

# A Semantic Space Partitioning Approach to Virtual Camera Composition

Marc Christie and Jean-Marie Normand

LINA - Laboratoire d'Informatique de Nantes Atlantique,  
FRE CNRS 2729  
2, Rue de la Houssinière,  
F-44322 Nantes France.

---

## Abstract

*In this paper, we present a semantic space partitioning (SSP) approach to the virtual camera composition problem. Virtual camera composition (VCC) consists in positioning a camera in a virtual world, such that the resulting image satisfies a set of visual cinematographic properties. Whereas most related works concentrate on numerically computing a unique camera position satisfying the problem, we offer to isolate identical possible solutions in 3D volumes with respect to their visual properties, and to propose them to the user. We introduce the notion of semantic volumes as an extension of visual aspects to characterize, compute and manipulate distinct solution sets. Our approach relies on (1) a space partitioning process derived from a study of possible camera locations w.r.t. to the objects in the scene and (2) local search numerical techniques to compute good representatives of each volume. This work is motivated by the lack of VCC tools in 3D software and the will to integrate cinematographic semantics in the description, solving and interaction processes. Experimental results illustrate the suitability of our approach for identifying and providing distinct solution sets. Furthermore, the exploitation of the semantic volumes lays the groundwork for natural and efficient user interaction by providing knowledge and reasoning on possible classes of solutions.*

---

## 1. Introduction

Positioning a virtual camera in a 3D virtual environment is generally a non-intuitive task when using a 2D input device such as a mouse. The user has a mental representation of the result in terms of what he wants to see on the screen, and has to apply a mental inversion process to determine the location, orientation and field of view parameters of the camera. 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 are based on complex mathematical notions (spline curves, velocity graphs) more or less hidden by high-level manipulators. Manipulators allow positioning and animating the camera, but lack correlation with well established cinematographic notions relative to camera composition (object fram-

ing, 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. current cinematographic notions.

### 1.1. Related work

In related literature, numerous approaches have utilized cinematographic properties such as subject size and location within the frame to assist users in camera composition and camera planning.

J. Blinn [Bli88] propose a vector algebra-based solution to pin two objects at given locations on the screen. Gleicher and Witkin [GW92] offer means to control 2D points directly on the display screen through differential manipulation techniques instead of controlling the camera parameters.

Blinn's algebraic method has laid the groundwork for cin-

ematographic idiom-based approaches, such as Christianson *et al.*'s compiler for the Declarative Camera Control Language (DCCL) [CAH\*96] or He's *et al.*'s Virtual Cinematographer [HCS96]. These approaches however suffer from the point representation of the objects that doesn't capture the occlusion properties and thus avoids one of the main problems in camera planning, namely occlusion. Moreover the "point-like" representation cannot take into account the real geometry of the scene.

Camera composition and camera planning can be viewed as constrained optimization problems in which the properties of the shot are expressed as numerical constraints on the camera variables (respectively the camera's path variables) and a broad range of solving procedures is available to compute solutions. The solvers differ in the way to manage over-constrained and under-constrained cases, in their complete or incomplete search capacities, in local minima management and possible optimization processes (generally finding the best solution w.r.t. an objective function).

In their CAMDROID system for automated camera planning [DZ95], Drucker *et al.* use an existing numerical constraint solver package (*CFSQP*) and have to compile the shot specifications in the input script used by this library. Unfortunately, the solving process is sensitive to the initial configuration and is subject to local minima failures. Bares *et al.* use a partial constraint satisfaction system named CONSTRAINTCAM [BGL98] in order to provide alternate solutions when constraints cannot be completely satisfied. This solution is based on a limited subset of cinematographic properties (viewing angle, viewing distance and occlusion avoidance), which limits the procedure to small problems.

P. Olivier *et al.* [OHPL99] propose a rich set of properties for composition purposes. The authors have developed the CAMPLAN system [HO00] that numerically solves the optimization problem via a metaheuristic search (genetic algorithms) method. The main shortcomings of this purely optimization-based technique is that the CAMPLAN genetic algorithm produces solutions in widely varying amounts of time and is also subject to the initial population of solutions. Gooch *et al.* [GRMS01] also use optimization procedures in order to produce images fitting artistic composition criterion like rules of thirds and fifths and the notion of *canonical viewpoint* (viewpoint leading to better identification of an object).

The CSP (Constraint Satisfaction Problem) framework has proven to succeed in some camera composition and motion planning approaches. In [BMBT00] Bares *et al.* propose a heuristic-based complete search algorithm. The process is applied inside promising 3D areas computed through simple geometric intersections. Although efficient, the approach avoids problems related to multiple solutions.

Unlike Bares *et al.* who utilize partial constraint satisfaction through cost functions, Jardillier & Langu  nou use pure

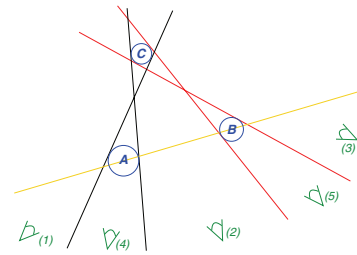


Figure 1: Possible distinct areas for viewing a couple of objects A and B (resp. on the left and right of the screen) w.r.t. to a third object C.

interval methods in *The Virtual Cameraman* [JL98] to compute camera paths which yield sequences of images fulfilling temporally indexed image properties. This idea has been improved by Christie *et al.* in [CLG02]. Unfortunately, the benefits of interval-based techniques that guarantee the fulfillment of the properties during the whole sequence are counterbalanced by the computational effort required, and the absence of any mechanism for constraint relaxation. However, whenever the method fails, the user has a guarantee that there are no solutions to his problem (due to completeness of interval-based approaches).

Some approaches combine constraints and optimization. One of these was presented by J. Pickering [Pic02] and is closely related to our work. In this solving method, the constraints are used to create feasible regions of space that will serve as bounds for an optimization procedure. The search space is subdivided by a "shadow-volumes" algorithm based on the properties of the image, and the feasible regions are then discretized and stored in an octree structure. Each node of the octree is then used as a starting point for a genetic algorithm that tries to find a solution to the problem. The main shortcomings of this approach lay in the computational effort required to create the octree, and the fact that multiple distinct solutions are ignored.

## 1.2. Overview

Most of the methods we mention rely upon optimization processes to compute satisfactory camera placements and therefore lead to a unique solution closely related to the objective function.

However, the description of a cinematic shot can possibly yield different visual solutions. Therefore, in 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 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 [DZ95, OHPL99, Pic02], or a reduced subset [JL98, CLG02] 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. Certainly, any classification process can be provided afterward, but requires an important computational effort.

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:

- describe the desired solution with a set of cinematographic-based properties,
- compute distinct classes of solutions satisfying the description with related cinematographic properties,
- explore and interact with the classes of possible solutions.

In the description phase, as in previous approaches [OHPL99, HO00, JL98, DZ95], 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, low and high angle). 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. The second process computes a *nice* representative 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.

This paper is organized as follows: Section 2 introduces our semantic space partitioning approach to virtual camera composition, Section 3 details the numerical solving process. The exploitation of the *semantic volumes* is presented in Section 4 and relevant results are then presented in Section 5. Finally Section 6 discusses future research directions and concludes.

## 2. A Semantic Space Partitioning Approach

Our approach to virtual camera composition (VCC) is based on the primary idea of *Binary Space Partition* (BSP) and can be considered as an extension of *visual aspects* [KvD79]

and closely related works such as *viewpoint space partitioning* [PD90] in the field of object recognition. 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 [Ari76] 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  $s_v = \langle S, \mathcal{V} \rangle$  where  $S$  is a conjunction of semantic tags (e.g.  $\text{LeftOf}(A) \wedge \text{MediumShotOn}(B) \wedge \text{Occludes}(A,B) \wedge \dots$ ) 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  $\text{Occlusion}(B,A)$ .

The following subsections present how the filtering operator  $G_f$  provides the semantic volumes related to each property. Below, we consider that the camera's *roll* degree of freedom (i.e. around the look-at vector) is restricted to interval  $[-\frac{\pi}{4}, \frac{\pi}{4}]$ , and the *tilt* angle is confined in  $[-\frac{\pi}{2}, \frac{\pi}{2}]$  (no upside-down cameras). Camera compositions seldom break these conventions.

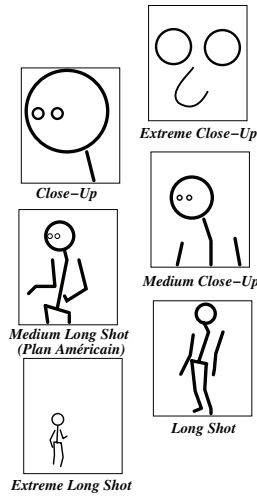


Figure 2: The six distances between the camera and a character according to Arijon [Ari76].

### 2.1. Projection Property

The projection property is based on the notion of scale shots in cinematography (cf. Fig 2). 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 optimal 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. In order to add some flexibility to the solving system, we compute an interval range related to this optimal value by subtracting and adding an epsilon to it (see [BMBT00]). The minimum and maximum bounds of the interval correspond to two distances defining the inner and outer radii of an hollow sphere that includes the set of consistent positions for the camera (cf. Fig. 3). Distance is trivially computed by the following equation:

$$distance = \left( \frac{sizeObject}{screenSize} \right) \times \left( \frac{1}{\tan\left(\frac{fov}{2}\right)} \right)$$

### 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

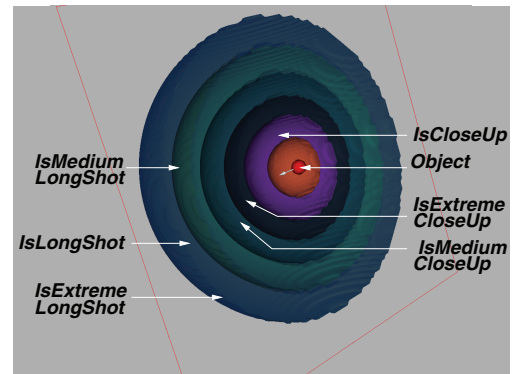


Figure 3: Semantic volumes leading to characteristics shots.

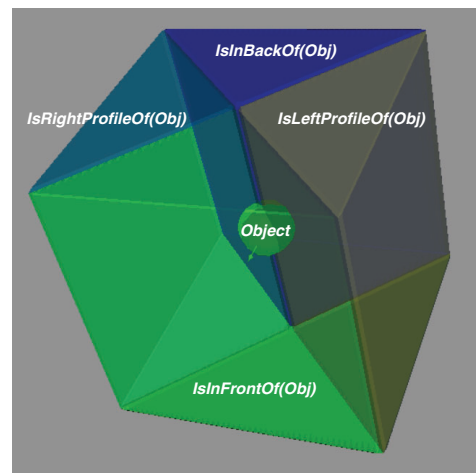


Figure 4: Four common relative viewing angles and related semantic tags.

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, w.r.t. the vector to consider (front, back, left, ...). Once again, a variation with the optimal orientation is accepted in order to avoid being too restrictive (cf. Fig. 4).

### 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

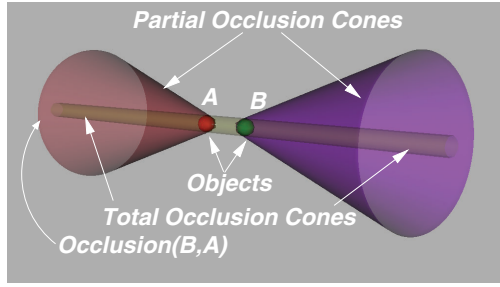


Figure 5: Occlusion cones computation (both partial and total occlusions).

property [DDP02]. 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).

## 2.4. Framing Property

The framing property allows the virtual cinematographer to constrain an object in a given frame inside, partially inside, or outside the screen space. This expressive property defines the relative locations and sizes of objects on the screen. It constrains altogether the distance shot and the camera locations and orientations. Moreover, total or partial occlusions can be derived from overlapping frames, and conversely non-occlusions can be derived from the absence of overlap (cf. Fig. 6 and 8).

The computation and characterization of the *semantic volumes* related to this property depend on the number of framing properties defined by the user and their relative location on the screen. We identify two cases: (1) whenever a single frame is defined, the framing property will perform like a projection property: the size of the frame and the size of the object in the scene lead to the computation of a volume that limits the camera-to-object distance. However all possible camera locations around the object are possible w.r.t. this distance; (2) if the user has specified two or more frames a specific study is to be carried out for every couple of objects in order to extract additional knowledge. The full study of all the distinct frame configurations is not provided here: below we propose a study over two cases leading to different *semantic volumes* considering overlapping and non-overlapping frames (Figures 8 and 6). Note that the *semantic volumes* computed at the end of the framing processing will only satisfy the spatial relations between the frames on the screen (e.g. object A is left of B), but not the exact locations of the objects in the frames; the numerical process will compute camera orientations to satisfy exact locations in the frames.

### 2.4.1. Non-overlapping configuration

Figure 6 illustrates a user input related to a non-overlapping configuration. Figure 7 provides a 2D topview of the scene

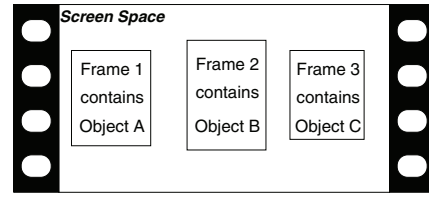


Figure 6: A non-overlapping description constraining three objects.

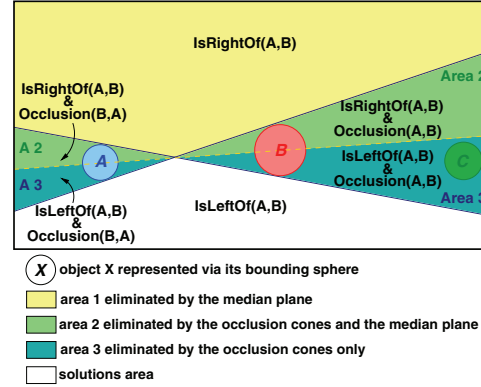


Figure 7: 2D Topview of the *semantic volumes* related to couple (A, B) when respectively framed on the left and right of the screen.

with locations of objects A, B and C. For clarity sake, the representation is two dimensional, but all the computation occurs in 3D. The first semantic volume is identified as Area 1 (behind A and B) and tagged with  $IsRightOf(A,B)$ . No possible camera location in this area can lead to a shot in which the object A lays on the left side of B. Second, as Frames 1 and 2 do not overlap on the screen, no occlusion should occur between A and B. Area 2 and 3 are built by computing the occlusion cones [DDP02]. Respective tags are  $Occlusion(A,B)$  where B hides A and  $Occlusion(B,A)$  when A hides B. Consequently, the last area (bottom of Figure 7) stands for the possible camera locations satisfying the user's description ( $IsLeftOf(A,B)$ ).

This process is repeated for every couple of objects in the screen, i.e.  $\{(A,C), (B,C)\}$ .

### 2.4.2. Overlapping configuration

Figure 8 illustrates an overlapping configuration. As the overlapping is partial ( $Frame1 \not\subseteq Frame2$  and  $Frame2 \not\subseteq Frame1$ ), areas 1, 2, 4 and 5 are not considered as possible camera locations and are tagged with  $TotalOcclusion(A,B)$  or  $TotalOcclusion(B,A)$ . Similarly, the area between A and B is not considered as possible ( $Between(A,B)$ ). Furthermore, Area 3 does not provide any overlapping configuration and is tagged with  $NoOcclusion(A,B)$ . All in all, the final areas containing possible camera locations are located at the left and right extremes of Figure 9. Two distinct volumes are



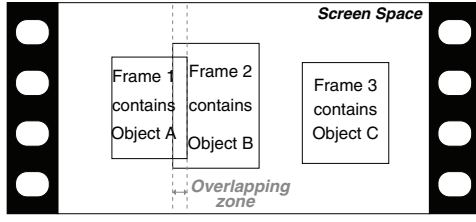


Figure 8: An overlapping description constraining three objects with an overlap.

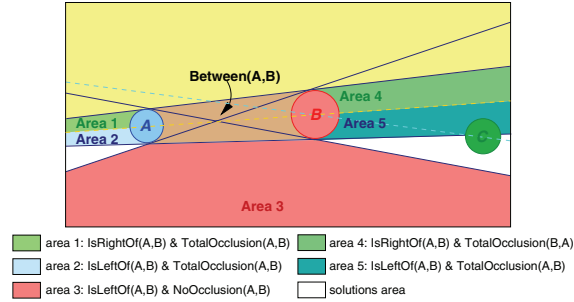


Figure 9: 2D Topview of *semantic volumes* related to overlapping frames on objects A and B.

proposed: if the camera is located in the right white solution space, B will appear closer than A in the final shot, whereas if it is positioned to the left, A will appear closer than B.

All the same, further semantic partitioning and characterization is offered by considering couples (A,C) and (B,C). The study of the whole set of distinct frame configurations follow the same process.

## 2.5. Relative positioning properties

Relative positioning properties allow to specify some relative positions between objects on the screen, *e.g.* one can stipulate that one wants to see the object A on the right hand side of B, above B, *etc.* These relative placements describe the spatial arrangement of objects without being forced to locate them precisely within frames.

## 2.6. Intersecting Semantic Volumes

Describing a visual composition as a conjunction of properties is naturally represented by a Boolean intersection of the *semantic volumes*. Therefrom, 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.

In order to implement the intersection of 3D volumes, we propose to rely on implicit surface representations [WW89]

of the semantic volumes rather than pure geometric B-rep implementations. Each primitive  $P_i$  (cone, sphere, plane) is the source of a potential field  $F_i(x, y, z)$ , the so-called *implicit function*. For each point  $M(x, y, z)$  of the Euclidian space, the fields of the primitives are combined:

$$F(x, y, z) = \sum_i F_i(x, y, z)$$

A 3D volume is generated when choosing a threshold value  $t$  such that  $\mathcal{V} = \{(x, y, z) \in \mathbb{R}^3 | F(x, y, z) \leq t\}$ . Using implicit functions provides us with the following assets: (1) exact intersections of volumes by composing the implicit functions (2) robust intersections in that we avoid degenerated vertices, edges or faces (3) possibility to test if a point lies in the surface via a fast and simple implicit function evaluation.

Our implementation relies upon the VTK library (<http://www.vtk.org>) that provides simple, robust and efficient means to create implicit surfaces by defining primitives for each object and intersecting them with the world cube bounding the whole 3D scene. For example, the possible camera locations representing the scale shots distances between the camera and an object (*cf.* Fig. 3) are described via implicit spheres. Similarly, the volumes illustrating the occlusion property are defined as implicit cones surrounding the objects (see Fig. 5), thus forbidding the regions of total or partial occlusions.

The major drawback to the use of implicit surfaces lies in the computational cost of tessellation. However our approach requires a unique tessellation at the final step in order to determine the number of non-connected components. The cost of the tessellation is driven by the number of subdivisions and for our purpose the computation of the non-connected components remains reliable with coarse subdivisions.

## 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 satisfies the framing properties and that maximizes each property  $p_i$  corresponding to a semantic tag of  $\mathcal{S}$  (*i.e.* , minimizing a cost function).

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

$$\begin{cases} \min \sum_i \text{cost}_{p_i}(c) \\ \forall j, f_j(c) \text{ is satisfied} \\ c \in \mathcal{V} \end{cases}$$

where  $\text{cost}_{p_i}(c)$  stands for the function cost associated to property  $p_i$ , and where  $f_j$  is the  $j$ -th framing property given by the user.

Table 1: Numerical *Local Search* procedure: given a semantic volume  $\langle S, \mathcal{V} \rangle$  it returns the “best” set-up in terms of constraints satisfaction w.r.t. a penalty function.

```

LocalSearch(in:  $\langle S, \mathcal{V} \rangle$ ; out: Configuration)
begin
  best  $\leftarrow$  initialConfiguration( $\mathcal{V}$ )
  current  $\leftarrow$  best
  nbTries  $\leftarrow$  0
  while (nbTries < maxTries) % Diversification Loop
    nbSteps  $\leftarrow$  0
    while (nbSteps < maxSteps) % Intensification Loop
      neighbours  $\leftarrow$  generateNeighbours( $\alpha$ , current,  $\mathcal{V}$ )
      current  $\leftarrow$  getBestNeighbour(neighbours,  $\mathcal{V}$ )
      if (isBetterConfiguration(current, best))
        best  $\leftarrow$  current
      end
      decrease( $\alpha$ )
      nbSteps ++
    endwhile
    nbTries ++
  endwhile
  return best
end

```

Classical optimization and constrained optimization techniques require a differentiable objective function (e.g. general gradient descent algorithm). In order to manage both non algebraic constraints such as  $c \in \mathcal{V}$  and algebraic constraints such as  $f_i(c)$ , 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 [AL97]. Table 1 presents our instantiation.

The algorithm relies on the exploration of the search space – here a *semantic volume*  $\langle S, \mathcal{V} \rangle$  – starting from an initial set-up, and exploring the neighbourhood around the current configuration. The `generateNeighbours( $\alpha$ , current,  $\mathcal{V}$ )` function randomly selects a set of neighbours in  $\mathcal{V}$  within a region around the current configuration. It introduces the notions of *diversification* and *intensification*. The *intensification* heuristic allows concentrating on promising regions of the search space by decreasing the neighbourhood’s size (here the contraction factor  $\alpha$ ). 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 of neighbours at each iteration of the stochastic `generateNeighbours` function. At each step, the best neighbour in terms of constraint satisfaction and cost function

minimization becomes the new current configuration.

As the user’s description may not lead to a solution (very precise framing is difficult to respect as objects have been positioned beforehand in the 3D scene), we propose to integrate the satisfaction of the framing constraints  $f_i$  in the cost function. Therefore,  $\text{cost}(c)$  is given by :

$$\text{cost}(c) = \prod_i f_i(c) \sum_j \text{cost}_{p_i}(c)$$

The cost function  $\text{cost}_{p_i}$  manages to maximize the respect of a property  $p_i$ . If considering the orientation property “In front of  $A$ ”, for example, the best representatives are located along  $A$ ’s front vector  $\vec{F}$  and such that the camera’s orientation is opposite to  $\vec{F}$ . This local search technique could be used as-is to compute solutions of camera composition problems similarly to Olivier *et al.*’s work [OHPL99]. However, the geometrical process maintains the solver in promising regions and reduces the importance of the weight factors associated to each property.

The non-algebraic constraint  $c \in \mathcal{V}$  is managed by a simple evaluation of the implicit function related to a *semantic volume*  $s_v$ . If negative, the configuration lies in  $\mathcal{V}$  and conversely if positive  $c$  is outside  $\mathcal{V}$ .

## 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 against 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 `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. `IsInFrontOf( $A$ )  $\wedge$  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

Property	Script
Framing	Frame Obj xmin xmax ymin ymax
Orientation	Orientation Obj viewing_angle
Projection	Projection Obj proj_distance
Occlusion	Occlusion Occder Occded typ
NoOcclusion	NoOcclusion Obj
On screen	OnScreen Obj
Left/Right	IsLeftOf/IsRightOf Obj1 Obj2
Above/Under	IsAboveOf/IsUnderOf Obj1 Obj2

Table 2: Scripted properties

volumes computed by tessellation. 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 :

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

where  $s_v$  is the previously computed semantic volume and  $G_f(\text{Front}(A))$  the geometric filter that computes the semantic volume related to the property  $\text{Front}(A)$ . If  $\text{Card}(s'_v)$  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  $A$  ( $\text{IsCloseUp}$ ,  $\text{IsMediumLongShot}$ , ...) can be expressed as follows :

$$\{s_{v1}, \dots, s_{vm}\} = \left\{ \begin{array}{l} s_v \cap G_f(\text{IsCloseUp}(A)), \\ \dots, \\ s_v \cap G_f(\text{IsExtremeLongShot}(A)) \end{array} \right\}$$

## 5. Results

The set of properties described in Section 2 is provided via a scripting language as an extension to the *Tcl/Tk* language. It is convenient for testing the application and evaluating the quality of the solutions given to the final user. Moreover it makes the insertion easier into existing softwares by allowing the development of dedicated *plug-ins* and graphical front-ends.

Table 2 exhibits the commands related to most properties. The framing property takes as parameters the coordinates of the bottom left and top right corners of the frame containing the object. Coordinate values less than  $-1$  or more than  $1$  frame objects out of the screen. The orientation property requires an object and a viewing angle among those presented in Section 2.2. The projection property involves an object and a shot class (cf. Figure 2). Occlusion properties associates occluders (*Occder*) with occludeds (*Occded*) and the occlusion type (total or partial).

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

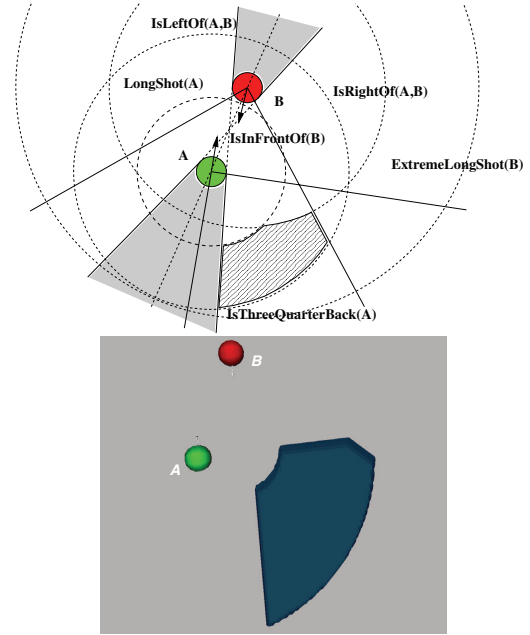


Figure 10: Top view of the search space computed by the over-the-shoulder shot (top) and tessellation of the implicit volume (bottom).



Figure 11: A result of the over-the-shoulder shot.

### 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 10 illustrates the related *semantic volume* and a result is presented in Figure 11. The shot is given by the following declarative script w.r.t. characters  $\$objA$  and  $\$objB$ :

```
#view A on the left
Frame $objA -0.8 -0.3 -1 1
#view B on the right at middle height
Frame $objB 0.3 0.8 -0.5 0.5
Orientation $objA BackRightThreeQuarter
Orientation $objB Front
Projection $objA LongShot
Projection $objB ExtremeLongShot
```



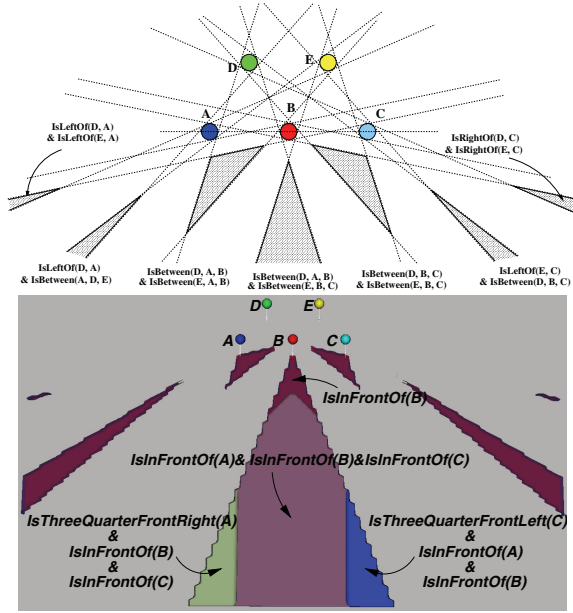


Figure 12: Top view of the *five-framing* shot (top) and tessellation related to possible camera locations (bottom) with further semantic information related to orientations of *A*, *B* and *C*.

## 5.2. The *five-framing* shot

The geometry of the scene is composed of 5 coplanar objects (see top-view Fig. 12). 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 occlusion. Some results are presented in Figure 13. All three shots satisfy the users description and illustrate different classes of solutions.

```
Frame $objA -1.0 -0.33 -1.0 1.0
Frame $objB -0.33 0.33 -1.0 1.0
Frame $objC 0.33 1.0 -1.0 1.0
Frame $objD -1.0 1.0 -1.0 1.0
Frame $objE -1.0 1.0 -1.0 1.0
NoOcclusion $objD
NoOcclusion $objE
```

Table 3 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; 25000 marching cubes are evaluated and examples 1 and 2 generate respectively 240 and 4036 polygons.  $T_N$  is the time spent in the local search with  $\text{nbSteps} = 25$  and  $\text{nbTries} = 25$ . 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.

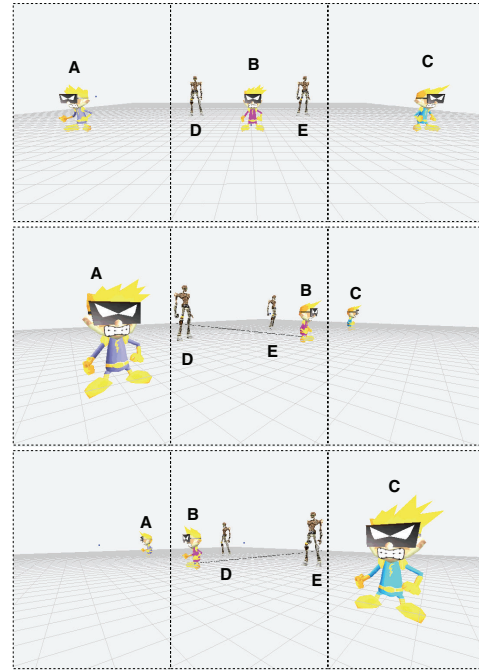


Figure 13: Three shots associated to the *Five framing* shot.

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
				$(0.53+7 \times 0.67)$

Table 3: 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: the cinematographic properties provide semantic volumes containing the possible solutions of the problem through a geometric process that filters non-satisfactory areas of the world. The computation of the boundaries relies on implicit surfaces and avoids intermediate volume calculation. 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 low computational cost.

The use of spheres as object boundaries in the occlusion computation can be criticized. 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. Efficient and precise results can be obtained through hardware buffer rendering capacities [HHS01, HO00] 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.

The computational cost of our approach is related to the number of objects in the scene and to the user's description. Most time is spent in the computation of the number of non-connected volumes which is related to the complexity of the implicit function. We plan to improve this process by computing the intersection of rough axis-aligned boundings of each semantic volume and then simply applying the tessellation on this intersection.

Finally, extending the SSP approach to dynamic scenes is our main objective since being able to characterize possible camera positions for a certain amount of time would greatly simplify the computation of camera paths. A temporal extension is under study featuring implicit sweep volumes. However the extension is not straightforward and the main difficulty lies in reasoning with the time criterion in each volume.

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.

## References

- [AL97] AARTS E., LENSTRA J. K. (Eds.): *Local Search in Combinatorial Optimization*. 1997.
- [Ari76] ARIJON D.: *Grammar of the Film Language*. 1976.
- [BGL98] BARES W. H., GREGOIRE J. P., LESTER J. C.: Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds. In *Procs of AAAI-98/IAAI-98* (1998), pp. 1101–1106.
- [Bli88] BLINN J.: Where am I ? What am I looking at ? *IEEE Computer Graphics & Applications* 8, 7 (Jul 1988), 76–81.
- [BMBT00] BARES W., McDERMOTT S., BOUDREAUX C., THAINIMIT S.: Virtual 3D Camera Composition from Frame Constraints. In *MULTIMEDIA '00: Procs. of the eighth ACM international conference on Multimedia* (2000), pp. 177–186.
- [CAH\*96] CHRISTIANSON D. B., ANDERSON S. E., HE L.-W., SALESIN D., WELD D. S., COHEN M. F.: Declarative Camera Control for Automatic Cinematography. In *Procs of AAAI Conference/IAAI-96, Vol. 1* (1996), pp. 148–155.
- [CLG02] CHRISTIE M., LANGUÉNOU E., GRANVILLIERS L.: Modeling Camera Control with Constrained Hypertubes. In *Procs of CP 2002* (September 9–13 2002).
- [DDP02] DURAND F., DRETTAKIS G., PUECH C.: The 3D Visibility Complex. *ACM Transactions on Graphics* 21, 2 (April 2002), 176–206.
- [DZ95] DRUCKER S. M., ZELTZER D.: CamDroid: a system for implementing intelligent camera control. In *Procs of the 1995 Symposium on Interactive 3D graphics* (1995), pp. 139–144.
- [GRMS01] GOOCH B., REINHARD E., MOULDING C., SHIRLEY P.: Artistic Composition for Image Creation. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (2001), pp. 83–88.
- [GW92] GLEICHER M., WITKIN A.: Through-the-Lens Camera Control. *Computer Graphics* 26, 2 (1992), 331–340.
- [HCS96] HE L.-W., COHEN M. F., SALESIN D. H.: The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. In *Procs of SIGGRAPH 96'* (1996), pp. 217–224.
- [HHS01] HALPER N., HELBING R., STROTHOTTE T.: A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence. In *Procs. of the Eurographics'2001 Conference* (2001), vol. 20, pp. 174–183.
- [HO00] HALPER N., OLIVIER P.: CAMPLAN: A Camera Planning Agent. In *Smart Graphics 2000 AAAI Spring Symposium* (March 2000), pp. 92–100.
- [JL98] JARDILLIER F., LANGUÉNOU E.: Screen-Space Constraints for Camera Movements: the Virtual Cameraman. In *Procs. of EUROGRAPHICS-98* (1998), vol. 17, pp. 175–186.
- [KvD79] KOENDERINK J., VAN DOORN J.: The internal representation of solid shape with respect to vision. *Biological Cybernetics* 32 (1979), 211–216.
- [OHPL99] OLIVIER P., HALPER N., PICKERING J., LUNA P.: Visual Composition as Optimisation. In *AISB Symposium on AI and Creativity in Entertainment and Visual Art* (1999), pp. 22–30.
- [PD90] PLANTINGA H., DYER C. R.: Visibility, Occlusion, and the aspect graph. *International Journal of Computer Vision* 5, 2 (Nov 1990), 137–160.
- [Pic02] PICKERING J. H.: *Intelligent Camera Planning for Computer Graphics*. PhD thesis, Department of Computer Science, University of York, September 2002.
- [WW89] WYVILL B., WYVILL G.: Field functions for implicit surfaces. *The Visual Computer* 5, 1&2 (1989), 75–82.