# Learning good views through intelligent galleries

THALES VIEIRA[1], ALEX BORDIGNON[1], ADELAILSON PEIXOTO[2], GEOVAN TAVARES[1], HÉLIO LOPES[1], LUIZ VELHO[3] AND THOMAS LEWINER[1]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Department of Mathematics — Universidade Federal de Alagoas — Maceió — Brazil
[3] Department of Computer Science — Instituto Nacional de Matemática Pura e Aplicada — Rio de Janeiro — Brazil
http://www.mat.puc-rio.br/~{thalesv, alexlaier, tavares, lopes, tomlew}.
adelailson@pos.mat.ufal.br. http://www.impa.br/~lvelho.

**Abstract.** The definition of a good view of a 3D scene is highly subjective and strongly depends on both the scene content and the 3D application. Usually, camera placement is performed directly by the user, and that task may be laborious. Existing automatic virtual cameras guide the user by optimizing a single rule, e.g. maximizing the visible silhouette or the projected area. However, the use of a static pre-defined rule may fail in respecting the user's subjective understanding of the scene. This work introduces *intelligent design galleries*, a learning approach for subjective problems such as the camera placement. The interaction of the user with a design gallery teaches a statistical learning machine. The trained machine can then imitate the user, either by pre-selecting good views or by automatically placing the camera. The learning process relies on a Support Vector Machines for classifying views from a collection of descriptors, ranging from 2D image quality to 3D features visibility. Experiments of the automatic camera placement demonstrate that the proposed technique is efficient and handles scenes with occlusion and high depth complexities. This work also includes user validations of the intelligent gallery interface.
**Keywords:** ***Learning. Camera Positioning. Virtual Camera. Intelligent Camera. Good View.***

**Figure 1:** *Overview of the intelligent gallery interface (from left to right): 1- The scene is displayed to the user as a gallery of views. 2 - From the user subjective selection of good and bad views, the machine learns user's preferences. 3- Upon validation, a new gallery is generated by genetic reproduction of the selected views. 4- Then, the machine automatically selects and orders the views in the gallery from its previous training. 5- The user can correct the automatic selection, repeating the reproduction until converging to a satisfactory final view.*

## 1 Introduction

Although we live in a three-dimensional world, our perception of physical realities remains mediated primarily by our visual senses, which are essentially bi-dimensional. Such dimensionality reduction usually implies in a loss of information. Humans deal with this sensorial limitation by resorting to knowledge and contextual data. A particular experience that we acquire as we explore the world is to position ourselves at vantage points which reveal the intrinsic 3D structure of objects of interest. Photographers develop this knowledge, learning where to place a camera to take a perfect shot.

With the advent of computers and advances in graphics technologies, we work more and more with three-dimensional data using only two-dimensional visual interfaces, e.g. positioning virtual 3D camera through 2D interaction. In this context, the photographer task of producing a good view can be laborious for both 3D specialists and inexperienced users: On one side, engineers and scientists using high-end graphics for CAD/CAM, medical imaging or simulations typically need to render large collections or sequences of 3D scenes, guaranteeing that each shot captures the relevant details for their specific applications. This re-

Preprint MAT. 09/08, communicated on March $31^{th}$, 2008 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of Eurographics 2009: Computer Graphics Forum, volume 28, number 2, pp. 717-726, Blackwell 2009.

petitive task unfortunately requires advanced knowledge. On the other side, 3D content has already come to the masses, pledging for more intuitive interfaces. A great help for those laborious tasks is a tool for the *automatic* generation of good views, i.e., 2D images that show 3D content with minimal loss, revealing the information desired by the user.

However, automatic viewpoint selection is a very hard task for two main reasons: first, there are many factors which must be taken into account to generate a good view, ranging from perceptual issues to geometric analysis of the objects in the scene; second, the "best" view heavily depends on what we want to see, and thus, it is both a user and application-dependent task.

In this paper we face the challenge of producing good 2D views by computer using the same approach as humans: *learning*! To do so, we introduce *intelligent galleries* (see Figure 1), an interface that learns from the user and evolves with continuous usage. It incorporates learning into design galleries, and can generate the desired views automatically or semi-automatically.

**Related Work**

Automatic camera placement has received a constant interest over the past decades [3], with mainly two strategies according to [25]: visibility maximization and view descriptor optimization.

The pioneering work of Kamada and Kawai [31] computes good views for 3D objects by minimizing the number of degenerated faces in orthographic projection. This simple criterion has been enhanced to entail a larger amount of visible three-dimensional details and extended for perspective projection in the works of Barral *et al.* [22] and Gómez *et al.* [27].

Then, two-dimensional visibility criteria have been introduced in Plemenos and Benayada [32], maximizing the non-occluded projected area. Extending that work, Vázquez *et al.* [34] propose a measure called viewpoint entropy, defining the goodness of a view as the amount of information it provides to the observer. This entropy considers both the projected area and the number of visible faces from the 3D model normal dispersion. However for terrains, where all the normals point in similar directions, this entropy loses its pertinence. To workaround this, Stoev and Straßer [33] propose to maximize an average of the entropy and the maximal scene depth. More complex definitions of distinctive regions have been proposed based on saliency learning [6] and direct database search [14] Similar descriptors have been used to generate a collection of representative views for a single scene, based on surface visibility [4] or on the notion of stable saliency-based views [21]. More view collections techniques are related to the next best view problem [17, 5]. However, this work focuses on the single best view problem.

More recently, Lee *et al.* [7] introduce mesh saliencies as a multiscale curvature-based measure of regional importance. As an application, they define good views by maximizing the total saliency of visible regions, with a heuristic gradient-descent optimization. Polonsky *et al.* [13] also measure the goodness of view by introducing 2D and 3D view descriptors such as projected surface area, curvature, silhouette and topological complexity. Different good views are obtained by maximizing these descriptors individually. Then, Sokolov *et al.* [19, 18] define good views minimizing a distance between the visible area distribution of the polygons in the image and the corresponding distribution of areas in the 3D scene. Podolak *et al.* [12] propose to use symmetry to define good projections and orientations of 3D scenes. Of particular attention to us, the authors of [13] conclude that combinations of such descriptors could lead to better results. Our approach deduces such combinations from a learning process.

Those techniques have been further derived using art and design criteria, which is particularly relevant in animation contexts. Blanz *et al.* [24] define factors that influence human preferred views for 3D objects, pointing out that the relevance of each factor depends on the task, the object geometry and the object familiarity. Gooch *et al.* [28] developed a computer system based on these factors and heuristics used by artists. Beyond static scene criteria, cinematography constraints have been proposed to generate good camera trajectories [23, 1, 26, 29, 30]. These criteria would be very useful to guide the user in choosing good views and producing more elegant animations, although we focus here on direct learning from user's experience rather than pre-established design rules.

Design galleries are described in the work of Marks *et al.* [8] as an efficient tool for assisting the user for highly non-linear problems with subjective goals. We extend here this proposal into *intelligent galleries*, which incorporates learning into design galleries. This allows assisting the user by incorporating its own experience.

**Contributions**

This work proposes to learn good views of a 3D scene from user's experience. As opposed to visibility or single view descriptor optimization approaches, it allows combining several criteria ranging from 3D scene content to 2D image arrangement. Our learning approach turns this combination context-dependent. In particular, it does not rely on predefined criteria, which are hard to define for such subjective problem as camera placement.

We propose here an intelligent gallery interface, which incorporates learning into design galleries. This leads to a simple interface that continuously learns from the user. The learning machine can then imitate the user to spare him from repetitive camera placement. The statistical learning is based on Support Vector Machines, and we further analyze its dependency on each view descriptor. From this analysis, we automatically adjust its parameters for each training set. Preliminary results show the effectiveness of the proposed interface in static and dynamic contexts.

The corresponding work was published in the proceedings of Eurographics 2009: Computer Graphics Forum, volume 28, number 2, pp. 717-726, Blackwell 2009.

## 2 Technical Overview

The definition of good camera from the user's experience relies on three main ingredients: an adapted statistical learning technique described in section 3; an efficient computation of image- and object-space descriptors detailed in section 4; and an intelligent gallery interface introduced in section 5.

***Descriptors.*** For each camera around an input 3D scene, a set of descriptors of the rendered view is computed. These descriptors either reflect the 2D image quality — e.g., the proportion of the image covered by the scene, the complexity of its silhouette curve or the alignment of the scene with the image directions — or the visibility of 3D features — e.g., mesh saliency, connected components or velocity for fluid simulations. We propose an efficient way to compute these descriptors, allowing interactive performances.

***Learning.*** The user's selections in the intelligent gallery serve as training set for the learning machine. More precisely, each view is represented as a vector $\mathbf{v}$ containing the descriptors values. Each selected view is associated to a class $\mathsf{s} = \pm 1$, depending whether the user marked it as a good or bad view. This way, we model the user selection as a discrete mapping $\mathbf{v} \mapsto f(\mathbf{v}) = \mathsf{s}$. We use an SVM formulation to incrementally build a function $\mathbf{v} \mapsto \hat{f}(\mathbf{v}) = \hat{\mathsf{s}}$ that statistically approximates the user selection. We analyze this function $\hat{f}$ and adjust its parameters for each specific training.

***Intelligent Galleries.*** The gallery initially collects views from a uniform sampling of the camera position space. A previously trained learning machine marks some of those views good or bad, which the user may correct. Upon validation a new gallery is generated by genetic reproduction of the good views, where the genes are the coordinates of the camera: position and orientation. This correction/validation process is repeated until the user obtains his best view of the 3D model, or until the genetic optimization converges according to the learning machine. The views in the gallery are ordered according to the learning machine, guiding the user to quickly select subjective good views.

With this formulation, the machine interacts with the gallery interface the same way the user would do and learns from his corrections. This can provide a fully automatic camera placement, which suits for repetitive camera placement such as animation frames. The same framework can work in semi-automatic mode, where the user controls and occasionally corrects the machine selections. Finally, it can be used only to guide the user, by proposing at privileged gallery locations the machine's best views.

## 3 Statistical Learning

Our intelligent gallery learns continuously from the user's choices. To implement the learning, we use a derivation of the *Support Vector Machine* (SVM) binary classifier, since it provides a non-linear classification and copes well with small training sets. SVM received a lot of attention in the machine learning community since it is optimal in the sense of the *VC* statistical learning theory [20]. We refer the reader to the book of Smola and Schölkopf [15] for a complete introduction to SVM techniques. This section briefly describes some basics of SVM and how we derived it for ordering views from a binary training. We further introduce a simple analysis of the SVM results, and an automatic parameter adjustment based on this analysis.

***SVM-based learning.*** During training, the SVM learning machine receives as input set of pairs $\{(\mathbf{v}_1, \mathsf{s}_1), (\mathbf{v}_2, \mathsf{s}_2) \ldots\}$, each pair corresponding to a view submitted to user selection. The vector $\mathbf{v}_i \in \mathbb{R}^k$ contains the view descriptors and the binary number $\mathsf{s}_i = \pm 1$ codes if the user marked the view as good or bad. The SVM produces a classifier $\mathbf{v} \mapsto sign\left(\hat{f}(\mathbf{v})\right) = \hat{\mathsf{s}}$ that imitates the user's behavior. This classifying function $\hat{f}$ corresponds to a linear function, but in a high or infinite-dimensional space $\mathcal{F}$:

$$\hat{f}(\mathbf{v}) = \textstyle\sum_j \alpha_j \, \mathsf{s}_j \left\langle \varphi(\mathbf{v}_j), \varphi(\mathbf{v}) \right\rangle + b \quad . \qquad (1)$$

The function $\varphi\colon \mathbb{R}^k \to \mathcal{F}$ maps the descriptors space non-linearly to the space $\mathcal{F}$, leading to a non-linear classifier $\hat{f}$ as shown in Figure 2. In this work, we used a Gaussian kernel, i.e.

$$\langle \varphi(\mathbf{v_1}), \varphi(\mathbf{v_2}) \rangle = \exp\left(-\tfrac{\|\mathbf{v_2}-\mathbf{v_1}\|^2}{2\sigma^2}\right)$$

with penalty weight $C = 1$ for outliers. Note that this classifying function uses all the components of the optimized vectors $\mathbf{v}_i$ at once, i.e. it combines all the different view descriptors in one function.

The SVM classifier $\hat{f}$ is optimized for its sign to approximate the user marks $\mathsf{s}$. However, we want to automatically order the views from the best one to the worst, and we thus need a continuous result. To do so, we directly use the function $\hat{f}$, interpreting that a high positive value of $\hat{f}(\mathbf{v})$ means that the view is good with high confidence.

***Classifying function analysis.*** Equation (1) defines the classifier $\hat{f}$ from the training set, and not directly from the descriptors. The dependency to a particular descriptor (i.e. a component of $\mathbf{v}$) is thus hard to interpret directly. We can observe this dependency visually, plotting the classification $\hat{f}$ versus the values of a particular descriptor on the training set (see Figure 3). Ideally, that plot for an influent descriptor would present a strong dependency (see Figure 3 (top)), while a less influent descriptor would have a random horizontal distribution of values (see Figure 3 (bottom)).

We would like to relate the non-linear function ($\hat{f}$) separately with each descriptor. However, non-linear analysis such as Kernel PCA [16] is not appropriate, since it would find relations with $\varphi\left(\hat{f}(x)\right)$. To avoid computationally expensive feature analysis or dimension reductions, we adopt a heuristic approach: linearizing $\hat{f}$ close to the average of the descriptors. This leads to a linear approximation of the non-linear dependency between a given descriptor and the classification. If the variance around the linear approximation is small, the dependency is well characterized by the

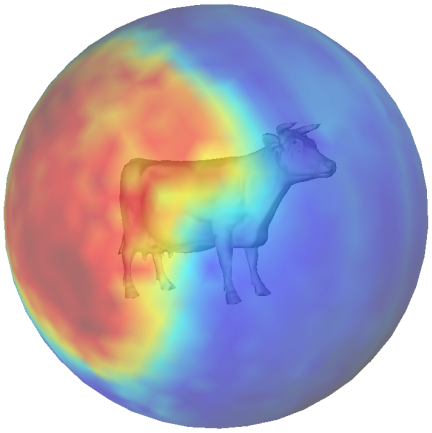


**Figure 2:** *Good camera positions (in red) from training the side view of the cow model (Figure 7). The estimation of the good cameras emphasize the non-linearity of the classifier.*

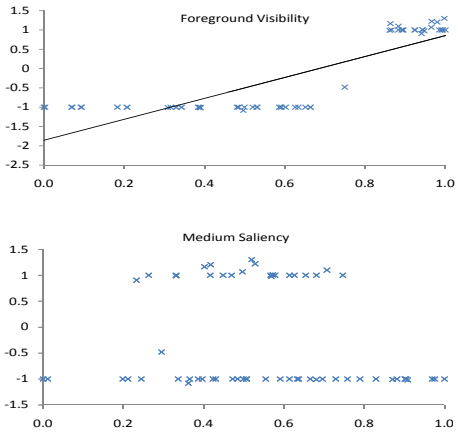



**Figure 3:** *SVM prediction $\hat{f}$ vs descriptor value, for the training of Figure 2: the saliency (bottom) has a low correlation with the $\hat{f}$, contrarily to the foreground visibility (top).*

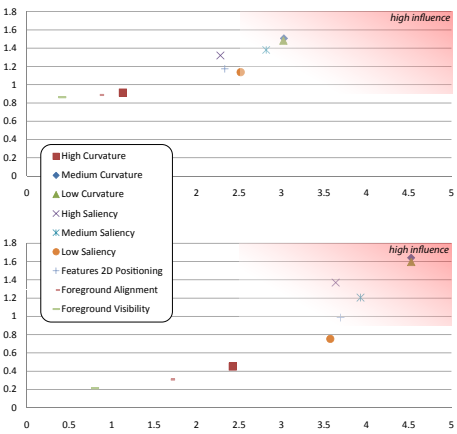



**Figure 4:** *Optimization of kernel parameter $\sigma$ for the training of Figure 9 (2−variance vs. correlation): $\sigma = 8$ (bottom) maximizes the descriptors' influences, compared to $\sigma = 1$ (top).*

slope of the regression line. If the variance is high, the linearization may not be valid. This leads to a characterization of the descriptor influence by the regression slope $\tau$ and the variance $\eta$ around it as $\tau(2-\eta)$ (see Figure 4).

***Automatic parameters adjustments.*** The adjustment of the parameter $\sigma$ used in the SVM classifier is done in order to maximize the influence of the parameters

$$\sum_{d \in Descriptors} \tau_d(2-\eta_d) \quad ,$$

turning the learning more robust by forcing to use more descriptors. To do so, we use a bisection method on the values of $\sigma$ to maximize the number of descriptors on the top right of the plots (see Figure 4). This process runs in less than a few seconds since the training set is fixed and small.

## 4 View Descriptors

In this section, we detail the collection of image-space (2D) and object-space (3D) descriptors used to qualify a view. Their values on a given view define the vector $\mathbf{v}$, which serves as input for learning. These view descriptors correspond to commonsense good views, and their combination obtained by the classifier may thus help characterizing subjective good views. Several of the following descriptors have already been used for camera positioning [11, 10, 13, 7]. We propose variations of those and further add application-specific descriptors such as velocity for simulation and groups in character animation. We further propose an efficient way to compute those descriptors for reaching interactive rates. The view descriptors refer either to the object-space, counting how much of geometric features is visible from the given camera, or to the image-space, evaluating the information contained in the rendered image (see Figure 5).

***Efficient Descriptor Computation*** Each object-space descriptor is a number associated to a feature, which is computed during pre-processing. For example, high curvature is a 3D feature of the mesh, and the associated descriptor counts how much of high curvature parts are visible for a given view. To do so, each vertex of the 3D scene is associated to a 32-bits `RGBA` color that encodes the 3D features present at that vertex. For example, if the curvature of a vertex is high and if the first bit of the red color encodes the high curvature feature, then the color associated to that vertex will have an odd red component. For a given camera, the scene is rendered with those colors. Note that several passes can be performed if the feature coding requires more than 32 bits. The object-space descriptor associated to a feature is then efficiently computed as the number of pixels in the rendered image whose `RBGA` color encodes that feature. This counting and the computation of the 2D descriptors are done by traversing the color buffer.

| property | feature | descriptor |
|---|---|---|
| curvature | low mesh curvature | $\Sigma$ visible low curv. |
| | medium mesh curvature | $\Sigma$ visible medium curv. |
| | high mesh curvature | $\Sigma$ visible high curv. |
| saliency | low mesh saliency | $\Sigma$ visible low sal. |
| | medium mesh saliency | $\Sigma$ visible medium sal. |
| | high mesh saliency | $\Sigma$ visible high sal. |
| patches | patch identifier | $\Sigma$ visible patches |
| connected components | component identifier | $\Sigma$ visible 2D area / / comp. 3D area |
| | component size | $\Sigma$ visible 2D area × × comp. 3D area |

**Table 1:** *Generic object-space descriptors.*

The corresponding work was published in the proceedings of Eurographics 2009: Computer Graphics Forum, volume 28, number 2, pp. 717-726, Blackwell 2009.

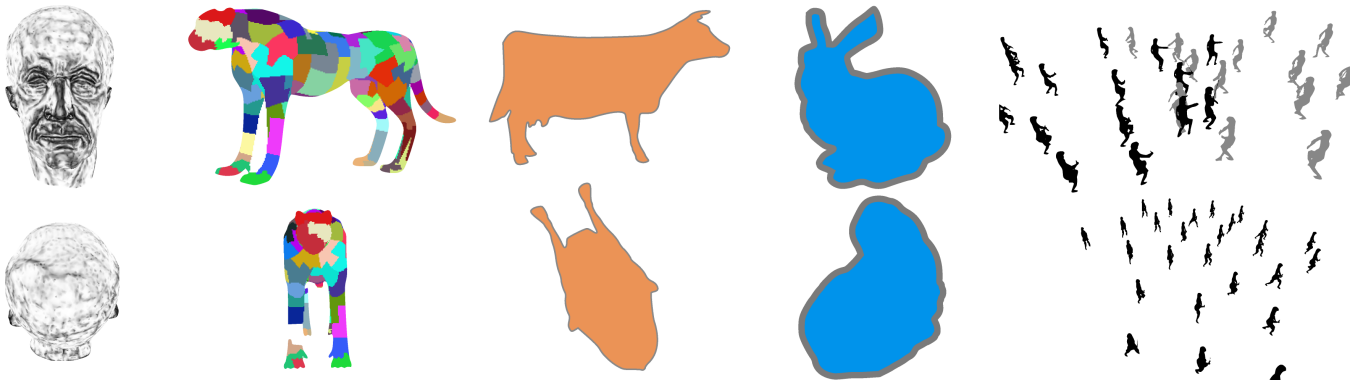


**Figure 5:** *The view descriptors help the learning machine in characterizing good views. From left to right: mean curvature (black to white scale), patches visibility (one color per patch), alignment in image plane, silhouette complexity and character groups in an animation context. The top row shows views with high values of each descriptor, and corresponds to commonsense good views, as opposed to the bottom row.*

**(a) Object-space descriptors**

In the current implementation of this work, we compute the following 3D features: mean curvature, saliency, connected components and visible patches (see Table 1 and Figure 5). For specific applications such as fluid simulation or animation, other properties such as velocity or human character groups are added to the descriptors.

***Geometric features.*** As geometric properties, we compute the mean curvature using the work of Meyer *et al.* [9] and the saliency using the work of Lee *et al.* [7]. For fluid simulation we add a 3D property corresponding to the magnitude of the velocity vector. Since those 3D properties have very different scales depending on the scene observed, we normalize them by equalizing their histogram on the first gallery and scaling their values to the unit interval. Each of these properties is then associated to three features: low values (0 to $\frac{1}{3}$), medium values ($\frac{1}{3}$ to $\frac{2}{3}$) and high values ($\frac{2}{3}$ to 1).

***Parts visibility.*** We use different partitions of the 3D scene. Connected components characterize the distinct scene objects, while patches of approximately equal area describe how much of the whole scene is visible (see Figure 5). The patches are computed with a greedy strategy: the set of patches is initialized to the set of triangles and the smallest ones are merged by union-find operations. Descriptors are obtained by counting the visible 2D patch size relative to the original 3D area, and also the 2D component size multiplied by the 3D area, the latter emphasizing big components. For animation, the character groups induce another partition of the scene. For normalization purposes, the set of parts visibility descriptors counts the visible pixels of each connected component, patch or character group, divided by its 3D area.

**(b) Image-space descriptors**

For image-space descriptors, we use the area of the projected scene, the complexity of the scene silhouette, the distribution of 3D curvatures in the image plane and the alignment of the projected connected components with the image's rectangular border (see Figure 5). These image-space descriptors are computed directly on the rendered image, where we identify the 3D connected components in the foreground from the color coding of the 3D features. We extract the foreground's contours by a simple continuation method.

The complexity of the silhouette is defined as the total integral of its curvature. The distribution of the 3D curvatures is computed by the average distance of the projected vertices to the center of the image, weighted by their mean curvature. A similar descriptor is obtained by replacing the curvature by the component size. Finally, the alignment of the connected components is obtained by linear regression from the projected vertices' position. This descriptor is the average of the regression lines inclinations, weighted by the number of vertices of each component.

## 5 Intelligent Gallery Interface

In this work, we introduce intelligent galleries, which extend design gallery interfaces [8]. This gallery interface is populated by a genetic reproduction strategy, and incorporates the learning machine described in section 3. In the gallery, the user marks good and bad views by a simple click, and the learning machine continuously evolves with those selections. The main actions can be reduced to four options (see Figure 6): 1 - load a new model; 2 - automatically select and order best views according to the current learning machine; 3 - let the user correct and validate the current selection and produce a new gallery from it; 4 - fine render a final view. The user can also use advanced features, such as saving and loading learning machines, load sequence of models for automatic camera placement, etc. This section details how the galleries are initialized and reproduced and how the user interacts with the learning machine.

***Initial population.*** When the user loads a model, the first gallery must offer him a good coverage of the 3D model. To do so with a reduced number of views in the gallery, we restrict the camera positions and orientation to a subset of the whole possible camera positions and orientation $\mathbb{R}^3 \times \mathbb{S}^2$. We distinguish terrain-like scenes, such as fluid simulation

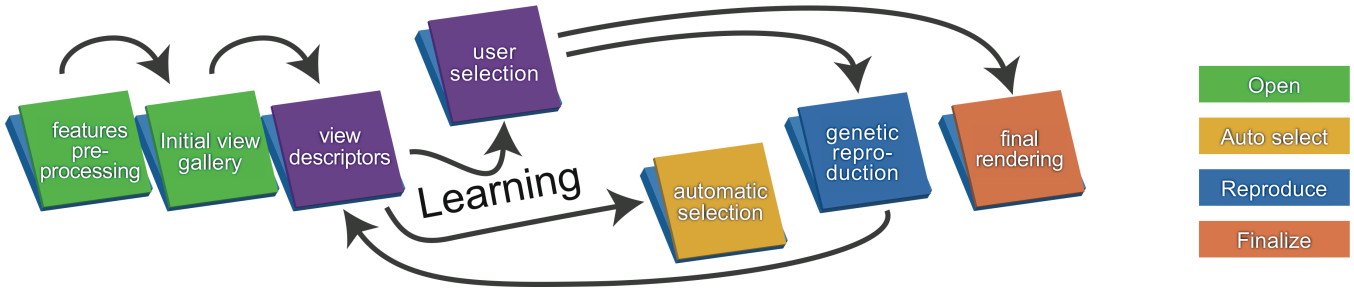



**Figure 6:** *The intelligent gallery interface allows the user to teach and correct a learning machine while quickly reaching his best view. It has mainly four actions: 1- open a 3D scene, which initializes the gallery; 2- automatically select and order good and bad views using the current training; 3- correct or validate the selection and generate a finer gallery by genetic reproduction; 4- use a particular view for final rendering. The training set accumulates all the user's selections.*

or character animation, from single objects, such as animal or vehicles models. For terrains, the camera positions are restricted to a parallelepiped above the terrain and the orientation are chosen inside a cone of $45°$ aperture. For single objects, the cameras are oriented towards the center of the object, and their positions are constrained to the surface of a sphere around the object. The radius of the sphere is set to twice the diagonal of the object's bounding box. In both cases, the cameras of the initial gallery are positioned on a regular lattice of these subsets of $\mathbb{R}^3 \times \mathbb{S}^2$. The perspective angle of the camera was fixed to $45°$.

***Interactive genetic optimization.*** The user can correct the automatically selected good and bad views in the gallery by a simple click. Upon validation, a new gallery is generated by genetic reproduction of the selected good views. More precisely, each camera position and orientation is represented by a chromosome vector. This chromosome has $5$ components for terrain-like models: 3 for the position and 2 for the orientation; and 2 components for single objects for the spherical parameterization of the camera positions. The cameras of the selected views are reproduced by generating random convex combinations of those chromosome vectors. Each selected camera randomly reproduces with four other cameras. Our genetic reproduction thus generates a new gallery of twice as much camera as previously selected good views. It is repeated until either the user gets satisfied or the learning machine's choices converges.

***Learning incorporation.*** The learning process tries to detect the views that the user would select. It must be trained first, so at least the first selection of the gallery must be manual or obtained by a pre-defined training. Then, at each selection validation, the views of the gallery are inserted into the training set and classified with $\mathtt{s} = \pm 1$, depending on whether the user marked the view as good or bad. The training is continuous, and can also be accumulated from different scenes. After each validation, the learning machine produces a new classifying function from the training set. Since the total number of views in the gallery is small, the SVM-based optimization is instantaneous, so that the user can use the automatic selection at any time. Finally, we guide the user by putting at the start of the gallery (top left) the best views according to the classifying function.

## 6 Results

The proposed method for learning good views from user's experience has been tested in three different contexts: 1- automatically reproduce user's experience on similar models; 2- check the interface's efficiency from recording inexperienced user interaction; 3- generate automatic camera trajectories for 3D animations from user's training on few frames.

We used three groups of models: static usual CG models (see Figures 7, 9 and 10), challenging scenes such as knots (see Figure 11) or high depth complex scene (see Figure 12), and frames of 3D animations (see Figures 8 and 14). The fluid simulation and character animation were generated using Blender [2] and each contains one per-vertex extra information: the fluid velocity and the character' identifier.

### (a) Automatic placement

***Reproducibility.*** We first check that the learning strategy combined with 3D and 2D descriptors is able to reproduce simple definitions of best views. To do so, we used the cow as a prototype for the animal models and navigated through the gallery to obtain a side view of it (see Figure 7). Then, we applied this training to the other animals in automatic mode (no user intervention). The results are satisfactory even for the horse, which has a slightly different pose. From the analysis of the learning described in section 3, the machine learned mainly from the foreground visibility and saliency.

**Figure 7:** *Automatic placement (no user intervention) from training for side views of the cow: the classifier illustrated in Figure 2 succeeds in automatically reproducing this best view from only one training model.*

The corresponding work was published in the proceedings of Eurographics 2009: Computer Graphics Forum, volume 28, number 2, pp. 717-726, Blackwell 2009.

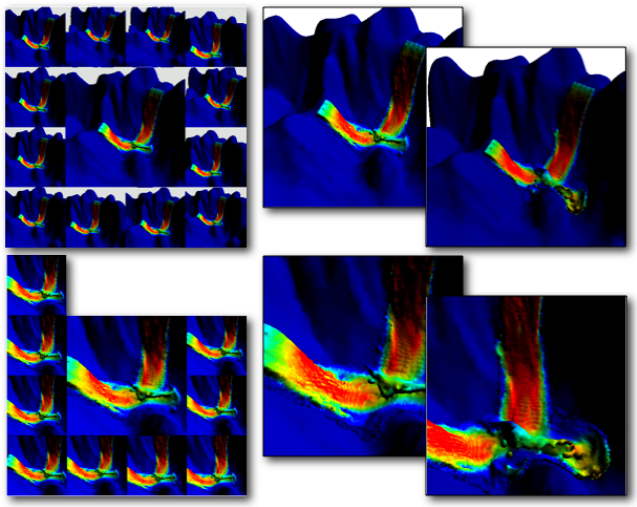


**Figure 8:** *Selections of best view of a fluid simulation from a designer (top left) and from a fluid researcher (bottom left) point of view. Each learning machine leads to automatic views of other frames of the simulation, according to the respective user's choice.*

We further checked the stability of the learning by counting the false positive and false negative when removing randomly 20% of the training set and testing on the removed samples (out-of-sample test). The average false positive for 10 random removals is 2.7% and 3.0% of false positive for the training of Figure 9. For the training of Figure 7, there are 5.7% false positive and 0.2% false negative.

***Subjectivity.*** We then check that our interface takes into account the user's subjectivity (see Figure 8). To do so, we submitted one frame of the fluid simulation to two different users: a graphic designer, letting him free to choose his best view and a fluid expert, asking him to focus on the fluid flow features. The resulting views and selection sequences obtained through the gallery interface were different. We then applied the training obtained from each user on another frame of the simulation and observed that the learning succeeded in selecting new views coherent with each training.

***Comparison.*** We illustrate some result of our technique and of previous works, from Polonsky *et al.* [13] and Lee *et al.* [7] in Figure 10. We used a unique training for all sets, obtained for the front view of the David's head and Max Planck's head (see Figure 9). Using this training in the intelligent galleries without user intervention leads to good automatic view selection. As compared to single criterion decisions [13, 7], descriptor combinations fits to a wider range of models.

***Challenging scenes.*** We further test our technique on more challenging scenes where simple criterion would be hard to determine explicitly. For example, 3D embeddings of knots are usually challenging to render, while our intelligent gallery is able to select clear views of complex knots from a single training on the trefoil knot (see Figure 11).

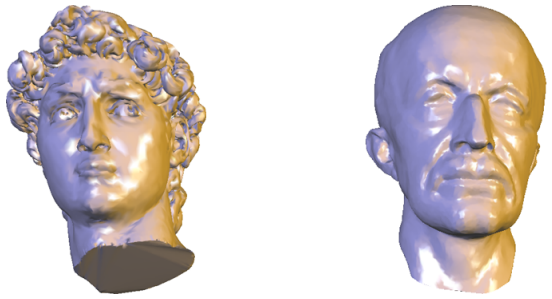


**Figure 9:** *A continuous training, started on David's head to reach the left view, and completed on the Max Plank's head.*

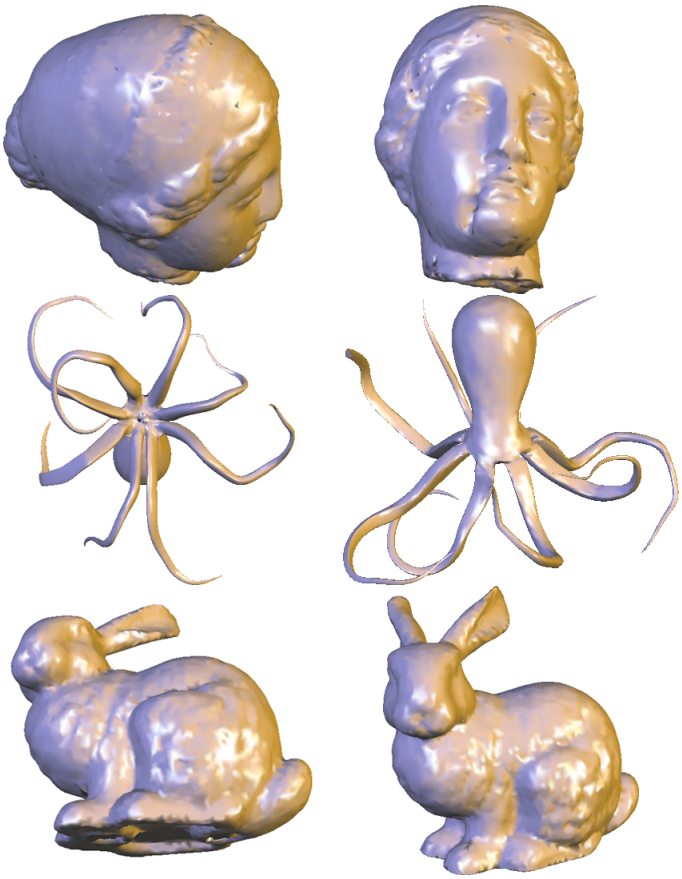


**Figure 10:** *Comparison of our automatic view selection (right column) from the training of Figure 9 with previous techniques: area maximization [13] (top left), silhouette length maximization [13] (middle left) and saliency-based camera positioning [7] (bottom left).*

Our technique handles nicely high depth scene and occlusion, for example when looking across a forest for the front of an occluded animal (see Figure 12): a single training even on a low density forest leads to good view selection of the animal front in more dense forests. Here, the occlusion is forced by maintaining the camera in a 2D plane. The combination of occlusion with the front constraint is a complex criterion, which is correctly interpreted by our intelligent gallery.

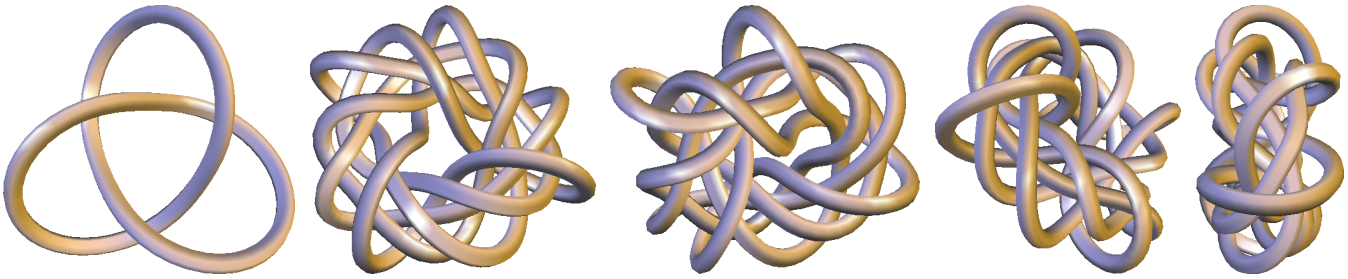


**Figure 11:** *Training on a single trefoil knot (left) leads to a coherent classification of good views for a more complex knot (decreasing view quality from left to right).*

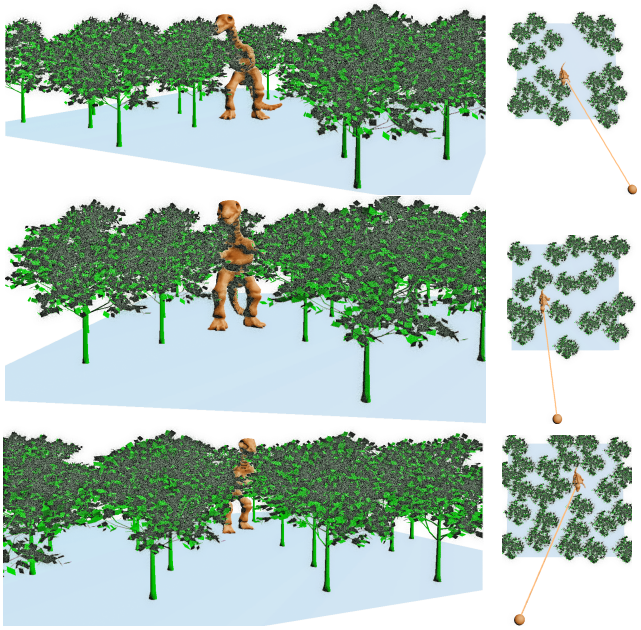
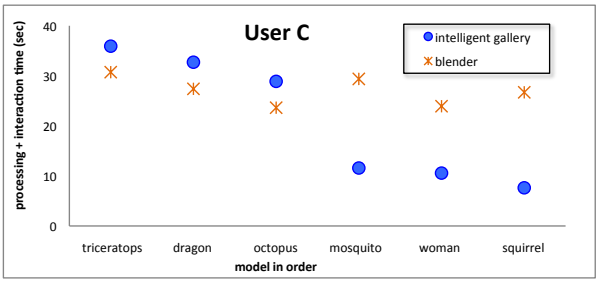


**Figure 12:** *Training on a high depth scene to catch the occluded dinosaur front (top). The automatic selection still catches the dinosaur's front, although the new random forest contains more trees (middle and bottom).*




| Model | # of Views | Pre-proc | Proc | Inter-act | Com-pare |
|---|---|---|---|---|---|
| cow (0) | 60 (4) | 2.140 | 8.280 | 73.8 | 41.3 |
| doberman (1) | 44 (2) | 4.218 | 6.595 | 50.5 | 40.9 |
| cat (2) | 20 (0) | 8.875 | 3.281 | 8.5 | 42.6 |
| dinosaur (3) | 20 (0) | 17.172 | 3.469 | 4.7 | 33.8 |
| cheetah (4) | 20 (0) | 4.703 | 3.234 | 6.3 | 41.3 |
| knot 3.1c (0) | 36 (2) | 6.890 | 4.671 | 56.1 | 29.2 |
| knot 7.7c (1) | 20 (0) | 6.672 | 3.235 | 10.0 | 22.5 |
| knot 6.13c (2) | 20 (0) | 3.375 | 3.250 | 11.1 | 26.6 |
| knot 138k (3) | 20 (0) | 3.735 | 3.250 | 7.1 | 21.9 |
| triceratops (0) | 28 (1) | 2.313 | 4.906 | 35.9 | 30.7 |
| dragon (1) | 24 (1) | 21.328 | 3.250 | 32.7 | 27.4 |
| octopus (2) | 24 (1) | 9.531 | 4.375 | 28.9 | 23.6 |
| mosquito (3) | 20 (0) | 5.141 | 3.234 | 11.6 | 29.4 |
| woman (4) | 20 (0) | 4.359 | 3.297 | 10.6 | 23.9 |
| squirrel (5) | 20 (0) | 5.281 | 3.328 | 7.6 | 26.7 |

**Table 2:** *Efficiency of our intelligent gallery interface. The total number of views generated by the galleries is completed by the number of galleries generated. The timings are expressed in seconds, and the comparison was done using a standard 3D interface [2] (crosses). The users adapted to the interface on two or three models (dots) and use it faster than traditional ones.*

### (b) Intelligent interface efficiency

We proposed the intelligent gallery interface to users with no particular experience in camera placement. The users chose themselves which models to work on. We asked them to select a good view of the model through the intelligent gallery, and then to obtain a similar view through a classical interface (i.e. Blender). The learning curves and timings (see Table 2) show that, after one or two models, the intelligent gallery accelerates the user's camera placement.

### (c) Application to 3D video generation

We further test our method on 3D animation sequences, where the camera placement must be repeated for each frame. We trained a few frames of the sequence and the intelligent gallery then automatically selected the best views for the key frames of the animation. For example in the 30 seconds character animation of Figure 14, we trained the first and last frame. We then applied it on key frames every 2 seconds. For testing purposes, we did not include any advanced constraints for the camera trajectory, such as respecting cinematographic rules. We only avoid the camera to oscillate by picking across key frames, among the automatically selected good views, the one that minimize the image displacement. Between key frames, the cameras are interpolated by cubic splines, using standard tools of Blender [2]. A

similar test has been performed on the fluid simulation of Figure 8, training on four frames adding the velocity magnitude and setting the key frame interval to 1.6 seconds (see the video).

**(d) Limitations**

The proposed interface has mainly two limitations. First, the intelligent gallery is designed to learn from the user interaction on a small set of models and expects the user to apply on models with similar objectives, e.g. a training set for a side view should not be applied to obtain a front view. For example on Figure 13, using the training of Figure 9 on the knot of Figure 11 or vice-versa leads to undesired views. Performing an out-of-sample test of the training of Figure 9 but applying on the samples of Figure 7 leads to $4.7\%$ false positive and $52.0\%$ of false negative: the front views are discarded in the cow model, while being praised in the other training. The reverse experiment leads to $8.5\%$ false positive and $53.5\%$ of false negative.

The second is related to the number of parameters ($k$) of the virtual camera. For the genetic reproduction to be able to generate all the possible cameras, it needs at least to have an initial population of size $2^k$ (the corner of the hypercube), and much more in practice. A complete camera with position, direction, perspective angle, viewport... would require an initial gallery too large for a reasonable interaction. To overcome this problem, new cameras need to be inserted independently of the genetic reproduction, e.g. sampling the view space [13] or using a next-best-view approach [17, 5].

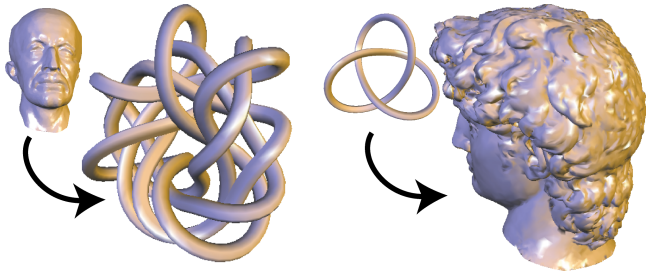


**Figure 13:** *When transferring a training across very different applications, here from the training of Figure 9 to the on of Figure 11 or vice-versa, the intelligent gallery may not weight correctly the model features.*

## 7 Next Steps

We proposed a new technique for automatic and semi-automatic determination of good views that learns from the user's subjective criteria. The introduction of intelligent galleries allows a smooth interface where the user incrementally teaches the machine while achieving his own best view. Unlike usual techniques, our learning approach can automatically obtain different good views of the same scene depending on the user and the application. We further used the proposed technique for camera placement in 3D sequences with simple trajectory restrictions, obtaining nice animations.

The actual implementation of the technique used a restricted set of descriptors for static 3D scenes. A greater set of descriptors, including existing ones from design research, would improve the learning process. Our framework can incorporate new descriptors with almost no alteration of speed since the classification is instantaneous. As further research, the use of intelligent galleries may improve other applications of design galleries, such as light sources positioning, transfer function design or isosurface selection. A complete user study will also be performed.

## Acknowledgments

The authors wish to thank Fabiano Petronetto for constructive discussions, Eduardo Telles, Debora Lima, Bernardo Ribeiro, Clarissa Marques and Afonso Paiva for their help with the interface, and Marcos Lage and Rener Castro for help with the coding. This work was supported in part by a grant from the CNPq (Universal MCT/CNPq, Productivity scholarship, Instituto Nacional de Ciência e Tecnologia), FAPERJ (Jovem Cientista) and CAPES.

## References

[1] W. H. Bares. A Photographic Composition Assistant for Intelligent Virtual 3D Camera Systems. In *Smart Graphics*, pages 172–183, 2006.

[2] Blender. Blender foundation. www.blender.org.

[3] M. Christie, P. Olivier and J.-M. Normand. Camera Control in Computer Graphics. *Computer Graphics Forum*, 27(7), 2008.

[4] S. Fleishman, D. Cohen-Or and D. Lischinski. Automatic Camera Placement for Image-Based Modeling. In *Pacific Graphics*, pages 12–20. IEEE, 1999.

[5] M. Hachet, F. Decle, S. Knodel and P. Guitton. Navidget for Easy 3D Camera Positioning from 2D Inputs. In *3D User Interfaces*, pages 83–89. IEEE, 2008.

[6] H. Laga and M. Nakajima. Supervised Learning of Salient 2D Views of 3D Models. In *Nicograph*, 2007.

[7] C. H. Lee, A. Varshney and D. W. Jacobs. Mesh saliency. In *Siggraph*. ACM, 2005.

[8] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims and S. M. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Siggraph*, pages 389–400. ACM, 1997.

[9] M. Meyer, M. Desbrun, P. Schröder and A. Barr. Discrete differential–geometry operators for triangulated 2–manifolds. In *VisMathematics*, pages 35–57, 2002.

[10] D. L. Page, A. F. Koschan, S. R. Sukumar, B. Roui-Abidi and M. A. Abidi. Shape analysis algorithm based on information theory. In *Image Processing*, volume 1, pages 229–232, 2003.

[11] O. E. M. Pastor. Visibility Preprocessing Using Spherical Sampling of Polygonal Patches, 2002.

[12] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz and T. Funkhouser. A planar-reflective symmetry transform for 3D shapes. In *Siggraph*, pages 549–559. ACM, 2006.

[13] O. Polonsky, G. Patanè, S. Biasotti, C. Gotsman and M. Spagnuolo. What's in an image? *The Visual Computer*, 21(8–10):840–847, 2005.

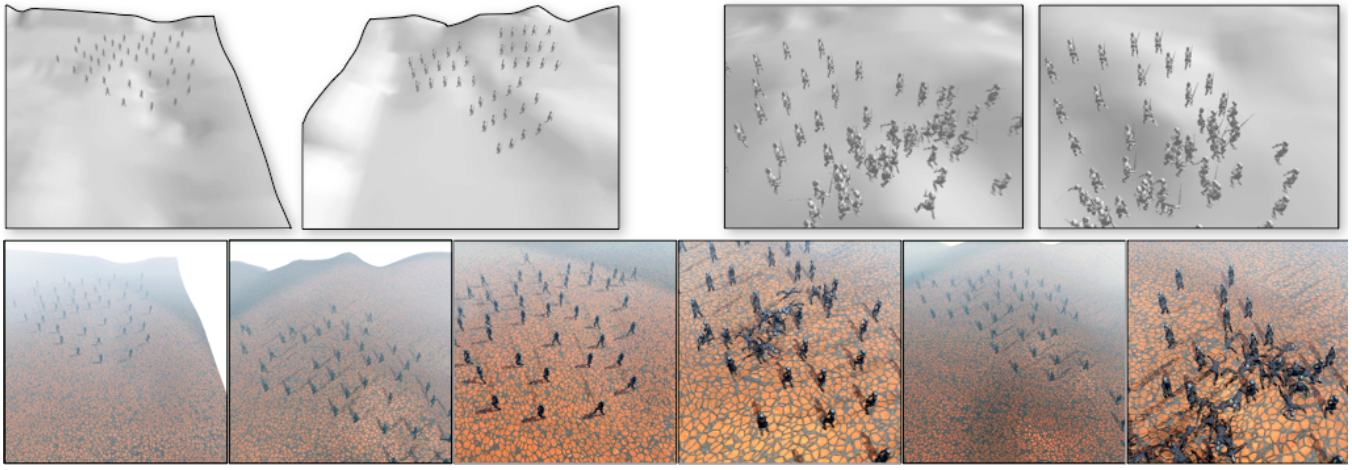


**Figure 14:** *Training of a battle scene from only two frames: the views of the first frame (top left), where the character groups are separated, are selected as good cameras; while the views of the last frame (top right) where the groups are equally mixed, are also selected as good cameras. With this only training, the cameras for the key frames of the animation are automatically positioned using our method.*


[14] P. Shilane and T. Funkhouser. Distinctive regions of 3D surfaces. *Transactions on Graphics*, 26(2):7, 2008.

[15] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT, 2002.

[16] B. Schölkopf, A. J. Smola and K.-R. Müller. *Kernel principal component analysis*, pages 327–352. MIT, 1999.

[17] K. Singh and R. Balakrishnan. Visualizing 3D scenes using non-linear projections and data mining of previous camera movements. In *Afrigraph*, pages 41–48. ACM, 2004.

[18] D. Sokolov and D. P. K. Tamine. Viewpoint quality and global scene exploration strategies. In *Grapp*, 2006.

[19] D. Sokolov and D. Plemenos. Virtual world explorations by using topological and semantic knowledge. *The Visual Computer*, 24(3):173–185, 2008.

[20] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[21] H. Yamauchi, W. Saleem, S. Yoshizawa, Z. Karni, A. Belyaev and H.-P. Seidel. Towards Stable and Salient Multi-View Representation of 3D Shapes. In *Shape Modeling and Applications*, page 40. IEEE, 2006.

[22] P. Barral, D. Plemenos and G. Dorme. Scene understanding techniques using a virtual camera. In *Eurographics*, 2000. Short paper.

[23] W. Bares, S. McDermott, C. Boudreaux and S. Thainimit. Virtual 3D camera composition from frame constraints. In *Multimedia*, pages 177–186. ACM, 2000.

[24] V. Blanz, M. J. Tarr, H. H. Bülthoff and T. Vetter. What object attributes determine canonical views. *Perception*, 28(5):575–600, 1999.

[25] M. Christie, R. Machap, J.-M. Normand, P. Olivier and J. Pickering. Virtual Camera Planning: A Survey. In *Smart Graphics*, pages 40–52, 2005.

[26] S. M. Drucker and D. Zeltzer. Intelligent Camera Control in a Virtual Environment. In *Graphics Interface*, pages 190–199, 1994.

[27] F. Gómez, F. Hurtado, J. A. Sellarès and G. Toussaint. Nice Perspective Projections. *Visual Communication and Image Representation*, 12(4):387–400, 2001.

[28] B. Gooch, E. Reinhard, C. Moulding and P. Shirley. Artistic Composition for Image Creation. In *Rendering Techniques*, pages 83–88. Springer, 2001.

[29] N. Halper and P. Oliver. CamPlan: A Camera Planning Agent. In *Smart Graphics*, 2000.

[30] L.-W. He, M. F. Cohen and D. H. Salesin. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *Siggraph*, pages 217–224. ACM, 1996.

[31] T. Kamada and S. Kawai. A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, Image Processing*, 41(1):43–56, 1988.

[32] D. Plemenos and M. Benayada. Intelligent Display in Scene Modeling: New Techniques to Automatically Compute Good Views. In *GraphiCon*, 1996.

[33] S. Stoev and W. Straßer. A Case Study on Automatic Camera Placement and Motion for Visualizing Historical Data. In *Visualization*. IEEE, 2002.

[34] P.-P. Vázquez, M. Feixas, M. Sbert and W. Heidrich. Viewpoint Selection using Viewpoint Entropy. In *Vision Modeling and Visualization*, pages 273–280. Aka, 2001.


The corresponding work was published in the proceedings of Eurographics 2009: Computer Graphics Forum, volume 28, number 2, pp. 717-726, Blackwell 2009.