3D Dynamic Grouping for Guided Stylization

Hedlena Bezerra, Elmar Eisemann, Xavier Décoret, Joëlle Thollot

To cite this version:

Hedlena Bezerra, Elmar Eisemann, Xavier Décoret, Joëlle Thollot. 3D Dynamic Grouping for Guided Stylization. NPAR '08: Proceedings of the 6th International Symposium on Non-photorealistic Animation and Rendering, Jun 2008, Annecy, France. pp.89-95, 10.1145/1377980.1377998 . inria-00345231

HAL Id: inria-00345231

https://hal.inria.fr/inria-00345231

Submitted on 9 Dec 2008

3D Dynamic Grouping For Guided Stylization

Hedlena Bezerra   Elmar Eisemann   Xavier Décoret[*]   Joëlle Thollot
Grenoble University/ARTIS[†]INRIA

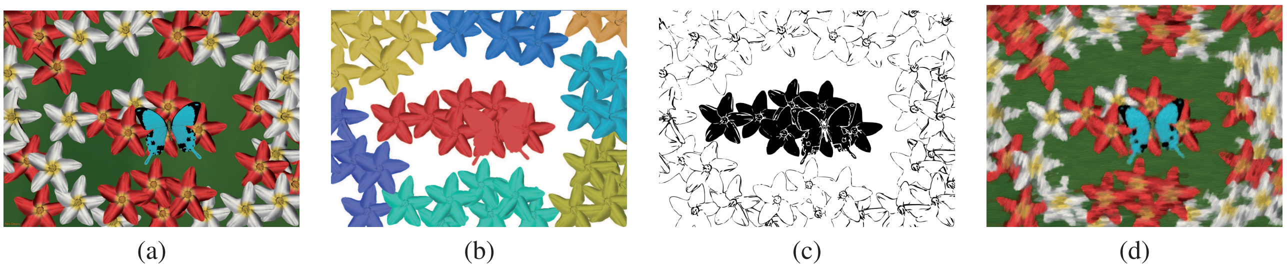

Figure 1: *An example illustrating our 3D clustering to drive two different stylizations: (a) In a meadow of flowers, we want a butterfly to influence surrounding objects. (b) We cluster the scene according to position. (c) Grouping information is used to drive a line rendering style where the butterfly makes its cluster appear black; (d) The same grouping information can be used to guide a completely different style: the shape of the clusters determines the size and orientation of the stylized strokes in this painterly rendering.*

Abstract

In art, grouping plays a major role to convey relationships of objects and the organization of scenes. It is separated from style, which only determines how groups are rendered to achieve a visual abstraction of the depicted scene. We present an approach to interactively derive grouping information in a dynamic 3D scene. Our solution is simple and general. The resulting grouping information can be used as an input to any "rendering style".

We provide an efficient solution based on an extended mean-shift algorithm customized by user-defined criteria. The resulting system is temporally coherent and real-time. The computational cost is largely determined by the scene's structure rather than by its geometric complexity.

1 Introduction

One fundamental question in expressive rendering (also called Non-Photorealistic rendering, or NPR) is: how to emphasize certain objects of the depicted scene? Answering this question allows the creation of efficient visual representations of a scene that communicate a specific message or are easier to understand.

Artists have done this for centuries. Depending on the medium, the artist has access to different methods of stylizations to produce such a representation. If the artist uses a camera, she can set the focus so that certain objects will be blurred (e.g. in the background). If she uses a pen-and-ink technique, she may decide to simplify the silhouette of certain objects, and to detail that of others with precise strokes. Furthermore, color can be used to make certain objects less visible by desaturation or averaging.

All these stylization techniques abstract or simplify the visual representation of the scene according to what we call a level of abstraction (LOA). Such an LOA is determined for each object *before* applying the stylization technique. The choice of LOA depends on the artist's goal and often the viewpoint. When dealing with interactive scenes, specifying the LOA can be very tedious. Our motivation is to address this problem with a system that accounts for dynamically varying scenes while allowing intuitive controls.

One possibility to derive an LOA uses the distance from the observer. The aerial perspective effect seen in nature is one example of this, where contrast decreases with distance. It is also common in artistic composition to give greater detail to foreground elements. These considerations motivated the use of depth in many previous techniques for abstraction and stylization [Barla et al. 2006; Kowalski et al. 2001; Markosian et al. 2000]. Although depth is a very common criterion for abstraction, it is not the only one that is involved in clustering processes. Information like normals, colors [Kolliopoulos et al. 2006], or region of interests according to the viewer [DeCarlo and Santella 2002] are combined to drive the process.

In this paper, we present a real-time technique to *cluster* a dynamic 3D scene. The user can devise clustering strategies taking into account any attribute of the scene. Once the clustering is done, a temporally coherent LOA is assigned to each group and used to drive the stylization. We show how to achieve an automatic, yet controllable output that can then be used with any rendering style. This stylization can be a function of the criteria used to cluster, but also of other object information available from the cluster's members. Figure 1 shows an example where clustering, LOA and stylization are based on different attributes.

In the following sections we review related work, describe the main steps of our algorithm in a simple situation and show how we can extend it to handle complex behavior. We conclude by discussing

[*]X.Décoret is now working at Phoenix Interactive
[†]ARTIS, is part of the LJK research laboratory and of INRIA Rhône-Alpes. LJK is UMR 5524, a joint research laboratory of CNRS, INRIA and Grenoble University.

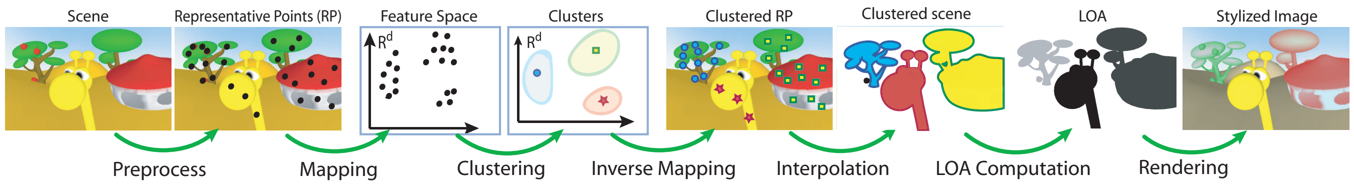

Figure 2: *Overview: The first step is done in a preprocess, all the others are dynamic. Finally, any stylization can be used for rendering.*

our method and show comparisons with previous approaches.

2 Related work

Abstraction has been involved in many NPR works and often in form of a particular style definition. An exhaustive overview would be well beyond the scope of this paper. We concentrate here on previous methods addressing the determination of levels of abstraction.

When dealing with images (either stills or videos), abstraction is usually done by segmenting the image into regions. This clustering in 2D can be used to control stylization. Examples can be found in [Wen et al. 2006] for color sketches, [Bousseau et al. 2000] for watercolor, and [Wang et al. 2004b; Winnemoller et al. 2006] for video stylization. In general, the level of abstraction is chosen manually. Several approaches provide ways to compute LOAs automatically. De Carlo and Santella [2002] use eye tracking, Lecot and Levy [2006] use a saliency map to guide the LOA computation. Bangham et al. [Bangham et al. 2003] associate LOAs based on the distance to the center of attention, which is estimated using a scale space approach. Orzan et al. [Orzan et al. 2007] also use a scale-space analysis to abstract a photograph according to a measure of importance. Although these approaches give interesting results, they do not lead to information about what parts constitute an object, or a group of objects, and they work on static data.

If the input is a 3D scene, one typical way to guide abstraction is to rely on depth. The farther an object is away from the viewpoint, the less detailed it is. Examples of usage can be found in [Markosian et al. 2000] based on graftals, or in [Barla et al. 2006] for toon shading. This type of simplification is classical in Computer Graphics when level of details are involved [Luebke et al. 2003]. Cole et al. [2006] present a temporally coherent system that relies on focal points or planes to deemphasize parts of a stylized 3D scene based on the distance in image or world space. This effectively guides the attention of an observer to important areas of the image. In contrast to a uniform distance, our approach can provide a scene adapted shape of the focus area and seeks to account for general criteria.

Several works rely on handmade segmentations of a 3D scene. In [Luft and Deussen 2004] geometric proximity is used to define bounding shapes for trees, where the size of the bounding volumes is chosen by the user. When several trees are present in the scene, meaningful groups are defined by hand. Balzer and Deussen [2007] present a clustering algorithm for graph visualization. Kowalski et al. [2001] show various techniques for abstraction, and use the notion of manually defined groups to produce convincing results. They also rely on the heuristic that objects far away should be grouped together. The user is able to modify the default behavior by adding some semantic importance information in the system. Our work extends this approach by offering an automatic clustering that could be taken as an input of their system. It generalizes the approach by giving more flexibility to the grouping strategies.

For most of these approaches, once the clustering is established, it does not evolve over time, since it is handmade or given as an input. In our approach, we concentrate on dynamic grouping.

Kolliopoulos et al.'s [2006] segmentation technique is similar to our method in that they also use clustering. They segment a rendered 3D scene in image space while taking into account 3D information (e.g. depth), and user provided object IDs. The major limitation of this method is that it does not take hidden geometry into account. Therefore regions that get disconnected by a nearby occluder can cause severe problems. Moreover the information of segmentation is not given at an object level, and thus it is harder to assure continuous evolution over time. Our work shares the same goal, and by using a fast object-space clustering, we are able to do this interactively and let the user design her styles according to any information available in the scene (including occluded geometry).

3 Overview

Figure 2 shows an overview of our approach. The input of our algorithm is an interactively manipulated 3D scene. The clustering is done on *representative points* that sample the scene and are selected in a preprocess (Sec 4.1). At run time, these points are clustered using a mean-shift algorithm (Sec 4.2) in a *feature space* specified by the user (Sec 6). Information from the clustered representative points is then remapped and interpolated for any point of the scene. Finally, the clustering information can be used as input to any renderer to obtain the final abstracted image (Sec 5).

4 Grouping

We begin by describing the clustering process in a simple case: how to cluster according to the position of objects in camera space. This is the most popular grouping in art because it takes into account position in x, y and depth of the projected scene. We demonstrate later that our method allows to cluster based on other attributes.

4.1 Representative Points

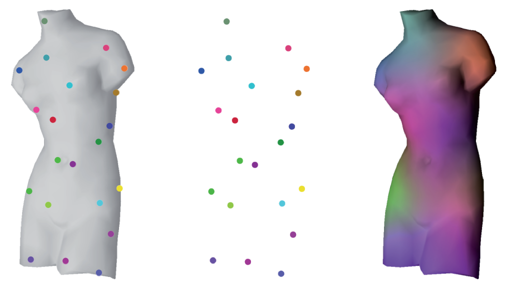

Figure 3: *Sampling (left), representative points (middle), weights on mesh (right).*

Clustering scenes with many points is a time consuming process.

To maintain interactive framerates, we work on a subset of scene points called *representative points*, computed during a preprocess.

The selection of these points is a trade-off between speed (directly related to the number of points) and accuracy of the final clustering. We propose to use a sampling method that is independent of the geometry (not related to the tessellation), and controlled by a single parameter. We select points on the objects surface using a Poisson sphere sampling (related to Poisson sphere and disc distributions [Cook 1986; Lagae and Dutré 2006]).

Starting with a radius r of the size of the object, we add samples until any additional sample would break the constraint that its distance to all other samples is at least r. Then we decrease the radius, and continue the process until we reach $r \leq \epsilon$. The resulting samples are our representative points. The single parameter ϵ controls the precision of the sampling.

The clustering itself is performed on representative points only. After grouping, the value v_P of a given attribute at a point P is obtained via a weighted interpolation of the attribute's values v_i at the representative points:

$$v_P := \frac{\sum w(P,R_i)v_i}{\sum w(P,R_i)}\text{ , with } w(P,R) := e^{-\|P-R\|^2} \tag{1}$$

where w is a weighting function that measures the proximity between two points (Figure 3, right).

Evaluating these weights at run-time would be quite costly. Our solution is to store the weights for each vertex in texture coordinates that can be used to efficiently evaluate the influence of each representative point. Currently, 32 RGBA texture coordinates can be used, allowing 128 weights per vertex, which proved largely sufficient in practice. Very large objects can be subdivided until 128 is enough for each sub-object.

4.2 Clustering

There exist many clustering algorithms in Computer Graphics. Many of them have been devised for acceleration purposes, such as octrees, KD-trees, regular grids, etc. They impose a mostly uniform structure, and cannot handle arbitrarily shaped clusters. Their goal is typically to divide the geometry in a balanced way to speed up hierarchical geometric tests. In our case, the goal of the clusters is to reveal the inherent structure of the scene. Thus, the resulting number of clusters cannot be known in advance, and should be derived for a given scale. Clusters of arbitrary shape should be handled. Furthermore, we will see that it is advantageous to be able to derive a cluster center. One possible approach that meets the above constraints is the mean shift algorithm [Comaniciu and Meer 2002]. We chose it because of its simplicity and potential to be accelerated in our context.

The mean-shift algorithm clusters points in a feature space. Thus, it is capable of finding groups of points that share sufficient similarity. Even though there is a more general formulation introduced in [Wang et al. 2004a] that represents an anisotropic extension, we focus on the original formulation.

Given n data points $x_i \in \mathbb{R}^d$, the original paper [Comaniciu and Meer 2002] defines a density function f. Based on a kernel function K, that describes the likelihood for points to cluster with their surrounding and a scope H determining the scale on which clusters will be established. Intuitively, it is a superposition of local weighting functions around sample points. Formally, it is given by:

$$f(x) := \frac{1}{nH^d}\sum_{i=1}^{n} K\left(\frac{x-x_i}{H}\right) \tag{2}$$

Even though several conditions are imposed on K, there is a vast variety of choices [Wand and Jones 1995]. We use a radially symmetric function:

$$K(x) := (2\pi)^{-d/2}e^{-\frac{1}{2}\|x\|^2} \tag{3}$$

In that case, H corresponds to the variance of the Gaussian.

The mean-shift moves each point iteratively along f's steepest ascent to a local maximum, called the *cluster leader* (or cluster mode). Interestingly, a locally weighted mean (depending on K) results in a position that lies along this gradient direction. An iterative process can thus be used, where the current position is replaced by the mean value of the neighborhood (hence the name *mean-shift*). The convergence of this method has been proved in [Comaniciu and Meer 2002]. Advanced solutions for a faster evaluation exist [Paris and Durand 2007] but are out of the scope of this article.

The mean-shift leads to clusters whose number does not have to be specified in advance, revealing the clustering information inherent to the scene at a given scope H. It can be seen as the scale at which we try to find clusters in the scene. Finding the right scale is a decision we leave to the user, because it is a semantic definition. An automatic approach to derive a reasonable size could be possible by exploring scale space techniques [Ter Haar Romeny 2003], but would be costly and might disagree with the artistic choice.

The advantage of performing clustering in feature space is that this definition is completely independent of the scene or animation. This is an elegant solution that has rather predictable behavior without further user interaction.

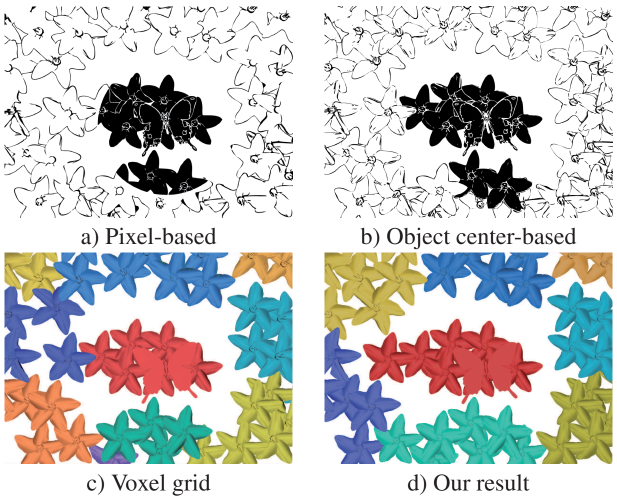

Figure 4: *Comparison of our clustering to naive approaches.*

Figure 4 shows a comparison between our clustering and several naive strategies. The scene is the one shown Figure 1 with a circular arrangement of flowers. A pixel-based (a), as well as an object center-based distance measure (b), do not isolate the circular arrangement. Clustering according to a grid (c) causes artificial separations between neighboring elements. Our approach (d) produces clusters that are more natural because they relate to the scene's structure.

Figure 5 shows an example inspired by Kolliopoulos et al. [2006] in order to compare our object space approach to their image space method. The style we use is similar to Kolliopoulos' cartoon style: the color of a cluster is the mean color of the objects contained in the cluster. Using an image space method creates two clusters in the background due to the occlusion. The chosen style has a large

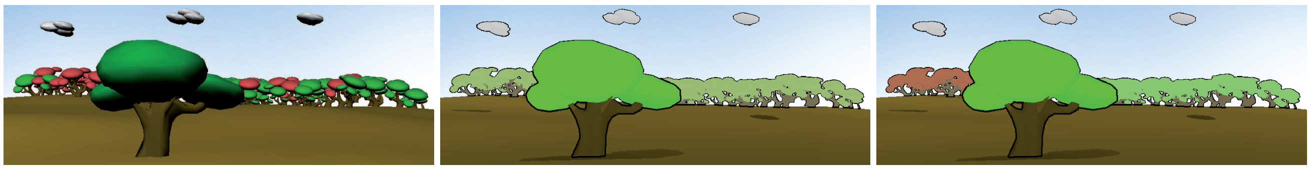

Figure 5: *Comparison of our clustering to an image-based approach: (Left) original scene; (Middle) our result; (Right) using only visible points creates two distinct clusters in the background.*

impact on the final image. With our approach the background trees stay clustered whatever may happen in the foreground.

Nevertheless, including invisible geometry may also produce undesirable results. For example, if a house contains people, should it behave differently in the clustering process than an empty house? There is no automatic way to predict what an artist would desire. The strength of our approach is to give the user the possibility to have more control over the grouping behavior. In the aforementioned example, one could exclude the people inside the house as long as its door remains shut and add them into the clustering process once the door opens. We present different ways to vary the influence of sample points in section 7.

4.3 Temporal coherence

It is important to note that, in general, the mean-shift clustering is not stable: a slight perturbation can lead to a different classification. Figure 6 shows this situation. There are two small clusters in blue and red. The white point in the middle does not provide enough density to create its own cluster, neither does it change the global density function in a way that would fuse the red and the blue cluster. This point can go either way, or even stay unclustered. A tiny perturbation can lead to a leap towards a rather distant position resulting in popping artifacts.

Nevertheless, ensuring temporal coherence in our case is relatively straightforward. We work in object space so we have information about all representative points and their history independently of their visibility. For image-based approaches, this step represents an important challenge. Our solution is to integrate a smoothing process with exponential decay. The problem of Figure 6 occurs only at local minima of the density function where the gradient vanishes. To ensure that a point does not directly jump from one cluster to the other without passing through an intermediate state, we examine whether it is located near a minimum. This can be done based on the computed mean-shift value. It is proportional to the gradient and we simply test whether the initial displacement is small enough to be neglected. In practice, a small constant value performs well as pointed out in [Wang et al. 2004b]. We used 10^{-5} throughout this paper, but it could be chosen according to the scene's total density.

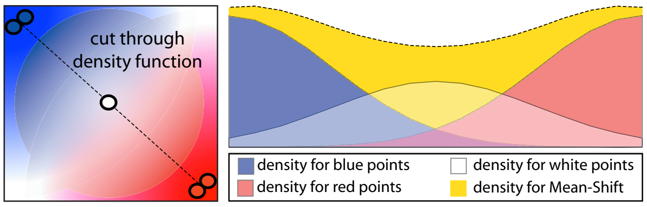

Figure 6: *Slight perturbations can lead to a different clustering. The white point could join the red or the blue cluster.*

5 Level of abstraction and stylization

As mentioned in the overview, once the grouping is established, for each cluster, a level of abstraction (LOA) is derived that guides the stylization. This value (vector or scalar) is a set of significant attributes, necessary to determine the final rendering. Its choice is application-dependent and thus left to the user.

Here, we present two strategies to derive LOAs along with a corresponding stylization example.

LOA computation using the cluster leader A simple way to compute the LOA is to use the cluster leader. Indeed, this feature point best represents the attributes of the group. It also lies inside the cluster defined by the density function. This is not necessarily the case for an average.

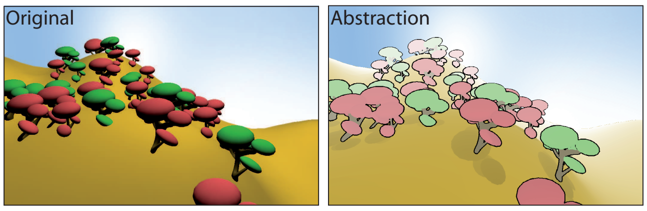

Figure 7: *The original scene is clustered using the position in camera space as feature space; The LOA is computed from the depth of the cluster leader; The style is cartoon-like with aerial perspective.*

Figure 7 shows an example that uses this strategy. The trees are clustered according to their camera space projections. The LOA of a cluster is the z coordinate of its leader. Two simple styles are used: color is desaturated according to the LOA and an outline is drawn around the clusters based on the discontinuities of the cluster LOA in image space.

LOA computation using a specific scene attribute A second strategy is to consider a specific attribute and merge it inside a cluster. A standard choice is to compute the average or the maximum. One direct application is to emphasize particular elements of the scene to attract the observer's attention. One way to achieve this is by attaching a special attribute to each object indicating its importance. The LOA of a cluster is then chosen based on whether it contains an important element.

Figure 8 shows an example. The butterfly is important and colorizes surrounding elements of the scene. This importance is encoded in an attribute (1 for the butterfly, 0 elsewhere). Clustering is performed in camera space. Due to the view space projection, objects farther away will create larger clusters, whereas objects near the viewpoint will be treated individually. This makes the butterfly act on larger groups at a distance. The LOA is given by the maximum of the importance value inside a cluster. A non-zero value indicates that color should be applied whereas zero results in a gray-scale output.

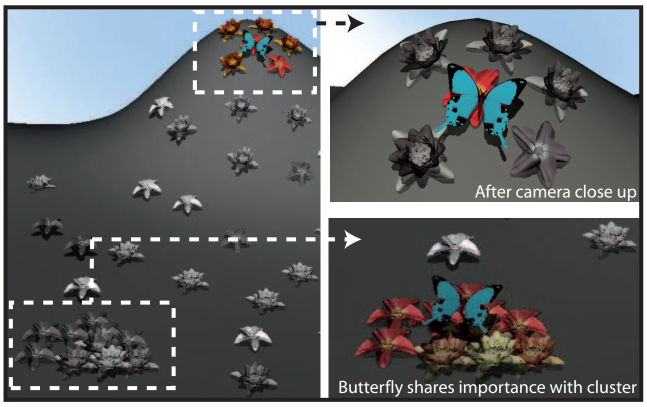

Figure 8: *In this example the LOA is based on the presence of the butterfly in the cluster. Using screen position as a feature, the clustering evolves according to the viewpoint. Camera zoom underneath the upper arrow: a flower bush when seen from far away is separated into individual flowers when zooming.*

6 Feature space

Although we illustrated the mean shift clustering in the context of position, it works on a feature space of arbitrary dimension. This allows our system to consider any attributes, or their combination, to define the clustering. For example, we can take color into account. Let's assume that feature points have (x,y,z,r,g,b) attributes. We can map them to a feature space using a perspective projection for position and LUV for color.

Combining attributes like color and position in a clustering process might not seem very intuitive, but by defining a mapping function, the clustering behavior can be predicted. For example, if color is supposed to be relatively more important than position, a scaling of the color dimensions takes this into account.

Figure 9 shows an example. Here, we use a simple style where the object's color is computed as the average of the objects colors in the cluster. Using only color leads to two clusters (red and orange on Figure 9-(b)). Similarly if only position is taken into account, three clusters are naturally obtained (Figure 9-(c)). When using both color and position, we end up with a compromise where a yellow apple among the red apples will neither be clustered with close apples (due to color), nor other yellow apples (due to distance) (Figure 9-(d)). This example demonstrates that multiple features can easily be taken into account in a controllable way.

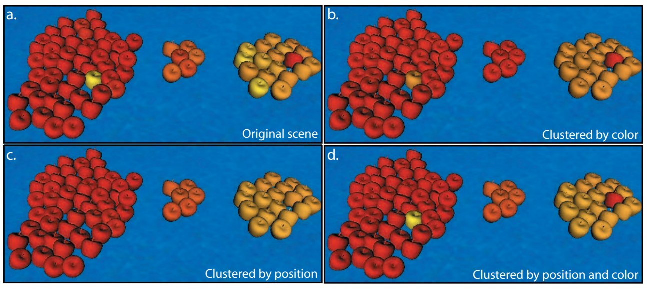

Figure 9: *(a) A scene with different colors, is clustered using different attributes: (b) color, (c) position, (d) color and position.*

7 Mean-Shift Extension

To allow more flexibility in the clustering process, we propose a slight modification of the original mean-shift to include weighting of points in the process. It adds two degrees of freedom. First, it makes the scale H vary with each point x_i. Second, a weight ω_i is defined for each point. These modifications are driven by the fact that the user should be able to specify that certain elements, with a particular semantics, are more likely to yield separate clusters, or will more or less "attract" their neighbors in feature space. This is typically useful in a scene that has elements at different scales. On a field with cows, each flower might be grouped with its neighboring flowers, but the scale is intuitively smaller than the level at which cows are grouped. Some elements might be important and imply the use of a larger radius; a king that attracts the attention of its servants earlier than a simple knight. We refer to the video for further examples of how these parameters can be applied.

The reformulated density function is:

$$f(x) := 1/n \sum_{i=1}^{n} \omega_i K\left(\frac{x-x_i}{H_i}\right) \tag{4}$$

With these weighted instead of classical means the method still converges. A proof is given in Appendix A.

Figure 10 shows an example for the influence of the weight. Here, the color of the cluster corresponds to the color of the leader defined by a ramp from green to red along the x-axis of the image. The only considered attribute is position and the top image shows the result when all characters have similar weights: we obtain one cluster with the leader in its center. The bottom image shows the result of increasing only the weight of the left giraffe. All the giraffes are clustered again but the leader is centered on the left. This mechanism is very intuitive and easily manipulable via an interface where the user selects an object and moves a slider to assign a weight. The modifications are directly visible allowing an interactive adjustment.

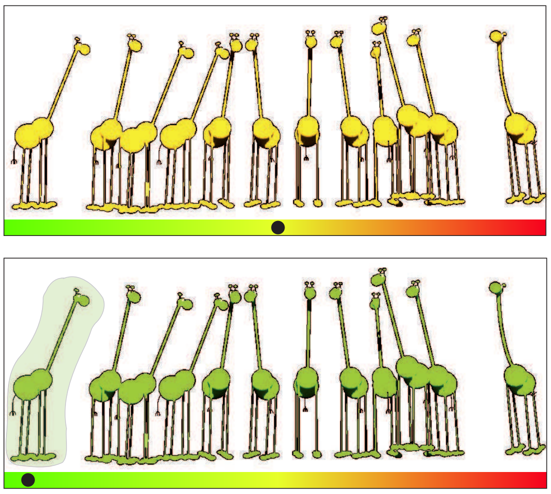

Figure 10: *The user can modify the weight of certain objects to influence the clustering. Black points indicate the cluster leader positions. Halos indicate stronger weights.*

8 Discussion

Our algorithm performs often well, running from 90 fps for the butterfly to 160 fps for the apple scene. The computation time depends on the number of representative points (we used up to around 500 points at real-time rates), the scale and the scene structure that can be more or less adapted to our acceleration grid. Of course, the theoretical performance of the mean-shift is expected $O(n^2)$ and this could be the case, but in practice, the grid data structure makes the algorithm fast enough for many complex scenes.

Our technique allows the user to control the clustering via several parameters. The first choice is the feature space that enables our system to decide which attributes have to be taken into account and in which proportion. It describes the similarity measure used for the clustering. Second, the user has to select the global scale of the clustering and can modify weights and scale per object. These parameters are accessible via a slider and can be changed interactively for easy tuning. In the future, we plan to explore learned-by-example clustering, and a system that automatically suggests "good" parameters to assist the user.

The LOA computation and stylization are currently implemented using shaders, letting the user make his own simple programs. But nothing prevents the creation of more intuitive user interfaces for given types of rendering that would use our clustering as a first step. We have shown that with simple styles and simple LOA strategies we can obtain quite complex behaviors.

Currently, representative points are chosen in a preprocess. This "sampling" is necessary to allow a fast clustering by treating a smaller set of points. However it is an approximation and might create artifacts for deformable objects because the samples may lose the uniformity of their initial distribution. We plan to investigate a real-time sampling method to address this issue.

In comparison to image based approaches like [Kolliopoulos et al. 2006], our algorithm shows strengths as well as weaknesses. Temporal coherence becomes simpler and invisible geometry can be treated (we discussed the implications in section 4.2). Another important point is that, usually, there are fewer representative points than pixels, accelerating the computations but also leading to coarser clusters. Image solutions directly take texture into account and our preprocess becomes unnecessary, which allows the easy integration of deforming objects. A combination of both techniques is a promising direction of future work.

9 Conclusion

We have presented a generic clustering approach that can be used to derive the inherent structure of a scene with applications to stylization. This is a common mechanism in the artistic production that we transfer in the context of dynamic 3D scenes. Our solution usually runs in real-time since it is independent of the scene complexity and it allows temporally coherent results. The user can control the clustering via interactive and intuitive handles thanks to our mean-shift extension.

The presented approach can be applied in contexts other than NPR. For example, we plan to use it to perform calculations for groups instead of each individual. The idea would be to perform a detailed computation for the cluster leader and coarse approximations for other elements, that are then combined to determine the complete simulation. Another application that our clustering is useful for, is to steer group behavior, where the style is animation. The accompanying video shows some of our results. All these possibilities are unified in one approach. In the future, we plan to apply the framework to artificial intelligence to help make decisions: based on group size, special members, distances to other clusters, etc.

Finally we believe that our decomposition of the image creation process opens new research avenues. We see such a computational model as a step towards a better formalization of the abstraction process leading to powerful drawing systems.

Acknowledgements We thank the reviewers for their helpful suggestions that improved the paper. Special thanks go to A. Orzan, K. Subr, and P.-E. Landes for their comments and help on the paper. We would also like to thank L. Boissieux for the giraffe models. The first author is supported by CAPES, Brazil.

References

Balzer, M., and Deussen, O. 2007. Level-of-detail visualization of clustered graph layouts. In *Asia-Pacific Symposium on Visualisation 2007*.

Bangham, J., Gibson, S., and Harvey, R. 2003. The art of scale-space. In *Proceedings of British Machine Vision Conference 2003*, 569–578.

Barla, P., Thollot, J., and Markosian, L. 2006. X-toon: An extended toon shader. In *Proceedings of NPAR 2006*, 127–132.

Bousseau, A., Kaplan, M., Thollot, J., and Sillion, F. X. 2000. Interactive watercolor rendering with temporal coherence and abstraction. In *Proceedings of NPAR 2006*, 141–149.

Cole, F., DeCarlo, D., Finkelstein, A., Kin, K., Morley, K., and Santella, A. 2006. Directing gaze in 3d models with stylized focus. In *Proceedings of Eurographics Symposium on Rendering 2006*, 377–387.

Comaniciu, D., and Meer, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 5 (May), 603–619.

Cook, R. L. 1986. Stochastic sampling in computer graphics. In *Proc. of SIGGRAPH*, ACM Press, 51–72.

DeCarlo, D., and Santella, A. 2002. Stylization and abstraction of photographs. In *Proceedings of SIGGRAPH 2002*, 769–776.

Kolliopoulos, A., Wang, J. M., and Hertzmann, A. 2006. Segmentation-based 3d artistic rendering. In *Proceedings of Eurographics Symposium on Rendering 2006*, 361–370.

Kowalski, M. A., Hughes, J. F., Rubin, C., and Ohya, J. 2001. User-guided composition effects for art-based rendering. In *Proceedings of Symposium on Interactive 3D Graphics 2001*, 99–102.

Lagae, A., and Dutré, P. 2006. Poisson sphere distributions. In *Vision, Modeling, and Visualization*, IEEE Computer Society. Accepted.

Lecot, G., and Levy, B. 2006. Ardeco: Automatic region detection and conversion. In *Proceedings of Eurographics Symposium on Rendering 2006*, 349–360.

Luebke, D., Reddy, M., Cohen, J., Varshney, A., Watson, B., and Huebner, R. 2003. *Level of Detail for 3D Graphics*. Morgan-Kaufmann, Inc.

Luft, T., and Deussen, O. 2004. Watercolor illustrations of plants using a blurred depth test. In *Proceedings of NPAR 2004*, 11–20.

Markosian, L., Meier, B., Kowalski, M. A., Holden, L. S., Northrup, J. D., and Hughes, J. F. 2000. Art-based rendering with continuous level of details. In *Proceedings of NPAR 2000*, 59–66.

Orzan, A., Bousseau, A., Barla, P., and Thollot, J. 2007. Structure-preserving manipulation of photographs. In *Proceedings of NPAR 2007*, 103–110.

Paris, S., and Durand, F. 2007. A topological approach to hierarchical segmentation using mean shift. In *Proceedings of IEEE conference on Computer Vision and Pattern Recognition 2007*.

Ter Haar Romeny, B. M. 2003. *Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, Written in Mathematica*. Springer.

Wand, M. P., and Jones, M. 1995. *Kernel Smoothing*. Chapman and Hall.

Wang, J., Thiesson, B., Xu, Y., and Cohen, M. F. 2004. Image and video segmentation by anisotropic mean shift. In *Proceedings of European Conference on Computer Vision 2004*, 238–249.

Wang, J., Xu, Y., Shum, H.-Y., and Cohen, M. F. 2004. Video tooning. In *Proceedings of SIGGRAPH 2004*, 574–583.

Wen, F., Q.Luan, Liang, L., Xu, Y.-Q., and Shum, H.-Y. 2006. Color sketch generation. In *Proceedings of NPAR 2006*, 47–54.

Winnemoller, H., Olsen, S. C., and Gooch, B. 2006. Real-time video abstraction. In *Proceedings of SIGGRAPH 2006*, 1221–1226.

A Convergence

In the paper [Comaniciu and Meer 2002], the proof for the iterative scheme was given for the original formulation. In our case, we can apply almost the same reasoning. Therefore we will mostly focus on the differences. Like in [Comaniciu and Meer 2002], we take a $k(x)$ such that $K(x) = k(x^2)$ and $g(x) = -k'(x)$. We can rewrite the gradient of f as:

$$\nabla f(x) \quad = \quad \sum_{i=0}^{n} c_i g\left(\|\frac{x - x_i}{H_i}\|^2\right)\left(\frac{\sum_{i=0}^{n} x_i c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)}{\sum_{i=0}^{n} c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)} - x\right) \quad (5)$$

where we denote the grouped constants c_i (including ω_i). Thus moving along the gradient is related to replacing our position by the weighted mean:

$$mean(x) := \frac{\sum_{i=0}^{n} x_i c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)}{\sum_{i=0}^{n} c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)} \quad (6)$$

As in [Comaniciu and Meer 2002], $y_{i+1} := mean(y_i)$ should converge for any y_0. The only property we need to apply the original proof, is that the functions $k(x^2/H_i^2)$ are all convex and $c_i g(\|x - x_i\|^2/H_i^2)$ is bounded. This is the case because of the choice of the kernel function K.