Global Search for Occlusion Minimisation in Virtual Camera Control

Paolo Burelli, *Student Member, IEEE* and Georgios N. Yannakakis, *Member, IEEE*

***Abstract*— This paper presents a fast and reliable global-search approach to the problem of virtual camera positioning when multiple objects that need to be within the reach of the camera are fully occluded. For this purpose, a comparative analysis of global-search algorithms is presented for the problem of maximising camera visibility across different tasks of varying complexity and within different real-time windows. A custom-designed genetic algorithm is compared to octree-based search and random search and results showcase the advantages of the genetic algorithm proposed with respect to efficiency, robustness and computational effort.**

I. Introduction

Camera control is a vital component of player experience and enjoyability in games [23]. A camera in games provides the player with a means for exploring the game world, getting feedback on her actions, and updating the state of the game. Given its critical importance in 3D virtual environments camera control may provide means of challenge for and justifiability of artificial intelligence. Camera settings for games are usually predefined by designers and potential camera visibility problems (e.g. in the case of occluded objects) are tackled via simple, nevertheless, unjustifiable and unrealistic solutions — i.e. the camera rapidly jumps to a selected non-occluded position towards the closest object of interest.

In this paper we propose a top-down global search approach to the problem of finding a non-occluded point when in-game *objects of interest* are fully occluded. Occlusion occurs when points (or objects) of interest that the camera needs to look at are fully, or partially, hidden by objects or walls of the designed scene geometry. Under these circumstances, the camera controller is required to find a path towards an occlusion-free point (ideally a fully non-occluded point) within a realistic time frame. If the controller is unsuccessful within a short time window (e.g. 200ms maximum), the reliability of game camera control is challenged and any immersion emerged by cinematographic game experience is lost.

The problem of camera visibility is challenging because (a) visibility heuristics are computationally expensive to calculate in real-time, and (b) the generated visibility function terrains are very rough for a search algorithm to explore. In particular, in order to evaluate the visibility "goodness" of a camera position, rays need to be casted towards the object(s) of interest; the designer often has to sacrifice visibility accuracy due to the computational cost of visibility estimation in real-time. Moreover, in the majority of occlusion situations met in computer games the objects of interest are much smaller than scene geometry features. This suggests that objects are either occluded or non-occluded; the situations where objects are partially occluded are rare. On that basis, visibility fitness generates rough search spaces the vast majority of which is covered by multimodal fitness plateaus of near-zero gradient.

Gradient search [7] and local search (e.g. artificial potential fields [9]) are bound to fail in such problems. Instead, robust and efficient global search algorithms are required to allocate fully, or partially, un-occluded camera positions. This paper introduces a comparative study of search algorithms for solving the problem of visibility occlusion in camera control. A custom-designed genetic algorithm (GA) for the investigated problem, octree-based search [1] (best-first and depth-first) and random search are evaluated in occlusion problems of increasing complexity and their speed is evaluated across different time windows. Algorithm performance is accessed via the amount of object visibility of the generated solutions, the number of times a global maximum is found and the time it took the algorithm to find the maximum.

Results show that the GA proposed demonstrates robustness and real-time efficiency across four dissimilar occlusion case studies containing three objects of interest and varying in complexity. It is also apparent that the GA approach performs well consistently with respect to real-time performance in all case studies examined. Random search performs well in complex problems but it performs poorly in simpler problems. Finally, octree-based search is outperformed by the aforementioned algorithms and performs well in simple problems only.

This paper is innovative in that it introduces an efficient and reliable GA solution to the problem of full occlusion in camera control; it examines complex case studies of multiple (three) objects of interest; and it provides a comparative analysis of search algorithms (including genetic and random search) with respect to problem complexity and real-time performance.

II. Background

The problem of automatically controlling the camera in virtual 3D environments has recently received significant attention from the research community [11]. The majority of the earliest approaches to camera control [24], [6], [13] focus on the mapping between user input and the degrees of freedom (DOF) of the camera in the 3D space. Direct

Authors are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark. Emails: {pabu, yannakakis}@itu.dk

978-1-4244-8126-2/10/$26.00

control of the several DOFs of the camera showed to often be problematic for the user [12] so researchers started to investigate how to automatically place and configure the camera.

One of the first examples of automatic camera control, was developed by Blinn already in 1988; it was an automatic camera control system for planet visualization in a space simulation at NASA [6]. Blinn's work has inspired many other researchers trying to produce more flexible autonomous camera systems and attempting to integrate aspects like camera motion and frame composition [2].

More generic approaches model camera control as a constraint satisfaction or optimisation problem. These approaches require the designer to define a set of required frame properties which are then modelled either as an objective function to be maximised by the solver or as a set of constraints that the camera configuration must satisfy. These properties describe how the frame should look like in terms of object size, visibility and positioning. Bares et al. [3] first introduced a detailed definition of these constraints.

Global optimisation based systems (see Halper's and Olivier's CAMPLAN [15] approach among others) model these requirements as a fitness function (a weighted sum of each required frame property) and search the space for the camera configuration generating the maximum fitness value. These approaches guarantee to find a result but their computational cost is high. On that basis, constraint satisfaction systems [17] are much more efficient but may not return any result if there is no configuration satisfying all the frame requirements at the same time. Bares and Lester [4] addressed the issue by identifying conflicting constraints and producing multiple camera configurations corresponding to the minimum number of non-conflicting subsets. Bourne and Sattar [7] extended Bares' and Lester's solution by adding a *weight* property to each constraint to define a relaxation priority. Another set of approaches [22], [10], [8], combines constraint satisfaction to select feasible volumes (therefore reduce the size of the search space) and optimisation to find the best camera configuration within these spaces.

Due to the complexity of evaluating a proper objective (fitness) function for camera control the aforementioned approaches have proven to be unsuitable for real-time interactive applications since their required computational time is too high to keep the camera updated at a reasonable rate (from 20 to 60 times per second). Moreover, in several cases the best camera configuration might not correspond to the global optimum of the fitness function, since the camera needs to maintain frame coherence (continuity between successive frames) [16].

Beckhaus et al. [5] investigated first the application of local search algorithms to camera control. Their system used Artificial Potential Fields (APFs) to guide the camera through a museum and generate smooth virtual tours. Bourne and Sattar [7] proposed a system that employed sliding octrees to guide the camera to the optimal camera configuration. Burelli and Jhala [9] extended these two approaches to include frame composition and support multiple-object visibility.

Local search approaches offer reasonable real-time performance and handle well frame coherence but often converge prematurely to local minima. This characteristic becomes critical when the camera control system has to optimise visibility of *objects of interest* since the visibility heuristic consists of many local minima areas with almost no gradient information available to guide local search (see Section IV).

A. Occlusion

Successfully handling object occlusion constitutes a vital component of an efficient camera controller [11]. Object visibility plays a key role in frame composition. For instance, an object satisfying all frame conditions (e.g. position in frame and projection size) does not provide any of the required visual information if it is completely invisible due to an occlusion.

The object occlusion problem can be separated in two dissimilar yet dependent tasks: occlusion evaluation/detection and occlusion minimisation/avoidance. Occlusion happens when the object of interest is hidden fully or partially by another object. A common technique to detect occlusion consists of casting a series of rays between the object of interest and the camera [8], [7]. A similar approach [21] generates a bounding volume containing both the camera and the object of interest and checks whether other objects intersect this volume. A third approach [16] exploits the graphic hardware by rendering the scene at a low resolution with a color associated to each object and checking the presence of the color associated to the object of interest. These techniques have been used for occlusion avoidance (i.e. maintaining an object visible in the frame) by reacting to incoming occluders [7], and via particle swarm optimisation [8]. Pickering [22] proposed a shadow-volume occlusion avoidance algorithm where the object of interest is modelled as a light emitter and all the shadow-volumes generated are considered occluded areas. However, the aforementioned approaches are not suitable for dynamic environments like games due to their high computational cost. Bourne and Sattar [7] devised an escape mechanism from occluded camera positions which forces the camera to *jump* to the first non-occluded position between its current position and the position of the object of interest. Their approach, however, generates undesired camera jumping behaviours within a game environment and considers just one object of interest.

B. The Camera Control System Utilized

We have developed CamOn [9], an autonomous camera system capable of generating smooth camera animations and solving camera composition tasks. Artificial Potential Fields (APFs) [19] are used to model camera constraints; each constraint is translated into a force vector attracting or repulsing the camera position, gradient search is then applied to animate the camera towards an optimal configuration. The APF camera system guarantees real-time performance and frame-coherence during camera animation. The CamOn occlusion avoidance system, inspired by [7], uses occlusion

information to modify the APF which, in turn, drives the camera out of the occluded position. Although this technique is very efficient, being a local search approach it is unable to escape local visibility optima (i.e. when the object of interest is fully covered by another object).

This paper is the first to propose an efficient yet cost-effective global search solution for minimising occlusion. The approach is also innovative in that it is able to handle visibility requirements for multiple targets rather than just one. Furthermore the paper introduces a comparative study of efficiency, robustness and effort cost of dissimilar global search algorithms for occlusion minimisation. The custom-designed genetic algorithm is found to be the most reliable approach for the problem.

III. Search algorithms

We employed four different global search algorithms to search for optimal un-occluded camera positions in dissimilar three dimensional scenes: a custom-designed genetic search algorithm, octree-based best first and depth first search and random search. In order to make a fair comparison among them, all implemented algorithms share the same search space, the same fitness evaluation method and the same termination conditions.

We assess the quality of a three dimensional camera position via the following fitness function, f:

$$f = \sum_{i=0}^{N} v_i w_i \tag{1}$$

where, v_i, is the visible fraction of the i-th object of interest in the frame; w_i, is the object's importance expressed as a weight value [9]; and, N, is the number of objects considered and need to co-exist within the camera's view-frame. In the experiments presented here, N equals 3 and w contains equal values ($1/N = 0.33$) for all three objects examined. On that basis, f values lie between 0 (no visibility of objects) and 1 (maximum visibility of all objects).

Visibility, v, is calculated by casting 9 rays from the camera's current position to the bounding box of each object. Eight rays are casted towards the corners and 1 towards the centre of the object. We chose ray-casting among other occlusion detection techniques due to their implementation ease, platform independency and hardware acceleration potential.

The search space that all four mechanisms consider is defined as a cube that surrounds the objects of interest. The size s of this cube is calculated as follows:

$$s = \max_i \{\vec{P}_i - \vec{C}\} \tag{2}$$

where, $\vec{P}_i$, is the position of the i-th object in the frame and, $\vec{C}$, is the centre of positions of all N objects considered which also defines the centre of the search cube; i.e. $\vec{C} = \sum_{i=0}^{N} \vec{P}_i/N$.

All algorithms are terminated if a camera position found generates an optimal fitness value ($f = 1.0$) or when a predefined time window is elapsed.

A. Genetic Algorithm

We implemented a modified version of a generational Genetic Algorithm (GA)[14] with custom crossover and mutation operators. Each chromosome, consisting of three real values, represents the location of the camera in the 3D space. The population, containing 120 individuals, is randomly placed into the search space via a uniform distribution and evaluated via (1) at each generation.

We use an elitism selection scheme where the 30 best chromosomes of each generation are selected for reproduction. Mating of parents is based on their ranking within the 30 best chromosomes; the first mates with the second, the third mates with the forth and so on. Ninety offspring are generated by applying a custom-designed crossover operator (with 100% probability) to each pair of selected parents. The crossover operator applied is a fitness-based weighted sum of the parents' position. The generated offspring $\vec{O}$ is defined as follows:

$$\vec{O} = \frac{\vec{P}_a f_a + \vec{P}_b f_b}{f_a + f_b} \tag{3}$$

where $\vec{P}_a$ and $\vec{P}_b$ are the two selected parents; and f_a and f_b are their corresponding fitness values.

Mutation is applied to all genes of the chromosome with a probability of 50%. The custom mutation operator applies a vector translation to the chromosome by adding a uniformly-distributed random vector to it. The maximum value of the dimensions of the translation vector equals 10% the size of the search space (i.e. $0.1s$). Generated offspring replace the 90 worst chromosomes of the current population.

B. Octree-based search

Octree generation [1] is a very popular technique of search within 3D virtual environments which makes it ideal for comparison against genetic and random search.

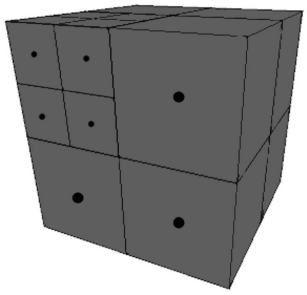


Fig. 1: Octree space subdivision

Specifically, we utilize octree spatial subdivision (see Fig. 1) and apply both best-first and depth-first search on the generated tree. Each node of the octree corresponds to a cubic subspace. During the octree exploration a fitness value is assigned at each subspace corresponding to the fitness value of its centre point.

The best-first algorithm picks the node of the octree with the highest fitness at each iteration and recursively subdivides its corresponding cube and searches within. Breadth-first

search operates a full search through all nodes of the octree: at each iteration all nodes are evaluated and ranked by fitness. Children nodes are generated and evaluated following the fitness ranking. In both search algorithms a node is selected randomly if two or more nodes have equal best fitness values.

C. Random Search

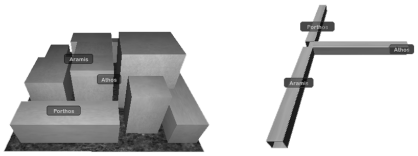
Random search generates a sequence of uniformly-distributed random points which are evaluated via (1). The algorithm stops if the termination conditions are satisfied (maximum expected fitness is reached) and the camera position with best fitness value is returned. Random search can be very efficient in very rough fitness landscapes and it is expected to work well in maximising camera visibility due to the morphology of the fitness terrains generated. This search mechanism is used as a control baseline.

IV. Object Occlusion Case Studies

This section presents the four test-bed scenes designed to test the efficiency and robustness of the four search algorithms; the complexity generated by each case study is also discussed.

The four scenes, namely, *Forest*, *Building*, *City*, and *Tunnels*, are illustrated in Fig. 2a, Fig. 2b, Fig. 2c and Fig. 2d, respectively. There are three objects (i.e. virtual characters) that need to be within the camera's viewpoint named *Athos*, *Porthos* and *Aramis*. We believe that those four case studies cover a wide range of scene geometry met in computer games; furthermore they provide a palette of testbeds that diversify the complexity of search.

(a) Forest (b) Building



(c) City (d) Tunnels

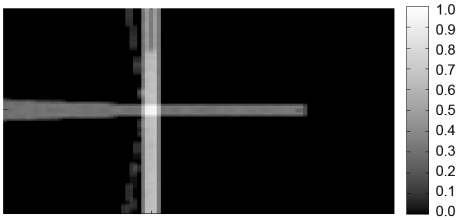
Fig. 2: The four test-bed scenes examined

The first test-bed scene (Fig. 2a) represents a forest-like environment, in which objects are surrounded by trees which act as occluders. Due to the narrow shape of the trees the areas where the camera is fully occluded are very few. Figure 3a shows also that the portion of space where all the three objects are visible is predominant. The second test-bed (Fig. 2b) consists of a house-like environment with walls separating the space into rooms and the objects placed in different rooms. The fitness landscape illustrated in Fig. 3b shows that the walls act as large occluders drastically reducing the portions of space where all the objects are visible at the same time. In the Building scene, there is one small area from where all three objects are fully visible.

(a) Forest

(b) Building

(c) City



(d) Tunnels

Fig. 3: Top-down view of the fitness landscapes of the four testbeds. The sample is calculated setting the camera position at a height of one meter from the ground level of the maps, This corresponds to half the height of the virtual character.

The third test-bed scene (Fig. 2c) is a city model with large buildings and narrow streets and the last test-bed (Fig. 2d) represents a three-tunnel crossing. The three objects are placed on the streets and in the tunnels of the City and the Tunnels scenes, respectively. It is worth noticing that there is no area from which all objects are visible in the City scene. That scene, however, has three local optima areas from where only two of the three objects are fully visible. In the Tunnels scene all objects are visible from the area of the

tunnel crossing.

The fitness landscapes of all scenes reveal some common characteristics. Due to the nature of the visibility problem and the way the fitness function is calculated, the landscape is split into several sub-areas with constant visibility values (plateaus) while the borders between these areas are very steep. This results to a search space with limited areas in which the gradient is non-zero. It is worth noting that the steepness of the areas between fitness plateaus is dependent on the size of the objects and the occluders. The three objects selected for the experiments presented here are virtual characters of realistic human-like dimensions. Each object is a cylinder 2 meters tall and 50 cm wide while all scenes are designed in an area of 17m $\times$ 17m (the ceiling in the Building scene is 4 meters tall). These scene types (small objects; large occluders) increase the complexity of the visibility problem and the need of global search algorithms to find an un-occluded camera position in the search space; local search is very likely to fail in such fitness terrains. The particular morphology of the fitness landscapes is an evident sign of high problem difficulty as confirmed by the Fitness Distance Correlation (FDC) [18] calculated for all four scenes which approximates zero values for all four testbeds (Forest: $-1.9 \cdot 10^{-6}$, Building: $-5.9 \cdot 10^{-6}$, City: $-2.5 \cdot 10^{-6}$, Tunnels: $-5.4 \cdot 10^{-7}$).

The above poses the question of whether the performance (fitness) function selected is appropriate for our problem. More rays toward an object would probably provide more information about the level of visibility and would assist in approximating visibility more accurately. However, ray-casting is computationally expensive and a designer has to maintain the right balance between computational time of fitness evaluation and approximation of fitness in the demanding task of camera control. Given that 200 ms is the largest acceptable time window available more rays would have an negative impact on the performance of all search algorithms.

A. Complexity Measure

To assess the difficulty of each case study a measure of complexity is required. While FDC can be a reliable measure of problem complexity we argue that it is inappropriate for the problem of object visibility since the fitness maps reveal that there is no sufficient fitness variation with respect to distance from the global optimum resulting in near-zero FDCs. Thus, alternatively, we propose the following measures of problem complexity: optimal space size, feasible space size, and fitness-based complexity.

- The *optimal space size* is calculated as the percentage of the global optima subspace over the total search space. This measure approximates the probability to hit a global optimum by chance; however, it does not consider the other partial solutions. The optimal space size value for the Forest, Building, City and Tunnel map is 40%, 0.5%, 0.1% and 0.1%, respectively.
- The *feasible space size* is calculated as the percentage of the subspace containing positions with non-zero fitness over the total search space. This measure approximates the probability of hitting any *feasible* solution by chance but it considers global optima solutions as feasible solutions. The feasible space size value for the Forest, Building, City and Tunnel map is 93%, 36%, 23% and 4%, respectively.
- The *fitness-based complexity* measure considers the average fitness value of a sufficient number of camera positions. Thus, scene complexity, c, is calculated as

$$c = 1 - \frac{\sum_{i=0}^{n} f_i}{n} \tag{4}$$

where f_i is the fitness value of the i-th position, and n is the number of camera positions considered for the calculation of c; n equals 10^5 in this paper and it is obtained by uniformly discretising the search space (100, 100 and 10 samples along the x, z, and y axis, respectively). The c value for the Forest, Building, City and Tunnel scenes is 0.4, 0.85, 0.89, and 0.91, respectively.

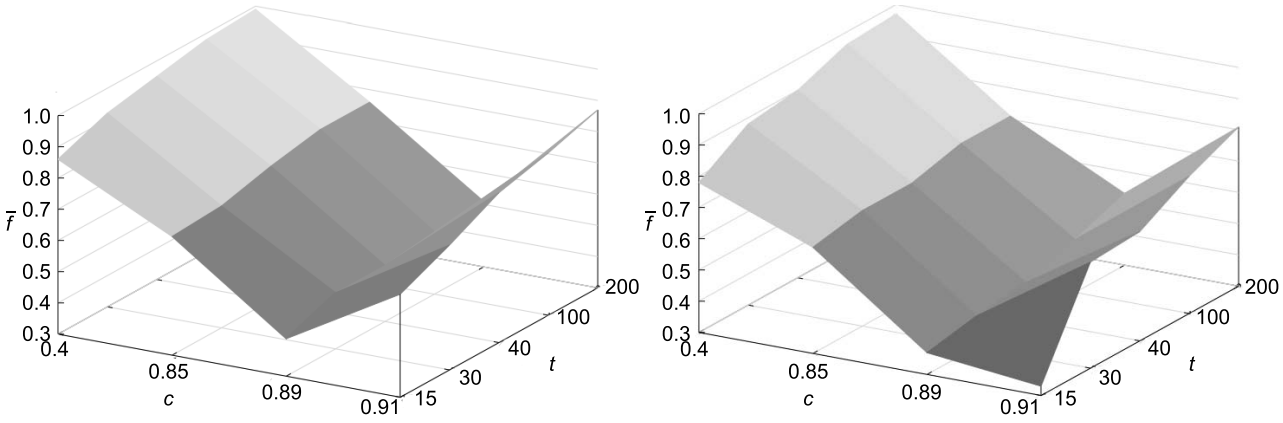
All above measures indicate that complexity increases from the Forest scene to the Building scene, and further to the City scene reaching a maximum value at the Tunnels scene. The difference of the complexity in between the scenes is dependent on the complexity measure considered. However, all three provide the same ranking of scenes with respect to complexity; in that sense all of them are appropriate measures for our scope. We pick the fitness-based value as a measure of complexity since it incorporates fitness information of the whole search space including both partial and global visibility areas.

V. Results

In this section we test the performance of the search algorithms with respect to the complexity of the task to be solved and the time taken for the algorithm to reach a solution. All experiments run on a Intel MacBook Pro, with a 2.0 GHz Core 2 CPU (the implemented algorithms use only one core) and 4 GB of RAM at 1067 MHz. The machine is capable of computing a maximum a 4000 fitness evaluations per second, but the average value (due to the system scheduler) is around 3750 evaluations per second.

A. Algorithm Performance Measures

Each test has been carried out 100 times for each case-study and each time window investigated. The average and standard deviation of the generated fitness values are calculated at the end of each test and used to assess the performance of each algorithm. Other measures of performance considered include the number of times the algorithm finds a global optimum and the time required to find the global optimum. The average fitness, $\overline{f}$, and the number of times the global maximum was found, g, provide measures of algorithm efficiency. The standard deviation of fitness defines a measure of algorithm robustness while the average time required to find a global maximum is suggested as a measure of computational effort.



(a) Genetic Algorithm

(b) Random Search

Fig. 4: Average fitness versus complexity, c, and time, t (in ms).

B. Time Windows

To examine how performance evolves over time all algorithms are terminated after some predefined time elapses. The real-time window values, t, selected in the experiments reported in this paper are 15, 30, 40, 100 and 200 milliseconds corresponding, respectively, to 60, 30, 25, 10 and 5 potential algorithm executions per second. These time-windows are important to evaluate how camera movement synchronises with the game rendering cycle, which is commonly executed between 20 and 60 Hz.

Camera positioning occurs in real-time and in response to users' actions; thereby the camera controller has to act rapidly so that the user does not perceive any delay between action and response. We believe that 200 ms is the maximum allowed time window in which the player does not perceive any camera positioning delay in a game. The time windows investigated here have been chosen to reflect this assumption.

C. Analysis

Table I contains the performance measures of average fitness and fitness standard deviation (within parentheses) for all four algorithms tested. Table columns present results obtained on the different testbeds ordered by complexity.

As a general observation all algorithms appear to perform equally or better when given more computational time. Another clear observation is that algorithms perform better at the easier tasks (e.g. Forest scene) than at the more complex tasks (e.g. Tunnels). The exception from this observation is the random search and the GA which both appear to perform better at the Tunnels scene than at the City scene. This exception is discussed at the dedicated GA and random search section that follows.

1) Octree-based Search: It appears that both octree-based search (OBS) algorithms (see Table I) generate high performances in low complexity testbeds, but their average fitness drops drastically as problem complexity increases. Both OBS algorithms perform poorly in complex scenes due to the high number of misleading local optima existent in the search space which guide the algorithm far from the global maximum. Despite of the deterministic nature of spatial subdivision, both algorithms show a little standard deviation within the experiment, this happens because the algorithms order randomly the nodes with equivalent fitness.

TABLE I: Average fitness

t	Forest	Building	City	Tunnels
Octree best-first search				
15ms	0.96 (0.01)	0.00 (0.00)	0.33 (0.00)	0.00 (0.00)
30ms	0.96 (0.01)	0.66 (0.11)	0.33 (0.00)	0.00 (0.00)
40ms	0.99 (0.01)	0.66 (0.00)	0.33 (0.00)	0.00 (0.00)
100ms	1 (0.00)	0.66 (0.00)	0.33 (0.00)	0.00 (0.00)
200ms	1 (0.00)	0.66 (0.00)	0.33 (0.00)	0.00 (0.00)
Octree breadth-first search				
15ms	0.96 (0.01)	0.00 (0.00)	0.33 (0.00)	0.00 (0.00)
30ms	0.96 (0.01)	0.49 (0.29)	0.33 (0.00)	0.00 (0.00)
40ms	0.99 (0.01)	0.66 (0.00)	0.33 (0.00)	0.00 (0.00)
100ms	1 (0.00)	0.66 (0.00)	0.33 (0.00)	0.00 (0.00)
200ms	1 (0.00)	0.66 (0.00)	0.33 (0.00)	0.00 (0.00)
Random search				
15ms	0.78 (0.13)	0.64 (0.09)	0.37 (0.08)	0.33 (0.23)
30ms	0.88 (0.10)	0.67 (0.04)	0.41 (0.08)	0.64 (0.14)
40ms	0.90 (0.08)	0.67 (0.05)	0.40 (0.10)	0.65 (0.14)
100ms	0.96 (0.04)	0.71 (0.09)	0.48 (0.07)	0.74 (0.13)
200ms	0.97 (0.02)	0.71 (0.09)	0.51 (0.06)	0.81 (0.15)
Genetic algorithm				
15ms	0.86 (0.10)	0.68 (0.08)	0.42 (0.09)	0.63 (0.15)
30ms	0.92 (0.07)	0.69 (0.08)	0.46 (0.08)	0.70 (0.14)
40ms	0.95 (0.04)	0.73 (0.11)	0.45 (0.08)	0.78 (0.14)
100ms	0.98 (0.02)	0.76 (0.12)	0.50 (0.06)	0.82 (0.15)
200ms	0.99 (0.02)	0.76 (0.12)	0.50 (0.07)	0.87 (0.14)

Results indicate that OBS algorithms are useful when a solution needs to be found rapidly in simple tasks (i.e. Forest scene) since the fitness they generate is significantly better

than the genetic and random search within the time-window of 15ms.

A t-test between OBS and random search (best-first p-value = 0.0; breadth-first p-value = 0.0) and between OBS and GA (best-first p-value = 2.7×10^{-11}; breadth-first p-value = 2.7×10^{-11}) reveal those statistically significant differences.

While performing well on fast and simple tasks, OBS algorithms perform poorly on typical occlusion minimisation problems like Building and City. Subsequently, our analysis will concentrate on the GA and random search approaches and their relationship since both appear to be more appropriate for the problem we investigate (see Table I).

2) GA versus Random Search: Figure 4 depicts the impact of time limit and the scene complexity on the average fitness of the GA and random search mechanisms. It appears that random search (see also Table I) reveals good performance in all scenes; however, as already mentioned, random search does not reach the performance of OBS approaches in the simple Forest scene. This result is somewhat expected since random search does not exploit any fitness information to identify the best areas to search within. The algorithm is performing very well in highly complex scenes such as City and Tunnels. On the contrary, the standard deviation of the fitness is the highest among all algorithms examined due to the stochastic nature of the algorithm.

The GA (see Fig. 4 and Table I) demonstrates efficiency (via average fitness performance) and robustness (via standard deviation of fitness) in all four testbeds and for all five time windows. It performs similarly to the octree-based algorithms (the standard deviation is higher due to the stochastic nature of the GA) in the Forest scene but it achieves consistently better performance in the other test-bed scenes for all the time limits.

Compared to random search, the GA performs significantly better (both in terms of average fitness and standard deviation) in the vast majority of scenarios independently of time window and problem complexity. The difference in terms of performance is more evident in the least complex test-bed while the performance of the two algorithms converges as complexity increases. A t-test between the two mechanisms shows a statistically significant difference of average fitness as the p-values calculated for the Forest, Building, and Tunnel scenes within the 200ms time window are, respectively 0.01, 0.01, and 0.0003. While the GA performs better than random search in the City scene given short time windows (i.e. 15ms – 40ms) the two mechanisms appear to perform similarly as time goes by (e.g. p-value for 200ms is 0.06).

It is also worth noticing that both mechanisms generate lower performance for the City scene than that of the Tunnels scene which is inconsistent with the complexity measures proposed. This suggests that other complexity measures that embed notions of multimodality could be more appropriate to classify scenes such as City. However, it should be noted that the c measure of complexity is consistent with OBS algorithm performance (see Table I).

TABLE II: Average time taken to find optimum (in ms)

t	Forest	Building	City	Tunnels
Random search				
15ms	10	15	13	6
30ms	19	25	19	12
40ms	14	40	29	15
100ms	38	62	47	63
200ms	124	82	72	103
Genetic algorithm				
15ms	11	7	8	7
30ms	18	22	15	11
40ms	13	21	18	21
100ms	42	60	28	65
200ms	126	68	32	77

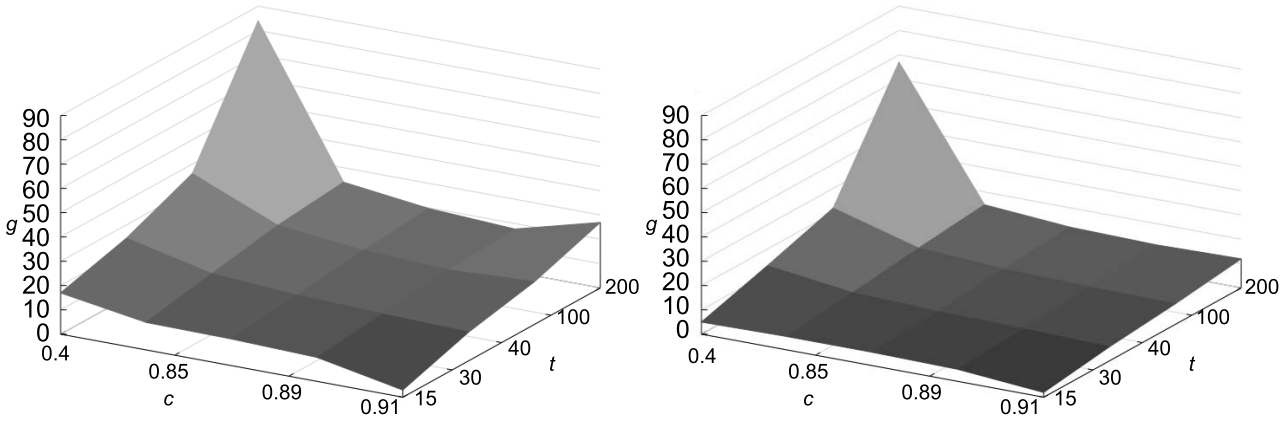
By observing the number of times the algorithms manage to find the global maximum (Fig. 5) and the average time it took them (Table II) it is clear that the GA is more efficient and faster in all scenarios examined while, in the City scene, it performs equally well with random search. As already mentioned, the City scene constitutes a rather hard problem for global search consisting of three global maxima and no area that provides full visibility of the three objects to the camera.

The performance obtained for the random search and the GA proposed reveals that both algorithms could be successful in camera positioning when objects are occluded. However, the GA is significantly better in low complexity case studies and showcases higher robustness and speed independently of complexity and time constraints. Such properties make it the most preferred algorithm among those examined in this study.

VI. Conclusions

This paper investigates search-based approaches for camera positioning in dynamic virtual environments. In particular we examine the problem of finding a non-occluded camera position for viewing multiple objects appearing in a 3D game environment. In game virtual words local search algorithms (e.g. artificial potential fields) fail evidently due to the complex search space landscapes generated by camera visibility measures. Fitness landscapes are generally multimodal consisting of zero gradient plateaus whose fitness distance correlation approximates zero.

Stochastic global search is expected to perform better in such fitness terrains and results obtained from this paper confirm our hypothesis. Four search algorithms (octree-based best-first, octree-based depth-first, random search and genetic search) are assessed in the comparative analysis presented. Algorithm performance is evaluated across dissimilar tasks varying in complexity and real-time taken to achieve the corresponding performance. Results suggest that the proposed genetic algorithm is significantly better than octree search in all case studies examined. Moreover, the GA outperforms



(a) Genetic algorithm

(b) Random search

Fig. 5: Number of times the global maximum was found, g, versus complexity, c, and time, t (in ms).

random search in problems of low complexity; however, the two algorithms generate similar performances in some tasks of high complication. The GA proposed is consistently the fastest, most robust and effective approach, from those investigated, to the problem of maximising object visibility.

As a future research step, alternative genetic search algorithms, suitable for constraint satisfaction problems, will be examined and compared with the proposed GA; the Feasible Infeasible Two-Population (FI 2-Pop) genetic algorithm [20] is a sensible choice to make with that regard. Another obvious future step of this research is the design of algorithms that will efficiently generate the shortest and/or most *scenic* path to the non-occluded position found by the search algorithm. Intelligent camera occlusion detection and avoidance will collectively advance the current state-of-the-art in camera control systems in games and enhance player experience during gameplay.

ACKNOWLEDGEMENTS

The authors would like to thank Julian Togelius for insightful discussions. The research was supported in part by the Danish Research Agency, Ministry of Science, Technology and Innovation; project name: *AGameComIn*; project number: 274-09-0083.

REFERENCES

[1] Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[2] Daniel Arijon. *Grammar of the Film Language*. Silman-James Press LA, 1991.

[3] William Bares, Scott McDermott, Christina Boudreaux, and Somying Thainimit. Virtual 3d camera composition from frame constraints. In *MULTIMEDIA '00*, pages 177–186. ACM, 2000.

[4] William H. Bares and James C. Lester. Intelligent multi-shot visualization interfaces for dynamic 3d worlds. In *IUI '99*, pages 119–126. ACM, 1999.

[5] Steffi Beckhaus, Felix Ritter, and Thomas Strothotte. Cubicalpath - dynamic potential fields for guided exploration in virtual environments. In *PG '00*. IEEE Computer Society, 2000.

[6] James Blinn. Where am i? what am i looking at? *IEEE Comput. Graph. Appl.*, 8(4):76–81, 1988.

[7] Owen Bourne, Abdul Sattar, and Scott Goodwin. A constraint-based autonomous 3d camera system. *Constraints*, 13(1-2):180–205, 2008.

[8] Paolo Burelli, Luca Di Gaspero, Andrea Ermetici, and Roberto Ranon. Virtual camera composition with particle swarm optimization. In *Smart Graphics*, pages 130–141. Springer-Verlag, 2008.

[9] Paolo Burelli and Arnav Jhala. Dynamic artificial potential fields for autonomous camera control. In *Artificial Intelligence and Interactive Digital Entertainment*, 2009.

[10] Marc Christie and Jean-Marie Normand. A semantic space partitioning approach to virtual camera composition. *Computer Graphics Forum*, 24(3):247–256, 2005.

[11] Marc Christie and Patrick Olivier. Camera Control in Computer Graphics. pages 89–113. Eurographics Association, 2006.

[12] Steven M. Drucker and David Zeltzer. Intelligent camera control in a virtual environment. In *Graphics Interface 94*, pages 190–199, 1994.

[13] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *Computer Graphics*, pages 331–340, 1992.

[14] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[15] Nick Halper and Patrick Olivier. Camplan: A camera planning agent. In *Smart Graphics 2000 AAAI Spring Symposium*, pages 92–100, 2000.

[16] Nicolas Halper, Ralf Helbing, and Thomas Strothotte. A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence, 2001.

[17] Frank Jardillier and Eric Languènou. Screen-space constraints for camera movements: the virtual cameraman. *Computer Graphics Forum*, 17(3):175–186, 1998.

[18] Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[19] O Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, 1986.

[20] Orla Kimbrough, David Harlan Wood, D.H. Wood, and Ming Lu. Introducing distance tracing of evolutionary dynamics in a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization, 2004.

[21] Erik Marchand and Nicolas Courty. Controlling a camera in a virtual environment. *The Visual Computer Journal*, 18:1–19, 2002.

[22] Jonathan Pickering. *Intelligent Camera Planning for Computer Graphics*. PhD thesis, University of York, 2002.

[23] D. Pinelle and N. Wong. Heuristic evaluation for games: usability principles for video game design. In *CHI'08: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1453–1462, New York, NY, USA, 2008. ACM.

[24] Colin Ware and Steven Osborne. Exploration and virtual camera control in virtual three dimensional environments. *SIGGRAPH*, 24(2):175–183, 1990.