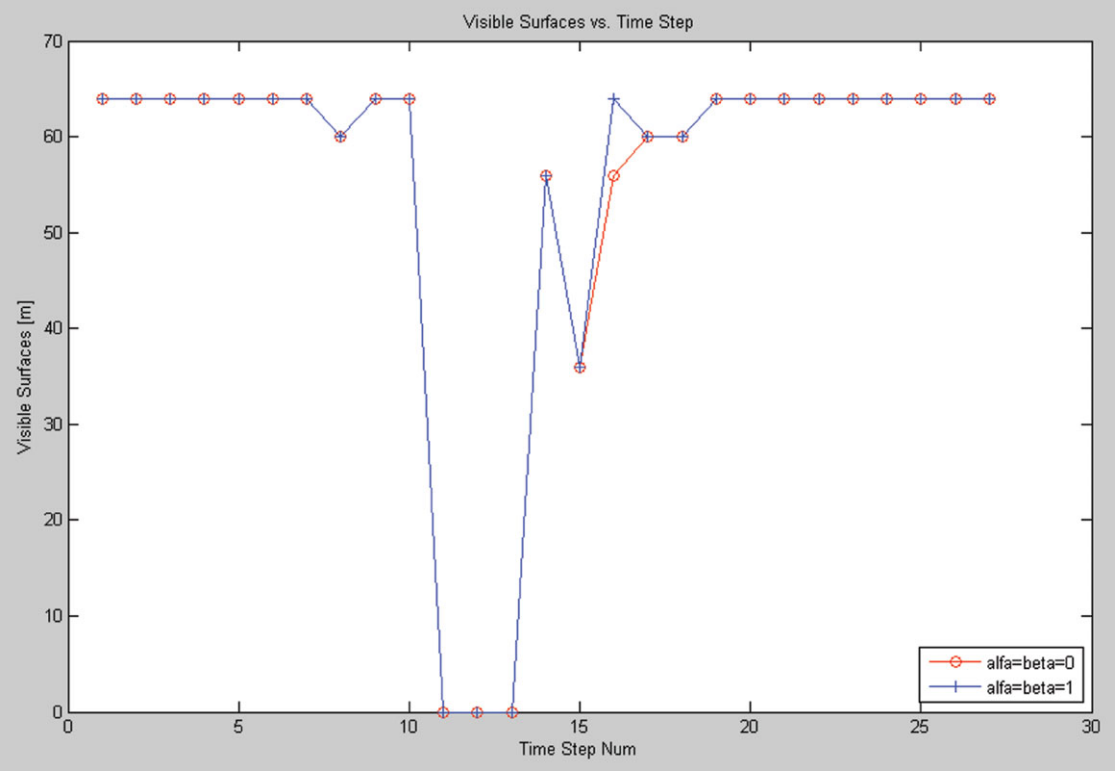


Fig. 14. (Colour online) Total sum of visible surfaces area during running time in two kinds of visibility characters.

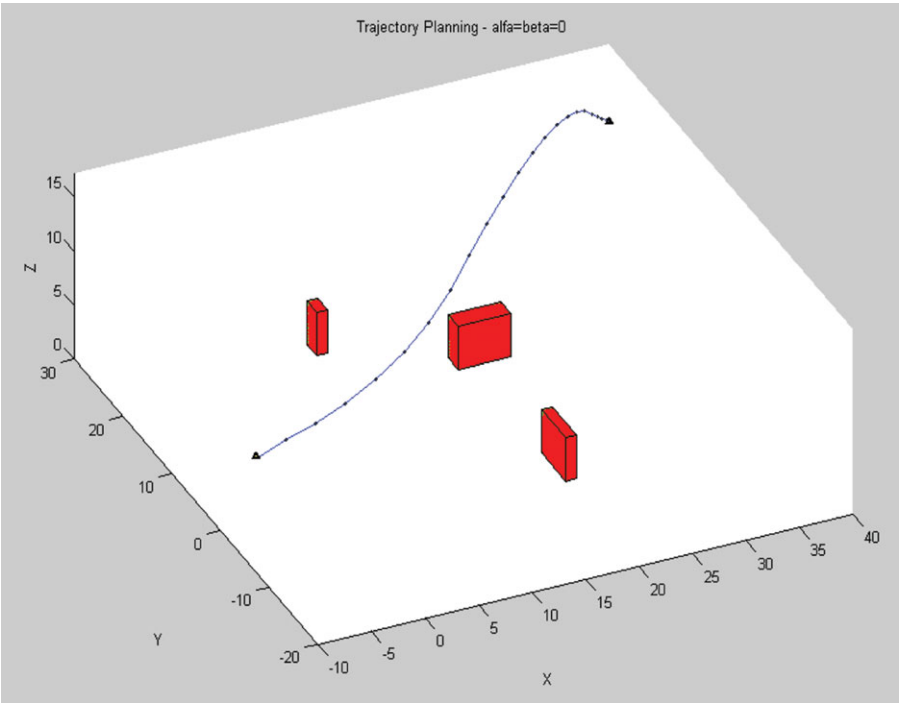


Fig. 15. (Colour online) Final planned trajectory from start to goal points with $\alpha = \beta = 0$.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

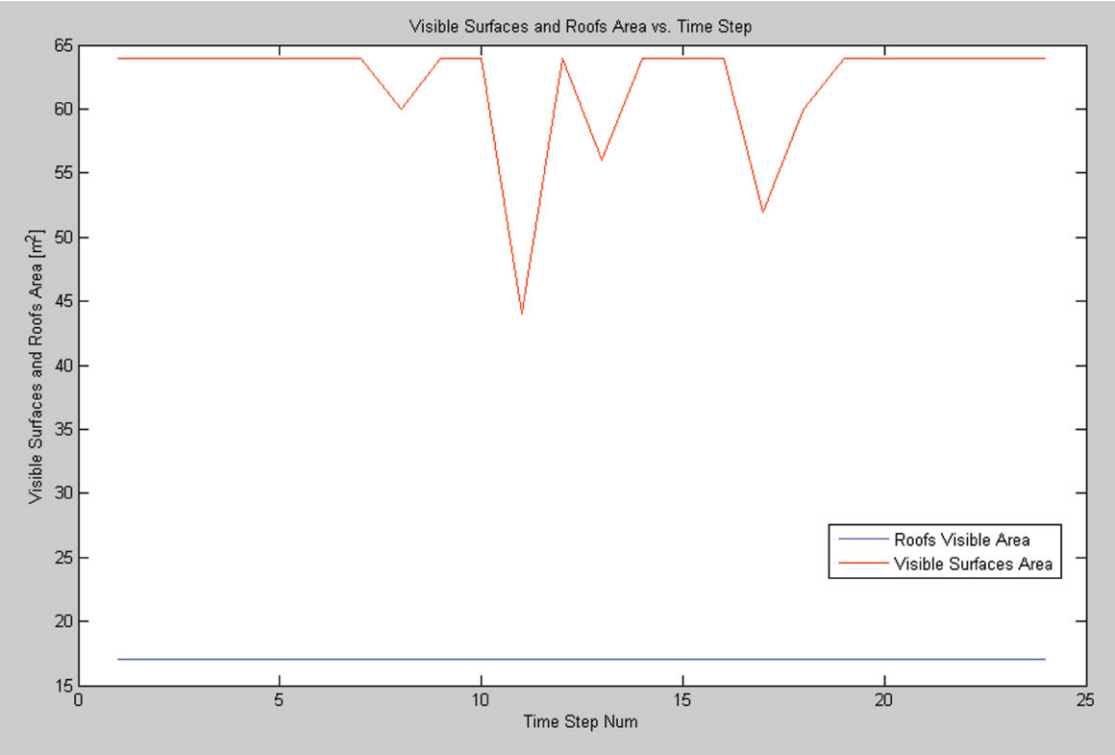


Fig. 16. (Colour online) Total sum of visible surfaces and roof area during running time in the case of $\alpha = \beta = 0$.

step. At each state, the planner computes the set of admissible velocities (AVs), $\tilde{U}(t+\tau)$, from the current UAV velocity,$U(t)$, as shown in Fig. 11. We ensure the safety of nodes by computing a set of VO.

In Fig. 11, nodes inside VO, marked in red, are inadmissible. Nodes out of VO are further evaluated; safe nodes are colored in blue. The safe node with the lowest cost, which is the next most visible node, is explored in the next time step. This is repeated while generating the most visible trajectory, as discussed in the next subsection.

Admissible velocities profile is similar to a trunked cake slice, as seen in Fig. 11, due to the Dubins airplane model with one time step integration ahead. Simple models admissible velocities, such as point mass, create rectangular profile.[4]

5.4. Cost function

Our search is guided by minimum invisible parts from viewpoint V to the 3D urban environment model. 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 is computed for each safe node $(q, \dot{q}) \notin VO$, i.e., node outside VO, considering UAV position at the next time step $(x(t+\tau), y(t+\tau), z(t+\tau))$ as viewpoint:

$$w(q(t+\tau)) = \alpha \cdot ISV(q(t+\tau)) + \beta \cdot IRV(q(t+\tau)), \tag{14}$$

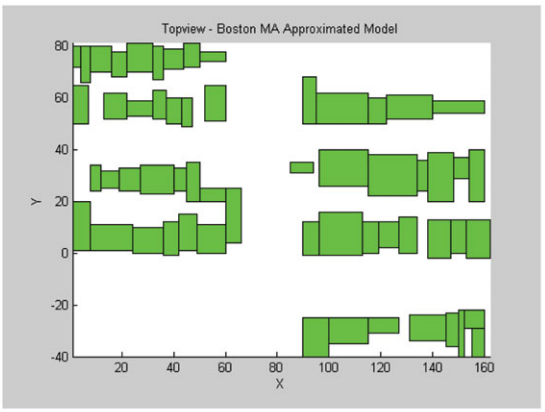
where α, β are coefficients, affecting the trajectory character. The cost function $w(q(t+\tau))$ produces the total sum of invisible parts from the viewpoint to the 3D urban environment, meaning that the velocity at the next time step with the minimum cost function value is the most visible node in our local search.

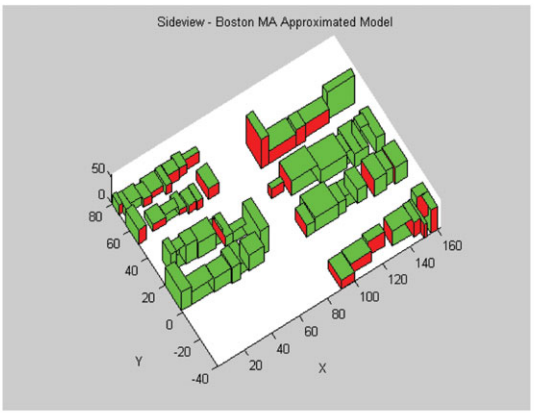
5.5. Planner pseudocode

The pseudocode of the UAV planner is as follows as can be seen in Fig. 12.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

(a)





(b) (c)

Fig. 17. (Colour online) Third scene based on real data of urban environments. (a) Gibson House Museum Region, Beacon St, MA, USA. (b) Top view modeling. (c) Side view from a random viewpoint.

6. Simulations

We have implemented the presented algorithm and tested some urban environments on a 1.8 GHz Intel Core CPU with Matlab. We computed the visible trajectories using our planner, with several types of trajectories consisting of roof and surfaces visibility, based on the introduced visibility computation method.

The initial parameters values are: $u_s(t=0) = 10\,(\mathrm{m/s})$, $u_z\,\theta(t=0) = 5\,(\mathrm{deg})$. UAV dynamic and kinematic constraints are $\phi^{\max} = \pi/4$, $u_z^{\max} = 0.3\,(\mathrm{m/s})$. $u_s^{\min} = 1\,(\mathrm{m/s})$, $u_s^{\max} = 15\,(\mathrm{m/s})$, time step was $dt = 1(\mathrm{s})$.

In the following figures, the start and goal points are marked in blue and yellow triangles (respectively), visible surfaces are marked in red, and invisible surfaces are marked in green.

In the first scene, we demonstrate our method on a simple case with three different buildings, as seen in Fig. 13(a). In Fig. 13(b), the same scenario is demonstrated as an opposite case of the locations of the start and the goal points. Start point set to (40, 30, 2) and the goal point set to $(-10, -18, 2)$.

In Figs. 13(c) and 13(d), UAV trajectories are represented by points, showing the visible and invisible surfaces at a specific point in the middle of the trajectory demonstrating visible and invisible parts at this point, from two different views.

The final trajectory can be seen in Fig. 13(e), where the purple trajectory was generated with $\alpha = \beta = 0$, which is the most visible trajectory, whereas the blue trajectory in the figure is with

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

Table I. Three test cases for the third scene with maximum visible area parameters.

Case number	Starting point	Goal point	Maximum visible area
Case 1	(220, 20, 20)	(−85, 4, 12)	$\alpha = 0, \beta = 0$
Case 2	(220, −40, 50)	(−120, 120, 50)	$\alpha = 0, \beta = 0$
Case 3	(−60, 90, 65)	(210, −75, 50)	$\alpha = 0, \beta = 1$

$\alpha = \beta = 1$. It can be noticed that UAVs avoid the close building at the start point due to the VO obstacle avoidance method.

As can be predicted, the total sum of areas of the visible surfaces in case of $\alpha = \beta = 1$ is larger than the total sum of the trajectory generated in the case of $\alpha = \beta = 0$, as seen in Fig. 14 (the blue line for the first case and the red line for the second).

We tested the second scene with the same previous scenario, but with a higher altitude of the start and target point (their Z values are higher than building's Z values) in order to analyze the influence of roof visibility in such cases. The start point is (40, 30, 10) and the target point is (−10, −15, 15) and the trajectory depicted in Fig. 15 is for the case of $\alpha = \beta = 0$.

In Fig. 16, we present the total sum of visible roof and visible surfaces for each point along the trajectory. The total sum of visible roofs area was a constant maximum value. It is obvious that the total area of visible surfaces at each point in this scenario (Fig. 16) is larger than the corresponding values for the previous scenario (Fig. 14).

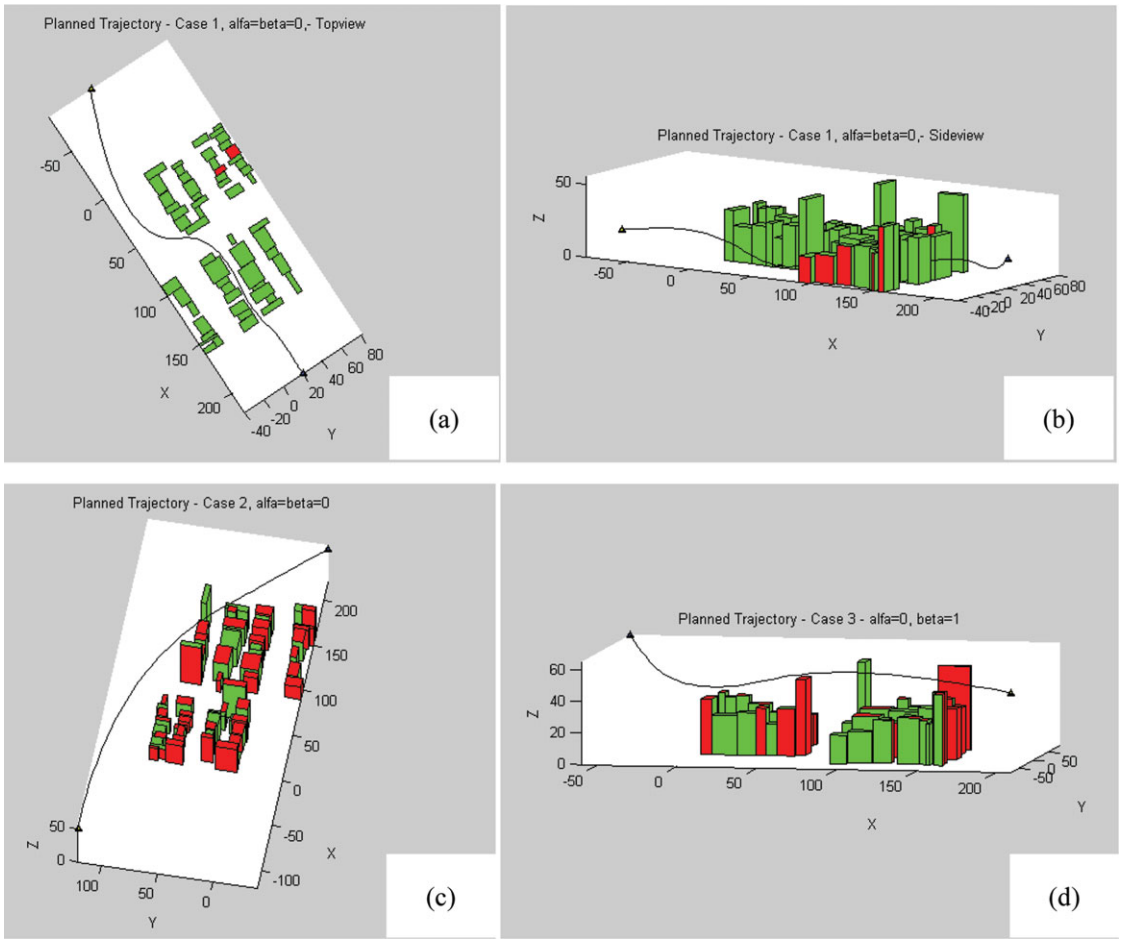


Fig. 18. (Colour online) Final planned trajectory from start to goal points in the third scene according to Table I: (a) Case 1: top view. (b) Case 1: side view. (c) Case 2: side view. (d) Case 3: side view.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

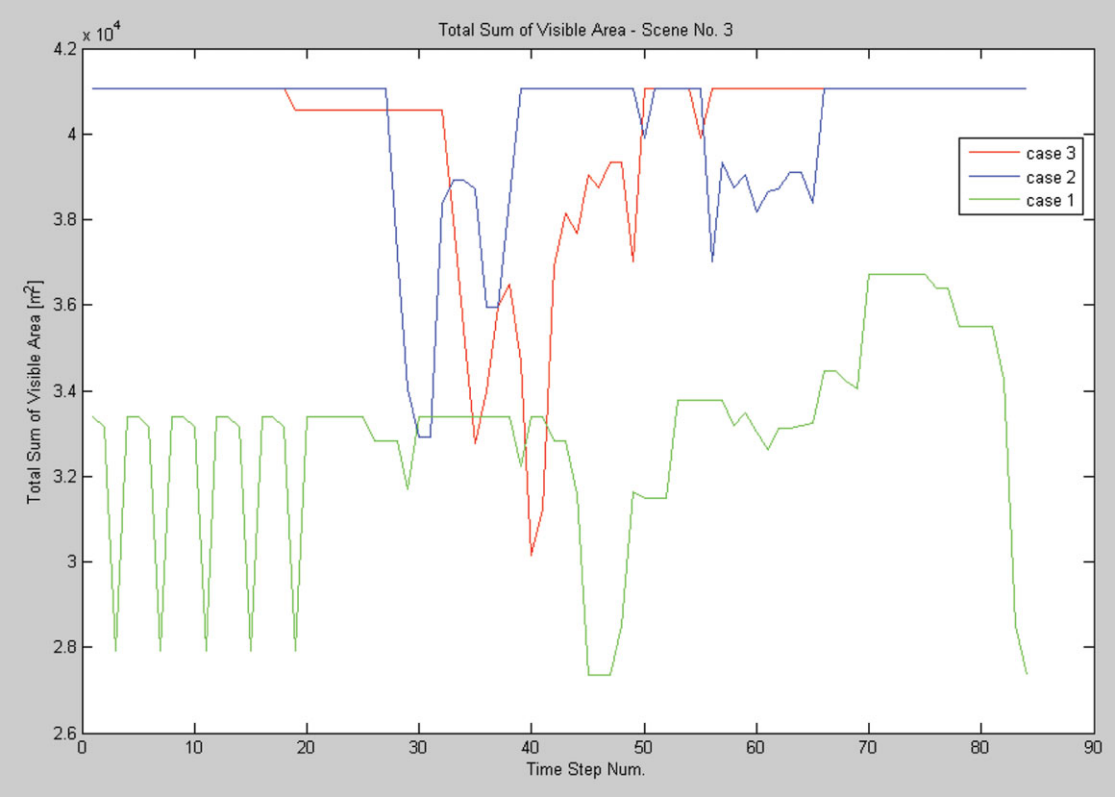


Fig. 19. (Colour online) Total sum of visible surfaces and roof area during running time for the third scene as detailed in Table I.

In the third scene, we demonstrate our method on a dense urban environment modeled from real data of the Gibson House Museum Region, Beacon St., MA, USA. The scene is depicted in Fig. 17(a) and was taken from Google Maps. This scene has been modeled, and a top view and a side view are depicted in Figs. 17(b) and 17(c), respectively.

We tested three cases of start and goal points with different coordinates and different parameters of α, βequals to one or to zero. The parameters that generated the maximum visible area are detailed in Table I, with start and goal points' location, including different Z coordinate values, for these cases.

In Figs. 18, we introduce three trajectories for each case, as detailed in Table I. The total sum of the visible parts (surfaces and roofs) for the different cases can be seen in Fig. 19. One can notice that the maximum total sum of visible area in most of the cases is achieved for the case of $\alpha = 0$, $\beta = 0$. In our case, with cubes building's model, "higher is better." Note that the influence of Z coordinate value is not necessarily better one as introduced in ref. [30], where roof's surfaces profiles are not rectangular.

Complexity analysis of our visibility algorithm is detailed in ref. [2] with performances comparison to the LOS method. The average running time for time step in the third scene is 0.7 s. We tested our method with Matlab to validate our concept. Further work will focus on converting our algorithm to real-time implementation by using GPU or parallel computation methods.

7. Conclusions and Future Work

We have presented an efficient UAV planning algorithm for visible trajectories in a 3D urban environment, modeling basic building structure with mathematical approximation for presentation of buildings' corners as continuous functions.

The planner takes into account dynamic and kinematic constraints using the Dubins airplane model. We compute at each time step the next UAV's attainable velocities and explore the most visible node, while avoiding buildings as static obstacles in the environments using the VO method.

https://doi.org/10.1017/S0263574713000787 Published online by Cambridge University Press

The main contribution of the method presented in this paper is that it does not require special hardware, and is suitable for UAV online visible trajectory planning based on the algorithms' performances. The visible trajectory is exact and allows us to configure the type of visible object, i.e., roof or surfaces visibility of the trajectory, and can be used for different kinds of applications. The algorithm was tested and verified in several kinds of simulations based on real urban environment models.

Further research will focus on modeling more types of roofs in urban environments and examining other global planning algorithms methods integrating other obstacle avoidance concepts and algorithms' real-time implementation. Future work will also include a real-time UAV platform testing our approach in experiments.

References

1. G. Elber, R. Sayegh, G. Barequet and R. Martin, "Two-Dimensional Visibility Charts for Continuous Curves," *Proceedings of Shape Modeling,* MIT, Boston, USA (2005) pp. 206–215.
2. O. Gal and Y. Doytsher, "Fast and Accurate Visibility Computation in a 3D Urban Environment," *Proceedings of the Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services*, Valencia, Spain (2012) pp. 105–110.
3. O. Gal and Y. Doytsher, "Fast Visibility Analysis in 3D Procedural Modeling Environments," *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications*, Washington DC, USA (2012).
4. P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.* **17**, 760–772 (1998).
5. Office of the Secretary of Defense, Unmanned Aerial Vehicles Roadmap, Tech. Rep., December (2002).
6. J. C. Latombe, *Robot Motion Planning* (Kluwer Academic Press, 1990).
7. M. Erdman, and T. Lozano-Perez, "On multiple moving objects," *Algorithmica* **2**, 477–521 (1987).
8. T. Fraichard, "Trajectory planning in a dynamic workspace: a 'state-time space' approach," *Adv. Robot.* **13**, 75–94 (1999).
9. S. M. LaValle and J. Kuffner, "Randomized Kinodynamic Planning," *Proceedings of the IEEE International Conference on Robotics and Automation,* Detroit, MI, USA (1999) pp. 473–479.
10. Z. H. Mao, E. Feron and K. Bilimoria, "Stability and performance of intersecting aircraft flows under decentralized conflict avoidance rules," *IEEE Trans. Intell. Transport. Syst.* **2**, 101–109 (2001).
11. J. Bellingham, A. Richards and J. How, "Receding Horizon Control of Autonomous Aerial Vehicles," *Proceedings of the IEEE American Control Conference*, Anchorage, AK, USA (2002) pp. 3741–3746.
12. B. Sinopoli, M. Micheli, G. Donata and T. Koo, "Vision Based Navigation for an Unmanned Aerial Vehicle," *Proceedings of the IEEE International Conference on Robotics and Automation* (2001).
13. J. Sasiadek and I. Duleba, "3D local trajectory planner for UAV," *J. Intell. Rob. Syst.* **29**, 191–210 (2000).
14. S. A. Bortoff, "Path Planning for UAVs," *Proceedings of the American Control Conference*, Chicago, IL, USA (2000) pp. 364–368.
15. H. Plantinga and R. Dyer, "Visibility, occlusion, and aspect graph," *Int. J. Comput. Vis*. **5**, 137–160 (1990).
16. Y. Doytsher and B. Shmutter, "Digital Elevation Model of Dead Ground," *Proceedings of the Symposium on Mapping and Geographic Information Systems (Commission IV of the International Society for Photogrammetry and Remote Sensing)*, Athens, Georgia, USA (1994).
17. F. Durand, 3D Visibility: Analytical Study and Applications *PhD Thesis* (Universite Joseph Fourier, Grenoble, France, 1999).
18. W. R. Franklin, "Siting Observers on Terrain," *Proceedings of 10th International Symposium on Spatial Data Handling*, Springer-Verlag (2002) pp. 109–120.
19. J. Wang, G. J. Robinson and K. White, "A fast solution to local viewshed computation using grid-based digital elevation models," *Photogramm. Eng. Remote Sens.* **62**, 1157–1164 (1996).
20. J. Wang, G. J. Robinson and K. White, "Generating viewsheds without using sightlines," *Photogram. Eng. Remote Sens.* **66**, 87–90 (2000).
21. C. Ratti, "The lineage of line: space syntax parameters from the analysis of urban DEMs'," *Environ. Plan. Plan. Des.* **32**, 547–566 (2005).
22. L. De Floriani and P. Magillo, "Visibility algorithms on triangulated terrain models," *Int. J. Geograph. Inf. Syst.* **8**, 13–41 (1994).
23. B. Nadler, G. Fibich, S. Lev-Yehudi and D. Cohen-Or, "A qualitative and quantitative visibility analysis in urban scenes," *Comput. Graph*. **5**, 655–666 (1999).
24. S. M. LaValle, *Planning Algorithms* (Cambridge University Press, Cambridge, UK, 2006).
25. M. Hwangbo, J. Kuffner and T. Kanade, "Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs," *Proceeding of the 2007 IEEE International Conference on Robotics and Automation, ICRA 2007*, Roma, Italy (Apr. 10–14, 2007).
26. http://www.asctec.de/uav-applications/research/products/asctec-hummingbird/.

27. A. Bhatia, M. Graziano, S. Karaman, R. Naldi and E. Frazzoli, "Dubins Trajectory Tracking Using Commercial Off-the-Shelf Autopilots," *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii (Aug. 18–21, 2008).
28. H. Chitsaz and S. M. LaValle, "Time-Optimal Paths for a Dubins Airplane," *Proceedings of the IEEE Conference on Decision and Control*, USA (2007) pp. 2379–2384.
29. S. Zlatanova, A. Rahman and S. Wenzhong, "Topology for 3D spatial objects," *Proceedings of the International Symposium Exhibition on Geoinformation* (2002) pp. 22–24.
30. W. R. Franklin and C. Ray, "Higher isn't necessarily better: visibility algorithms and experiments," **In**: *Advances in GIS Research: Sixth International Symposium on Spatial Data Handling* (T. C. Waugh and R. G. Healey, eds.) (Taylor & Francis, Edinburgh, 1994) pp. 751–770.