Combining Local and Global Optimisation for Virtual Camera Control

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

***Abstract*— Controlling a virtual camera in 3D computer games is a complex task. The camera is required to react to dynamically changing environments and produce high quality visual results and smooth animations. This paper proposes an approach that combines local and global search to solve the virtual camera control problem. The automatic camera control problem is described and it is decomposed into sub-problems; then a hierarchical architecture that solves each sub-problem using the most appropriate optimisation technique is proposed. The approach is compared to pure local search solutions to showcase the advantages of the proposed architecture in terms of visual performance and robustness.**

I. Introduction

Camera control has a deep impact on player experience and enjoyability in games [24]. The camera represents the point-of-view of the player through which she perceives the game world and gets feedback on her actions.

Camera settings for games are usually directly controlled by the player or statically predefined by designers. Direct control of the camera by the player increases the complexity of the gameplay interaction and reduces the designer's control on game storytelling (e.g. the player might point the camera towards an area which reveals unwanted information). On the other hand, a designer-driven camera control releases the player from the burden of controlling the point of view, but often generates undesired camera behaviours (e.g. the player is hidden behind an object). Moreover, if the content of the game is procedurally generated, the designer might not have any information to define a priori the camera positions and movements.

Automatic camera control aims to define an abstraction layer that permits the designers to instruct the camera with high-level and environment-independent rules. The camera controller should dynamically and efficiently translate these rules into camera movements while the player plays the game.

Several techniques for automatic camera control have been proposed in the past — the reader is referred to [12] for a comprehensive review. The most common approaches model the camera control problem as a constraint satisfaction or optimisation problem. These approaches allow the designer to define a set of requirements on the frames that the camera should produce and on the camera motion. Depending on the approach, the controller positions and animates one or more virtual cameras that attempt to satisfy the predefined requirements.

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

Finding the best camera positions and movements that satisfy the designer's requirements in a dynamic three dimensional environment is a complex task. Evaluation of frame properties, such as object visibility, is computationally expensive and the evaluation functions often generate terrains that are very rough for a search algorithm to explore [10]. Moreover, the camera is required to react in real-time at dynamic change of the environment or player action which forces the computation time to be constrained within small time intervals (e.g. 16.7 ms for 60 fps).

Pure global optimisation approaches [23], [11], [8], [17] are capable of producing well composed shots with respect to designer requirements. However, their high computational cost makes them inappropriate for real-time interactive applications such as games.

Local search approaches [4], [6], [9], on the other hand, offer real-time performance and allow the designer to control also camera motion parameters such as speed and acceleration but they tend to stick to local optimum solutions.

This paper proposes an approach to the automatic camera control problem that employs local search, global search and path planning to generate smooth camera animations and well composed shots. The camera control problem is decomposed into smaller tasks and different techniques are used to perform different tasks overcoming the limitations of the pure approaches.

The approach proposed extends and draws upon one of the author's earlier work on local search based camera control [9] and on visibility optimisation [10]. The new camera controller is built upon a layer of three components: the local search algorithm proposed in [9] is used to find the best camera configuration; a stochastic population-based global search algorithm is employed to avoid premature convergences to local minima and a real-time efficient path planning algorithm is designed to generate smooth camera animations and to control camera movements.

The *CamOn* camera control system, that embeds all three modules, is evaluated through three case studies varying in complexity. Results show that the approach proposed demonstrates robustness and high visual performance across all the three case studies. It is also apparent that the computational cost of this combined approach does not differ significantly from pure local search, but the visual performance, both in terms of composition and animation, is significantly higher.

This paper is innovative in that it introduces an efficient and reliable hybrid solution to the automatic camera control problem coupling local with global search. The proposed approach successfully combines different optimisation methods

978-1-4244-6297-1/10/$26.00 ©2010 IEEE

thereby fully exploiting their combined advantages which, in turn, help towards avoiding algorithm-dependent limitations.

II. Related work

Automatically controlling the camera in virtual 3D dynamic environments is an open research problem. Earliest approaches [25], [5], [15] focused on the definition of virtual camera properties and investigated the mapping between input devices and 3D camera movement. Direct control of the several degrees of freedom of the camera has shown to be problematic for the user [14] so researchers started to investigate how to automatically place and configure the camera. Christie and Olivier [12] classified different approaches to automatic camera control into three main categories according to the modelling algorithm: algebraic systems, reactive systems and generalised approaches.

The first example of an algebraic camera control system, was developed by Blinn in 1988; it was an automatic camera control system for planet visualisation in a space simulation at NASA [5]. Blinn's work inspired many other researchers trying to produce more flexible camera control systems and attempts to integrate aspects like camera motion and frame composition [1].

Gleicher and Witkin proposed a reactive technique inspired by visual-servoing called Through-the-lens camera control [15], this technique permits the user to manipulate the camera by constraining some projected image features. Through-the-lens based systems are computationally efficient so they are ideal for tasks such as object manipulation; their aim, however, is to maintain specific image features (i.e. keep an object in the centre of the screen) and require a preliminary camera initialisation.

A. Generalised Approaches

Generalised approaches model the camera control problem as a constraint satisfaction or optimisation problem. These approaches require the designer to define a set of required frame properties and camera motion properties. The defined properties 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. Bares et al. [2] first introduced a detailed definition of these properties and how to evaluate them.

Jardillier and Langunou [19] developed a first hard-constraint based system; their approach progressively prunes the search space for each constraint until the solution space is found. Although computationally efficient, this system is not able to find any solution in case of conflicting constraints (i.e. no solution satisfies all the constraints). Bares and Lester [3] addressed this issue by identifying conflicting constraints and produce multiple camera configurations corresponding to the minimum number of non-conflicting subsets. Bourne and Sattar [6] extended their solution by adding a *weight* property to each constraint to define a relaxation priority. Other researchers [23], [11], [8] combined constraint satisfaction with global optimisation. According to that approach, hard constraints are used to select a feasible solution volume (therefore reduce the size of the search space) and global optimisation is then applied to find the best camera configuration within this space. Pure global optimisation based systems, like Halper and Olivier's CAMPLAN [17], demonstrate more reliable performance and guarantee to find the best camera configuration for the given requirements but their computational cost is high.

Due to the complexity of the fitness function evaluation global search based approaches have proven to be unsuitable for real-time 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, since the camera needs to maintain frame coherence [18], the best camera configuration might not always correspond to the global optimum of the fitness function (i.e. the camera should not jump every time the global optimum changes). Using local search to optimise the function permits to avoid this kind of behaviour and to control the camera motion. Beckhaus et al. [4] first introduced local search algorithms to camera control; their system employs Artificial Potential Fields to generate collision-free camera paths through a virtual environment. Bourne and Sattar [7] proposed a system that employs 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 tracking.

Local search approaches offer reasonable real-time performance and permits the control of dynamic camera parameters such as speed and acceleration but often converge prematurely to local optima. Premature convergence becomes critical when the camera control system is required to optimise visibility of *objects of interest* since the visibility fitness landscape includes many local optima areas with almost no gradient information available to guide local search away from local visibility minima [10].

B. Occlusion

Successfully avoiding object occlusion constitutes a vital component of an effective camera controller [12]. Object visibility plays a key role in frame composition: 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 occlusion problem consists of two dependent tasks: occlusion evaluation/detection and occlusion minimisation/avoidance. Occlusion happens when the object of interest is hidden fully or partially by one or more other objects.

A common technique to detect occlusion consists of casting a series of rays between the object of interest and the camera [8], [7]. Marchand and Courty [22] generate a bounding volume containing both the camera and the object of interest and check whether other objects intersect this volume. A third approach [18] exploits the graphic hardware by rendering the scene at a low resolution with a colour associated with each object and checking the presence

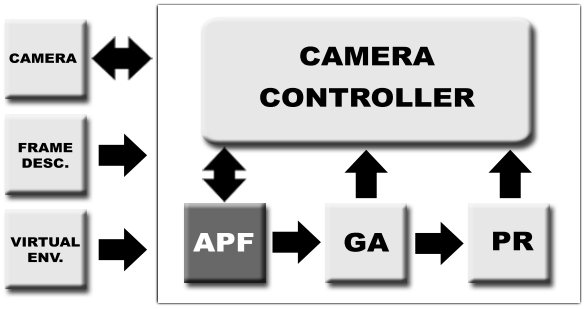


Fig. 1. The CamOn architecture

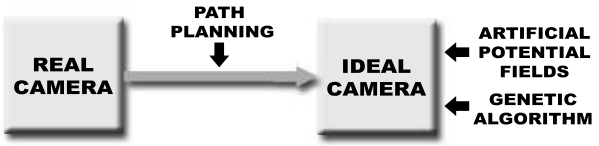


Fig. 2. Ideal and real camera

of the colour associated to the object of interest. These techniques have been used in local search approaches for occlusion avoidance (i.e. maintaining an object visible in the frame) by reacting to incoming occluders [7], and in global search approaches for occlusion minimisation through particle swarm optimisation [8].

Pickering [23] proposed a shadow-volume occlusion avoidance algorithm where the object of interest is modelled as a light emitter and all the non-illuminated areas of the scene generated are considered occluded areas.

Occlusion minimisation approaches suffer from the same limitations global search approaches do. Moreover, occlusion avoidance is ineffective when the subject is fully occluded and no local improvement to visibility is available. 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 jumps within the game environment and is unable to handle more than one object of interest.

We propose an approach to camera control that combines global search, local search and path planning that overcomes the limitations of both local and global search with respect to camera control.

III. Camera Control System

CamOn is an autonomous camera system capable of generating smooth camera animations and solving camera composition tasks. We extend our previous implementation [9] by embedding it into a high level architecture (Fig. 1) that combines local search through an Artificial Potential Fields (APF), global search through a custom-designed Evolutionary Algorithm (EA) and path planning implemented using Probabilistic Roadmaps (PR).

CamOn animates two cameras simultaneously: the *real camera* and the *ideal camera* (Fig. 2). The *ideal camera* is the result of the optimisation process and the *real camera* is the one that is used for the rendering and follows the *ideal camera* at a controlled speed which depends on the desired motion camera motion properties.

The system iteratively animates the two cameras and at each iteration it performs the following steps:

1) CamOn takes the current camera configuration, the frame description and the current virtual environment state as input.
2) It calculates and sets the *ideal camera* orientation.
3) It calculates the new *ideal camera* position through local optimisation.
4) If the camera position converged to a local optimum or the current number of visible objects is less than a user defined threshold, a new position is calculated through global optimisation.
5) An available path connecting the current *real camera* and *ideal camera positions* is generated.
6) If a path is found in step (5), *real camera* position and rotation are animated towards the *ideal camera* ones. If such a path is not available the *real camera* is placed and rotated like the ideal one.
7) CamOn returns the *real camera* position and rotation as output.

The *ideal camera* orientation is procedurally calculated and depends on the desired visible subjects and their desired position on the screen. The *real camera* rotates to match the *ideal camera* orientation at the desired rotation speed.

Regarding the position of the camera, the *ideal camera* is animated through APF to optimise the given frame requirements; if a local optimum is detected by the camera controller, a position that maximises the targets visibility is derived through a custom EA. The result of the global search will also be utilised to identify the targets which cannot be included in the frame (i.e. all the targets which are not visible from the position calculated by the global search module). These targets will not be considered any more as targets until new global search is required and their properties will not be take into account by the local search module.

CamOn constantly computes the shortest path between the two cameras' positions, and animates the *real camera* through this path. If no path is available or the current number of visible objects is less than a user defined threshold, the *real camera* is forced to jump to the ideal position.

A. Camera Properties and Fitness Function

CamOn allows the designer to control the camera through a set of properties; these properties can be separated in two groups: *camera motion properties* and *frame composition properties*.

Motion properties control the camera motion dynamics. CamOn currently supports the following properties:

- **Camera Movement Speed**: Defines the speed in space units per second at which the *real camera* moves to follow the *ideal camera* position.
- **Camera Rotation Speed**: Defines the speed in degrees per second at which the *real camera* rotates to match the *ideal camera* rotation.
- **Frame Coherence**: Defines the threshold value (minimum percentage of visible targets) for triggering global search — i.e. if the current visible surface of all the targets is below the fraction defined by frame coherence, CamOn will look for a new position through global search.
- **Obstacle Avoidance**: A predefined boolean value that controls whether the camera should or should not avoid the objects in the scene during its motion.

Frame composition properties describe the disposition of the visual elements in the image [1]; following the model proposed by Bares at al. [2] we have defined a set of properties each of which may be applied to an object of interest for the camera.

CamOn supports the following four frame composition properties.

- **Object Visibility**: Defines whether an object (or a part of it) should be visible in the frame. Object Visibility is a composed property which affects both the camera position to avoid occlusion and camera orientation to include the object in the frame.
- **Object Projection Size**: Defines the size of object in the frame; size is defined as the quotient between frame height or width and the relative longest side of the object's projected bounding box.
- **Object View Angle**: Defines the angle from which the camera should view the object. The view angle is defined using spherical coordinates.
- **Object Frame Position**: Defines the cell position (within a 3x3 grid) that the projected image of the object should have in the frame.

For an in-depth description of the frame composition properties and the way they are evaluated the reader is referred to our previous study on Artificial Potential Fields based camera control [9].

Camera motion properties are used by the camera controller to animate the *real camera* towards the *ideal camera* position. This position is calculated by optimising a fitness function which is proportional to the satisfaction level of the required frame composition properties. Given that N is the number of composition properties used to describe the frame and c_i is the satisfaction value of the i_{th} property, the fitness f is calculated as:

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

where w_i is the predefined *importance weight* of the i_{th} property. While the APF based local search attempts to maximise this fitness function, when the camera controller requires a global search for a new camera position another fitness function is optimised. The fitness function used for global search considers only *Object Visibility* properties of eq. 1, importance weight values are zero for all other properties.

B. Camera Orientation

Two of the previously mentioned composition properties contribute to the *ideal camera* orientation: Object Visibility and Object Frame Position. Each of these properties defines an ideal camera look-at point (the position which the camera should look at) the camera orients towards the centre of mass of these points. Given that V equals to the number of Object Visibility and Object Frame Position properties and $\vec{l_i}$ equals to the ideal look-at point of the i_{th} property, the resulting look-at position $\vec{L}$ is defined in eq. 2.

$$\vec{L} = \frac{\sum_{i=0}^{V} \vec{l_i} m_i}{\sum_{i=0}^{V} m_i} \tag{2}$$

The mass of each point m_i is defined as $m_i = w_i \times r_i$ where r_i is the radius of the object bounding sphere.

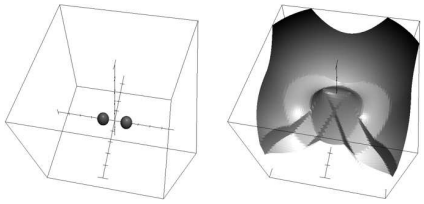
C. Local Optimisation

The local optimisation module is based on APF [9]: frame composition properties are used to generate the potential field and the camera position is animated towards the optimal position. Artificial Potential Fields [20] is a local search method commonly employed in the area of robotics to control the navigation of robots in dynamic environments. Robots are modelled as particles moving in a field of potentials attracted by low potential areas and repulsed by high potential areas; the goal position thus generates an attractive force (a low potential zone) and obstacles generate repulsive forces (high potential zones). At each iteration the particle moves along the force resulting from the sum of all the repulsive and attractive forces influencing current particle position; the particle continues to move until it reaches a stable state.

Each frame constraint produces one force attracting or repulsing camera position.

An example of the potential field generated by a combination of two *Object Visibility* and two *Object Projection Size* properties can be seen in Fig. 3. In this example two objects of interest (spheres) have to be fully visible while their projected image should cover half of the screen and the shot needs to be taken from the front. The potential field shown is a sample of the 3D field measured along the horizontal plane passing through the spheres centre, the low (light) areas are attractive positions while the high (dark) areas are repulsive.

Converting frame composition properties to APF requires the identification of the position goals corresponding to each property. Ideal camera positions for each property are modelled as low potential zones; other parts of the search space have a potential proportional (the exact relation varies between properties) to the distance from the ideal position and to the property satisfaction of the corresponding camera position [9].



(a) Spheres (b) Potential Field

Fig. 3. Example potential field produced for a front shot of two spheres

The net force value at each point equals to the weighted sum of all the forces produced by the different composition properties. Each weight is the predefined importance of the corresponding property, as previously seen in eq. 1.

D. Global Optimisation

Visibility fitness generates rough search spaces mostly covered by fitness plateaus [10]; gradient based search is bound to fail in optimizing such a fitness function, making the camera unable to escape these plateaus. When the camera is positioned in a visibility plateau, one or more subjects are potentially non visible preventing any desired visual information about that subject to be communicated to the player. In order to escape such local optima the camera control system employs global optimisation to find a better solution when a complete occlusion is detected.

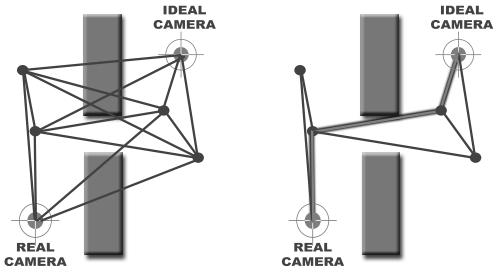
Our previous study on visibility optimisation using global optimisation [10] analysed the performance of different algorithms over different case studies and a custom designed EA demonstrated the highest performances on all the tests performed. For this paper, we also considered the use of the FI-2POP EA algorithm [21] as an alternative global search method but it demonstrated no performance improvement when compared to the performance of the custom-designed EA introduced in [10]. Thus, we decided to adopt the aforementioned EA as our global optimiser for CamOn. The algorithm is presented briefly below.

According to this modified version of a generational EA [16], each chromosome represents a point in the 3D solution space consisting of three real values. The population, containing 120 individuals, is randomly placed into the search space via a Gaussian distribution and evaluated via (1) at each generation. The search space is defined as a box surrounding the target objects. The size s of this cube is calculated as:

$$s = \max_{i=0}^{T}\{\vec{P}_i - \vec{C}\} \tag{3}$$

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

The algorithm terminates if a camera position found generates an optimal fitness value ($f = 1.0$) or when a predefined timeout is reached (the timeout threshold used in this paper equals 200 ms).



(a) Graph generation (b) Shortest path

Fig. 4. Probabilistic Roadmaps

The algorithm employs a selection scheme in which the 30 best chromosomes of each generation are selected for reproduction. Mating of parents is based on their ranking; the first individual mates with the second, the third mates with the forth and so on. From each couple 3 offspring are generated by applying a custom-designed recombination operator (with 100% probability). The recombination operator applied is a fitness-based weighted sum of the parents' position. The generated offspring $\vec{O}$ is calculated via eq. 4.

$$\vec{O} = \frac{\vec{P}_a f_a + \vec{P}_b f_b}{f_a + f_b} \tag{4}$$

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

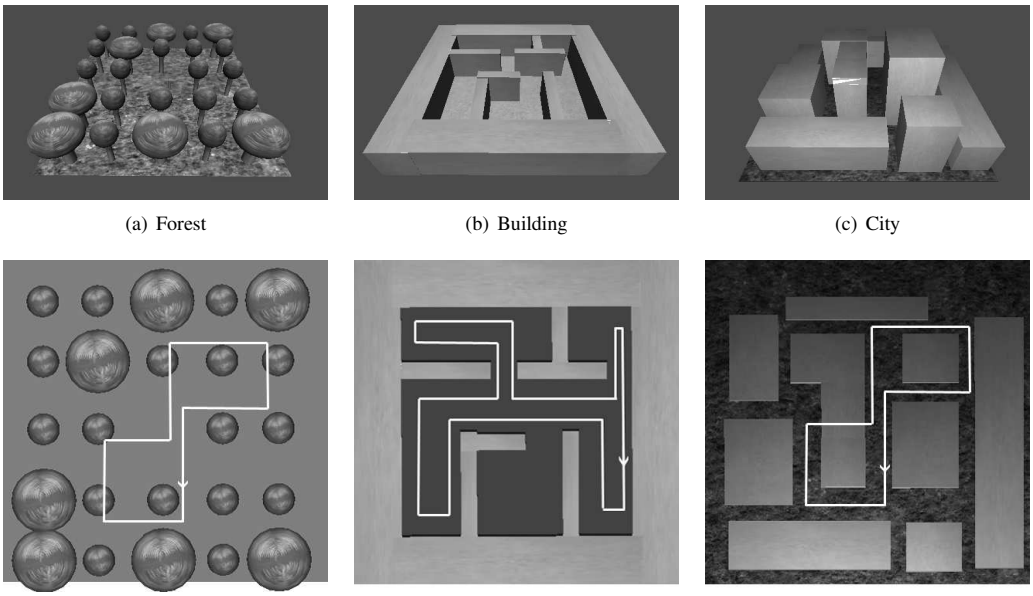
Mutation is applied to all genes of the chromosome with a probability of 50%. The custom mutation operator applies a vector translation to the chromosome by adding a vector with normally distributed random length to it. The average value of the distribution is 0 and the standard deviation equals to the 10% of the length of the vector represented in the chromosome to which the mutation is applied. Generated offspring replace the 90 least-fit chromosomes of the current population.

E. Path Planning

At each iteration, CamOn looks for a valid path that connects the current *real camera* position with the current *ideal camera* position. If such path is available the *real camera* is translated along this path. If the path is not available (e.g. the *real camera* is in a closed room) the *real camera* is translated directly to the *ideal camera position* generating a jump. The path between the two cameras is calculated using Probabilistic Roadmaps (PR) [13].

According to the PR approach, a set of random (normally distributed) points is generated in the space between the two camera positions; the mean of the mixed-Gaussian distribution is located halfway between the two positions and its standard deviation equals to half of their distance.

All the generated points as well as the *real camera* position and the *ideal camera position* define the nodes of a graph which is fully interconnected. All the point-connectors which cross a virtual object are removed and the shortest path



(a) Forest (b) Building (c) City

(d) Forest Animation Path (e) Building Animation Path (f) City Animation Path

Fig. 5. Case Studies

between the node containing the *real camera* position and the one containing the *ideal camera* position is calculated (see Fig. 4) and used to move the *real camera*.

IV. Performance Evaluation

Our previous study [9] showed that APF based camera control is capable of generating smooth camera animations and well composed shots with real-time performance. The evaluation performed in that study directly applies to the proposed solution since the integration of global optimisation and path planning does not influence the computational efficiency. However, the visual performance, in terms of composition and camera animation, could potentially differ significantly. In order to assess the degree of those performance differences we designed an experiment that measures the camera visual quality in a set of game-like virtual environments; both the experiment designed and the case studies examined are presented in the remainder of this section.

A. Experiment

The camera control system is instructed to follow and keep a target visible while it moves around a virtual environment. The camera control profile includes the following visual properties:

- **Object Visibility** for the full object's bounding box with corresponding importance weight value of 1.0.
- **Object Projection Size** with expected projection size equal to 1.0 and corresponding importance weight value of 1.0.
- **Object View Angle** with expected horizontal and vertical angle equal to 0 degrees and corresponding importance weight value of 0.1.

The camera movement and rotation speed are set to 1.0 and frame coherence is set to 100%. The target is a cylindrical object that moves at variable speed along a predefined path, illustrated in Fig. 5(d), 5(e), 5(f). The object's movement speed varies between 0.1 and 2 meters per second and each experiment runs for 60 seconds.

Three different configurations of the camera control system are evaluated to compare their performance. The configurations include a pure APF based implementation of the camera control system, a combined APF-EA implementation and an implementation embedding all three main components of the camera controller (i.e. APF, EA and PR).

B. Case Studies

Each configuration is evaluated on three virtual environments, namely, *Forest*, *Building*, and *City*, illustrated in Fig. 5(a), Fig. 5(b) and Fig. 5(c). These three virtual environments represent a wide range of level geometry met in computer games; furthermore they provide a palette of testbeds with different search space complexity. The geometric structure of these environments shows evident differences which influence the complexity of the search space. An effective complexity measure for this problem is based on the average fitness value of the sampled search space [10]. According to this complexity measure ($c = 1 - \bar{f}$) the environments have the following c values: *Forest* 0.35, *Building* 0.76, *City* 0.85.

The first and least complex environment (Fig. 5(a)) represents a forest scene in which the target is surrounded by trees which act as occluders. Due to the narrow shape of the trees the portions of space in which the camera is fully occluded are very few and most of the search space contains valid camera positions. The second environment

(Fig. 5(b)) is a house-like model with walls separating the space into rooms and doors connecting them. The walls act as large occluders drastically reducing the portions of space containing potential optimal positions. The last and most complex virtual environment is a city model with large buildings and narrow streets. The target object selected for the experiment represents a virtual character of realistic human-like dimensions. It is modelled using a cylindrical mesh 2 meters tall and 50 cm wide, while all levels are designed in an area of 17m $\times$ 17m (the ceiling in the Building level is 4 meters tall).

V. Results

This section presents the results of the performance tests for the three camera control system configurations mentioned earlier in section IV. The performance is analysed with respect to the complexity of the task presented to the CamOn system configuration. All experiments run on an 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 visibility fitness evaluations per second, but the average value (due to the system scheduler) is around 3750 evaluations per second. Moreover, the experiments run at a fixed frame-rate of 30 frames per second, and the APF solver runs once per frame.

A. Performance Measures

Each test has been carried out for 60 seconds for each case-study and each configuration examined. The average visibility over time is 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 *real camera* jumps to a new position and the average fitness over time. The average visibility and fitness provide a measure of the ability of the system to generate high quality frames, and the number of jumps gives an indication of the system's ability to generate smooth camera animations.

B. Analysis

Tables I, II and III contain, respectively, the performance measures of average visibility, average fitness and number of jumps for all three configurations tested. Table rows present results obtained on the different testbeds ordered by complexity.

TABLE I
Average Visibility

	APF	APF, EA	APF, EA, PR
Forest	0.97	0.97	0.97
Building	0.90	0.97	0.96
City	0.75	0.97	0.95

A clear observation is that all configurations perform better at the easier tasks (e.g. Forest environment) than at the more complex tasks (e.g. City).

In terms of average visibility the three configurations perform equally in the Forest environment. In the Building environment the APF-based controller has an evident drop of the visibility value which deteriorates further in the City environment (see table I). The average visibility decreases since, in the Building and City environments, the occluders are much larger than in the Forest environment, which results to increased times spans for the APF to locate the target again after a full occlusion.

TABLE II
Average Fitness

	APF	APF, EA	APF, EA, PR
Forest	0.82	0.79	0.79
Building	0.76	0.79	0.78
City	0.67	0.78	0.78

The other two configurations show almost equal average visibility on all the test environments; however the configuration which does not include the path planning module has slightly higher average visibility since the camera does not spend time to move to the new unoccluded position found using global search, but it jumps directly there. The path covered by the camera to reach the global visibility optimum does not consider any fitness thus the time spent for animation slightly decreases both the average fitness and the average visibility.

It should be remembered that the evolutionary algorithm does not consider the full fitness function given by eq. 1 but only its *Object Visibility* properties. This explains why the average fitness is slightly higher for the pure local optimisation approach than the average fitness of the other configurations in the Forest environment. The average fitness decreases (as the average visibility does) when the complexity of the task increases, making CamOn configurations that include the EA component more appropriate with regards to this performance measure (see Table II).

TABLE III
Total Number of Jumps

	APF	APF, EA	APF, EA, PR
Forest	0	9	1
Building	0	11	3
City	0	14	2

The gradient search of APF generates smooth camera animations in its attempt to find the ideal camera position; this results to a non jumping camera behaviour when attempting to follow a moving target. On the other hand, running global search every time a local optimum is found, forces

the camera to jump to the global position fragmenting the camera movement (see Table III). The number of jumps generated by the camera controller which integrates directly local and global optimisation is constantly higher than 0 and increases with respect to task complexity. Introducing a path planner to animate the camera to the position found by global search sensibly decreases the number of jumps significantly and makes the camera movement smoother. The controller still makes the camera jump a number of times; this happens when the path finding algorithm is unable to find a path between the *real camera* and the *ideal camera*.

VI. Conclusions

This paper proposes a hybrid approach to automatic camera control that combines local optimisation, global optimisation and path planning. We describe the limitations of the single approaches and we show how each approach can be used to tackle a specific sub-problem and how they can be combined under a camera control system we name CamOn.

The proposed solution employs Artificial Potential Fields to generate smooth camera animation towards the optimal camera position. If the local search converges prematurely to a local optimum (i.e. a position where one or more subjects are completely occluded) the system uses a custom designed Evolutionary Algorithm to find the position which provides the best visibility over the objects of interest. Probabilistic Roadmaps are used to compute the path between the current camera position and the new one found either using local or global search.

We evaluated the solution's performance through an experiment that measures the camera visual quality in a set of game-like virtual environments. The results of the evaluation show that the system demonstrates robustness and is able to generate high quality shots and smooth camera movements in all the three case studies.

In the least complex case study the proposed hybrid approach shows similar performance to pure local search. However, the other two case studies show a progressively higher difference between the performances, with the hybrid approach outperforming pure local search.

The current approach allows control of camera motion within limits. In particular, apart from camera speed and frame coherence, no other parameters are available to influence the camera trajectories (e.g. motion smoothness, animation length). Moreover, a better local minimum detection heuristic could be implemented to reduce the number of unnecessary global searches.

As a future research step, the impact of automatic camera control to the player's experience should be investigated; a series of experiments involving human players should be designed and performed to measure how different camera control paradigms affect the various aspects of the player's interaction.

Furthermore, a study on what the player is looking at while he is playing and how this information relates to the different player types and games, might provide important insight for the design of a set of standard camera controller settings. This information could also be used to adapt dynamically these standard settings to each player according to his playing style.

References

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

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

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

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

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

[6] Owen Bourne and Abdul Sattar. Applying constraint weighting to autonomous camera control. In *AIIDE '05*, pages 3–8, 2005.

[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] Paolo Burelli and Georgios N. Yannakakis. Global search for occlusion minimization in virtual camera control. In *Proceedings of the Congress on Evolutionary Computation*, WCCI 2010.

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

[12] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera Control in Computer Graphics. In *Computer Graphics Forum*, 2008.

[13] Jean claude Latombe, Petr Svestka, Mark Overmars, Lydia Kavraki, and Lydia Kavraki. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *In IEEE International Conference on Robotics and Automation*, page 171, 1994.

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

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

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

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

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

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

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

[21] Steven Orla Kimbrough, Gary J. Koehler, Ming Lu, and David Harlan Wood. On a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research*, 190(2):310 – 327, 2008.

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

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

[24] 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.

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