Computer Graphics and Constraint Solving: An Application to Virtual Camera Control.

Jean-Marie Normand

LINA - Laboratoire d'Informatique de Nantes Atlantique,
FRE CNRS 2729
2, Rue de la Houssinière,
F-44322 Nantes France.
normand@lina.univ-nantes.fr

Abstract. Since Ivan Sutherland's Sketchpad system in 1963, constraint solving techniques have been used with success in computer graphics. This success leans on the following key elements: (i) computer graphics applications manage a high number of graphical primitives; a natural user-interaction process consists in setting relations and properties between these primitives, (ii) constraint solving techniques offer a flexible modelling framework to express relations and properties between entities and (iii) a broad set of solving techniques can assist the user in computing and maintaining these relations and properties, possibly augmented by valuable properties such as robustness or performance.
However, shall be it be in declarative or constructive approaches, constraint techniques generally lose the semantics of the initial problem. The relations and properties are translated into constraint systems and the solving processes compute one or more solutions completely or partially satisfying the constraint system. The interpretation of the solution is left to the user; questions naturally arise on the representativeness of the solutions (is there a unique solution, how many solutions or classes of solutions are there ?) and on the quality of the solution (how many properties and which properties are satisfied?).
In this work, we illustrate the need for a semantic dimension through an application that embeds constraint solving techniques and show how this can improve high-level interaction. Our application consists in positioning a virtual camera in a 3D virtual environment so that the resulting image fulfils a set of visual properties.

1 Introduction

Modelling, animation and rendering have dominated research in computer graphics yielding increasingly rich and complex virtual worlds. In order to manage the complexity of these new worlds, some applications have successfully relied on the use of *CSPs* (Constraint Satisfaction Problems) and their associated constraint solving techniques. Indeed such techniques seems highly appropriate to solve difficult problems (such as computer graphics applications) thanks to their

declarative properties as well as the whole set of methods available that will ensure the efficiency of constraint solving approaches. However, actual constraint based techniques are somehow limited in the interaction offered to the user. This is especially true when considering that the modelling stage actually loose the semantics contained in the properties given by the user when transforming them into constraints. Indeed the constraints obtained by this step are solved without any consideration with respect to (w.r.t.) their membership to a property or another. That means that each property will be transformed into a list of constraints without any relation to each other. Some *semantic* decomposition involving geometric constraints involve a semantic dimension and are able to give explanations in case of failure (for a broad overview of these decompositions see [14]). Nevertheless, this is restricted to geometric constraints and cannot be seen as a very general method. When solving the CSP, information that could be gathered from the knowledge of the violated constraints is lost and nothing is provided to the user regarding the inconsistence of the problem. This is not a problem when the problem is well-constrained, but what if the problem is over-constrained and the solving method has to relax constraints in order to find a solution ? What if the problem is under-constrained and present different classes of solutions, each one having different properties? In the first case the user will neither be aware of which constraints were relaxed nor why. In the latter, the algorithm will choose a solution in one of the different sets (usually the first solution found is returned) without giving any further information on the other classes of solutions to the user. Main difficulties arise from the fact that one cannot determine the nature of a CSP (over, under or well constrained) before solving it, and that the solving methods should be chosen according to this nature. The loss of information that comes from the inadequacy of the solving algorithm chose w.r.t. the nature of a problem could be prevented by maintaining semantics of the problem throughout the solving process.

This would also lead to replace the usual measure of similarity between two solutions (usually the Euclidean distance) by a closeness w.r.t. constraints satisfaction which would capture more accurately the difference among solutions. We believe that the most important issue when solving CSPs does not "simply" consist in providing a solution to the user but in supplying meaningful information on it. In the case of an under-constrained problem for instance, instead of returning a unique solution, we believe that returning a solution of each class of solutions and giving information on the differences between them is much more interesting for the user.

In order to illustrate how we propose to maintain semantics during the whole solving process, we have chosen to rely on the problem of virtual camera composition (VCC) that consists in positioning a camera in a virtual world such that the resulting image satisfies a set of visual cinematographic properties. This task is generally achieved through a tedious and time-consuming process requiring a succession of "place the camera" and "check the result" operations. Current 3D modelers surprisingly lack integration of tools to assist the user in this task, despite the fact that cinema, in more than a hundred years, has provided a rich

grammar that allows a director to unambiguously describe shots. Modelers allow positioning and animating the camera, but lack correlation with well established cinematographic notions relative to camera composition (object framing, distance shot specification, relative viewing angles, occlusions). In this paper we propose a new approach to virtual camera composition that fully relies on this grammar, relieves the user from low-level parameter manipulation and offers him classes of possible solutions w.r.t. cinematographic notions.

This application seems well fitted to the purpose of maintaining semantics in the solving process since it is very easy to either describe an over-constrained shot (which will correspond to an over-constrained problem) or a scene that could lead to lots of different solutions (stating that one wants an object to be on the screen will lead to a under-constrained problem for instance). By maintaining semantics throughout the solving process our method will be able to provide further information to the user on the solutions of the problems, *i.e.* which properties are fulfilled by the solution in the case of an over-constrained problem or what are the differences between two (or more) solutions in the case of an under-constrained problem.

Before going further on, we will present which constraint solving techniques are commonly used for graphical applications and what are their main differences with our approach. Commonly, constraint solving techniques are classified between CSP-based approaches, metaheuristics, local propagation, mathematical approaches and deductive approaches [21]. Deductive approaches consist in using an expert system that will solve the set of constraints by rewriting them via symbolic transformations. While being useful when dealing when relatively small problems, deductive approaches seem inefficient for 3D problems involving many constraints and for problems where the constrains graph is cyclic. Such problems tend to be difficult for local propagation techniques as well. Indeed those techniques, although generally efficient for graphical applications because they maintain dynamically the consistency of a constraint network (citations?), are inappropriate for cyclic as well as under-constrained problems both being relatively common in graphical applications.

Another kind of constraint solving techniques are mathematical approaches, that can be distinguished between symbolic and numeric methods. Those techniques are also quite unappropriate for our purpose since symbolic methods (based on Gröbner basis decompositions) are totally inefficient while mathematical numeric methods while being relatively fast present the inconvenient of giving always the same solution for a given problem (deterministic behaviour). A lot of graphical applications have chosen to rely on metaheuristics techniques for solving their constraints systems *e.g.* [12, 16, 22]. Among those techniques, we can identify different families of methods, namely *neighborhood* techniques [16] such as simulated annealing or tabu search; *evolutionary* techniques (genetic algorithms [12], ant colony, *etc.*) and hybrid techniques that consist in mixing the two previous families. Those methods present the advantages of being easily implementable, any-time (one can stop the algorithm whenever one wants and still get a pseudo-solution), adapted to huge search spaces; but they

also suffer from the following drawbacks: sensibility to local minima, impossibility to prove neither the inconsistency of the problem nor the optimality of the solution found and necessity of a fine tuning of the parameters to get good results. The last family of constraint solving technique usually encountered is the CSP framework. It enjoys the completeness property which ensures that no solution is lost during the solving process, therefore this framework has been widely used in computer graphics applications, particularly within a declarative modelling context. CSP-based techniques are usually based on the *backtracking* algorithm which enforces the exploration of the whole search space. The search is performed along a search-tree whose branches are being pruned thanks to constraint propagation algorithms. On top of CSP-based techniques, we can find what we call incomplete CSP-based techniques that consist in a hybridization of complete constraint propagation algorithms (that will prune inconsistent part of the search space) with metaheuristics procedures (that will guide the search towards interesting areas of the search space), an example of such techniques can be found in [] (mettre un papier pertinent ici genre pickering ou notre algo mi csp mi local). Main drawback of CSP-based methods reside in the relative slowness of this family of algorithms w.r.t. metaheuristics.

The paper is organized as follows, we next present a brief overview of our method before detailing the semantic space partitioning approach w.r.t. to visual properties on the image. Section 3 will be consecrated to the description of our numeric algorithm which is based on a continuous extension of a local search algorithm. The exploitation of the semantic volumes is presented in section 4. Finally results are presented in section 5 before drawing a conclusion.

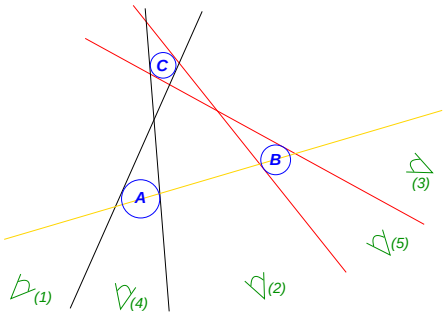

Fig. 1. Possible distinct areas for viewing a couple of objects A and B (resp. on the left and right of the screen) w.r.t. to a third object C.

1.1 Overview

Most of camera positioning methods rely upon optimization processes to compute satisfactory camera placements and therefore lead to a unique solution closely related to the objective function (see [8] for a broader overview).

However, the description of a cinematic shot can possibly yield different visual solutions. Therefore, by computing the set of semantically distinct solutions w.r.t. cinematographic properties, one provides the user meaningful results. Figure 1 presents a top view of a simple scene containing three objects A, B and C. Whenever the user describes a shot in which he constrains A and B respectively to lay on the left and on the right sides of the screen, it clearly yields three possible classes of camera configurations: area (1) object C is on the left of A and B on the screen, (2) object C is between A and B, and (3) object C is on the right of A and B. Moreover, when considering possible occlusions, two classes can be added (4) A occludes C and (5) B occludes C. In such cases, classical optimization and incomplete CSP-based approaches fail in that a unique solution [9, 19, 20], or a reduced subset [13, 7] of solutions is proposed, whereas all classes of solutions should be equally considered. These approaches actually lose the semantics of the problem while relying upon pure numerical approaches. Once a solution is computed, no further information on its characteristics or differences with other possible solutions is provided.

In this paper, we propose to integrate a semantic dimension in the solving and interaction processes to assist the user in his camera placement tasks. We follow a threefold declarative approach: (1) describe the desired solution with a set of cinematographic-based properties, (2) compute distinct classes of solutions satisfying the description with related cinematographic properties and (3) explore and interact with the classes of possible solutions.

In the description phase, as in previous approaches ([19, 12, 13, 9]), a high-level grammar is offered including composition properties (framing objects on screen surface, relative object orientation and size) and shot properties (close shot, establishing shot, *etc.*). The geometry of the scene (locations and orientation of objects) is considered as an input provided by the user.

In the computational phase, two processes are combined. The first process partitions the search space according to cinematographic properties (*e.g.* area such that A occludes B on the screen) and builds the intersection of the space partitions (indeed the conjunction of the visual properties will correspond to an intersection of the *semantic volumes cf.* section 2.4). The second process computes a *nice* representative (one that maximizes the objectives functions of the local search algorithm) of each possible class of solutions via a continuous domain implementation of a local search metaheuristic algorithm.

Finally, in a third phase, the user navigates in the possible solution sets and interacts with the semantic information provided in each area. This paper concentrates on the first two phases and offers solid foundations for high-level interactions with the user.

2 A Semantic Space Partitioning Approach

Our approach to virtual camera composition (VCC) is based on the primary idea of *visual aspects* [15]. The idea behind *visual aspects* is to gather all the viewpoints of a single polyhedron that share similar topological characteristics

on the image. A change of appearance of the polyhedron with changing viewpoint, gives rise to boundaries in the search space. Computing all the boundaries enables the construction of regions of constant aspect, namely *viewpoint space partitions*.

In this paper, we propose an extension of viewpoint space partitions to multiple objects and replace the topological characteristics of a polyhedron by cinematographic properties such as occlusions, relative viewing angles, distance shots and relative object locations.

We introduce the notion of *semantic volume* as a volume of possible camera locations that give rise to qualitatively equivalent shots w.r.t. to cinematographic properties, *i.e.* semantically equivalent shots. Each volume is characterized by a set of semantic tags issued from film grammar [3] and each tag is associated to a satisfied property in the volume. Tags are either related to a single object such as viewing angle, or to a couple of objects such as occlusion and relative image location. Therefore, the entire space of possible camera locations is thoroughly partitioned for each object, and each couple of objects.

We then derive from the user's description the subset of volumes to be considered and intersect them. This process leads to a set of non-connected regions of which each represents a different class of solutions in terms of visual aspect.

The computation of *semantic volumes* can be formalized as follows. We define the geometric filtering operator G_f that inputs a property p and provides a semantic volume s_v, which is defined by $s_v = \langle \mathcal{S}, \mathcal{V} \rangle$ where $\mathcal{S}$ is a conjunction of semantic tags (*e.g.* $\mathsf{LeftOf(A)} \wedge \mathsf{MediumShotOn(B)} \wedge \mathsf{Occludes(A,B)} \wedge \ldots$) and $\mathcal{V}$ is a subset of possible camera locations in $\mathbb{R}^3$. Every camera location inside s_v possibly satisfies the property p, whereas every camera location outside s_v certainly violates p. Possible camera orientations are to be further computed by the numerical process (see Section 3). The filtering operator is complete in that it does not lose any correct camera locations.

The operator G_f computes (possibly) non-connected volumes by pruning the space of most of the inconsistent camera locations w.r.t. p and associates a semantic tag to the resulting volumes. For example, the user description "A occludes B" (see Fig. 5) leads to a cone-shaped volume V coupled with the semantic tag $Occlusion(B, A)$.

The following subsections present how the filtering operator G_f provides the semantic volumes related to some properties.

2.1 Projection Property

The projection property is based on the notion of scale shots in cinematography. It allows the artist to specify a viewing shot for an object. There are basically six different kinds of shots : the *Extreme Close-Up*, the *Close-Up*, the *Medium Close-Up*, the *Medium Long Shot* (or *Plan Américain*), the *Long Shot*, the *Extreme Long Shot*. Related semantic tags reflect all six kinds of shots (ExtremeCloseUp(Object) to ExtremeLongShot(Object)).

The underlying *semantic volume* is computed given the position of an object and a cinematographic scale shot specified by the user. One can deduce an op-

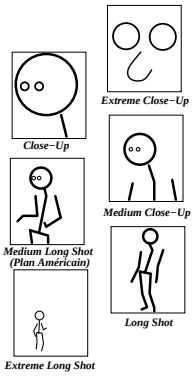

Fig. 2. The six distances between the camera and a character according to Arijon [3].

timal size corresponding to each scale shot presented in Figure 2. The object's bounding sphere and desired area in the frame are used to determine the range of camera distances. The minimum and maximum bounds of the interval correspond to two distances defining the inner and outer radiuses of an hollow sphere that includes the set of consistent positions for the camera (*cf.* Fig. 3).

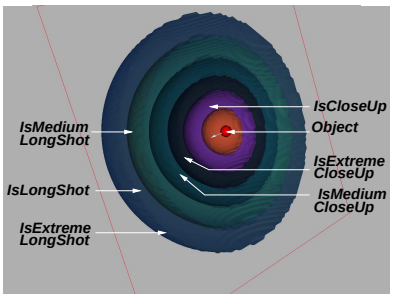

Fig. 3. Semantic volumes leading to characteristics shots.

2.2 Orientation Property

The orientation property lets the virtual cinematographer specify the viewing angle required to shoot an object or a character. A common set of 8 viewing angles is offered (*e.g.* relative to object A, there are IsLeftProfileOf(A), IsRightProfileOf(A), IsInFrontOf(A), IsInBackOf(A), IsThreeQuarterFrontLeft(A) up to IsThreeQuarterBackRight(A)) and each can be composed with high and low relative angles IsHighAngle(A) and IsLowAngle(A).

Computing orientation *semantic volumes* consists in building a prism-shaped

volume of possible camera locations (*cf.* Fig. 4), w.r.t. the vector to consider (front, back, left, *etc.*).

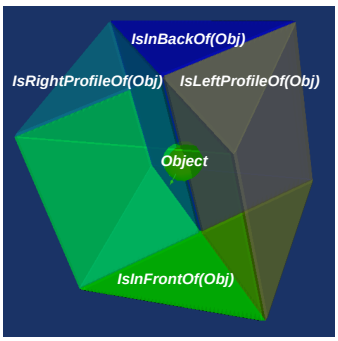

Fig. 4. Four common relative viewing angles and related semantic tags.

2.3 Occlusion Property

The occlusion property gives the user the opportunity to specify some visibility constraints between two objects of the scene. The cinematographer can characterize a total occlusion of an object by another, a partial occlusion or an absence of occlusion between two objects. A partial occlusion occurs when a part of an object's projection overlaps some part of the second object's projection.

The *semantic volumes* induced by an occlusion property are computed given characteristic cones defined with respect to the positions of the two objects involved in the occlusion property [10]. The inside bounds of the cones define the volumes of partial occlusion. The outside bounds of the cones define the volumes where no possible occlusion can occur (*cf.* Fig. 5).

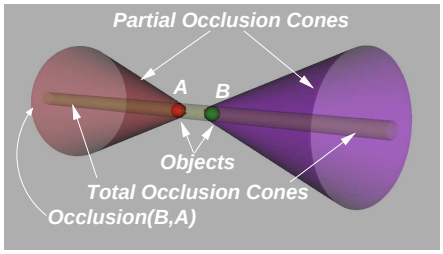

Fig. 5. Occlusion cones computation (both partial and total occlusions).

2.4 Intersecting *Semantic Volumes*

Describing a visual composition as a conjunction of properties is naturally represented by a Boolean intersection of the *semantic volumes*. From there, a set of p_i properties provided by the user results in the following computation :

$$\begin{aligned}\bigcup_i p_i &= \bigcap_i G_f(p_i) \\ &= \bigcap_i \langle \mathcal{S}_i, \mathcal{V}_i \rangle \\ &= \langle \bigwedge_i \mathcal{S}_i, \bigcap_i \mathcal{V}_i \rangle\end{aligned}$$

The intersection process may lead to an empty result, a unique volume or to a set of non-connected volumes.

3 Numerical solving stage

The output of the space partitioning approach consists in a semantic volume $s_v = \langle \mathcal{S}, \mathcal{V} \rangle$ containing possible camera locations. From there, the numerical stage computes a *nice* representative of each disjoint volume in $\mathcal{V}$. This consists in choosing in $\mathcal{V}$ a consistent camera configuration (location, orientation and focal distance) that maximizes each property p_i corresponding to a semantic tag of S (*i.e.* minimizing a cost function).

The problem therefore comes down to determine a septet of variables c, such that :

$$\begin{cases} \min \sum_i \mathsf{cost}_{p_i}(c) \\ c \in \mathcal{V} \end{cases}$$

where $\mathsf{cost}_{p_i}(c)$ stands for the function cost associated to property p_i.

Classical optimization and constrained optimization techniques require a differentiable objective function (*e.g.* general gradient descent algorithm). In order to manage our constraints we rely on stochastic-based *Local Search* techniques.

3.1 The *Local Search* framework

The underlying principles of the *Local Search* framework can be found in [1]. Algorithm 1 presents our instantiation.

The algorithm relies on the exploration of the search space – here a *semantic volume* $\langle \mathcal{S}, \mathcal{V} \rangle$ – starting from an initial set-up, and exploring the neighborhood around the current configuration. The $\mathsf{generateNeighbours}(nb_{\text{neigh}}, \text{current}, \mathcal{V})$ function randomly selects a set of nb_{neigh} neighbors in $\mathcal{V}$ within a region centered at the current configuration. It introduces the notions of *diversification* and *intensification*. The *intensification* heuristic allows concentrating on promising regions of the search space. Conversely, *diversification* prevents being stuck in local minima. When no improvement has been made for a while a random reset of the initial assignment is performed to allow a wider investigation of the search space.

The *Local Search* procedure is parameterized by the maximum number of iterations (maxTries and maxSteps) and the number nb_{neigh} of neighbors at each iteration of the stochastic $\mathsf{generateNeighbours}$ function. At each step, the best

Alg. 1. Continuous Local Search Algorithm.

```
1  CLS(in: ⟨S, V⟩; in: maxTries,maxSteps,nb_neigh ∈ ℤ;out: Configuration)
2  begin
3  best ← initialConfiguration(V)
4  current ← best
4  nbTries ← 0
   % Diversification Loop
5  while nbTries < maxTries
6    nbSteps ← 0
     % Intensification Loop
7    while nbSteps < maxSteps
8      neighbors ← generateNeighbors(nb_neigh, current, V)
9      current ← getBestNeighbor(neighbors, V)
10     if (isBetterConfiguration(current, best)) then
11       best ← current
12     end
13     nbSteps++
14   endwhile
15   nbTries++
16 endwhile
17 return best
18 end
```

neighbor in terms of constraint satisfaction and cost function minimization becomes the new current configuration.

3.2 Continuous Extension of Local Search

In order to take into account the intrinsic continuous nature of the variables of our application, the Local Search Algorithm had to be adapted. Indeed in the general case, Local Search algorithms are applied for problems with discrete variables. Only a few works tried to apply local search methods in continuous domains [17, 6, 5, 2, 4]. In all these works, configurations are, like in the discrete case, continuous instantiations of the variables, *i.e.* points in $\mathbb{R}^n$. This leads in reconsidering the neighborhood and the heuristic parts of the algorithm in order to handle the huge size of the domains taken into account ($\mathbb{R}$ discretized). Our approach is somehow orthogonal : we consider intervals of $\mathcal{I}^n$ instead of points in $\mathbb{R}^n$. Our algorithm is then centered on managing boxes-configurations and the continuous nature of the constraints we encounter. The metaheuristic part or tabu search of our algorithm is quite simple for now. This method is closer to work from Barichard and Hao [4] on PICPA, a hybrid method which metaheuristic part is a genetic algorithm. Individuals are boxes selected among

a multi-objective criterion. In our algorithm, configurations are boxes selected w.r.t. a min-conflict criterion.

In our continuous local search framework, as virtual camera composition requires to handle continuous variables in a wider way, the configurations are defined by a Cartesian product of intervals with floating points bounds (namely a box). We thus rely on interval arithmetics [18] to evaluate the configurations on both constraints and penalty functions. The main advantage of the interval arithmetics is that it ensures the *containment property*, *i.e.* it guarantees that the evaluation of a box contains the evaluation of every point in the box. Thus, while using interval arithmetics the evaluation of the costs functions will not be obtained as reals but as intervals.We then compare the interval representing the evaluation of each box (*cf.* Fig. 6) in order to determine the best configuration.

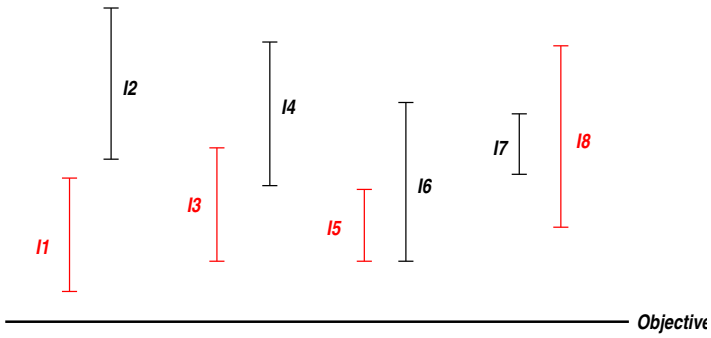

Fig. 6. Interval Comparison. In this scheme intervals are compared by twos, grey interval being the best.

4 Exploitation of Semantic Volumes

The main contribution of this paper is to offer a semantic basis for exploring and interacting with the volumes. We illustrate two possible interactions: making requests on the computed *semantic volumes* and making requests on the whole 3D scene w.r.t. the computed volumes.

4.1 Making requests on the volumes

For a given description, each computed volume provided by the geometric solver stores some knowledge related to the satisfaction of the properties. From here, the characterization of each distinct volume can be semantically augmented according to properties the user has not mentioned. For example, if a semantic volume s_v is characterized by the relative location $\mathsf{IsLeftOf}(A, B)$ tag, some further characterization of s_v can be computed by considering the orientation properties related to A and B (*e.g.* $\mathsf{IsInFrontOf}(A) \wedge \mathsf{IsInBackOf}(B)$). As a consequence,

the user can select two semantically augmented volumes s_{v1} and s_{v2} from the same description and request for the differences between them. This request boils down to compute all tags in s_{v1} and in s_{v2} that do not belong to $s_{v1} \cap s_{v2}$.

4.2 Making requests on the scene

As each object and couple of objects in the scene generate their respective *semantic volumes*, it is trivial to make requests on a computed volume s_v against any possible object or property. The request comes down to computing a new geometric intersection and checking the number of connected volumes computed by tesselation. For example, in Problem 2 of Section 5 (the five-framing shot), one can ask if there exists a possible camera location such that A, B and C can be viewed simultaneously from the front relative angle :

$$s'_v = s_v \cap G_f(\mathsf{Front}(A)) \cap G_f(\mathsf{Front}(B)) \cap G_f(\mathsf{Front}(C))$$

where s_v is the previously computed semantic volume and $G_f(\mathsf{Front}(A))$ the geometric filter that computes the semantic volume related to the property $\mathsf{Front}(A)$. If s'_v is empty, the answer is no. Contrarily, s'_v contains all possible volumes answering the request. Splitting a computed volume w.r.t. a property follows a similar scheme. For example, the splitting of a volume s_v into all the possible shot distances relative to an object A ($\mathsf{IsCloseUp}$, $\mathsf{IsMediumLongShot}$, ...) can be expressed as follows :

$$\begin{aligned}\{s_{v1}, ..., s_{vn}\} = \{\ & s_v \cap G_f(\mathsf{IsCloseUp}(A)),\\ & ...,\\ & s_v \cap G_f(\mathsf{IsExtremeLongShot}(A))\}\end{aligned}$$

5 Results

Two examples illustrate our approach. First the classical over-the-shoulder shot implying two characters and then a framing shot with 7 possible classes of solutions.

5.1 The *over-the-shoulder* shot

The classical over-the-shoulder shot is commonly encountered when filming a dialogue between two or more actors and consists in laying the camera behind one actor while framing the other. Figure 7 illustrates the related *semantic volume* and a result is presented in Figure 8.

5.2 The *five-framing* shot

The geometry of the scene is composed of 5 coplanar objects (see top-view Fig. 9). The user frames objects A, B and C respectively in the left, middle and right of the screen, and constrains D and E to belong to the screen without any

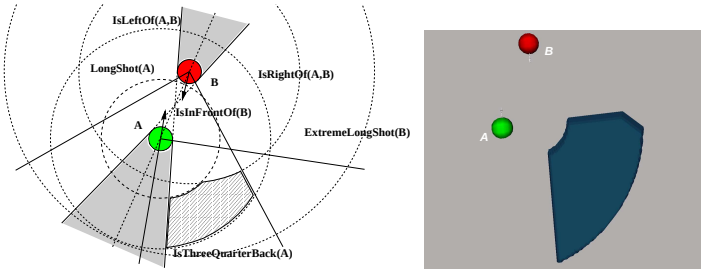

Fig. 7. Top view of the search space computed by the *over-the-shoulder* shot (left) and tessellation of the implicit volume (right).

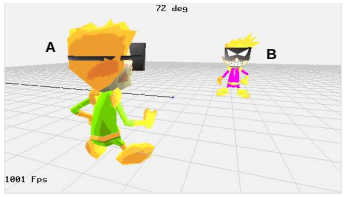

Fig. 8. A result of the over-the-shoulder shot.

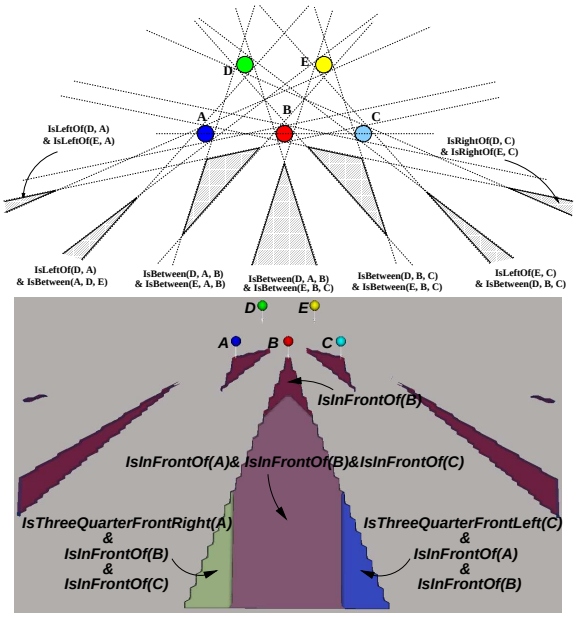

Fig. 9. Top view of the *five-framing* shot (top) and tessellation related to possible camera locations (bottom) with further semantic information related to orientations of A, B and C.

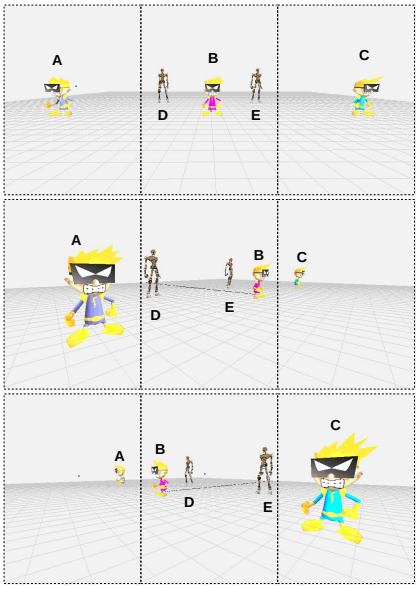

Fig. 10. Three shots associated to the Five framing shot.

occlusion. Some results are presented in Figure 10. All three shots satisfy the users description and illustrate different classes of solutions.

Table 1 presents the time spent during the geometrical (T_G) and numerical (T_N) steps in computation of one representative of each semantic volume. T_G is directly related to the granularity of the tessellation (examples 1 and 2 generate respectively 240 and 4036 polygons). T_N is the time spent in the local search with nbSteps $= 25$ and nbTries $= 25$. Such values provide a very good time-to-quality ratio. Although the total time to compute all representatives seems important ($7 \times 0.67 = 4.69$ seconds), time per representative is around 0.7 seconds which is quite acceptable for interaction purposes.

Example	Nb Vol	T_G	T_N	Total time
1	1	0.21	0.60	0.81
2	7	0.53	0.67	5.22
				(0.53+7×0.67)

Table 1. Time for computing a representative of each volume. T_G and T_N represents the time spend in the geometrical and numerical processes (Time in seconds on Linux OS, Pentium M 2GHz, 512Mo)

6 Discussion

The semantic space-partitioning approach offers the following features: cinematographic properties provide semantic volumes containing the possible solutions of the problem through a geometric process that filters non-satisfactory areas of the world. Whenever the intersection process leads to an empty result, there is a guarantee of contradiction in the user's specification. The numerical process offers a *good* representative of each volume at relatively low computational cost.

The main interest of maintaining semantics throughout the whole solving process (as shown is the case of positioning a virtual camera) of a problem, is that the user is provided with much more meaningful and interesting information on the solutions of his problem. Unlike other approaches we offer solutions as well as additional knowledge on the different classes of solutions (if existing). This allows the user to make choices and reason on the problem instead of "only" being given a solution to a set of constraints he had not modelled. Additional information could also be provided on constraints the user had not even modelled (*cf.* section 4).

As for our system, one can criticize the use of spheres as object boundaries in the occlusion computation. Indeed they present the major drawback of being greatly inappropriate for bounding objects that are mainly designed along one axis (like walls or pillars for example). This could lead to forbid large regions of the search space that could possibly lead to correct shots during occlusions management. Nevertheless by using trees of bounding spheres we are still capable of representing relatively complex shapes such as a human body and integrating them in our scenes. Still, it is possible to obtain efficient and precise results through hardware buffer rendering capacities [11, 12] but seem delicate to integrate in our approach. Happily, finer boundary representations can be integrated by computing shadow volumes provided an implicit representation of such volumes is available.

To conclude, in this paper we have presented an original approach to virtual camera composition that identifies classes of distinct solutions, provides means to characterize them and computes good representatives. By extending the notion of visual aspects, we introduced the notion of *Semantic Volumes* as a set of possible camera locations that share a same set of cinematographic characteristics. Experimental results show the suitability of our approach and opens exciting perspectives in providing natural and intuitive interfaces to virtual camera composition. This work also lays the groundwork to a new kind of solving processes that maintain semantics of the problem while solving it in order to provide the user with useful information on the solutions (or pseudo-solutions).

References

1. E. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. Wiley and Sons, New York, 1997.
2. A. Anglada, P. Codognet, and L. Zimmer. An adaptive search for the NSCSPs. In *CSCLP 2004*, Lausanne, 2004.

3. D. Arijon. *Grammar of the Film Language*. Silman-James Press, 1976.
4. V. Barichard and J-K Hao. A population and interval constraint propagation algorithm. *LNCS*, 2632:88–101, 2003.
5. R. Battiti and G. Tecchiolli. The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. *Annals of Operations Research*, 62, 1996.
6. R. Chelouah and P. Siarry. Tabu search applied to global optimization. *European Journal on Operational Research*, 2000.
7. M. Christie, E. Languénou, and L. Granvilliers. Modeling Camera Control with Constrained Hypertubes. In *Procs of CP 2002*, September 9-13 2002.
8. M. Christie and P. Olivier. Camera control. In *Eurographics 2006, State of the Art Reports*, September 2006.
9. S. M. Drucker and D. Zeltzer. CamDroid: a system for implementing intelligent camera control. In *Procs of the 1995 Symposium on Interactive 3D graphics*, pages 139–144, 1995.
10. F. Durand, G. Drettakis, and C. Puech. The 3D Visibility Complex. *ACM Transactions on Graphics*, 21(2):176–206, April 2002.
11. N. Halper, R. Helbing, and T. Strothotte. A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence. In *Procs. of the Eurographics'2001 Conference*, volume 20, pages 174–183, 2001.
12. N. Halper and P. Olivier. CAMPLAN: A Camera Planning Agent. In *Smart Graphics 2000 AAAI Spring Symposium*, pages 92–100, March 2000.
13. F. Jardillier and E. Languénou. Screen-Space Constraints for Camera Movements: the Virtual Cameraman. In *Procs. of EUROGRAPHICS-98*, volume 17, pages 175–186, 1998.
14. C. Jermann, G. Trombettoni, B. Neveu, and P. Mathis. Decomposition of geometric constraint systems: a survey. *Internation Journal of Computational Geometry and Applications (IJCGA)*, 2006.
15. J.J. Koenderink and J. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.
16. M. Larive, O. Le Roux, and V. Gaildrat. Using Meta-Heuristics for Constraint-Based 3D Objects Layout . In Dimitri Pléménos, editor, *The Seventh International Conference on Computer Graphics and Artificial Intelligence, 3IA'2004*. Dimitri Pléménos, General Chair of the Conference, mai 2004.
17. Z. Michalewicz. Genetic Algorithms, Numerical Optimization and Constraints. *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 151–158, 1995.
18. R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, N.J., 1966.
19. P. Olivier, N. Halper, J. Pickering, and P. Luna. Visual Composition as Optimisation. In *AISB Symposium on AI and Creativity in Entertainment and Visual Art*, pages 22–30, 1999.
20. J. H. Pickering. *Intelligent Camera Planning for Computer Graphics*. PhD thesis, Department of Computer Science, University of York, September 2002.
21. O. Le Roux. *Modélisation déclarative d'environnements virtuels : contribution à l'étude des techniques de génération par contraintes* . Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 2003.
22. S. Sanchez, O. Le Roux, H. Luga, and V. Gaildrat. Constraint-Based 3D-Object Layout using a Genetic Algorithm . In *The Sixth International Conference on Computer Graphics and Artificial Intelligence, 3IA'2003*. Procedings edited by Dimitri Pléménos, General Chair of the Conference, mai 2003.