Computing 3D From-Region Visibility Using Visibility Integrity

Jixuan Zhi[1], Yue Hao[1], Christopher Vo[2], Marco Morales[3], and Jyh-Ming Lien[1]

***Abstract*—Visibility integrity (VI) is a measurement of similarity between the visibilities of regions. It can be used to approximate the visibility of coherently moving targets, called group visibility. It has been shown that computing visibility integrity using agglomerative clustering takes $O(n^4 \log n)$ for n samples. Here, we present a method that speeds up the computation of visibility integrity and reduces the time complexity from $O(n^4 \log n)$ to $O(n^2)$. Based on the idea of visibility integrity, we show for the first time that the visibility-integrity roadmap (VIR), a data structure that partitions a space into zones, can be calculated efficiently in 3D. More specifically, we offer two different offline approaches, a naive one and a kernel-based one, to compute a VIR. In addition, we demonstrate how to apply a VIR to solve group visibility and group following problems in 3D. We propose a planning algorithm for the camera to maintain visibility of group targets by querying the VIR. We evaluate our approach in different 3D simulation environments and compare it to other planning methods.**

***Index Terms*—Motion and Path Planning, Computational Geometry, Motion Control, Visual Tracking, Real-time Planning**

I. Introduction

THE problem of determining if two points in the space are visible from each other is one of the most fundamental problems in robotics. Visibility algorithms and data structures are used to support various robotic system components such as the camera system [1], [2], [3] and the motion planner [4]. For example, robots often use the visibility of other objects in the environment to determine and parameterize their behavior [5]. *From-region visibility* is a problem that involves determining the visible part of a space from a given region. It is the basis of many pursuit-evasion problems [6]. The computation of from-region visibility is known to be expensive and often involves hardware acceleration through GPUs [7]. Therefore, it is not suitable for many real-time 3D applications [8].

As the computation of from-region visibility is costly [7], in [9] we introduced a concept called *Visibility Integrity* in order to build a *Visibility Integrity Roadmap* (VIR), a data structure that encodes visibility information that can be pre-computed entirely *offline*. Here, we further develop *Visibility Integrity* to significantly reduce the time complexity of computing a VIR from $O(n^4 \log n)$ to $O(n^2)$ for n samples. This makes the VIR more applicable to from-region visibility problems that require solutions in real-time. We also propose two methods to build a VIR offline; we apply VIRs to 3D environments and provide metrics to compare it with other methods.

As an application of VIR, we investigate the *group visibility* problem where we need to find the area of the space that maximizes the visibility of a coherent group of moving targets. We demonstrate the proposed group visibility computation method in the context of *group following* which heavily uses visibility information to compute the trajectory for a traveling camera to maximize the number of targets in a group. The group following problem appears in robotics, for instance, when a mobile robot or an unmanned aerial vehicle carries a camera to monitor a group of ground vehicles [10], humans, or animals [11], [8]. Even when the target is a single entity, the group following problem is still relevant, such as when this entity is not a point but a body composed of several articulated parts or limbs, or when the body is deformable [10]. This also happens when the target is a set of estimates of an uncertain state [12].

Historically, there are two approaches to determine the visibility of regions in the space, one decides it from a point (*from-point* methods) and another does it from a region (*from-region* methods). While the *group visibility* problem is similar to from-region methods, the "regions" that we are interested in are those visible to a group of constantly moving targets rather than static surfaces, such as walls, that are usually the "regions" in the from-region methods. In addition, most from-region methods approximate the visibility between different areas through simple shapes (e.g., discs or rectangles) that collectively cover the space and their construction is somewhat arbitrary. Consequently, such strategies frequently suffer from over-partitioning or under-partitioning, thus having a negative effect on the ability to follow a group of objects.

Main Contributions In this paper, we improve several aspects of our previous work in [9] to build a visibility integrity roadmap VIR for a known workspace. Specifically, we present:

- An analysis of *visibility integrity* (Section III-A).
- Two methods to compute a VIR: a naive clustering-based method that takes $O(n^4 \log n)$ time and a kernel-based iteratively-expanding algorithm that takes $O(n^2)$ time, where n is the number of samples created to probe workspace visibility (Section III-B).
- A kernel-based method for group following in 3D environments to plan 3D camera positions and orientations

Manuscript received: February, 24, 2019; Revised May, 19, 2019; Accepted June, 26, 2019.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was in part supported by AFOSR FA9550-12-1-0238 and by Academia Mexicana de Cultura A.C.

[1]Jixuan Zhi, Yue Hao and Jyh-Ming Lien are with Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030 USA, {jzhi,yhao3,jmlien}@gmu.edu

[2]Christopher Vo is with the Sentien Robotics, Inc. and the AES Corporation, Arlington, VA, USA, vo@irixmedia.com

[3]Marco Morales is with Departamento de Sistemas Digitales, Instituto Tecnológico Autónomo de México, Río Hondo 1, México City, 01080, México, marco.morales@itam.mx

Digital Object Identifier (DOI): see top of this page.

that maximize the number of visible targets subject to speed and motion smoothness constraints. We show that the computational cost and the average travel distance of the VIR camera are more favorable comparing to the existing methods (Section IV.B).

II. Related Work

A vast majority of the visibility related problems are studied in the context of computer graphics and computational geometry [13]. A simplified approach to solve the from-region problem is to partition the entire environment into regular grids. For instance, Bittner et al. [7] use samples of ray tracing to determine whether a set of given view cells can see the set of objects in a scene.

Many existing methods for camera motion planning attempt to maintain the visibility of a single reference point (e.g., the center of mass) [14], [15], [16]. However, the assumption (explicit or implicit) that a single reference point (from-point) can track the entire target often over-simplifies the problem. In contrast, we model the targets as a cluster of points and we find a sequence of camera positions to maximize the number of visible points. Therefore, our approach can be applied to tracking a coherent group of agents, feature points in a rigid or deformable body, or joints of an articulated character.

Our approach is more in line with those methods that encode visibility information through data structures built based on different metrics, such as visibility graph [17], mutual visibility graph [18], reduced visibility graph [19], corridor map [16] and pursuit fields [20]. These approaches however were designed to follow a single target, thus only considered from-point visibility.

Our camera motion planning approach is motivated by the visibility-aware roadmap (VAR) [14]. This approach precomputes a graph (or roadmap) to capture the connectivity and visibility information of the workspace. The roadmap consists of overlapping spheres and the visibility between them. Vo and Lien [21] introduced the monotonic tracking regions and adapted VAR for tracking multiple targets using their centers. Unfortunately, the result is suboptimal, limited to 2D, and the quality of the roadmap depends on the type of environment and the size of the discs, which is user-defined.

Another motivating approach for us is proposed by Christie et al. [22]. In contrast to approaches which use ray casting, they quickly evaluate visibility utilizing the principle of soft shadow generation. This method aggregates the visibility information for multiple targets over time and determines the visibility information for all the targets, not one. However, every target in the group is handled separately. In contrast, our method clusters the group members into regions with similar visibility. Therefore, our approach is more efficient for computing the visibility of groups.

We introduced the concept of *visibility integrity* to build a VIR for a known workspace in [9]. The vertices of the roadmap represent regions with *similar visibility*, and its edges connect adjacent visibility regions. An algorithm to compute a VIR offline and to query it for planning in 2D was sketched out and experimentally shown to provide better visibility than VAR. Here we further develop the VIR method to 3D environments.

III. Visibility-Integrity Roadmap (VIR)

As introduced in [9], a visibility-integrity roadmap (VIR) partitions the space into visibility-integrity regions (nodes) that are connected to adjacent ones through edges. We use the cell centers of the grid or sampled points to represent the regions in the workspace. A visibility-integrity region is a cluster of points that can see similar things in the workspace.

A. Measuring Visibility Integrity

We measure the *visibility integrity* $\mathbf{vi}(C)$ of a cluster of points C as an estimate of how similar is the visibility of the elements of C. A low $\mathbf{vi}(C)$ means that some locations can be seen only from the minority of points in C. Figure 1 shows some point sets with high and low visibility integrity values.

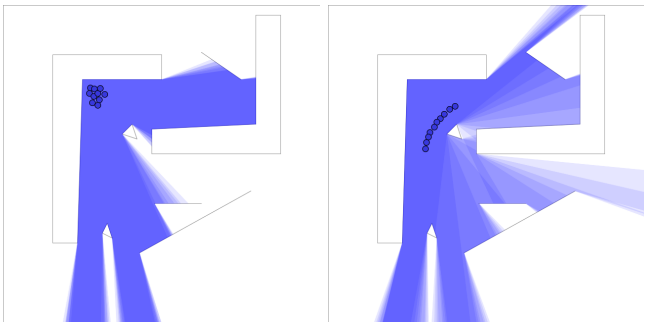


Fig. 1. A set of 10 points with high visibility integrity shown on the left and another set of 10 points in the same environment have low visibility integrity on the right. The lighter area means it is only visible from a few points.

Now, we formally define visibility integrity. We start with the visibility $V(p)$ of a point p as all the points visible from p. Let C be a cluster of points, then $V(C)$ is the union of all points sampled in workspace visible from the points in C, i.e., $V(C) = \bigcup_{\forall p \in C} V(p)$.

Then we introduce the *visibility ratio* of any point q, not necessarily in $V(C)$, with respect to C as the ratio of C that can see q. We define a function named $\mathbf{vr}(q, C)$ to measure the ratio. More specifically,

$$\mathbf{vr}(q, C) = \frac{||V(q) \cap C||}{||C||} \tag{1}$$

where $V(q) \cap C$ is a subset of C visible by q. The ratio is bounded between 0 and 1. When $\mathbf{vr}(q, C)$ is 0, then no point in C can see q. Likewise, if $\mathbf{vr}(q, C)$ is 1, all points in C are visible from q.

The visibility between regions is usually approximated by the ratio between the visible pairs and the total pairs of sampled points representing the regions. Based on this, we denote this visibility ratio between two point sets C and D as

$$\mathbf{vr}(C, D) = \frac{\sum_{q \in D} ||V(q) \cap C||}{||C|| \cdot ||D||} = \frac{\sum_{q \in D} \mathbf{vr}(q, C)}{||D||} \tag{2}$$

Finally, the visibility integrity of C is the average of visibility ratio of all points in $V(C)$, i.e.,

$$\mathbf{vi}(C) = \frac{\sum_{q \in V(C)} ||V(q) \cap C||}{||C|| \cdot ||V(C)||}. \tag{3}$$

and applying Eq. 2 to Eq. 3 we have

$$\mathbf{vi}(C) = \mathbf{vr}\left(C, V(C)\right) \ . \tag{4}$$

The value of $\mathbf{vi}(C)$ is also between 0 and 1. Clusters with high visibility integrity facilitate pursuing a set of objects, because $C \subset V(C)$ for any C and high visibility integrity means most of the points in $V(C)$ are visible for the points in C. In such case, if a majority of the objects are in $V(C)$, the camera can stay in any position of C to maintain a high visibility of all the objects.

The visibility integrity of C is 1 when there is only one point in C and when C is the kernel of a star-shaped polygon. (see Section III-B2 for the definitions of kernel and star-shaped polygon). Note that the next equality also holds:

$$\frac{\sum_{q\in V(C)} \mathbf{vr}(q, C)}{||V(C)||} = \frac{\sum_{p\in C} \mathbf{vr}(p, V(C))}{||C||} \ .$$

B. Constructing a Visibility Integrity Roadmap

To construct a VIR, we start from an initial roadmap composed of points sampled from the space and edges connecting pairs of nearby and mutually-visible points. The initial samples can be obtained from regular grid cell centers (grid-based) or randomly (sampling-based). It can be a simple probabilistic roadmap [23] or even a fuzzy version [24]. Consequently, all nodes q in this initial VIR have $\mathbf{vi}(q) = 1$ Then, a merging operation iteratively combines pairs of connected nodes and reduces the overall visibility integrity.

Let n be the number of centers for the grid-based method or the number of samples for the sampling-based method. Here, we present two methods to compute a VIR.

1) A Naive Method: For each neighborhood point sets U and V, we define the connection $e = \{U, V\}$ associated with the visibility integrity $\mathbf{vi}(U \cup V)$. We process point sets based on the order of their visibility integrity using agglomerative clustering. More specifically, our implementation maintains neighborhood connections in a max-heap. Each element in the heap stores a neighborhood connection $e = \{U, V\}$ and is ordered in the heap by its visibility integrity $\mathbf{vi}(U \cup V)$. The clustering method iteratively collapses neighborhood connections (i.e., merge clusters) with the highest visibility integrity. The clustering process guarantees to finish because of the visibility integrity cannot increase after merging. Note that every pair of clusters merged will also affect the visibility integrity of existing cluster pairs in the heap.

This merge operation is repeated until the highest visibility integrity in the heap is lower than the value specified by the user. An example of the clustering results can be found in Figure 2 (a and b).

Unfortunately, this clustering approach is computationally expensive, in particular for 3D environments. The time complexity is $O(n^4 \log n)$ for fully connected n samples. As the initial VIR can have n^2 edges, and, in every iteration, merging two nodes will affect the visibility integrity of $O(n)$ edges, and the cost of recomputing the weights of these edges (i.e., the visibility integrity of the cluster pairs connected by the edge) is $O(n \log n)$ (see Theorem III.1 and its proof.) Therefore, the total cost is $O(n^2 \cdot (n \cdot n \log n)) = O(n^4 \log n)$.

2) Kernel-based Method: To reduce the time complexity, we present a significantly more efficient approach to build a visibility integrity roadmap. First, let us define an α**-kernel**. The word kernel is a traditional term in computational geometry used to describe a set of points that can see all points in a polygon P. A polygon P with non-empty kernel $\kappa(P)$ is called star-shaped.

It is known that the kernel $\kappa(P) \subset P$ is a single connected component of convex shape. Here, we relax the definition to approximate the idea of a kernel for point sets, and we also use sample points to represent the polygon region P. We define an α**-kernel** to be a set of points $\kappa_\alpha(P)$ such that each can see at least $\alpha|P|$ number of points in P and collectively can see all points in P. Moreover, we enforce $\kappa_\alpha(P) \subset P$. Now, we let α be the desired visibility integrity and the point set $V(C)$ be the visible points from a cluster C, then we have $C = \kappa_\alpha(V(C)) \subset V(C)$. In the rest of this section, we will discuss how we can efficiently compute $\kappa_\alpha(P)$.

For convenience, let us rewrite the visibility integrity of C previously defined in Equation 3 as:

$$\mathbf{vi}(C) = \frac{N(C)}{||C||\cdot||V(C)||} \ , \tag{5}$$

where $N(C) = \sum_{p_i \in V(C)} n_{p_i}(C)$, and $n_{p_i}(C)$ is the number of visible points in C from point p_i.

Based on this definition, we derive the following theorem.

Theorem III.1. *Given the definition of visibility integrity in Eq. 5, the visibility integrity of the union of two clusters C and D can be written as*

$$\boldsymbol{vi}(C \cup D) = \frac{N(C) + N(D)}{(||C||+||D||)||V(C) \cup V(D)||} \ ,$$

and can be computed using the union-find operation in $O((||C||+||D||)\log(||C||+||D||))$ time.

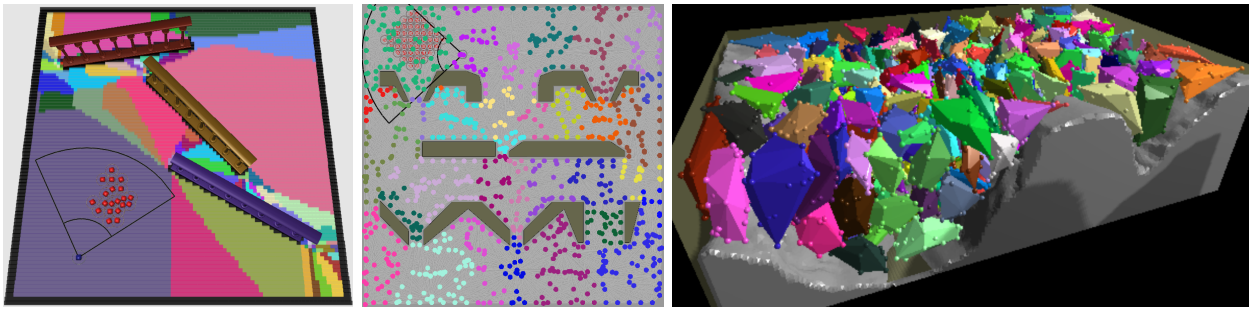
Proof. Simply because, C and D must be disjoint and if a point p_i belongs to both $V(C)$ and $V(D)$, then $n_{p_i}(C \cup D) = n_{p_i}(C) + n_{p_i}(D)$. Similarly, if $p_i \notin V(D)$, then no points in D can see p_i. Therefore, $n_{p_i}(C \cup D) = n_{p_i}(C)$ and if $p_i \notin V(C)$, then $n_{p_i}(C \cup D) = n_{p_i}(D)$. Therefore the total sum of $N(C \cup D)$ is simply the sums of $N(C)$ and $N(D)$ for all points in C and D. Therefore, if we know the sums $N(C)$ of all $n_{p_i}(C)$ and the sum $N(D)$ of all $n_{q_j}(D)$ for C and D, then $N(C)+N(D) = \sum_{p_i \in V(C)} n_{p_i}(C) + \sum_{q_j \in V(D)} n_{q_j}(D)$ can be computed in constant time. Therefore the computation of $\mathbf{vi}(C \cup D)$ is dominated by the operation $V(C) \cup V(D)$, which takes $O((||C||+||D||)\log(||C||+||D||))$ time using the union-find algorithm. □

We can further improve the time complexity if we only make sure that the visibility integrity in the cluster is greater than a user defined value. In particular, we note that $\mathbf{vi}(C \cup D)$ for two clusters can be upper bounded by:

$$\mathbf{vi}(C \cup D) \leq \frac{N(C) + N(D)}{(||C||+||D||)\max(||V(C)||, ||V(D)||)} \ .$$

This upper bound can be computed in constant time. Moreover, the lower bound is:

$$\mathbf{vi}(C \cup D) \geq \frac{N(C) + N(D)}{(||C||+||D||)(||V(C)||+||V(D)||)} \ .$$



(a) Grid-based VI (b) Sampling-based VI (c) Sampling-based VI in canyon 3D of of Fig. 4

Fig. 2. The visibility integrity regions in three different environments. (a) Grid-based visibility integrity region computed by naive method. (b) Sampling-based visibility region computed by naive method. (c) Convex hulls of the clusters acquired by applying the kernel-based method on sampled points. In all figures, each visibility integrity region has a visibility integrity of 0.5.

Therefore, through these bounds, we design an efficient greedy method to compute $\kappa_\alpha(P)$ by expanding an existing kernel to iteratively include the adjacent samples until no more points can be added without violating the definition of $\kappa_\alpha(P)$, i.e., until α drops below the desired visibility integrity. As each expansion of the α**-kernel** will take only constant time, a significant improvement over the $O(n^2 \log n)$ time for each iteration of the agglomerative clustering, the computation time for constructing VIR using α**-kernel** is dominated by the initial visibility check between all pairs of n samples and its time complexity is $O(n^2)$.

IV. Motion Planning using Visibility Integrity

Here, we present how one can use the visibility integrity regions as constructed above to plan motions for a camera to follow a group of targets. We first give a formal statement of the problem, then we apply the kernel-based method that relies on the offline construction of VIR in 3D environments and analyze the results.

A. Problem Statement

In this section, we give a formal problem statement of the group following problem. We assume that the targets are a group of coherent objects moving in a 3D workspace with known obstacles that can block the camera's view. The targets tend to stay together when moving, but they can also split into several small groups if obstacles are in the way. The targets have omnidirectional view and have a bounded velocity.

We consider the following scenario: the target group is constantly moving toward a goal unknown to the camera. If they reach the goal, a new random goal appears. Therefore, the camera cannot predict the trajectories of the targets. The camera is a point in the 3D workspace whose configuration is composed of its position and view direction. The camera also has a maximum velocity that is equal to the targets' maximum velocity. Given the positions of the camera and the group targets, our goal is to efficiently evaluate the camera configurations so that we can maximize the number of targets in the camera's view. *Therefore, the problem of group following can be reduced to finding a sequence of camera configurations that maximize the number of visible targets during a specified period of time subject to the speed constraints of the camera.*

We assume that initially, there is at least one target in the camera's view. If there is no target in the camera's view, the camera keeps its position and randomly changes its orientation until it finds a target and resumes following the group.

B. Motion Planning with an offline-built VIR

Once we have a visibility integrity roadmap, we can use it for a camera to follow the group targets T. We propose a planning algorithm for the camera to maintain visibility of the targets by querying the VIR. At each time step, our planner proceeds as follows. First, it samples potential locations for each target and the camera. Then, it associates sampled positions to the visibility integrity regions in the VIR. Finally, it chooses a camera position to maximize the target visibility by querying the VIR. The details are discussed below.

Our planner first predicts n_t potential future locations for each target $t \in T$. We sample these potential locations based on a multivariate normal distribution of the distance from the target and the angle from the target's heading direction.

Then, our planner identifies the visibility integrity regions containing each of the n_t potential future locations for each target $t \in T$. Let us first discuss the case that VIR is constructed using sampled points (see Fig. 3). Given a predicted target position s, we find the k closest sampled point sets S_s in VIR to s, and we identify R_s as the visibility regions that contain S_s, with k being a user-defined number. The probability of s belonging to a VI region $R \in R_s$ is

$$prob(s \in R) = \frac{\mathbf{vi}(s \cup R)}{\sum_{\forall R' \in R_s} \mathbf{vi}(s \cup R')} , \tag{6}$$

as illustrated in Fig. 3. Therefore, our planner identifies the k VI regions associated with all the predicted target positions. Let $prob(t, s)$ be the likelihood that the target t will be at position s in the next time step, a time step means one iteration. Then the probability of the target t moving into the VI region R is defined as the sum of all predictions $\{s_i\}$:

$$prob(t, R) = \sum_{\forall s_i} prob(t, s_i) \cdot prob(s_i \in R) , \tag{7}$$

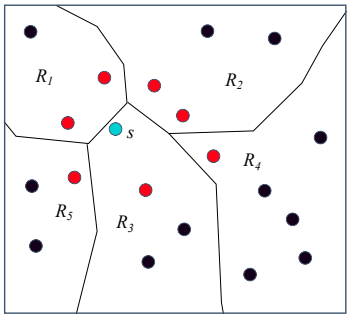


Fig. 3. Estimating the likelihood of point s (blue) to its neighboring VI regions. Point s can be associated with VI regions $\{R_i\}$ through 7 closest samples S_s (red). The probability of association of s with each $\{R_i\}$ estimated by Eq. 6. Note: We cannot use the boundary of VI regions shown in this figure to locate s since our method does not explicitly compute it.

TABLE I
AVERAGE VISIBLE RATIO AND TIME COST (MILLISECONDS IN PARENTHESIS) ACHIEVED BY DIFFERENT METHODS

Env.	Reactive	IO [25]	VIR	Ideal
tower	0.62 (0.54)	0.82 (12.01)	0.73 (2.99)	0.85 (10.69)
pachinko	0.39 (1.31)	0.65 (09.48)	0.47 (3.03)	0.71 (08.33)
ruins	0.50 (1.32)	0.78 (23.46)	0.72 (4.45)	0.78 (19.36)
canyon	0.91 (0.44)	0.99 (17.34)	0.96 (3.20)	0.99 (15.34)

TABLE II
AVERAGE DISTANCE TRAVELLED PER (SIMULATION) STEP

Environment	Reactive	IO [25]	VIR
tower	4.274	4.905	1.837
pachinko	4.674	5.775	2.336
ruins	6.443	4.444	0.684
canyon	5.951	4.191	0.246

Finally, our planner selects a position for the camera to maximize the visibility of targets. To do so, we sample n_c potential future locations for the camera from a ball centered at the current position of the camera with a radius of the distance that the camera can reach in one time step without colliding with obstacles. We evaluate the potential location c as the number of targets that can be seen from c (associated with the VI region R_c) as approximated by:

$$eval(c, R_c) = \sum_{\forall t \in T} \sum_{\forall R_t} prob(t, R_t) \cdot \mathbf{vr}(R_t, R_c) \ . \quad (8)$$

The camera prediction c may be associated with multiple VI regions. If so, the evaluation function accumulates scores from all regions: $eval(c) = \sum_{\forall R_c} prob(c, R_c) \cdot eval(c, R_c)$. The potential position with the largest score will be the next position of the camera. As the targets are continuously moving, our planner can find a smooth and collision-free path for the camera to track the targets.

When building a VIR from a grid, the estimation is simpler. Given a target or camera prediction s, we determine the cell containing s and its corresponding VI region: the one with the highest visibility integrity to s. $prob(s \in R)$ in Eq. 6, $prob(s_i \in R)$ in Eq. 7 and $prob(c, R_c)$ are simply 1 or 0.

C. Performance Evaluation

To evaluate our planner with a kernel-based VIR construction using sampled points, we performed a total of 32 simulation runs on the environments shown in Figure 4 For each environment, the number of sampled points and the value of desired visibility integrity (kernel value) are different. For all experiments, the maximum linear speed of the camera v_C^{max} and the targets v_T^{max} is constant. In each scenario the size of targets $|T|= 20$, the number of samples is set to 25 and 50 for each target and the camera, respectively.

1) visible ratio and time cost: We measure the performance of the various methods by calculating a visible ratio as the proportion of the visible targets compared to the total targets during the entire simulation. For the time cost, we sum up the total time during the entire simulation with 10,000 time steps.

Table I shows the average visible ratio and the average time cost (in milliseconds) per simulation step in various environments. For the VIR method, we only compute the simulation phase time cost (camera behavior for each time step), because the time cost of initialization phase (construct a VIR) for these four environments only takes up to one percent of the simulation phase. We compare our approach with the reactive camera, the IO camera and the ideal camera. The reactive camera selects its next position by following the center of the currently visible targets. The IO camera also uses a sampling strategy [25], and the ideal camera is the ground truth which selects the best camera position based on the predicted positions of both visible and invisible targets.

As shown in the results, the planner using VIR outperforms the baseline reactive method in maintaining the visibility of the target. Compared to the IO and ideal camera, VIR achieves a slightly lower visibility but at a fraction of their time. The VIR method has a good balance between visible ratio and time cost making it suitable for real-time applications.

2) Lazy property: VIR camera is also "lazy". This means that the camera prefers to move as little as possible in order to follow the targets, and it only reacts to situations in which targets may cross the boundary of the visibility-integrity regions. This property is consistent with the theory: considering that $C \subset V(C)$ and if a majority of the targets are in $V(C)$, the camera stays in its position in C and follows the targets in $V(C)$, and if the targets move to $V(C')$, the camera has to move to C' to track the targets.

To measure the "lazy" property, we propose a new metric that averages the moving distance of the camera for each time step during the whole simulation. The distance travelled is based on the user-defined linear velocity. Since the camera remains stationary when it loses all the targets, we only consider the time steps when the visible ratio is above 0. Table II shows the results obtained from different environments. Compared to the reactive method and IO method, the camera with the VIR method moves significantly less than the other two. Especially in large environments such as ruins and canyon, the average distance travelled by the VIR camera is far shorter than the other two. This indicates that there exists some larger visibility integrity region in those environments. The "lazy" property is useful for UAVs to track moving objects because it helps preserve energy.

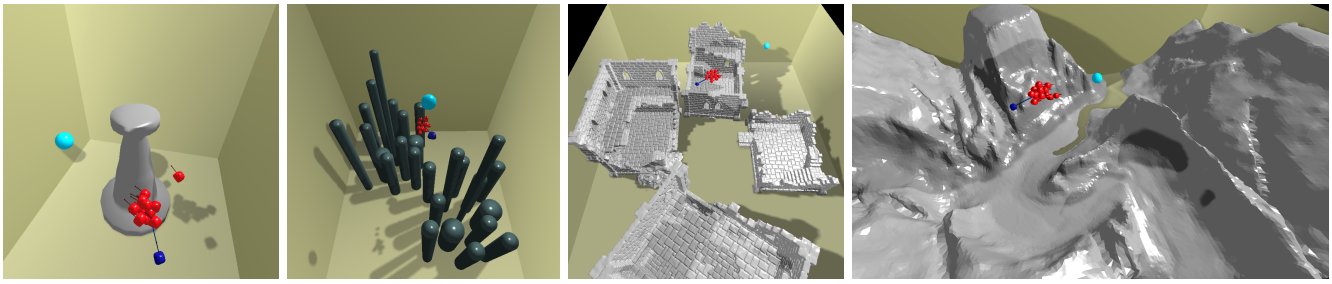


Fig. 4. Images of the four 3D environments for the kernel-based offline experiments. From left to right: tower, pachinko, ruins, canyon. In all environments, all targets are in red, the camera is in dark blue, and the random goal is in sky blue.

V. Conclusion

In this paper, we propose improvements to effectively compute the visibility integrity roadmap data structure that can be used in the group following problem. We show experimental simulations by querying this data structure to calculate a camera's motion to maximize its visibility of group targets. According to the experimental results, the proposed kernel-based offline method achieves similar visibility performance to its predecessors with a significant reduction in the time cost; the algorithm also shows a "lazy" property which means the camera applying this method moves shorter average distances than its predecessors. The proposed method still has limitations. For instance, we found that in environments with many narrow areas, the proposed approach still performs poorly. In addition, even though we drastically reduced the computation time of VIR, it remains expensive for large 3D environments. We are currently developing a strategy that can incrementally build VIR in the regions visited by the targets and the camera.

References

[1] D. Nieuwenhuisen and M. H. Overmars, "Motion planning for camera movements," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 3870– 3876. 1

[2] M. Christie, P. Olivier, and J.-M. Normand, "Camera control in computer graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197 – 2218, 2008. 1

[3] R. Geraerts, "Camera planning in virtual environments using the corridor map method," in *The Second International Workshop on Motion in Games*, S. L. N. in Computer Science (LNCS), Ed., no. 5884, 2009, pp. 194—209. 1

[4] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979. 1

[5] D. M. Bourg and G. Seemann, *AI for Game Developers*. O'Reilly Media, Inc., 2004. 1

[6] S. Sachs, S. M. LaValle, and S. Rajko, "Visibility-based pursuit-evasion in an unknown planar environment," *Int. Journal of Robotics Research*, vol. 23, no. 1, pp. 3–26, 2004. 1

[7] J. Bittner, O. Mattausch, P. Wonka, V. Havran, and M. Wimmer, "Adaptive global visibility sampling," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 94:1–94:10, Aug. 2009, proceedings of ACM SIGGRAPH 2009. 1, 2

[8] S. Rodriguez and N. Amato, "Behavior-based evacuation planning," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 350–355. 1

[9] J.-M. Lien and Y. J. Kim, "Follow moving things in virtual world," in *Human Computer Interaction Korea (HCIK)*, South Korea, Jan 2016, best Paper Award. 1, 2

[10] J. F. Harrison, C. Vo, and J.-M. Lien, "Scalable and robust shepherding via deformable shapes," in *the Third International Conference on Motion in Games*, Zeist, Netherlands, Nov. 2010. 1

[11] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol. 21, July 1987, pp. 25–34. 1

[12] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters," *The International Journal of Robotics Research*, vol. 22, no. 2, pp. 99–116, 2003. 1

[13] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand, "A survey of visibility for walkthrough applications," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 9, no. 3, pp. 412–431, 2003. 2

[14] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross, "Visibility transition planning for dynamic camera control," in *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009, pp. 55–65. 2

[15] D. Nieuwenhuisen and M. H. Overmars, "Motion planning for camera movements," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 3870– 3876. 2

[16] R. Geraerts, "Camera planning in virtual environments using the corridor map method," in *The Second International Workshop on Motion in Games*, S. L. N. in Computer Science (LNCS), Ed., no. 5884, 2009, pp. 194—209. 2

[17] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A sampling-based motion planning approach to maintain visibility of unpredictable targets," *Auton. Robots*, vol. 19, no. 3, pp. 285–300, 2005. 2

[18] I. Becerra, R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J.-P. Laumond, "Maintaining strong mutual visibility of an evader moving over the reduced visibility graph," *Autonomous Robots*, vol. 40, no. 2, pp. 395–423, Feb 2016. 2

[19] R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J.-P. Laumond, "A complexity result for the pursuit-evasion game of maintaining visibility of a moving evader," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 2657–2664. 2

[20] M. Zhang and S. Bhattacharya, "Multi-agent visibility-based target tracking game," in *Distributed Autonomous Robotic Systems*. Tokyo: Springer Japan, 2016, pp. 271–284. 2

[21] C. Vo and J.-M. Lien, "Following a large unpredictable group of targets among obstacles," in *The Third International Conference on Motion in Games*, Zeist, Netherlands, Nov. 2010. 2

[22] M. Christie, J.-M. Normand, and P. Olivier, "Occlusion-free camera control for multiple targets," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012, pp. 59–64. 2

[23] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996. 3

[24] C. L. Nielsen and L. E. Kavraki, "A two level fuzzy prm for manipulation planning," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 1716–1721. 3

[25] C. Becker, H. González-Baños, J.-C. Latombe, and C. Tomasi, "An intelligent observer," in *The 4th International Symposium on Experimental Robotics IV*. London, UK: Springer-Verlag, 1997, pp. 153–160. 5