A Benchmark for Virtual Camera Control

Paolo Burelli and Georgios N. Yannakakis

Department of Architecture, Design and Media Technology,
Aalborg University Copenhagen, Denmark and
Institute of Digital Games, University of Malta, Malta
Email: pabu@create.aau.dk, georgios.yannakakis@um.edu.mt

Abstract. Automatically animating and placing the virtual camera in a dynamic environment is a challenging task. The camera is expected to maximise and maintain a set of properties — i.e. visual composition — while smoothly moving through the environment and avoiding obstacles. A large number of different solutions to the problem have been proposed so far including, for instance, evolutionary techniques, swarm intelligence or ad hoc solutions. However, the large diversity of the solutions and the lack of a common benchmark, made any comparative analysis of the different solutions extremely difficult. For this reason, in this paper, we propose a benchmark for the problem of virtual camera control and we analyse a number of different problems in different virtual environments. Each of these scenarios is described through a set of complexity measures and, as a result of this analysis, a subset of scenarios is selected as the core of the benchmark.

1 Introduction

The virtual camera, in an interactive 3D environment, represents the point of view of the user and it deeply influences her way to perceive the environment and her ability to effectively accomplish any task. In applications such as 3D modellers and computer games, the virtual camera provides a mean for interacting with the virtual environment and has a large impact on the usability and the overall interactive experience [1]. Moreover, in 3D computer games the presentation of the game events largely depends on the camera position and movements; thus, virtual camera control has a significant impact on aspects such as gameplay and storytelling.

What is the optimal way of controlling the virtual camera is a an open research question: on one side of the control spectrum, direct control of the camera by the player increases the complexity of the interaction and eliminates the designer's control over game storytelling; on the other side, statically predesigned camera animations release the player from the burden of controlling the point of view, but they cannot guarantee a correct visualisation of all possible player actions. Furthermore, such approach is not applicable in multi-player games or in games where the content is procedurally generated as the designer has potentially no information a-priori to define the camera positions and movements.

Automatic virtual camera control aims at studying effective and efficient methods to control the camera behaviour through high-level and environment-independent requirements, such as the visibility of a particular object or the size of that object on the screen.

Based on these requirements the method should dynamically and efficiently calculate a near optimal (or even optimal) path for the camera. Over the years a number of different approaches has been proposed, ranging from early ad hoc solutions [2] to more recent and flexible approaches [3], which tackle the task as an optimisation problem and apply different variations of search algorithms. However, while it is intuitively possible to detect a gradual improvement in the efficiency and effectiveness of the algorithms proposed, it is still not possible to precisely compare the algorithms and quantify the results achieved.

Contrarily to other fields, such as computer vision [4] or optimisation [5], in virtual camera control, there is a lack of a common well defined benchmark problem. As a result all researchers that proposed a contribution to the field had to design custom test problems in custom virtual environments, making a comparison between experiments and findings nearly impossible. For this reason, in this article, we present an analysis of the virtual camera control problem from an optimisation perspective and, based on this analysis, we introduce a set of test problems representing different challenges commonly met in virtual camera composition applications such as computer games. Each problem is evaluated according to a number of metrics about optimisation complexity and, from the results of this evaluation, a smaller subset of the most representative problems is selected to compose the benchmark. Finally, in an attempt to make the benchmark easily accessible and testable, we released it as an open source project on GitHub[1]. The project, along with the analysis presented in this paper, is intended as a starting point towards a normalisation of the evaluation practices in virtual camera control.

2 Background

With the introduction of three-dimensional computer graphics, virtual camera control immediately appeared as a challenging task. The first studies on virtual camera [6] focused primarily on designing interaction metaphors for manual camera control. However, direct control of the several degrees of freedom of the camera proved often to be problematic for the user [7], leading researchers to investigate the automation of camera control.

In 1988, Blinn [2] showcased one of the first examples of an automatic camera control system. Blinn designed a system to automatically generate views of planets in a space simulator of NASA. Although limited in its expressiveness and flexibility, Blinn's work inspired many other researchers trying to investigate efficient methods and more flexible mathematical models able to handle more complex aspects such as camera motion and frame composition [8]. More generic approaches model camera control as a constraint satisfaction problem. These approaches require the designer to define a set of desired 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 constraints describe how the frame should look like in terms of object size, visibility and positioning. Bares et al. [9] presented a detailed definition of

[1] Available at: https://github.com/paoloburelli/VirtualCameraSandbox

these constraints, which became the standard input of most automatic camera control methods.

Since Bares' definition, the problem of finding one or more camera configurations that satisfy a given set frame constraints has been tackled by a number of researchers [10]. A multitude of algorithms based, for instance, on genetic algorithms [11], particle swarm optimisation [12] or hill climbing [13] have been proposed and evaluated. However, while most of the articles composing the-state-of-the-art include an empirical evaluation of the algorithms, those evaluations have been carried out with different metrics and on different test problems. Furthermore, only few articles include a comparative analysis of the results of a new algorithm against other algorithms already introduced [3, 14, 15].

The work by Burelli et al. [12], for instance, includes an evaluation on three static camera control problems in an outdoor urban environment and lists convergence time and best solution quality as success criteria. Following a completely different approach, Lino et al. [13] use an animated scene from the film 1984 on which the test multiple shots. Their success criterion is based on the average frame-rate at which the system is able to calculate the camera solutions. In an attempt to create a uniform evaluation method, in this paper, we introduce a set of general problems for camera control and we suggest a series of metrics to evaluate the complexity of the problems and the performance of the algorithms.

3 Optimising the Virtual Camera

To describe virtual camera control as an optimisation problem we need to identify its basic characteristics, i.e. what is the solution space and what is the objective function that needs to be optimised.

The solution space contains all possible camera configurations and, according to the standard perspective camera model in OpenGL [2], a virtual camera is defined by six parameters: position, orientation, field of view, aspect ratio, near plane and far plane. Camera position is a three-dimensional vector of real values defining a Cartesian position. Camera orientation can be defined either using a quaternion, a set of three Euler angles or a combination of two three-dimensional vectors describing the front direction and the up direction. With the last four parameters, the domain of the virtual camera control objective function is at least 10-dimensional.

However, some parameters such as near and far planes are commonly constant, while other parameters are tightly related to the shot type. In particular, parameters such as field of view and aspect ratio are used dynamically and statically to express certain types of shots [8]. For this reason, we consider the field of view, the aspect ratio and the two planes as fixed parameters; the search space that composes the domain of our objective function is, therefore, six-dimensional and it contains all possible combinations of camera positions and orientations.

The objective function of the optimisation problem follows the concept of *frame constraints* introduced by Bares et al. [9] as properties that can be used to describe

[2] Open Graphic Library - http://www.opengl.org

how the camera should behave. Through these constraints it is possible to define such behaviour in terms of how the frame generated by the camera should look like. Every frame constraint is converted into an objective function that, in a linear combination with all the constraints imposed, defines a camera composition objective function [16]. In this article, we consider three different constraints: *Vantage Angle*, *Object Projection Size* and *Object Visibility* that serve as representatives of all the constraints listed by Bares et al. [9].

3.1 Vantage Angle

The vantage angle constraint binds the camera position to the position and rotation of a target object. It requires the camera to be positioned so that the angle between the target object front vector and the front vector of the camera equals to a certain value. A vantage angle constraint is defined by three parameters: the target object, the horizontal angle and the vertical angle and its objective function is defined as follows:

$$\begin{aligned} f_\theta &= f_\alpha \cdot f_\beta \\ f_\alpha &= 1 - \frac{-\arctan(\frac{\boldsymbol{C}_x}{\boldsymbol{C}_z}) - \alpha}{180} \\ f_\beta &= 1 - \frac{|\arcsin(\boldsymbol{C}_y) - \beta|}{180} \end{aligned} \tag{1}$$

where α is the desired horizontal angle between the camera and the target's front vector, β is the desired vertical angle between the camera and the target's front vector and $\boldsymbol{C}$ is the normalised camera position transformed in target relative space — i.e. the pace defined by the target's front, right and up vectors.

This frame constraint is equivalent to *OBJ VIEW ANGLE* constraint of the Bares et al. [9] list. However, from an optimisation perspective, it also equivalent to *CAM POS IN REGION*, since this constraint requires the camera to be positioned in a specific subspace. Thus, both constraints limit the camera into a continuous region of space and affect only camera position.

3.2 Object Projection Size

The object projection size constraint binds the camera position and rotation to the position and size of a target object. It requires the area covered by the projection of a target object on the screen to have a specific size. The object projection size constraint is defined by two parameters: the target object and the fraction of the frame size that the projection should cover. The target object is approximated using a proxy geometry, e.g. an object aligned bounding box, and the satisfaction function is defined as follows:

$$\begin{aligned} f_\sigma &= 1 - |\sigma_c - \sigma_d| \\ \sigma_c &= \begin{cases} T_y/S_y & \textit{if } T_y/S_y > T_x/S_x \\ T_x/S_x & \textit{otherwise} \end{cases} \end{aligned} \tag{2}$$

where σ_d is the desired projection size, S_x and S_y are the width and height of the screen, and T_x and T_y are the projected width and height of the target object's proxy geometry. This frame constraint corresponds to the *OBJ PROJECTION SIZE* constraint of the Bares et al. [9] list.

3.3 Object Visibility

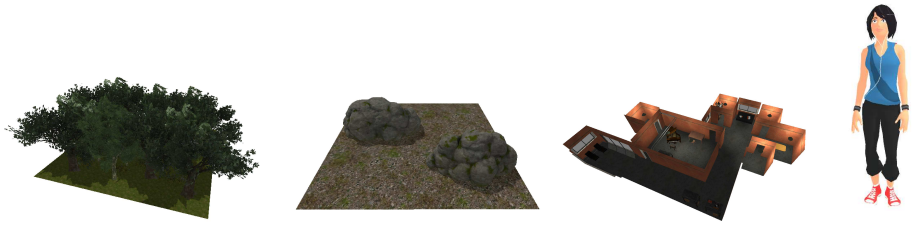
The object visibility constraint binds the camera position and rotation to the position and size of a target object. It requires the target object to be included in the frame and not hidden by any other object; both conditions are necessary to identify the target object as visible. In order to respect these two requirements the camera should be placed at a sufficient distance from the target and oriented in order to frame the target. Moreover, the volume between the camera and the target should not contain obstacles — i.e. any non-transparent object — that hide the target.

The objective function f_γ of this frame constraint quantifies the ratio between the actual visible area of the projected image of the object and its total projected area and it is defined as:

$$
\begin{aligned}
f_\gamma &= 1 - |\gamma_d - \gamma_c| \\
\gamma_c &= \frac{\sum_{i=1}^{N} infov(\boldsymbol{v}_i)}{N} \frac{\sum_{i=1}^{5} (1 - occ(\boldsymbol{e}_i))}{5} \\
infov(\boldsymbol{x}) &= \begin{cases} 1 & \textit{if } \boldsymbol{x} \textit{ is in the view frustum,} \\ 0 & \textit{otherwise.} \end{cases} \\
occ(\boldsymbol{x}) &= \begin{cases} 1 & \textit{if } \boldsymbol{x} \textit{ is occluded,} \\ 0 & \textit{otherwise.} \end{cases}
\end{aligned}
\tag{3}
$$

where γ_c is the current visibility value of the target object, γ_d the desired visibility value, $\boldsymbol{v}_i$ is the position of the i^{th} vertex of the object's mesh, N is the number of vertices of the mesh, function $infov(\boldsymbol{x})$ calculates whether a point is included in the field of view or not, $\boldsymbol{e}$ is the list containing the positions of the four extreme vertices in field of view — i.e. the top, bottom, left and right vertices on screen — and the one closer to the center of the projected image. The $occ(\boldsymbol{x})$ function calculates whether the point $\boldsymbol{x}$ is occluded by another object or not. The implemented version of the function is optimised not to calculate the second part of the function if the first part is equal to 0. The occlusion check is implemented by casting a ray towards the point defined by the vector $\boldsymbol{x}$ and then checking whether the ray intersects any other object other than the target. The $infov(\boldsymbol{x})$ function is implemented by checking whether the point defined by $\boldsymbol{x}$ is included within the six planes composing the view frustum.

The object visibility constraint includes the *OBJ IN FIELD OF VIEW*, *OBJ OCCLUSION MINIMISE*, *OBJ EXCLUDE OR HIDE* and *OBJ OCCLUSION PARTIAL* constraints of the list proposed by Bares at al. [9].



(a) Forest **(b)** Clearing **(c)** House **(d)** Subject

Fig. 1: The three test environments and the human-like model used in the benchmark.

3.4 Objective Function

The complete virtual camera control objective function is a linear combination of the three aforementioned objective functions in which the objective function value of each constraint is defined within the interval [1,0]. The weights can be used by the designer of the composition task to prioritise the satisfaction of some frame constraints over others. This equation does not take into account the interplays between frame constraints; for instance, it might be argued that any constraint related to a subject should be deactivated if the subject is out of view. However, any consideration of this sort requires a semantic understanding of the shot description, which is out of the scope of this article.

4 The Benchmark

The starting point of the benchmark we propose in this article is a set of 36 virtual camera problems designed to represent virtual camera control tasks of different complexities. This initial set includes three different virtual environments which are used to test 12 different composition problems. The 12 composition problems include the optimisation of a Vantage Angle constraint, a Projection Size constraint, a Visibility Constraint and the combination of all three constraints. These four combinations are evaluated with one, two and three subjects (see Figure 1d). In each test problem, the subjects are positioned and move around the environment in a random fashion. The subjects' speed is also randomised and varies from 0 to the average human walking speed (1.38 m/s). The subjects are shaped as stylized human beings, have human-like proportions and height (approx. 1.8 meters tall) and are composed by 3800 triangles each.

The three test environments have been selected to provide a wide range of geometrical features commonly present in computer games and interactive virtual environments. Moreover, they have been designed to incorporate a wide variety of camera composition challenges with different levels of complexity. They include one indoor and two outdoor environments with different geometrical characteristics: a forest, a house and a rocky clearing. The *forest* environment is an outdoor virtual environment composed by a cluster of 42 plants of different sizes and shapes, made of approximately 244000 triangles; the subjects are placed between these trees that act as partial occluders and

scattered obstacles. Such environments feature a highly multi-modal objective function landscape, due to the fact that the tree trunks, as other possible sparse-occluders, are thin obstacles that produce a slicing effect in the visibility objective function landscape. The second environment, the *house*, is an indoor environment with closed spaces separated by solid walls, open doors and transparent windows. The environment contains 182 objects and it is made of approximately 210000 triangles. As described in [17], walls act as large occluders inducing large areas of the objective function landscape to have little or no visibility gradient. The last environment, the *clearing*, is the simplest one from a geometrical perspective — i.e it is made of approximately 10100 triangles. It contains two medium sized obstacles and the rest of the space is empty. This environment is expected to be the simplest in terms of optimisation complexity, as the lack of obstacles tends to produce objective function landscapes with smooth gradients and a small number of modalities.

The environments are composed only by static structures; however, the targets of the desired shots move along random paths, making the problem dynamic and unpredictable. Moreover, the test problems incorporating more than one subject at a time include a further challenge since each subject acts as a dynamic obstacle for the camera and the other subjects. The test problems include either one, two or three moving subjects. For each configuration of subjects, the three frame constraints defined in Section 3 are tested both independently and combined all together. We believe that the provided range of problems covers a large number of the possible virtual camera control tasks in terms of optimisation challenges. In each test problem, the objective function is a weighted sum of the objective functions of the different frame constraints included in the problem; all frame constraints have the same weight — i.e. no priority is given to any subject or particular property — and the overall objective function is bounded between 0 and 1.

4.1 Complexity

To fairly analyse and compare the performance of any algorithm, we need to define common complexity measures [18] within common test problems [19]. In turn, the choice of appropriate case studies is a key aspect of any comparative analysis since those affect both the validity and the extensibility of any findings. In our effort to measure the complexity of each scenario, we employ a set of complexity measures proposed by Törn et al. [18] and we extend them in order to tackle both dynamic and static cases.

Törn et al. classify global optimisation problems into four categories (unimodal, easy, moderate and difficult) based on three features of the objective function:

- The probability that the region of attraction of the global optimum — i.e. the area of the objective function that features a gradient converging towards the optimum — is missed during sampling.
- If and how embedded the global optima are — i.e. if there exist local optima near the global optima region of attraction, so that sampling around these leads to the detection of better solution and, eventually, of a global optimum.
- The number of local optima.

A global optimum in optimisation is the best solution to the problem, while a local optimum is a solution to the problem which maximises/minimises it locally but not globally. Based on the above criteria the first measure of complexity that corresponds to the *probability of missing the global optimum region*, P, is defined as follows:

$$P = (1 - p^*)^{N_f} \tag{4}$$

where p^* is the percentage of the whole search space being a region of attraction of global optima and N_f denotes the affordable number of objective function evaluations within one unit of time. In our analysis, we take 16.6 ms — the frame rendering duration at a 60 fps frame-rate — as a reference unit of time; this way, P is roughly inversely proportional to the probability to find the optimal camera configuration within one frame. However, any unit of time could be adopted as long as it is coherent among the different problems and it is proportional to the time scale of the problem.

The *embeddedness* measure of complexity, E, is given by:

$$E = 1 - \frac{1}{nD_{max}} \sum_{i=1}^{n} D_{min}(x_i) \tag{5}$$

where $D_{min}(x)$ is the distance between solution x and the closest global optima region of attraction and D_{max} is the maximum distance between two points in the search space. In the experiment's search space, this distance equals to the length of the parallelepiped's diagonal.

The last complexity measure considered in this paper which corresponds to the *number of local optima*, NoL, is defined as follows:

$$NoL = \frac{A_{lo}}{A_{go}} \tag{6}$$

where A_{lo} and A_{go} are the search space sizes containing local and global optima, respectively.

Finally, to be able to capture the complexity of the problems also from a dynamic optimisation perspective, we need to extend these measures with an estimation of how the fitness landscape changes over time. Such a measure should capture the speed and the magnitude of the changes in the objective function and should give a comprehensive measure for the whole objective function landscape. For this purpose, we define a measure D of the objective function dynamism calculated as follows:

$$D = \frac{1}{K} \frac{1}{T} \sum_{i=0}^{K} \sum_{j=0}^{N-1} |f(x_i, t_{j+1}) - f(x_i, t_j)| \tag{7}$$

where K is the number of samples taken from the objective function at each sampling, T is the duration of the animation, N is the number of times the objective function is sampled during the animation, $f(x_i, t_j)$ is the objective function value of the sample x_i at the j^{th} time step and $f(x_i, t_j + 1)$ is the objective function value of the sample x_i at time $j + 1$. The D measure is a discrete approximation of the average absolute change in the objective function for all points in the solution space over time. High D values correspond to highly dynamic problems, while a D value that equals 0 corresponds to a static optimisation problem.

	Forest				House				Clearing			
	P	E	NoL	D	P	E	NoL	D	P	E	NoL	D
Angle 1	**0.03**	**1.00**	**2.50**	**1.35**	0.00	1.00	0.75	1.32	0.00	1.00	2.00	1.18
Angle 2	**0.04**	**1.00**	**2.00**	**1.27**	**0.18**	**1.00**	**2.25**	**1.21**	**0.33**	**0.88**	**2.25**	**1.09**
Angle 3	0.7	1.00	3.45	1.23	0.20	1.00	3.57	1.07	0.37	0.91	4.25	1.08
Size 1	0.49	0.91	5.63	0.13	0.72	0.93	5.96	0.13	0.38	0.94	5.38	0.15
Size 2	**0.69**	**0.71**	**10.20**	**0.13**	0.84	0.69	6.30	0.12	0.92	0.77	34.86	0.16
Size 3	0.91	0.73	23.89	0.03	0.77	0.67	16.58	0.02	0.90	0.55	30.31	0.05
Visib. 1	0.65	1.00	8.32	0.03	0.87	0.95	15.28	0.02	0.59	0.98	2.98	0.03
Visib. 2	0.83	0.59	91.04	0.02	0.98	0.79	31.82	0.01	0.80	0.92	8.52	0.03
Visib. 3	**0.94**	**0.56**	**415.65**	**0.02**	**0.94**	**0.81**	**1305.63**	**0.01**	0.78	0.88	8.18	0.02
All 1	**0.76**	**0.85**	**59.75**	**0.22**	**0.80**	**0.99**	**48.57**	**0.21**	0.43	0.94	20.28	0.15
All 2	0.64	1.00	68.35	0.16	0.76	0.99	54.00	0.19	0.78	0.89	26.56	0.15
All 3	0.83	0.95	70.50	0.14	**0.83**	**0.75**	**72.00**	**0.16**	**0.73**	**0.95**	**41.75**	**0.16**

Table 1: Complexity measures for each scenario. P is the probability that the region of attraction is missed during random sampling, E is the degree of how embedded the global optima are within local optimal solutions, NoL is the ratio between local and global optima, and D is the degree of dynamism. The most representative problems are highlighted.

4.2 Categorisation and Selection

Table 1 presents the values of the aforementioned complexity measures for each test problem investigated. All the values have been calculated by discretising the solution space with a resolution of 0.5 meters on the X and Z axis, 1 meter on the Y axis and by sampling linearly 32 rotations per position. All the problems have been evaluated by sampling the objective function each half second for 10 seconds while the subjects move in the environment. This process has been repeated 20 times for each test problem to minimise the effects of the initial random placement of the subjects.

From a static optimisation perspective, the test problems are sorted into the four categories proposed by Törn et al. [18]: *unimodal*, *easy*, *moderate* and *hard.* Following this categorisation, the vantage angle test problems with one subject are categorised as *unimodal* or *easy*. These test problems have a smaller global optima attraction area and higher evaluation times resulting in P values greater than 0. The projection size and visibility test problems with one subject fall as well in the *easy* category in all test environments, The projection size and visibility test problems with more than one subject are categorised as *moderate* in the rocky clearing virtual environment, while they are categorised as *difficult* in the other two environments. This is due to a higher P value and the more embedded local optima.

When the problems are analysed with respect to their dynamism, a different picture emerges: the visibility and projection size problems have all a significantly lower D value than the angle problems, revealing that the latter is the most complex set of problems from a dynamic optimisation perspective. The reason for such a difference in terms of dynamism is expected, as the angle objective function, depends on the subjects' orientation; therefore, even a little orientation change in terms of degrees, has an

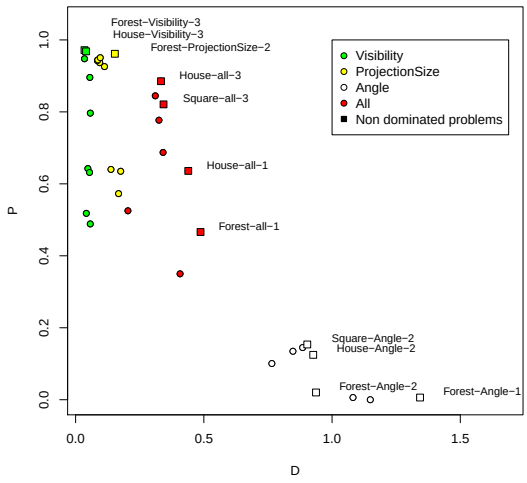


Fig. 2: Test problems sorted in a scatter plot according to their dynamism factor D and their landscape complexity P. The Pareto front identified by the squares contains all the non dominated problems in terms of complexity and dynamism.

amplified effect on the objective function landscape proportional to the distance to the subject.

Due to the contradiction between static and dynamic complexity, we need to sort the problems in a two dimensional space, defined by P and D. Sorting the problems in this manner allows us to reduce the number of test problems to be analysed by selecting a subset of non-dominated problems — i.e. the set of problems that are the most complex both in terms of dynamism and static optimisation complexity (See Figure 2). To perform this selection, we can employ the concept of Pareto optimality: a solution is defined as Pareto optimal if it cannot be improved with respect to any aspect without worsening at least one other aspect. The subset having such characteristic is highlighted in Table 1 and it contains the following problems:

- Two visibility and one projection size problem categorised as *difficult* in terms of static optimisation complexity with extremely low dynamism. We name these problems as *difficult-slow*.
- Four problems with all frame constraints with moderate static optimisation complexity and moderate dynamism. We name these problems as *average*.
- Four vantage angle problems with low static optimisation complexity and very hight dynamism. We name these problems as *easy-fast*.

5 Performance Metrics

Another fundamental aspect of the benchmark is the definition of one or more success criteria and performance measures. For virtual camera optimisation in static environ-

ments, e.g. for the generation of a single shot, there are two well established performance measures, which have been already adopted in a number of previous studies: best solution quality and convergence time [3,12]. The first measure expresses how good the algorithm is in finding the right camera configuration and, thus, generate a good quality shot. The second one expresses how efficient the algorithm is in its process.

A different situation exists for camera optimisation in dynamic environments as there is no established performance measure for the algorithms. We suggest three different performance measures inspired from the literature in dynamic optimisation: accuracy, reliability and initial convergence time. The first measure describes how close the current best camera configuration found by the algorithm is to the best existing camera configuration at any given moment. To measure such aspect, we employ the average best function value measure suggested by Schönemann [20] to evaluate dynamic optimisation algorithms. The second measure, reliability, describes how often the algorithm succeeds to provide an acceptable solution during the optimisation. The measure employed to quantify this aspect is the percentage of time in which the algorithm is able to track accurately the moving optimum within a certain acceptance range. The value of such acceptance range is dependent on the perception of how the frame looks similar to the ideal frame so it can be any percentage of the optimum. The last measure describes how much time it takes the different algorithms to converge to an optimal camera configuration which is then tracked during the rest of the execution. The measure corresponds to the time taken initially by the algorithm to reach the point in which the solution found stops to improve; therefore, the algorithm stops to converge.

6 Conclusions and Future Work

The purpose of this article is to define a standard benchmark for future research in virtual camera control in games and beyond. For this reason, we have identified the building blocks of the camera control problem and we studied the complexity of each one of them. In particular, we isolated three principal objective functions, we made their associations to the state-of-the-art definition of frame constraints and we systematically analysed their complexity. Furthermore, we designed a suite of virtual environments and problems which have been analysed for their complexity both as dynamic and static optimisation problems. As a result of this analysis, we identified 11 test problems. These problems compose the core set of the benchmark and cover the most relevant issues identified in the analysis. Finally a number of possible performance measures has been proposed and described for different analysis scenarios.

The source code and the models composing the problems analysed in this article have been published as an open source project in the GitHub platform. This article and the open source project contribute towards the establishment of standard experimental practices in the field for virtual camera control. Based on the results presented, we envision a number of future research directions such as the identification of stereotypical animation patterns or the introduction of custom benchmarks for different applications such as storytelling or specific game genres. Furthermore, we believe that the thorough analysis of camera control complexity conducted in this paper can contribute to further development of more efficient virtual camera control algorithms.

References

1. David Pinelle and Nelson Wong. Heuristic evaluation for games. In *CHI*, CHI '08, pages 1453–1462, New York, New York, USA, 2008. ACM Press.
2. James Blinn. Where Am I? What Am I Looking At? *IEEE Computer Graphics and Applications*, 8(4):76–81, 1988.
3. Roberto Ranon and Tommaso Urli. Improving the Efficiency of Viewpoint Composition. *IEEE Transactions on Visualization and Computer Graphics*, 2626(c):1–1, 2014.
4. Jia Deng Jia Deng, Wei Dong Wei Dong, R. Socher, Li-Jia Li Li-Jia Li, Kai Li Kai Li, and Li Fei-Fei Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
5. H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17:619–632, 1991.
6. Colin Ware and Steven Osborne. Exploration and virtual camera control in virtual three dimensional environments. *ACM SIGGRAPH*, 24(2):175–183, 1990.
7. Steven M. Drucker and David Zeltzer. Intelligent camera control in a virtual environment. In *Graphics Interface*, pages 190–199, 1994.
8. Daniel Arijon. *Grammar of the Film Language*. Silman-James Press LA, 1991.
9. William H. Bares, Scott McDermott, Christina Boudreaux, and Somying Thainimit. Virtual 3D camera composition from frame constraints. In *ACM Multimedia*, pages 177–186, Marina del Rey, California, USA, 2000. ACM Press.
10. Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera Control in Computer Graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218, 2008.
11. Nicolas Halper and Patrick Olivier. CamPlan: A Camera Planning Agent. In *International symposium on Smart Graphics*, pages 92–100. AAAI Press, 2000.
12. Paolo Burelli, Luca Di Gaspero, Andrea Ermetici, and Roberto Ranon. Virtual Camera Composition with Particle Swarm Optimization. In *International symposium on Smart Graphics*, pages 130–141. Springer, 2008.
13. Christophe Lino, Marc Christie, Fabrice Lamarche, Guy Schofield, and Patrick Olivier. A Real-time Cinematography System for Interactive 3D Environments. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 139–148, 2010.
14. Paolo Burelli and Georgios N. Yannakakis. Combining local and global optimisation for virtual camera control. In *Computational Intelligence and Games CIG 2010 IEEE Symposium on*, pages 403–410, 2010.
15. Mike Preuss, Paolo Burelli, and Georgios N. Yannakakis. Diversified Virtual Camera Composition. In *European Conference on the Applications of Evolutionary Computation*, Malaga, 2012.
16. Patrick Olivier, Nicolas Halper, Jonathan Pickering, and Pamela Luna. Visual Composition as Optimisation. In *Artificial Intelligence and Simulation of Behaviour*, 1999.
17. Paolo Burelli and Georgios N. Yannakakis. Global Search for Occlusion Minimisation in Virtual Camera Control. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Barcelona, 2010. IEEE.
18. A. Törn, M.M. Ali, and S. Viitanen. Stochastic Global Optimization: Problem Classes and Solution Techniques. *Journal of Global Optimization*, 14(4):437–447, 1999.
19. Christodoulos A. Floudas and Panos M. Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer-Verlag New York, September 1990.
20. Lutz Schönemann. Evolution Strategies in Dynamic Environments. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 51–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.