Computer-Aided Control for Expressive Rendering

Hedlena Maria de Almeida Bezerra

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE
Spécialité **« Mathématiques, Sciences et Technologies de l'Information, Informatique»**

Arrêté ministériel : 7 août 2006

Présentée et soutenue publiquement par

Hedlena Maria DE ALMEIDA BEZERRA

le **05 Novembre 2010**

LE RENDU EXPRESSIF ASSISTÉ PAR ORDINATEUR

Thèse dirigée par **François SILLION** et codirigée par **Joëlle THOLLOT**

JURY

M. Georges-Pierre BONNEAU	Professeur - Université de Grenoble	Président du Jury
M. Bernard PÉROCHE	Professeur - Université Claude Bernard Lyon 1	Rapporteur
M. Mario COSTA SOUSA	Professeur - University of Calgary	Rapporteur
M. Doug DECARLO	Professeur - Rutgers University	Examinateur
M. François SILLION	Directeur de Recherche INRIA Rhône-Alpes	Directeur
Mme. Joëlle THOLLOT	Professeur - Université de Grenoble	Co-directrice

Thèse préparée au sein du **Laboratoire Jean Kuntzmann** dans **l' Ecole Doctorale MSTII**

0.1 Résumé

L'édition, la création, et le design digital mènent la flexibilité des ordinateurs vers l'acte de création. La quasi-absence de conséquences par l'expérimentation peut être à la fois une bonne et une mauvaise chose. Bien que la flexibilité des ordinateurs encourage l'exploration d'idées artistiques, elle peut également la freiner, en proposant trop d'options mal-adaptées. Un grand défi pour le développement des logiciels graphiques aujourd'hui est de tirer parti de la flexibilité sans précédent des ordinateurs, tout en offrant à l'utilisateur le niveau de contrôle adapté à ses besoins.

La difficulté quand on développe un tel système est fortement influencée par l'attente que l'utilisateur place dans le travail effectué par l'ordinateur. L'ordinateur étant considéré comme un assistant dans le processus de création, on attend de lui qu'il soit 'assez intelligent' pour mériter ce rang. Habituellement, nous n'acceptons pas bien quand le résultat d'une telle assistance n'est pas satisfaisante. Cependant, le niveau d'attente et, par conséquence, de satisfaction, varie en fonction de l'utilisateur. Les utilisateurs novices ont tendance à être moins exigeants et sont satisfaits quand ils peuvent créer des résultats intéressants avec peu d'intervention humaine. Les utilisateurs professionnels, d'un autre côté, ont des besoins différents. Ils recherchent des logiciels assez performants pour automatiser leurs taches, mais qui leur laissent assez de contrôle sur le résultat obtenu. Pour cette catégorie d'utilisateurs, plus l'impact du logiciel sur le résultat est important, plus ils veulent avoir le contrôle sur la façon dont est obtenu ce résultat.

Dans cette thèse, nous démontrons que l'ordinateur peut être un assistant précieux dans le processus de création visuel du moment que des systèmes d'interaction adaptés existent. L'adaptation est donc un aspect important car, à différents niveaux, les utilisateurs veulent pouvoir contrôler le résultat afin d'exprimer leur style et leur créativité. La thèse présente deux scénarios différents qui démontrent que la contrôlabilité joue un rôle majeur dans l'expressivité : une technique temps-réel pour grouper des scènes 3d dynamiques pour obtenir automatiquement un résultat sur lequel on peut facilement appliquer un style de rendu arbitraire; et plusieurs dérivées d'une primitive de dessin vectoriel appelée Diffusion Curves qui contraignent et contrôlent la création de dégradés de couleurs complexes. La thèse propose, d'un coté, des représentations mathématiques et algorithmiques adaptées avec différents niveaux de contrôle sur les algorithmes de rendu expressif qui manipulent des scènes bi et tridimensionnelles; et, d'un autre coté, de traduire ces outils en interface intuitive pour l'utilisateur.

0.2 Abstract

Digital edition, creation, and design bring the computer's extraordinary flexibility to the act of creation. The near absence of penalty for experimentation can be, however, both a blessing and a curse. Although the unprecedent flexibility of the computer encourages exploration of artistic ideas, it also can be a factor of great distraction to the artist. A great challenge for the development of graphics softwares today is, thus, to take advantage of computer's unprecedented flexibility, while offering to the user the adapted level of control that she or he is looking for.

The difficulty when designing such adapted system is highly influenced by the expectations users place on the job performed by computers. As the computer places itself in the role of an assistant in the creation process, we expect it to be 'smart enough' to deserve this place. Usually, we do not accept well when the result of such assistance is not satisfactory. Nevertheless, the level of expectation, and consequently, satisfaction, is not the same for all users. Novice users tend to be less exigent and are often happy when the computer help them to create interesting results without the need of a strong user intervention. Professional users, on the other hand, usually have different needs. They are looking forward to softwares that are good enough in automatizing tasks, but also that provide intuitive controls to manipulate their results. For these users, the bigger is the impact of the work generated by the computer in the final result, the more these tasks are subjected to have control parameters.

In this thesis, we demonstrate that the computer can be placed as an incomparable assistant into the visual creation process once well-adapted interaction systems are provided. Adaptation is thus an important aspect because, in different levels, users want to be able to have the control over the result in order to give their touch and express their creativity. The thesis presents two different scenarios demonstrating that controllability plays a major role in expressiveness: a real-time technique to cluster a dynamic 3D scene in order to achieve an automatic, yet controllable output that can then be used as input to any rendering style; and a number of adaptations to a drawing vector graphics primitive called Diffusion Curves that constrain and control the creation of complex color gradients. The thesis proposes, on one hand, mathematical and computational well-adapted representations to provide the user with different levels of control over algorithms of expressive rendering in order to manipulate two- and tridimensional scenes; and, on the other hand, to reproject this technical support into user-friendly software solutions.

0.3 Agradecimentos

Agradeço a Deus por ter sido tão 'mimada' por Ele. A São José e Nossa Senhora, que sempre estiveram ao meu lado. Agradeço a meus pais, em especial minha mãe, Maria Helena, o coração mais terno do mundo. A meus irmãos, Helenilson, Hednilson, Nilshelena e Nillena, tão numerosos e tão valiosos. À minhas cunhadas Tiana e Dani, tão lindas. A meus sobrinhos Gabriel, Lucas, Isabela, Lara e Felipe, que enchem nossa família de alegrias. À *voinha*, a quem esta tese é dedicada. E a todos aqueles de Dona Ana ou de Seu Neto Avelino.

À família Eisemann, Freya, Hans, Almuth, Martin, *Oma*, Jezebela (*in memoriam*) e em especial a Elmar, ator mais importante da minha vida dos últimos 5 anos. Sem sua inteligência e busca pela perfeição, essa tese não teria tamanha qualidade. Sem seu apoio e carinho, eu não sei nem se teria conseguido... Elmar, à você, esta tese.

À Joëlle. Obrigada por ter acreditado em mim e transformado minha vida. Se hoje me considero alguém melhor, mais madura, à ela devo o aprendizado. À François Sillion pela oportunidade.

Aos meus fieis escudeiros Erika, Gi e Matheus. Meus irmãos por opção. À Beriê, e seu entusiasmo pela computação gráfica. À Palu e recentemente Dani pelo apoio. À Cêça 'tramontina' que me fez redescobrir a beleza da amizade brasileira. À Suzanita e Marcelinho meus irmãos cariocas. À Paulinha, que me ensinou o encanto pela computação gráfica. A Maluquete e suas histórias malucas. À AdeliJ, corpo diferente, mas destino copiado. Modelo vivo da Barbie Software Engineer! À Betuca, pelas discussões e ensinamentos sobre justiça social. À Bruno, Eraldo e Pablito, minhas melhores lembranças do Rio. À meu querido Serginho Krakowski. À Tristan, meu amigo longe dos olhos, mas perto do coração. À Tallitha e Elaine, minhas fortalezas. A Paulo e Bela e ja adianto Samuel. À Carlos, que sempre esteve comigo.

À Hai and Valentin, primeiros e grandes amigos dessa jornada na França. À todos do CTM, memórias de tantos anos. A Manu pela amizade e exemplo de bom gosto. À Frankoko, por todas as *palavras do mês*, discussões futebolísticas e até ouvido para meus momentos *Barbie*. À Pierrot, sempre *classe* em tudo, Minha admiração e profundo carinho. À Kartic, Ajita and Atreya, além de tudo meus amigos. À Nassimo, Pierre-Neigel, Olivier, Cyril Crassin, Cyril Soler, Fabrice, Nicolas, Charles, Isabelle, J-D, Laurent, Eric, Mahdi, Laurence, Alex, Adrien, Thierry, Kaleigh, Kiran, Vasile. À Daniel Sykora, meu amigo hoje mais real que virtual. À Patricia e Imma, pelo carinho tão imenso. À Adeline Pihuit, pela amizade mais que feminina tão necessária no INRIA. A Danielzinho, chefe de todos os chefes. À Laurentiu, por me fazer acreditar nos meus *27 anos*.

À *Dri*, viestes na hora exata com ares de festa e luas de prata. À Javier, meu irmão de continente e de fé.

Mencionar todo mundo é tão difícil, mas meu coração, eternamente grato a todos que por aqui passaram.

Contents

0.1 Résumé ii
0.2 Abstract iii
0.3 Agradecimentos iv

1 Introduction **1**
1.1 Content Creation 1
1.2 User control 5
1.3 Goals and Contributions 6
1.3.1 3D Grouping of Scenes 6
1.3.2 Controllable Diffusion of Colors 7
1.3.3 Interpreting Shape from 2D Drawings 7

2 Dynamic Grouping in 3D Scenes **9**
2.1 Criteria for Grouping 10
2.2 Related Work 14
2.2.1 Object-based NPR 16
2.3 Overview of Our Approach 18
2.4 Grouping 19
2.4.1 Representative Points 19
2.4.2 Clustering 20
2.4.3 Temporal coherence 23
2.5 Level of Abstraction and Stylization 24
2.6 Feature Space 26
2.7 Mean-Shift Extension 27
2.7.1 Convergence 27
2.8 Results 29
2.9 Discussion 30

3 Controllable Diffusion For Vector Graphics **31**
3.1 Raster Graphics Images 32
3.2 Vector Graphics Images 33
3.3 Diffusion Curves 37
3.4 Controllable Diffusion Curves 39
3.5 Related Work on Diffusion 42
3.6 Mathematical Background 44
3.6.1 Diffusion Process 45
3.6.2 Reformulating the Diffusion Process 45
3.7 Diffusion Control via Constraint Systems 47
3.7.1 Diffusion Barriers 47
3.7.2 Anisotropic Diffusion 49
3.7.3 Color Strength 52
3.8 Beyond Local Constraints 54

3.9 Results . . . 55
3.10 Discussion . . . 57

4 Shape and Tone Depiction from 2D Drawings 59
4.1 Computer-Assisted Drawings . . . 61
4.2 Lines and the Perception of Shape . . . 64
4.2.1 Shape Reconstruction from Labeling . . . 65
4.2.2 Perceptual Experiments . . . 67
4.3 Shading and the Perception of Shape . . . 68
4.3.1 Manipulating 3D Scene Depiction . . . 69
4.3.2 Manipulating 2D Scene Depiction . . . 70
4.4 Shading from Line Drawings . . . 72
4.4.1 Enriching Cartoon Drawings . . . 74
4.4.2 Image-based Methods . . . 75
4.5 Ongoing Work on Shading from Curves . . . 78
4.5.1 “Under”, “Over” . . . 78
4.5.1.1 Diffusion Barriers . . . 79
4.5.2 Mathematical Lines . . . 80
4.5.3 Surface Reconstruction from Line Properties . . . 81
4.5.4 Anisotropic Diffusion . . . 83
4.5.5 Animated System . . . 83
4.6 Discussion . . . 84

5 Conclusion 85

1 Introduction

1.1 Content Creation

Since the early times, reality, visual perception, and imagination have stimulated the human mind to express (by creating) and/or to interpret (by observing) visual content. Prehistoric men were driven by the belief that images would protect them against other powers, which were, to them, as real as the force of nature. We cannot hope to understand these strange beginnings of art unless we try to enter into the mind of such primitive peoples and find out what kind of experience it is which made them think of pictures, not as something nice to look at, but as something powerful to use [Gom61]. It is fascinating to think how, in such remote times, the need for visual creations was already present even though the medium available to express them was so primitive. Nonetheless, prehistoric art pieces are far from simplistic paintings one might expect and often reflect a tremendous creative mind capable of exploiting the combination of basic materials to produce variances in color, chroma, and intensity. Even though limited ranges of colors and shades were available, expressive results were possible, for example, the impressive contrasts between light and dark placed in order to achieve a sense of volume when drawing three-dimensional subjects as seen on Figure 1.1.

Figure 1.1 *A bison from Altamira cave (Spain) aged between 25,000 and 35,000 years. Already at that time, artists used charcoal and other materials to create images on walls.*

Thousands of years after the charcoal images left by humans on cave walls, the Egyptians used a wider variety of media to create images in order to decorate, to represent real life, or even to accompany individuals in their after-life. Figure 1.2 (left) shows an image of a garden with a pound created at 1400 BC in order to represent a vivid picture of life in Egypt at that time. Many advances occurred throughout the years, and new supports appeared, such as human-made buildings, textile, paper, pottery. Art pieces depicted new color spectra that originated from new materials and the way art was perceived changed. In the 15th century,

Figure 1.2 *Art pieces at different stages of history. Left: The garden of Nebanum, c.1400 BC. Wall painting from a tomb in Thebes. Middle: Albrecht Dürer Hare, 1502. Watercolor and gouache on paper. Right:* Le baiser de l'hôtel de ville *by Robert Doisneau, 1950, Photograph.*

artists were capable of reproducing contemporary objects with an incredible amount of detail through oil and watercolor techniques. The richness of those media and the popularity gained by art pieces encouraged artists to bring in colors of incomparable fidelity and they managed to produce astonishing renditions of their own worlds as, for example, the faithful drawing of the Dürer's hare where tiny details were carefully recorded (Figure 1.2 middle).

New discoveries lead to new forms of media which were slowly introduced and understood as new art forms. From the first experiences with the pinhole camera in the 5th century BC to the appearance of the modern cameras in the early decades of the 19th century, photography introduced a new possibility for visual communication. Photography seemed to be able to capture more detail and information than traditional media such as painting and was incredibly faster. The control during creation moved away from ink, brushes, and canvases to focus on aperture, composition, and capture. In the 20th century, photography joined the popularity hall of expressive art and remarkable pieces can be cited as *Le baiser de l'hôtel de ville* by Robert Doisneau in 1950 (Figure 1.2 right).

It is interesting to notice that photography paved the way to a new perception of art. Photography's incredible ability to capture details from reality in a matter of seconds reflected the enormous pace at which technology was growing and, underneath it, the need for speed society was looking for.

This need for performance and accuracy also became evident in other fields. The need for efficient numeric calculations inspired Charles Babbage in 1823 to start building his *difference engine*, a complex mechanical machine for calculating logarithms and trigonometric functions that were important for the military for performing in ballistic calculations and motivated the construction of complex calculating machines. Such machines soon found application in many fields, e.g., highly specialized solutions enabled to compute the 1890 U.S. census in a matter of months - while the previous census had taken almost 10 years [Wik10].

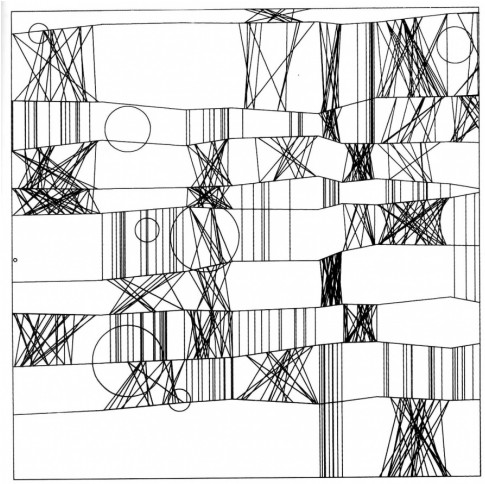


Figure 1.3 *Frieder Nake, plotter drawing, ink on paper, 1965.*

With the commercialization of computer-driven plotters in the early 1960's, not only typing became possible, but the enormous computational potential could finally also be exploited for drawing. At this point, computer art was a reflection of both; the power of the computer and its limitations. Although some pioneering artists tried out the possibilities of the medium, producing art with a computer required a strong familiarity with its mathematical vocabulary. The visual language was dominated by geometric shapes and functions, and compositions were made up of rotated and scaled copies [Spa98]. Not by coincidence, many artwork was produced by the direct cooperation between artists and researchers, or solely by those scientists who also developed artistic skills. The latter is the case of the artwork created by the German professor Frieder Nake, whose art piece is depicted in Figure 1.3. During this early stage, questions had been raised concerning how much the work produced by computers could be considered as *art* [Nol67, Nol66]. However, as pointed out by Csuri and colleagues [CS68], the importance aspect of art lies in its conception. This view was adopted wildly, as the potential of the computer was discovered, and they acted increasingly as an assistant, rather than a creator. Csuri and colleagues believed that the mathematical transformations made possible by the computer presented a new dimension of art. Indeed, in the following years, society was faced with a revolution: personal computers and interactive graphics software systems.

Enabled via the incredible development of computers, the history of content creation was influenced drastically. Computers became faster and more versatile enabling their applications to a large variety of domains and purposes and computer art took a giant leap forward. Affordable cost, intelligent tools, high performance computation, and a new feeling of freedom, by surpassing the limitations of previous media, attracted many artists to explore the computer as a creative tool. Powerful content creation software products were developed and facilitated human-computer interaction more intuitive. Soon, the help provided by computers made it possible for artists to better focus on the visual content instead of controlling every process related to the creation. For example, interesting artwork like the *Doll House* by Michele Turre presented in Figure 1.4 (left) illustrates well he advantages of the new medium (the computer together with software and input tools). Here, scaling and composition mechanisms were enabled by the computer to allow the artist to concentrate on the creative process.

The flexibility of this new medium was the basis that enabled its breakthrough and lead to an omnipresence. From tiny cell phone screens, over astonishing dynamic special effects

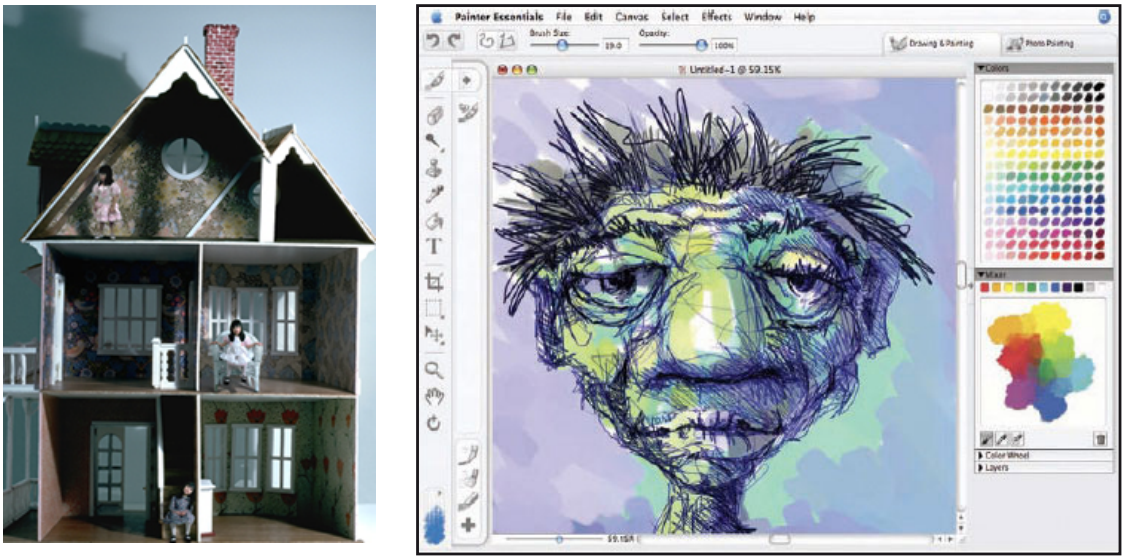


Figure 1.4 *Left: Image produced with the help of photo edition softwares.* Doll House *by Michele Turre, created with the help of photo edition softwares. Right: Snapshot from a painter software simulating watercolor and pencil mediums. Image from Corel Painter© tutorials.*

displayed in the cinema, to even art galleries. Today, many traditional artists feel comfortable to designing and experimenting with the computer before picking up their brush and canvas. This is made possible due to the use of adequate software that immerses the artist in a creative environment that adapts to their needs. Such software simulates the appearance and behavior of traditional media that artists are familiar with. Many different tools are only a simple button click away. Nonetheless, different software packages might still specialize on particular task sets. For instance, tools such as Adobe Photoshop© for photo editing, usually have a different set of features and vocabulary than tools for drawing and painting, like Corel Painter©. Figure 1.4 (right) shows an example of a software interface specialized on watercolor and pencil-like media simulation.

Beyond still images, significant advances were also achieved in the field of video and animation. Computer animation and video-oriented motion graphics benefit from a wide range of tools designed to automatize and help artists to create dynamic scenes. Even for traditional cel animation, the use of software is increasingly involved in the creation process. Possibilities, such as digital in-betweening (a process to add intermediate frames to render more fluid continuous movements), to quasi-3D effects such as the ballroom scene in *Beauty and the Beast* by Disney© (see Figure 1.5 left). This development went even further, *Toy Story* (1995), by Disney-Pixar©, was the first fully and high successful computer-generated feature film (Figure 1.5 middle). More than just enhancing the appearance of traditional animation, the computer made it possible to create and animate fully believable 3D worlds. Even photo-realistic human actors like those presented in the movie *Final Fantasy: The Spirits Within* produced by Square Pictures© became possible (Figure 1.5, right).

Although the amount of work involved in such a movie production is still considerable, it is undeniable that computers brought interesting opportunities to the process of content creation.

Figure 1.5 *Left: Ballroom scene from* Beauty and the Beast *(1991) by Disney©. Middle: Buzz Lightyear character from* Toy Story *(1995), the first fully computer-generated feature film. Disney-Pixar©. Right: Ming-Na's character from* Final Fantasy *(2001) by Square Pictures© was designed to be as realistic as possible.*

1.2 User control

Digital edition, creation, and design bring the computer's extraordinary flexibility to the act of creation. The near absence of a penalty for experimentation can, however, be both: a blessing and a curse. This has been underlined by Anne Spalter in her book "The Computer in the Visual Arts" [Spa98]. According to Spalter, although the unprecedented flexibility of the computer encourages the exploration of artistic ideas, it also reduces the structure that a physical medium often enforces. Therefore, the freedom of the medium can be a factor of great distraction to the artist. A great challenge for the development of graphics software today is, thus, to take advantage of the versatility of the medium provided by the computer, while adapting the level of control to the artist's expectations.

The difficulty when designing such an adaptive system is highly influenced by the expectations users place on the job performed by computers. As the computer places itself in the role of an assistant in the creation process, we expect it to be 'smart enough' to deserve this place. Usually, we consider it unacceptable when the result of such assistance is unsatisfactory. Nevertheless, the level of expectation, and consequently, satisfaction, might vary from one user to the other. Novice users tend to be less demanding and are often satisfied when the computer helps them to create an interesting result without much effort and little user intervention. Professional users, on the other hand, usually have different needs. They are expecting that the software is smart enough to automatize tasks, but it should also provide intuitive controls that allow them to manipulate their results. For such users, the bigger the impact of the computer on the final result, the more control parameters are expected.

1.3 Goals and Contributions

In this dissertation, we argue that the computer is an incomparable assistant for the visual content creation process, provided that well-suited interaction systems are made available. Having appropriate control handles is an important aspect because these empower the user in their creative process, and enable the creation of more expressive content. Our goal is to exploit the large expressive potential lying in adequate computer algorithms and to conceive general, yet intuitive tools. By building upon mathematical and computationally well-adapted representations of the artistic expressiveness, it is possible to formalize this aspect. However to truly achieve intuitive behavior, we also need to consider user-friendly software solutions. Our thesis is that this goal can be reached by building upon intuitive, yet tedious, but easy to perform operations by the machine, that are controlled on a higher level by the artist. In such a way, we achieve a predictable algorithmic behavior, while ensuring that the artistic freedom remains untouched and also avoid drowning the artist in an abundance of possibilities and settings that Spalter warned about [Spa98].

We illustrate our principle via three different cases of expressive rendering for which control handles are introduced that bridge generality and intuitiveness.

1.3.1 3D Grouping of Scenes

The first contribution is a grouping approach for elements in dynamic 3D scenes that helps conveying and leveraging group relationships. We show that, by using clustering strategies that take into account various attributes of the scene, we offer control handles that help the artist to exploit the scene's inherent structure in stylization processes.

Our contribution comprises the following elements:

- A general automatic grouping approach able to take many features (position, color, velocity, etc) into account;
- An intuitive user interaction mechanism with portable and predictable composition behavior;
- A temporal coherence system to ensure grouping information varies consistently over time;
- An efficient approach based on a fast computation method to interactively manipulate the scene.

This approach is presented in Chapter 2.

1.3.2 Controllable Diffusion of Colors

Our second work addresses methods to create and manipulate two-dimensional drawings. We present a technique to enable vector-based primitives to depict complex color gradients. In particular, we extend the Diffusion Curve primitive by Orzan and colleagues [OBW+08]. While the original tool is particularly appreciated by novice users, professional artist complain about a lack of expressive freedom. Our algorithm aims at increasing the flexibility in order to enhance usability and expressiveness by providing the user with control over the color diffusion process. More specifically, our contributions are:

- A more flexible curve primitive that blocks diffusion (but do not emit color);
- A control mechanism for diffusion anisotropy and orientation;
- A control mechanism for diffusion strength (speed);
- A generalized solution method for non-local diffusion.

This approach is presented in Chapter 3.

1.3.3 Interpreting Shape from 2D Drawings

Scientists have explored many different avenues for understanding and representing shapes based on line drawings. These efforts resulted in many techniques addressing the problem from different perspectives such as the extraction of expressive lines from 3D surfaces, the understanding of surface perception and the extraction of surface descriptions from hand-made and computer-generated drawings. We analyze the importance of line and scene features and links these to computational algorithms. Particularly, we discuss a whole series of possible improvements over previous approaches for the extraction of approximating surface normals from two-dimensional line drawings. While not all of our suggestions are integrated in a software solution, we do illustrate how normals can be used to enrich the drawing with shading information and show some examples for our improvement in Chapter 4.

Overview As pointed out above, Chapters 2 and 3 describe our computational and mathematical contributions to control the creation of expressive content, supported by prototype implementations. Chapter 4 gives an overview of several shape depiction techniques and discusses possible improvements for the particular case of cartoon-drawing illumination. Finally, Chapter 5 presents the conclusion of the dissertation linking these contributions together and summarizing potential directions for future work.

2 Dynamic Grouping in 3D Scenes

One fundamental question in expressive rendering is: how to emphasize certain objects of the depicted scene? Answering this question allows the creation of efficient visual contents 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 stylization to produce different representations of the scene (see Figure 2.1). If the artist uses a camera, the focus can be set so that certain objects will be blurred (e.g., in the background) (Fig. 2.1, left). If a pen-and-ink technique is used, he/she may decide to simplify the silhouette of certain objects, and to detail that of others with precise strokes (Fig. 2.1, middle). Furthermore, color can be used to make certain objects less visible by desaturation or averaging (Fig. 2.1, right).

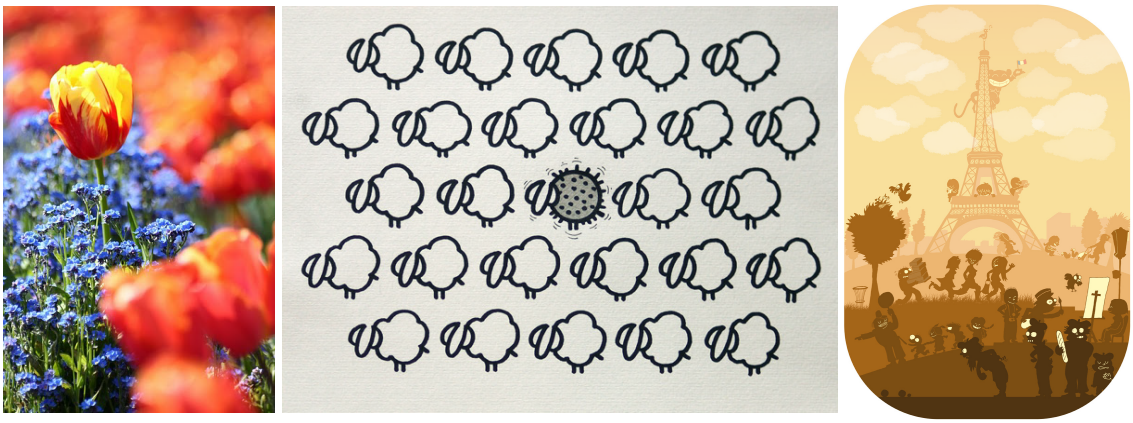


Figure 2.1 *Several techniques to emphasize content. Left: Focus manipulation. Photo by* Betuca. *Middle: simplify/improve line depiction by adding/removing details. Right: Color desaturation.*

Grouping elements together is also part of the choice made to decide what areas should be emphasized in the scene. Depicting objects as belonging to different groups plays a major role to convey relationships between them and the organization of the scene, i.e. overall composition. As a side-effect, the most salient or structured parts of the scene can be identified, and the purpose of the image more efficiently communicated. One observation is that elements in the group are often depicted in the same style. The criteria for grouping are multifaceted. Depth of field is only one strategy. Others include semantics, proximity or even orientation or common movement. Rather than visual items simply being aggregated, items adhering together in a group suggest that they serve together as argument to a richer scene interpretation [Fel03].

This chapter presents 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. We propose

an approach where clustering strategies can be devised taking into account any attribute of the scene. Once the clustering is done, a temporally coherent *Level of Abstraction* 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. Our choice of 3D scenes as input is motivated to assure temporal coherence and to benefit of camera movement. In the following sections, we propose an efficient solution based on an extended mean-shift algorithm customized by user-defined criteria. The approach was published at the 6th International Symposium on Non-photorealistic Animation and Rendering, NPAR 2008 [BEDT08].

2.1 Criteria for Grouping

Grouping and perceptual organization have proven to be remarkably difficult problems, and the subjective judgments from human observer's are still far superior to the best known computation techniques [Fel03]. It is clear, therefore, that this mechanism cannot be completely automated, and that the user should keep some sort of control over it. There are several criteria according to which objects can be grouped, the most common used being spatial distance and depth position. In general, groups that bind objects at similar distances (by similar color, proximity in the image, similar lighting, or similar blurring) make the spatial organization of the scene to pop up more effectively [Dur02]. Figure 2.2 depicts two comic book scenes from *Spiderman* by Marvel Comics©. On the left, we easily perceive two distinct groups of characters: one which elements are painted in a colorful manner, and another painted in plain blue. The depiction of the scene suggests a clustering mechanism based on the character's depth position. Characters placed in the first depth plane are depicted with great amount of detail as it is commonly the case. The same criterion for clustering can be perceived in the image on the right, but here, characters falling in a cluster with a higher depth position are this time brought into attention.

Although spatial distance and depth seem to be more general choices for clustering, some groups are based on complex criteria that can play a role individually, or in combination with others. In *The Death of the Superman* illustration by DC Comics© depicted in Figure 2.3, groups of elements can be easily identified. Nevertheless, the criteria which influenced the clustering structure are rather complex. The scene's arrangement suggest that the grouping standard was based on the character's movement direction. Consequently, characters following the funeral are distinguished from those simply watching it. In addition, the relative depth that follows the axis of movement seems to be also taken into account. It explains why characters depicted at the back of the procession are part of another distinguished group. Nevertheless, many aspects on this image are based on semantical choices that can only be retrieved from the user, not the scene itself. The dimension of each group, the relationship between them, and the rules by which they are illustrated are some examples of the artist's personal touch and proper use of visual perception cues.

The range of criteria for grouping is, therefore, multifaceted. Other Gestalt grouping

Figure 2.2 Spider-Man *is a fictional Marvel© Comics superhero. Characters are often grouped into clusters and rendered with different colors in order to bring the audience's attention to interactions between them.*

principles like, for example, collinearity, shape similarity, color, or velocity can be considered *individually or in combination* in order to derive the interpretation of the scene. The grouping approach becomes even more challenging when performed in animated 3D scenes. The problem is harder than for a static image, because the layout of objects varies with time. How would the Superman's funeral scene (Figure 2.3) be depicted if the point of view in which the scene was drawn moved with time? It would be more than tedious for an artist to specify the grouping information at each frame, especially if temporal coherence must be enforced. Even if he/she could do so, it would work only for static scenes, and not for interactive scenes, where the viewpoint changes in real time.

The question of relating objects to each other helps to depict *what* to render, and it is separated from the style, which only determines how these groups are rendered to achieve a visual perception of the depicted scene. The stylization process can, nevertheless, use the grouping information to evidence, enhance, the relationship of elements falling in the same cluster. The extent to witch elements in these groups are detailed are here named **Level of Abstraction**, or simply LOA. In the *Spiderman* example of Figure 2.2 (left), characters falling in the cluster with a higher depth position were depicted in an abstracted manner, therefore high LOA. Respectively, characters in the first depth plane were illustrated with great amount of the detail, thus low LOA. The opposite situation is illustrated in Figure 2.2 (right). These two examples show that, once groups are retrieved, different LOAs can be assigned to them according to the artist's desire of emphasis/de-emphasis the subject.

Closely related to the problem of retrieving grouping information, the choice of LOA also depends on the artist's goal and often the viewpoint. Nevertheless, when dealing with interactive scenes, specifying the LOA can be very tedious. In this chapter, we propose to relate the derivation of LOAs to the grouping information, since it makes sense that all members in a cluster should be equally perceived. One possibility is then to automatically

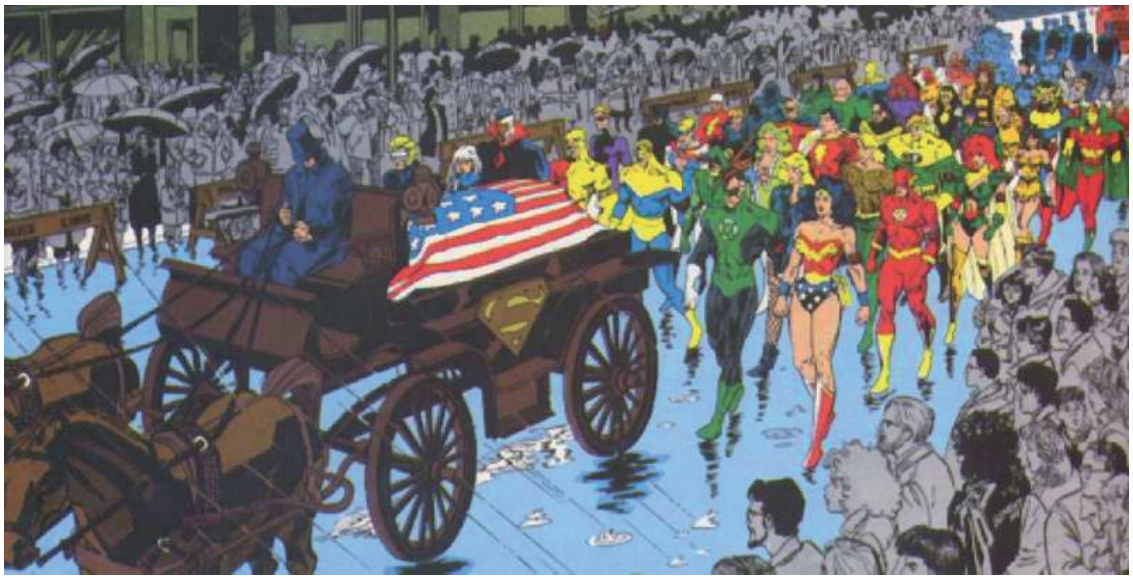


Figure 2.3 The Death of Superman *is a 1992 comic book storyline that occurred in DC Comics©'s Superman titles.*

derive LOAs using the average position of elements in a cluster to compute the distance from the group to 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 provide greater detail to foreground elements as seen in Figure 2.2 left. These considerations motivated the use of depth in many previous techniques for abstraction and stylization [BTM06, KHRO01, MMK+00]. We see in Section 2.2 that, although depth is a very common criterion for performing abstraction, it is not the only one that is involved. Information like normals, colors [KWH06], or region of interests according to the viewer [DS02] can also be combined to drive the process.

Spatial distance was the criteria used to cluster flowers and the butterfly depicted in Figure 2.4. The consequently stylizations shown on (c) and (d) were performed taking the extracted group information (b) into account. In Figure 2.4 (c), all the flower groups placed in the exterior circular arrangement of the image were depicted using the same LOA, while the group in the center of image containing the butterfly was depicted using a lower LOA value, therefore being more evidenced. In Figure 2.4 (d), the same grouping information was used to guide a completely different style: all the groups have the same LOA, but the position of the clusters determines the size and orientation of the stylized strokes in this painterly rendering style.

Grouping information plays an important role into the creation of efficient visual representations. This process is, nevertheless, subject to several levels of controllability: First, the attributes on which groups will be based on; Second, the influence of each of these attributes into the clustering process; Third, the influence of these groups during the stylization process (LOA). Therefore, we present an approach to interactively derive grouping information of scenes that has the following properties:

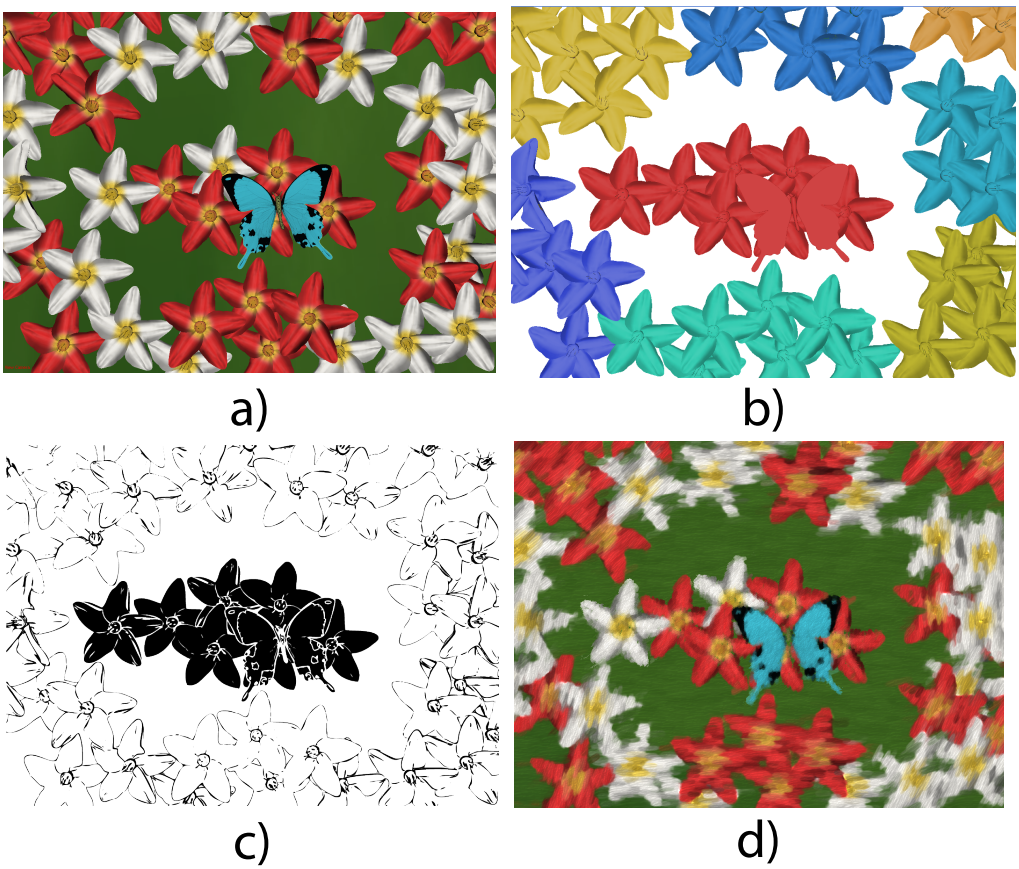


Figure 2.4 *An example illustrating our 3D clustering to drive two different stylizations: (a) A meadow of flowers and a butterfly. (b) We cluster the scene according to position. (c) Grouping information is used to drive a line rendering style where the group containing the butterfly appears black; (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.*

- Generality: automatic grouping should be able to take many features (position, color, velocity, etc) into account;
- Flexibility: the user's interaction should be intuitive and the composition behavior portable and predictable;
- Temporal Coherence: grouping information should vary continuously over time;
- Interactivity: computations must be fast enough for interactive manipulation of the scene.

2.2 Related Work

Grouping and abstraction have been proposed in many NPR works and often in form of a particular style definition. In this Section, we concentrate on previous methods addressing stylization processes that use clustering techniques as starting points to the determination of LOA and/or to perform abstraction.

When dealing with images, abstraction is usually done by segmenting the image into regions. Computing a segmentation of an image means partitioning it into a set of regions with uniform visual aspect. By 'uniform', different methods mean different definitions, ranging from low-level properties (e.g., color) to higher-level properties and visual cues from Gestalt and perception theory (e.g., texture) [LL06a]. This clustering in 2D can be used to control stylization. Examples can be found in [WQL+06] for producing color sketches from 2D input images (see Figure 2.5). To obtain sketchy look, the authors preprocess the input image segmenting it into a set of regions via a mean-shift algorithm which performs a clustering on the color space of neighboring pixels. Then, a two-stage stylization algorithm is used to simulate properties of the artists' free-hand drawing style. Similarly, Bousseau *et al.* [BKTS00] present an interactive watercolor rendering technique that recreates the special visual effects of lavis watercolor. Their technique is organized in two steps: an abstraction step that recreates the uniform color regions of watercolor and an effect step that filters the resulting abstracted image to obtain watercolor-like images. The authors suggest that applying the watercolor filter to detailed images produce results that appears too detailed compared to a painting done by hand. Therefore, a color abstraction step is needed and aims at reproducing uniform color regions by segmenting the original image. The aforementioned works suggest, therefore, that the segmentation of an image into clusters of similar colors is of major importance to a broad range of abstraction techniques.

Figure 2.5 *Color Sketch Generation by [WQL+06]. (a) Input image. (b) Results of an image segmentation. (c) Final results after abstraction.*

In the context of video, the work of Wang *et al.* [WXSC04] describes a system for transforming an input video into a highly abstracted, spatio-temporally coherent cartoon

animation with a range of styles. Also here, in order to obtain abstracted regions, the authors make use of a mean-shift algorithm on the color space, but this time, it is used to create three-dimensional semantic regions by interpolation between the keyframes. It suggests, therefore, that clustering techniques can be used in higher order domains as 2D color spaces in regard to time. A more complex technique to abstract videos by Winnemöller *et al.* [WOG06] uses a class of filters, called anisotropic diffusion filters, which have the desirable property of blurring small discontinuities and sharpening edges, as guided by a diffusion conduction function that varies over the image. The abstracted image resulting from this technique is then used to perform an automatic, real-time video and image abstraction framework that abstracts imagery by modifying the contrast of visually important features, namely luminance and color opponency. We see that complex mechanisms can drive different clustering techniques that are very often the base of content simplification goals, later used in various stylization methods.

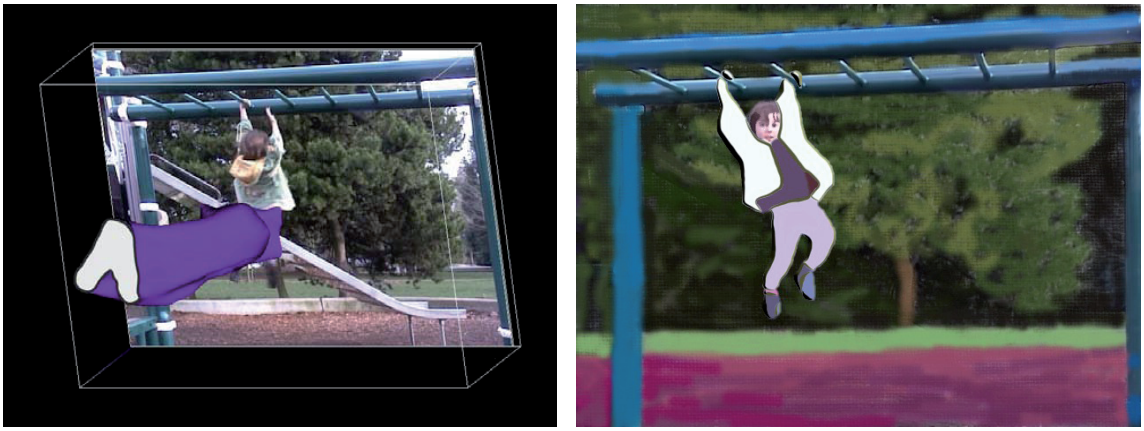


Figure 2.6 *Video Tooning by [WXSC04]. Left: A smoothed semantic region sliced at time. Right: A stylization example.*

In general, the LOA is chosen manually as it is the case for all the previously mentioned works. Nevertheless, several approaches provide ways to compute LOAs automatically. De Carlo and Santella [DS02] use eye tracking to derive the user area of interest emphasizing them by adjusting their resolution, i.e., in areas of emphasis drawing regions with higher resolution while in other areas drawing regions with lower resolution. Lecot and Levy [LL06a] use a saliency map to guide the LOA computation and abstract the image using higher-order color gradients. Bangham *et al.* [BGH03] associate LOAs based on the distance to the center of attention, which is estimated using a scale space approach. Orzan *et al.* [OBBT07] 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.

2.2.1 Object-based NPR

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 [MMK+00] based on graftals. In this work, smaller-scale changes in LOA (e.g., number of strokes, as well as stroke length, width, and color) varies view-dependently according to a continuously changing measure of desired LOA. Consequently, certain graftals, called tufts, have a multi-resolution structure, so that a single graftal, seen from a distance, transforms into several graftals when viewed from nearby as suggested by the camera movement depicted in Figure 2.7. Because of their multi-resolution nature, tufts can provide some measure of efficiency by reducing the cost of processing graftals that are ultimately not drawn, e.g., due to their distance from the camera. That is, a single tuft might control all the blades of grass in a region of ground. When the camera is sufficiently far away, the tuft can determine with a single computation that no blades need to be drawn without visiting individual blades.

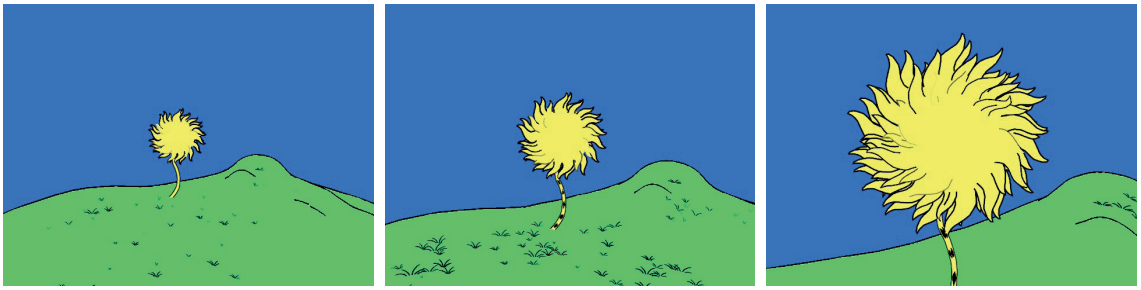


Figure 2.7 *Art-based Rendering with Continuous Levels of Detail by [MMK+00]. The image is rendered with different levels of details according to the distance of the object to the camera. The image shows 3 different stages of a camera movement. When far away, few details are rendered; the more the camera approaches from the objects, the more details are revealed.*

Also using depth as feature to guide the LOA, Barla *et al.* [BTM06] describe a toon shader that supports view-dependent effects as a notion of tone detail, so that tone varies with depth or orientation relative to the camera. Their technique also performs abstraction of the shape perceived through the shading by using a modified normal field defined by interpolating between normals of the original shape and normals of a highly abstracted shape. The normal interpolation is also computed as function of depth position or orientation relative to the camera. This type of simplification is classical in Computer Graphics when level of details are involved [LRC+03]. Cole *et al.* [CDF+06] 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 as seen in Figure 2.8. In contrast to a uniform distance as

used by these techniques, we propose an approach that can provide a scene adapted shape of the focus area and seeks to account for general criteria.

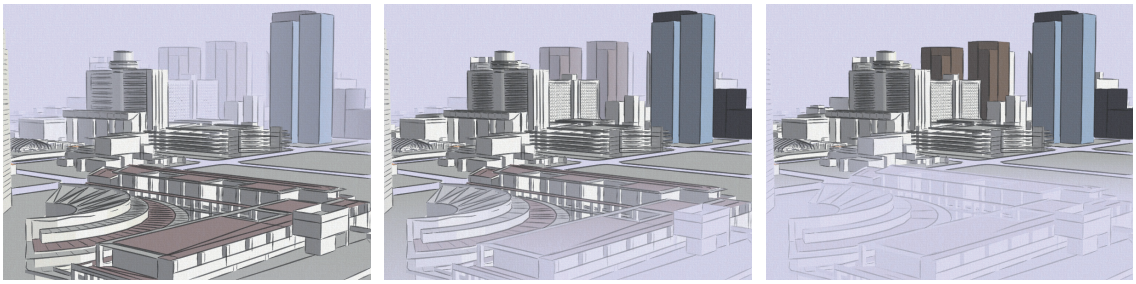


Figure 2.8 *Directing Gaze in 3D Models with Stylized Focus by [CDF+06]. The authors propose a temporally coherent control of emphasis for interactive 3D NPR applications. In this image, focal plane moves gradually from foreground to background, yielding to an automatic change in emphasis.*

Several works rely on handmade segmentations of a 3D scene. In [LD04] 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 [BD07] present a clustering algorithm for graph visualization. Kowalski *et al.* [KHRO01] 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. To illustrate this technique, Figure 2.9 (a) shows the group at a close distance, where grouping has not yet taken effect. Every mountain draws all of its outlines. In (b), the camera has moved back and the initial effects of the grouping can be perceived. The interior boundaries of the group are beginning to fade away. Finally, in part (c), grouping is completed. All interior outlines have disappeared, leaving only the exterior silhouette of the mountains. Part (d) shows the result without grouping. 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.

The work of Kolliopoulos *et al.* [KWH06] introduces a segmentation technique similar to our method in that they also use clustering. They segment a rendered 3D scene in image space while taking into account, for each pixel, 3D information (e.g. depth, normals), scene information (e.g., colors), and user provided object IDs. When dealing with animated scenes, temporal coherence is encouraged by segmenting adjacent frames together. Given a segmentation, the technique can then render the scene in a variety of styles that explicitly make use of the segments as seen in Figure 2.10. Nevertheless, the major limitation of this method is that it does not take hidden geometry into account. Therefore regions that get dis-

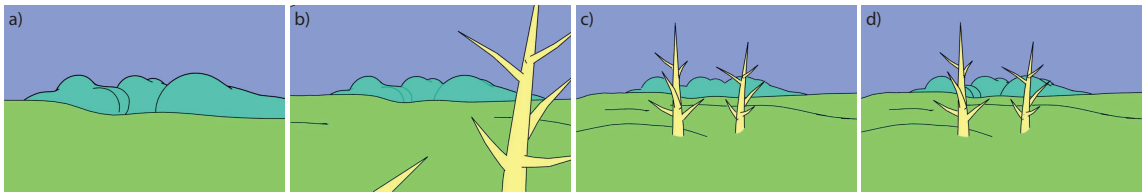


Figure 2.9 *User-Guided Composition Effects for Art-Based Rendering by [KHRO01]. As the camera moves, groups are detected and the rendering styles is impacted through the addition or suppression of outlines.*

connected 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 his/her styles according to any information available in the scene (including occluded geometry). We provide an analysis of the advantages and disadvantages of object and image-space clustering in Section 2.8.

Figure 2.10 *Segmentation-Based 3D Artistic Rendering by [KWH06]. Simple scenes rendered in various artistic styles based on 2D segments. (a) Scene segmentation. (b) Toon shading with segment-boundary contours. (c) Painterly strokes without relief texturing. (d) Stippling.*

2.3 Overview of Our Approach

Figure 2.11 shows a pipeline overview of our approach with its main components. 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 2.4.1). At run time, these points are clustered using a mean-shift algorithm (Sec 2.4.2) in a *feature space* specified by the user (Sec 2.6). 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 2.5).

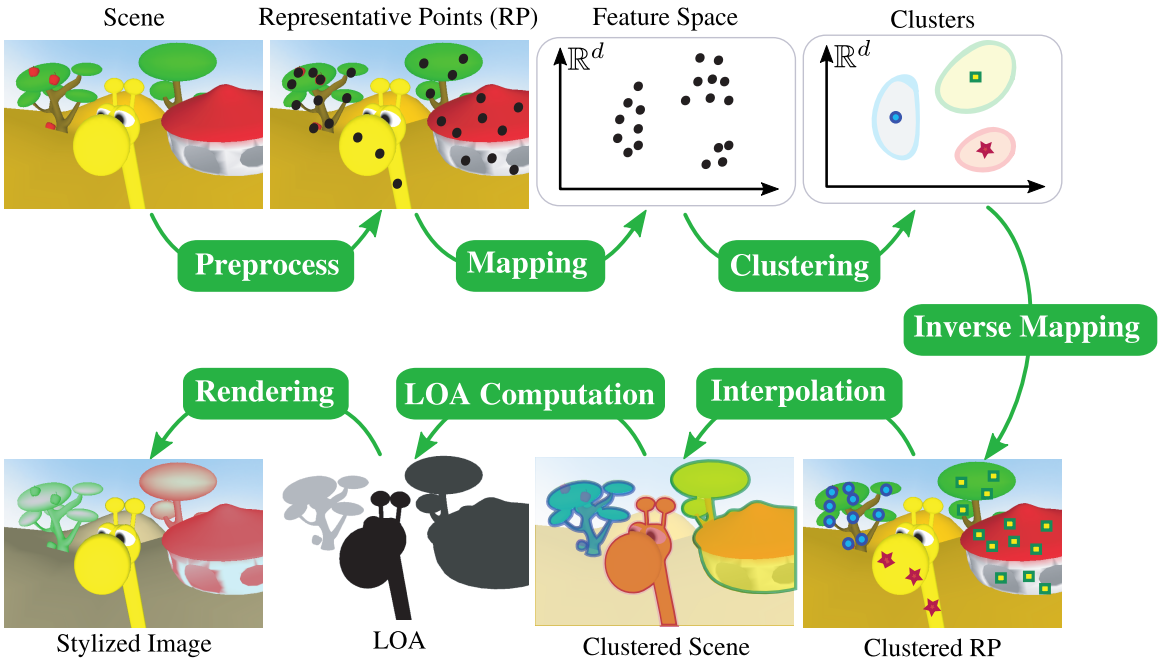


Figure 2.11 *Overview: The first step is done in a preprocess, all the others are performed at run-time. Finally, any stylization can be used for rendering.*

2.4 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.

2.4.1 Representative Points

Clustering scenes with many points is a time consuming process. To maintain interactive frame rates, 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 [Coo86, LD06]).

Figure 2.12 *Sampling (left), representative points (middle), weights on mesh (right).*

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:

$$v_P := \frac{\sum w(P, R_i) v_i}{\sum w(P, R_i)}, \text{ with } w(P, R) := e^{-\|P-R\|^2} \tag{2.1}$$

where w is a weighting function that measures the proximity between two points (Figure 2.12, 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.

2.4.2 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. In most cases, they impose 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 [CM02]. 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 [WTXC04] that represents an anisotropic extension, we focus on the original formulation.

Given n data points $x_i \in \mathbb{R}^d$, the original paper [CM02] 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:

$$f(x) := \frac{1}{nH^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{H}\right) \tag{2.2}$$

Even though several conditions are imposed on K, there is a large variety of choices [WJ95]. We use a radially symmetric function:

$$K(x) := (2\pi)^{-d/2} e^{-\frac{1}{2}\|x\|^2} \tag{2.3}$$

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 [CM02]. Advanced solutions for a faster evaluation exist [PD07] but are out of the scope of this thesis.

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 [Ter03], 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 2.13 shows a comparison between our clustering and several naive strategies. The scene is the one shown in Figure 2.4 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 neigh-

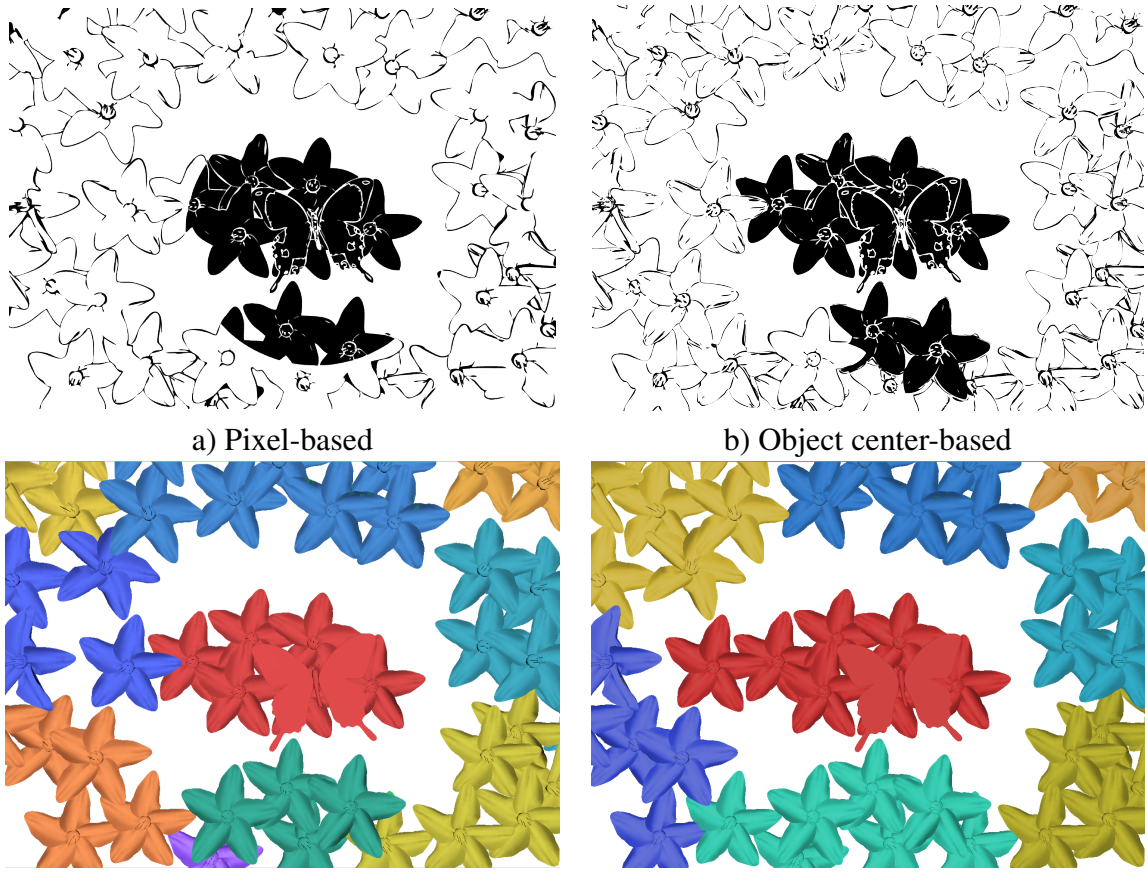


Figure 2.13 *Comparison of our clustering to naive approaches.*

boring elements. Our approach (d) produces clusters that are more natural because they relate to the scene's structure.

Figure 2.14 shows an example inspired by Kolliopoulos *et al.* [KWH06] 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 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 2.6.

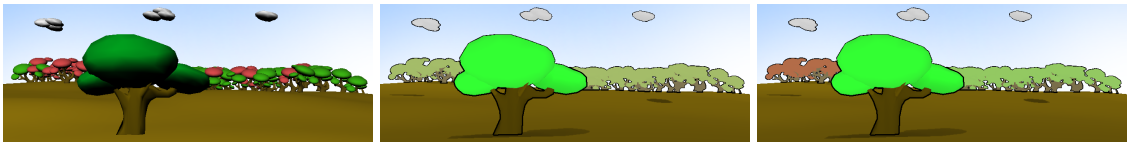


Figure 2.14 *Comparison of our clustering to an image-based approach: (Left) original scene; (Middle) our result; (Right) using only visible points creates two distinct clusters in the background.*

2.4.3 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 classifications. Figure 2.15 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 2.15 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 [WXSC04]. We used 10^{-5} throughout this paper, but it could be chosen according to the scene's total density.

We refer to the resource video [1] for the demonstration of temporally coherent examples.

2.5 Level of Abstraction and Stylization

As mentioned in the overview of our approach, 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.

[1] `http://artis.inrialpes.fr/Members/Hedlena.Bezerra/resources/DynamicGrouping.avi`

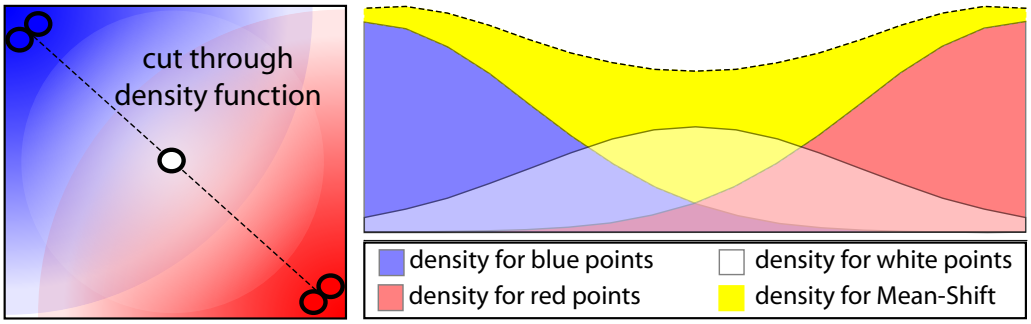


Figure 2.15 *Slight perturbations can lead to a different clustering. The white point could join the red or the blue cluster.*

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.

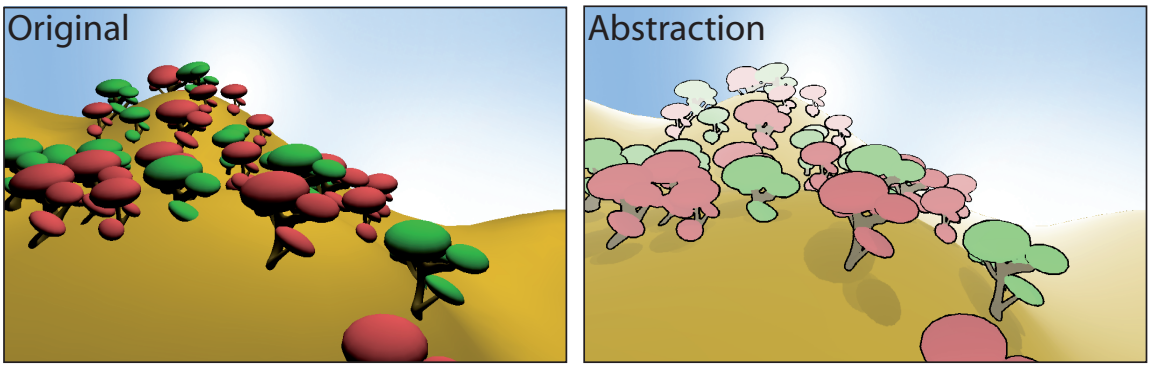


Figure 2.16 *The original scene is clustered using the position in camera space as feature space; The LOA is computed from the depth of the cluster leader; The style is cartoon-like with aerial perspective.*

Figure 2.16 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.

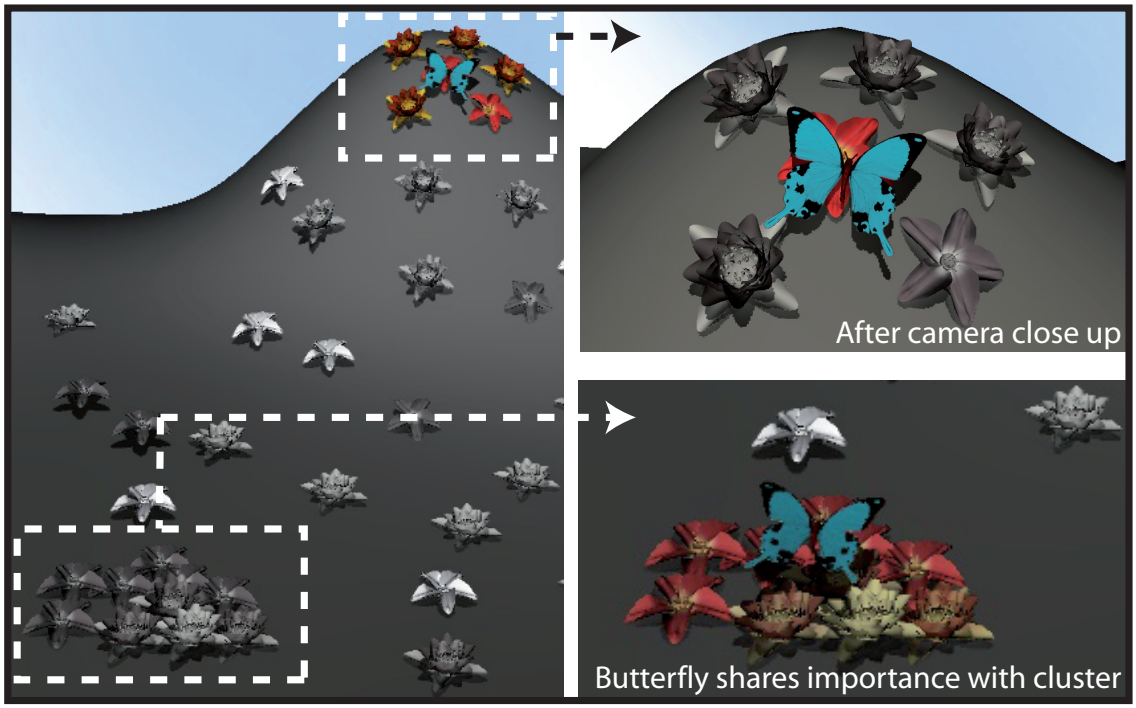


Figure 2.17 *In this example the LOA is based on the presence of the butterfly in the cluster. Using screen position as a feature, the clustering evolves according to the viewpoint. Camera zoom underneath the upper arrow: a flower bush when seen from far away is separated into individual flowers when zooming.*

Figure 2.17 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.

2.6 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 2.18 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 in Figure 2.18-(b)). Similarly if only position is taken into account, three clusters are naturally obtained (Figure 2.18-(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 2.18-(d)). This example demonstrates that multiple features can easily be taken into account in a controllable way.

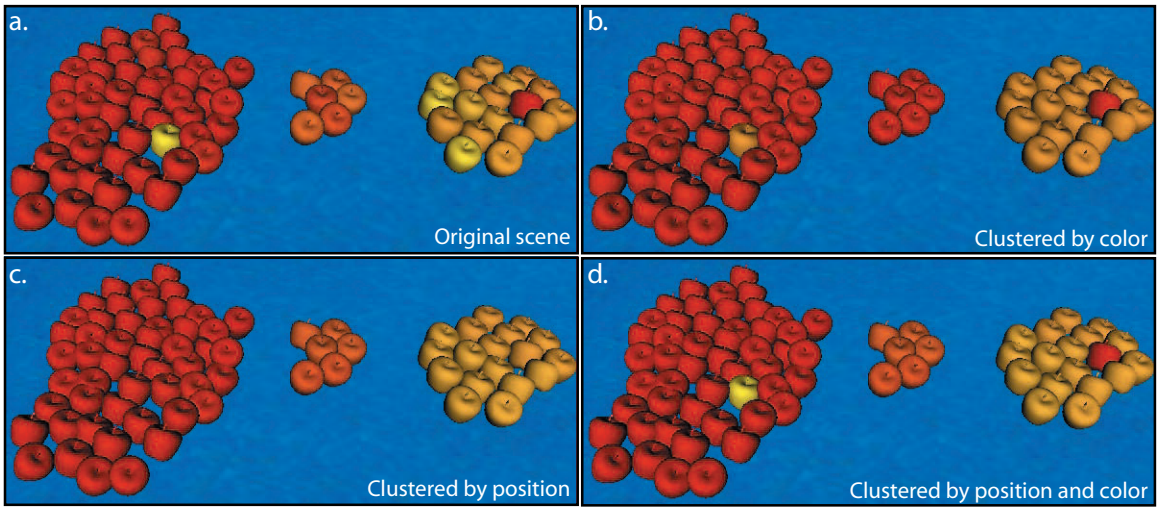


Figure 2.18 *(a) A scene with different colors, is clustered using different attributes: (b) color, (c) position, (d) color and position.*

2.7 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.

The reformulated density function is:

$$f(x) := 1/n \sum_{i=1}^{n} \omega_i K(\frac{x - x_i}{H_i}) \tag{2.4}$$

With this weight instead of classical means the method still converges. A proof is given in Section 2.7.1.

Figure 2.19 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 attribute considered 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 positioned 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 it a weight. The modifications are directly visible allowing an interactive adjustment.

2.7.1 Convergence

In [CM02], 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 [CM02], 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:

$$\nabla f(x) = \sum_{i=0}^{n} c_i g\left(\|\frac{x - x_i}{H_i}\|^2\right)\left(\frac{\sum_{i=0}^{n} x_i c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)}{\sum_{i=0}^{n} c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)} - x\right) \tag{2.5}$$

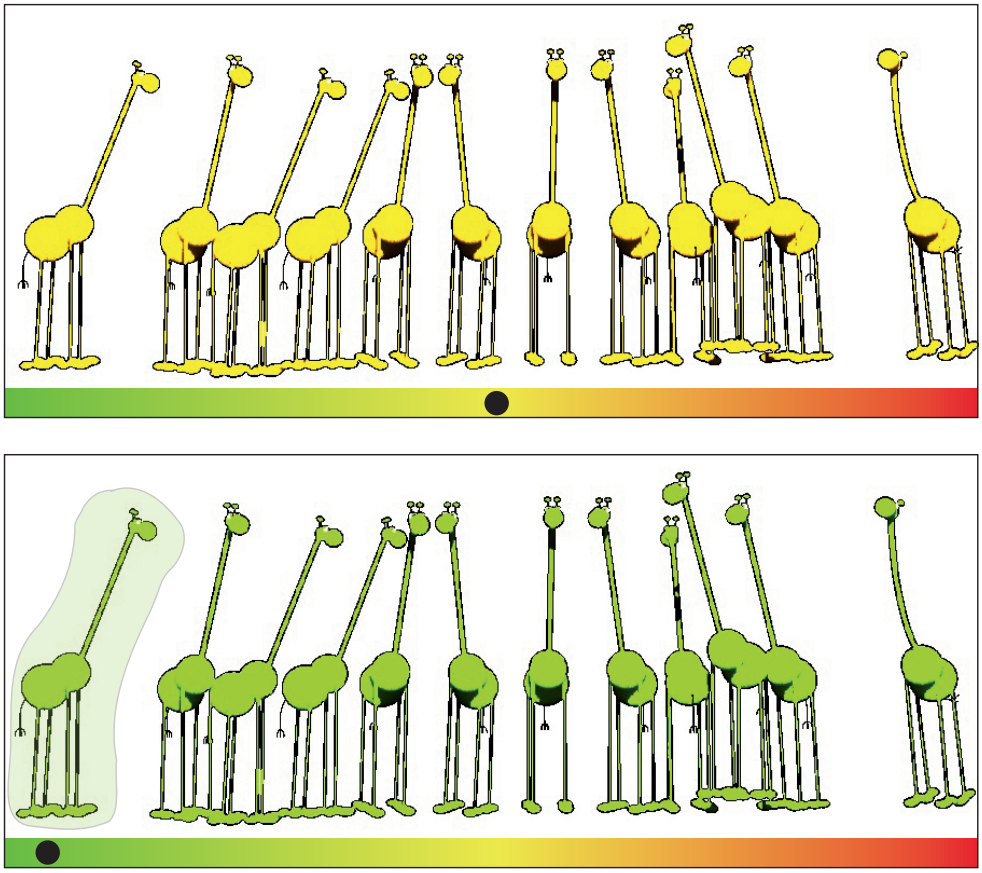


Figure 2.19 *The user can modify the weight of certain objects to influence the clustering. Black points indicate the cluster leader positions. Halos indicate stronger weights.*

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:

$$mean(x) := \frac{\sum_{i=0}^{n} x_i c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)}{\sum_{i=0}^{n} c_i g\left(\|\frac{x-x_i}{H_i}\|^2\right)} \tag{2.6}$$

As in [CM02], $y_{i+1} := mean(y_i)$ 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.

2.8 Results

This chapter presented a real-time technique to *cluster* a dynamic 3D scene. 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 can be changed interactively for easy tuning. Interesting improvements are possible, for instance, exploring 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 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. One interesting venue of future work in this direction is to investigate a real-time sampling method to address this issue.

In comparison to image based approaches like [KWH06], our algorithm shows strengths as well as weaknesses. Temporal coherence becomes simpler and invisible geometry can be treated (we discussed the implications in Section 2.4.2). 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.

2.9 Discussion

Group information can be very effective in revealing the structure of a scene. When different parameters are taken into account, different scene features can be revealed. We have raised a series of control features that reflect a set of 'handles' in which the user can hold to modify the scene representation to create comprehensible images. For all features here exploited, default attributes can be derived, e.g., the extraction of clusters according to the object's positions and the automatic derivation of LOA according to the group's average position. In this way, it can be adapted to different level of user expertise and artistic composition goals.

The presented approach can be applied in contexts other than NPR and graphics. For example, to perform simulation 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. All these possibilities are unified in one approach. Our technique can also be used for 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 computer-generated illustration systems.

Because grouping is fundamentally a perceptually driven task, many subjective criteria can potentially be taken into consideration. Expressing arbitrary subjective criteria in computationally meaningful forms requires mathematically formulated heuristics. Nevertheless, heuristics might not always produce the expected behavior, particularly in highly subjective perceptions of grouping (such as 'similarity'). However, given that our approach is decoupled from the actual algorithms used in computing the feature space, we envisage that as heuristics get better at expressing subjective criteria, the approach will yield more interesting results in expressive rendering and other fields.

3 Controllable Diffusion For Vector Graphics

A medium in art may refer to the type of material used to create artwork. In this sense, no limit exists on what defines an art medium. Traditional media, as well as any arbitrary object used to create a piece of artwork, are considered an art medium [Sto10]. Different types of media provide artists with a great flexibility in expressiveness. Many options exist in which one may wish to explore in creating original artworks. In Figure 3.1 (left), the artist Romero Britto[1] uses acrylic on canvas to create an impression that reflects his optimistic faith in the world around him. Many of his paintings decorate hospital facilities and have been found to be very successful among children. Various other art techniques provide other art media. Photography falls into the artwork category and the medium it uses traditionally is paper and film. However, with the move to digital technology for printing, film has been replaced by the computer. The intense gaze of an Indian girl photograph depicted in Figure 3.1 (middle) was captured using a digital camera. This kind of medium provides the user with enormous flexibility greatly due to its near zero cost of experimentation and real-time display of the result. In the present days, a list of art media includes digital files and printed material, as more technology based projects are accepted as art.

Figure 3.1 *From left to right: Lost of Innocence by Romero Britto©. Intense gaze of Indian girl (image thanks to* Betuca*). Computer illustration by Ken Wong's electronic portfolio accessible on the Internet.*

Technology has also played a major role in producing visual contents with the advent of digital design and layout programs. The main benefit of using a computer is the flexibility it offers to people in the field of illustration or design. Digital creation has become a great

[1] http://www.britto.com/

source of illustrations which are, nowadays, easier to share and expose due to the advent of the Internet. Figure 3.1 (right) depicts an illustration made using graphic software and shared by an electronic portfolio created by the artist Ken Wong [2]. If an artist wants to pursue sharing their artwork electronically, digital drawing can save some time over having to digitize the work using a scanner, where readjustments might be necessary depending on how the scan turned out. The popularity of the personal computers also became possible thanks to intuitive input mechanisms that often imitate drawing tools. Images drawn on devices like tablets are easily transferred into drawing programs like CAD or edition programs. With such devices, even degrees of shadings can be achieved depending on the pressure sensitivity of its surface. Nevertheless, the process of including the computer into the creation process raises important design concerns about the representation, and manipulation processes. The latter two are strongly related to each other since different image representations can facilitate or prevent manipulations and therefore a careful analysis is necessary.

This chapter first discusses methods to represent and manipulate two-dimensional drawings. It gives an overview over raster and vector graphics representations, focusing on their characteristics to enable the creation and manipulation of complex color illustrations. Being a recent concern in the Graphics community, we will present approaches to enable vector-based primitives to depict complex color gradients. Especially, we will review the Diffusion Curves primitive by Orzan *et al.* [OBW+08]. Then we present a novel algorithm to increase the usability and expressiveness of this technique by providing the user with control over the color diffusion process. This algorithm was developed as a part of this dissertation, and was recently published at the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR 2010 [BEDT10].

3.1 Raster Graphics Images

Raster Graphics is the two-dimensional representation of an image as a finite set of points defined by numerical values of a matrix structure where each entry is a pixel. Typically, each point in an image is represented by three or four component intensities such as red, green, blue and alpha. Modern systems encode pixel color values by devoting some bits for each of the RGBA separate components. By using an appropriate combination of these elements, million colors can be represented, though not necessarily distinguishable by the human visual system. Raster images are suitable for the representation of complex variations of shapes and colors, like photographs. They are also easily displayed and used in many electronic devices. Several software solutions are available to create, manipulate, and store raster images and, usually, they provide a large variety of manipulation tools to alter a single or a set of pixels at once (e.g., Gaussian blurs, edge detectors, image sharpeners, or artistic filters).

[2] http://www.kenart.net/

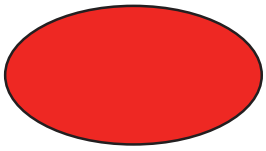
The possibility to store arbitrary colors in each pixel makes raster images very suitable to represent any kind of visual content, but it comes with a cost. First, image files are usually large and directly related to the quality of the representation (resolution). Second, due to its sample-based representation, pixels are isolated entities and information about objects constituting the image are difficult to retrieve. This drawback makes many manipulation techniques very difficult, for example, the deformation of objects. A third important issue is related to image resizing. Since the image width and height are fixed values, image scaling operations usually result in artifacts. An example is depicted in Figure 3.2. The original image (left) was scaled by a factor of seven and the result is depicted in the middle image. This operation degrades greatly the quality of the image as it reveals its pixel structure.

Figure 3.2 *Left: Original image. Close-up after image scaling using raster (middle) and vector (right) representations.*

3.2 Vector Graphics Images

Vector Graphics is an image representation that uses geometric shapes to represent its content. Objects in the image are represented by a set of elements (such as rectangles, circles, ellipses, Bézier curves), or text. All shapes have attributes specific to that shape for positioning and sizing. They also have presentation attributes which affect operations and attributes like fill and stroke color, border width and more. Figure 3.3 illustrates an ellipse object and its main shape and presentation attributes used to render it. Vector graphics programs use these mathematical formulae to construct the screen image, building the best quality image possible given the screen resolution. Therefore, when a scaling is performed on a vector graphic version of the previous example of a flower depicted in Figure 3.2 (left), the result is still a sharp and fine rendition of its content (right), in opposition of its raster graphics counterpart (middle). This occurs because the mathematical formulae determine where the dots that make up the image should be placed for the best results when displaying the image. Since these formulae can produce an image scalable to any size and detail, the quality

of the image is limited only by the resolution of the display, and the file size of vector data generating the image stays the same.



```
cx="100"  cy="100"
rx="100"  ry="54"
stroke="black"
stroke-width="2"
fill="red"
```

Attribute	Explanation
cx	The x-axis center of the ellipse
cy	The y-axis center of the ellipse
rx	The length of the ellipse's radius along the x-axis
ry	The length of the ellipse's radius along the y-axis
Presentation Attributes	Color, FillStroke, Graphics

Figure 3.3 *A Vector Graphics element: an example of an ellipse and its attributes.*

For artists it is often beneficial to work with vector-based applications because objects can be described directly via primitives that represent the shape instead of having to work with pixels. The image can be manipulated by editing screen objects which are then saved as modifications to their mathematical formulae. Most geometric-based software (for example, Corel CorelDRAW©, Inkscape©) provide intuitive graphical user interface to group, stylize, transform and composite elements into previously rendered objects. The feature set includes nested transformations (objects can be moved, scaled, rotated, distorted, or flipped), clipping paths, alpha masks, filter effects and template objects.

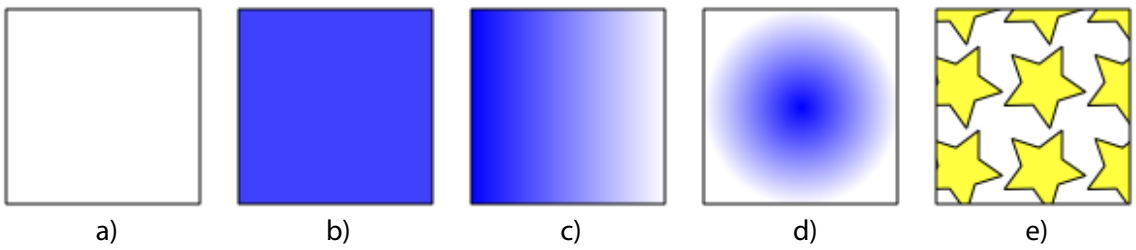


Figure 3.4 *Vector Graphics color fill. a) No paint (transparent). b) Flat (solid) color. c) Linear Gradient (a smooth transition between two or more colors). d) Radial Gradient (a smooth transition between two or more colors in a radial direction). e) Pattern (filled with a repeating pattern).*

To control the appearance of objects, there are a number of different options for the *fill* paint attribute. A fill is the only way to create a solid area of color. Examples of the different options are shown in Figure 3.4. Easy to manipulate, expressive results can be achieved even when only flat colors are used, as shown in Figure 3.5 (left). Gradients allows users to imitate shading effects and bring three dimensional look to pictures. Basically, a gradient is a smooth transition between two or more colors. Vector graphics supports two types of gradients: linear (along a line, as shown in Figure 3.4 (c)), or radial (away from a center, with possibly unequal axes and a noncentral focus point, shown in Figure 3.4 (d). Interesting shadings can also be created by composing gradients and flat colors, as shown in

Figure 3.5 (middle). Patterns and textures (Figure 3.4 (e)) are other expressive options that allow texture looking into pictures. As any other filling element, they can be used alone or overlapping other effects like gradient and make possible to increase the tridimensional look of images, as shown in Figure 3.5 (right). Unfortunately, despite the expressiveness power offered by these features, some challenges remain. Traditional vector graphics tend to look less detailed than their pixel-counterparts, as it is visible when compared to real photographs. The advent of more advanced color fills helped to close this gap. New solutions like *gradient meshes* (Adobe Illustrator©, Corel CorelDRAW©) are capable of representing almost photo-realistic images by interpolating colors on a planar quadrilateral mesh.

Figure 3.5 *Filling attributes can bring expressiveness into the creation process. Left: Vector image created with flat colors. Middle: Gradients and flat colors can be composed to achieve tridimensional look. Right: Patterns allow texture mapping look to bidimensinal images.*

With gradient mesh, designers can create complex and photorealistic art using carefully laid out gradients. A gradient mesh can be thought of as net placed over a solid filled object, as illustrated in Figure 3.6 (left). Using the various points where the mesh intersect, punctual colors can be added and are then interpolated over the entire mesh by a bicubic interpolation algorithm. By adding colors and manipulating the mesh structure itself, flat colors or complex gradients that imitate highlights or shadows can be depicted and the result be impressive realistic shades of colors as for the Yukio Miyamoto [3] illustration depicted in Figure 3.6 (right).

Unfortunately, creating a gradient mesh manually requires skill and is labor intensive. To help to bridge the gap, Sun *et al.* [SLWS07] described a semi-automatic optimization process for fitting a gradient mesh to a given image region. With this approach, a resulting gradient mesh can then be obtained in a matter of minutes instead of the exhaustive hours that it would require for a skilled user to obtain the same result. Nevertheless, this algorithm requires user assistance both to segment the image into regions, and to construct the mesh

[3] `www.khulsey.com/masters_yukio_miyamoto.html`

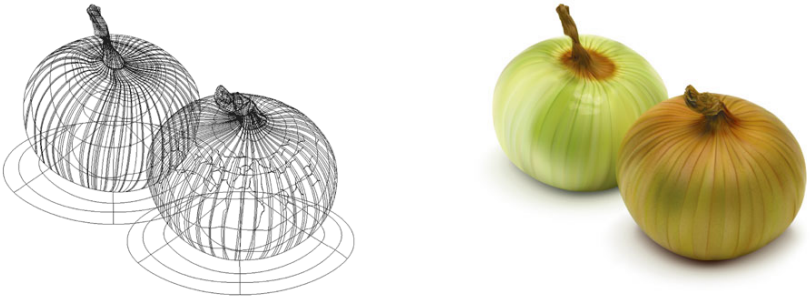


Figure 3.6 *Photorealistic design with gradient mesh tool. Left: net support for color interpolation. Right: result after interpolating.*

(the user must choose the grid corners and usually the spacings of the grid lines for each region). Then later, in the same spirit, Lai *et al.* [LHM09] presented a fully automatic algorithm to represent an arbitrary image region using a single mesh. Their mesh representation handles the creation of complex geometry and even holes, issues presented in the previous approach (see Figure 3.7). Their results are general, compact, and compatible with commercial software. Although these automatic approaches can dramatically increase the time necessary to create a mesh, gradient meshes are inefficient and difficult to manipulate since too many small patches are generated in smooth regions. In addition, the introduction of such a grid adds more complexity to the vector illustration, trading off some of its editing advantages against a richer representation.

To improve the creation of complex vector-based color gradients while keeping the process simple and intuitive, Orzan *et al.* [OBW+08] introduced a new vector primitive called *Diffusion Curves*. We'll present the details of this technique in the next section.

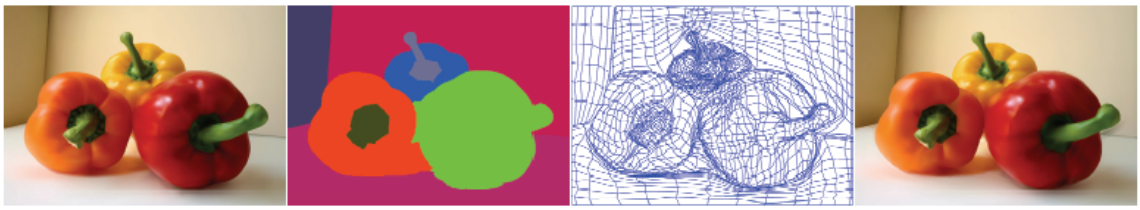


Figure 3.7 *An automatic algorithm to produce gradient mesh from pictures [LHM09]. From left to right: original photograph, image segmentation, generated mesh, vector result.*

3.3 Diffusion Curves

To understand the concept of diffusion curves, two important observations need to be discussed: the importance of edges to the recognition of images, and the process of creating 2D drawings.

In an image context, lines are often important features and encode much of the initial scene information, which is exploited by many vision-related algorithms [Eld99, Car88]. These studies have demonstrated that edges, augmented by color and blur information, constitute a near-complete and natural primitive for encoding images. Consequently, some authors, as Elder *et al.* [EG01], have exploited these studies and introduced image manipulation techniques based on deleting, copying or pasting edges. Further, it was also demonstrated that edges can be used to compress images [EZ98] and manipulate photographs [EZ96, OBBT07].

On the broader level, we verify that the process of producing traditional 2D freehand drawings can be roughly divided into two major stages: the drawing process and the painting process. Once drawings are traced onto the canvas, colors are applied in the resulting areas of the image. Therefore, the relation of lines and colors are of major importance into the creation process. Although filling regions with solid colors can be very expressive, complex color gradients more effectively convey the illusion of volume and illumination, enabling the depiction of a broad range of effects.

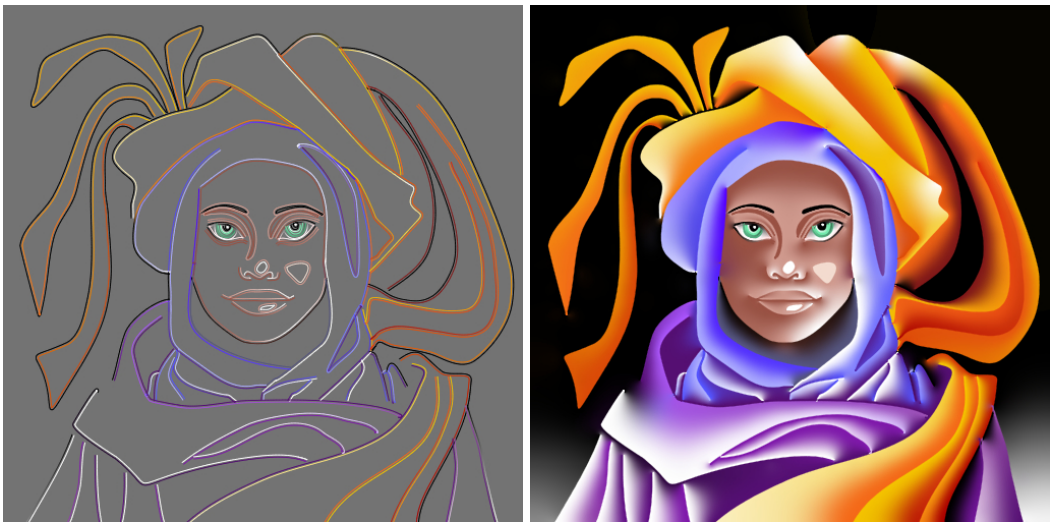


Figure 3.8 *Diffusion Curves approach. Left: Image encoded by curve primitives. Right: Image after diffusion process.*

This environment of edges and colors filling images laid the groundwork for Diffusion Curves [OBW+08], a vector representation of edges and their attributes. Diffusion curves greatly increase the manipulation capabilities suggested by Elder *et al.* [EG01] to include shape, color, contrast, and blur operations to generate images allowing the creation of rich colored areas. The process is simple and intuitive. The formulation of diffusion curves allows for the flexible creation of vector graphics images from a set of curves and colors:

a diffusion process fills out the parts of the image that are away from curves, as shown in Figure 3.8.

Figure 3.9 depicts the representation of diffusion curves. Technically, a Diffusion-Curve image is defined by a set of Bézier curves (a). For each curve, the user specifies two sets of color control points (b), one at each side, corresponding to color constraints on the right and left half space of the curve. By specifying a set of blur points along the curve (c), an artist can also create smooth color transitions across curve boundaries. Color and blur attributes can vary along a curve to create rich color transitions. This variation is guided by an interpolation between the attribute control points in attribute space. Then, color curves are diffused outwards by a solving a Poisson Equation (more details are presented in Section 3.6.1) and are combined with the blur diffusion to define the final image (d). The diffusion curves themselves hold an intuitive meaning and are very compact since the colors and blur values are only specified at control points of the curves.

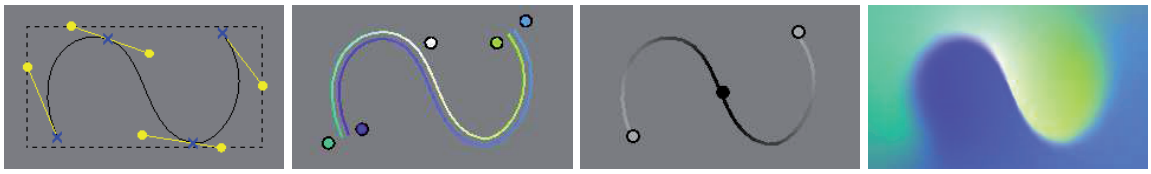


Figure 3.9 *A Diffusion curve: (a) A geometric curve described by a Bézier spline. (b) Arbitrary colors on either side, linearly interpolated along the curve. (c) A blur amount linearly interpolated along the curve. The final color image (d) is obtained by diffusion and reblurring.*

The fact that the diffusion curves representation relies on a small number of simple entities makes it particularly well-suited for artists. It's representation is compact and compatible with vector graphics representations because image can be stored by sets of control point attributes as curve structure, colors, and blur. The diffusion process employed by this technique allows for highly expressive results. Later, Winnemöller *et al.* [WOBT09] also proved that more complex attributes as normals and texture coordinates can be easily attached to curves in order to be diffused and enrich 2D drawings. Several works have also improved the diffusion process in order to obtain fine grained results [JCW09a], and extended the diffusion curves idea to render 3D surfaces details [JCW09b].

Diffusion is, indeed, a key component that enables a rich, yet simple definition of resolution-independent illustrations. However, despite many advantages, the Diffusion Curve model has limitations in certain situations and does not always seem to agree with how an artist wants to use the them in an illustration software system. First, the diffusion itself cannot be controlled, only the colors. Further, the fact that color needs to be defined everywhere along the curve can lead to tedious and nonintuitive interactions. In the following sections, we present how this lack of controllability affects the creation process and propose a reformulation of diffusion curves in order to provide the user with a finer iteration with the

resulting image. We will present a number of adaptations to diffusion curves that constrain how color is spread across the image. Specifically, we argue for the utility of controlling the speed and direction of the color diffusion, and the ability to have barriers that can be defined without the need to specify a particular color along these curves. We also describe how this can be implemented by solving a linear system, and demonstrate the effectiveness of our solution on a number of examples.

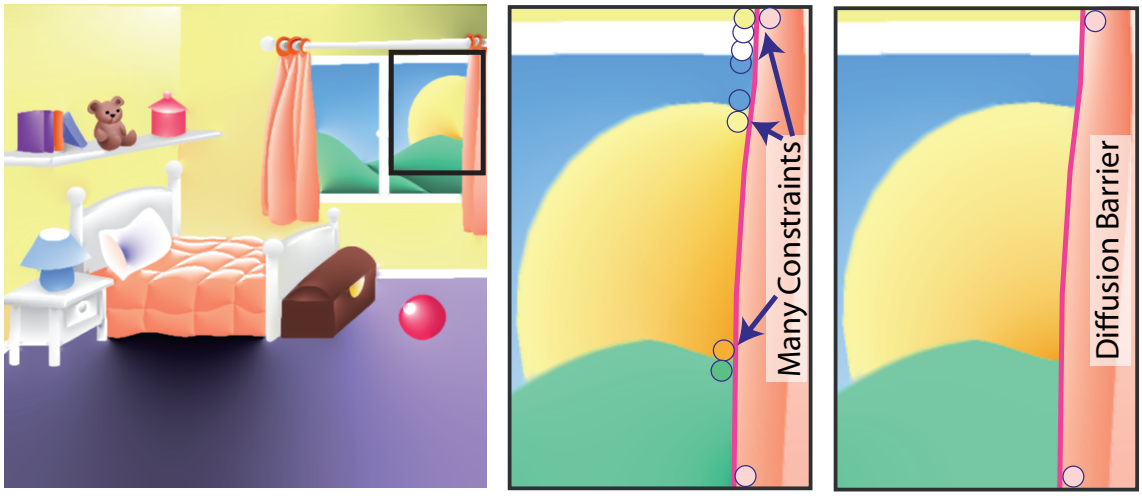


Figure 3.10 *Diffusion Curves allow us to draw vector images with a rich set of color gradients (left). Our work reduces the number of color definitions (middle) for an equivalent output (right).*

3.4 Controllable Diffusion Curves

To illustrate the importance of controlling a color diffusion process, we will start taking a look at some examples. Johnston [Joh02] pointed out that not all curves should diffuse values to both sides. In particular, along occlusion boundaries, diffusion should typically only occur on the occluding part. To handle these exceptions, a blending mask is manually created that limits the extent of diffusion. On the other hand, diffusion curves do not offer this kind of control, and many illustrations exhibit unwanted halos or modifications of the color gradients, because the artist is forced to specify diffused colors on *both sides of each curve*, even if there is no need for them. An example is illustrated in Figure 3.10. The curtain needs color constraints even on the occluding boundary, leading to a large number of additional color constraints whose presence affected the radial appearance of the sun's gradient. Instead, the approach presented in this chapter allows the definition of *diffusion barriers* that block diffusion without emitting colors. No color constraints need to

be defined on the blocked side, which avoids the inconsistencies, as illustrated in the inset, that can arise from such unnecessary color constraints that do not perfectly match up with the geometry. Diffusion barriers can also be used to define shapes that can simply be filled with colors via diffusion curves, that are applied similarly to a paint-bucket tool.

Another example is a gradient where colors from different places intervene with differing strengths. Figure 3.11 illustrates how the shadow underneath the bed also strongly influences the surrounding floor. Controlling the color strength directly provides influence over the impact of a color. Previously, the artist was obliged to place additional curves to simulate a non-linear diffusion behavior. Not only is this tedious, but any color change also implied that all these curves would need to be adapted. Influencing the diffusion via color strength is independent of the associated colors.

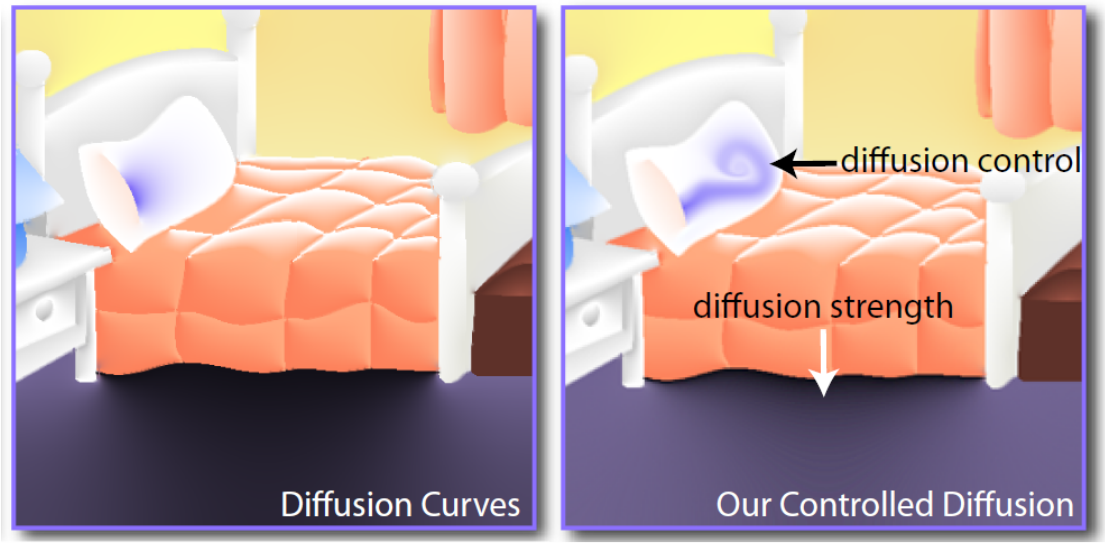


Figure 3.11 *Our work also allows the control the diffusion strength of certain colors (right, floor), or even influence diffusion directions (right, cushion).*

We also introduce a control over diffusion orientation which helps to guide color locally (almost like a smearing tool). It allows complex shapes and color variations that cannot be achieved with the uniform diffusion available from previous work. In Figure 3.11 (right) this technique was used to create a complex pattern on the cushion, and we show in the following sections how expressive results are possible while keeping the representation vector-based.

Finally, in some situations, one needs to use a color resulting from the diffusion process at a different location of the scene; e.g., if a shape represents a thin occluder and one wants to guarantee a smooth color transition on the occludee. An example of this situation is depicted in Figure 3.12. On the left, we show that, to give the impression of a continuous color diffusion across the wall, the user was forced to place color constraints on the left

and right sides of the curtain hanger. Not only this can be a tedious a process, but care must be taken in order to choose colors that match the diffusion and give the impression of a continuous wall paint. On the right, we show the same result made with non-local diffusion. Here, the colors defined on one side of the hanger were diffused through its right side without the need of any extra color constraint. This is made possible because we allow colors to be diffused to arbitrary regions of the image and no longer restrained into it's neighborhood.

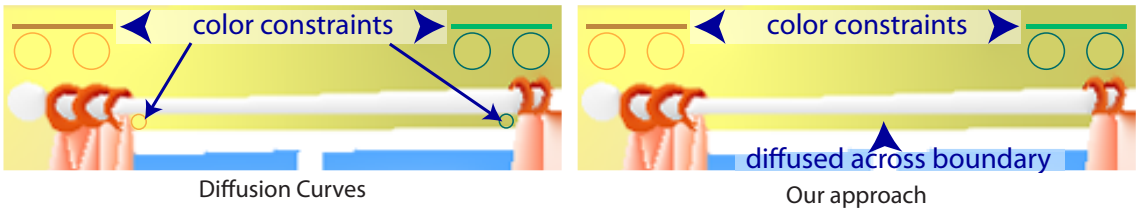


Figure 3.12 *The artist decided to add a color gradient to the wall. Usually it would be blocked by the curtain rod, but by virtually connecting the two regions, colors are transferred from one side to the other. Further, the color gradient was conveniently defined in the interior region of the wall, similar to a paint-bucket fill. This was enabled by setting some boundaries to diffusion barriers.*

We will show in the next sections how our technique avoids such problems by enabling control over the diffusion process and related diffusion constraints while maintaining the simplicity of the original Diffusion Curves. The new degrees of artistic freedom allow for further expressivity and enable an intuitive design of complex illustrations and color gradients. Figure 3.13 shows the entire bedroom image after applying our diffusion extensions. Precisely, our contributions are:

- Diffusion barriers that block diffusion (but do not emit color);
- Control over diffusion anisotropy and orientation;
- Control over diffusion strength (speed);
- A generalized solution method for non-local diffusion.

Figure 3.13 *The figure shows Figure 3.10 (left) that was modified using the diffusion curve extensions presented in this chapter. Our solution offers more control over the diffusion process and makes several tasks simpler.*

3.5 Related Work on Diffusion

As previously discussed, edges are often important feature to the recognition of objects through the natural vision processes. This fact motivated many vision-related techniques to compress images [EZ98] or manipulate photographs [EZ96, OBBT07]. These tasks often underly the principle of the Poisson equation which has found applications in many contexts of computer graphics, e.g., image editing, where it enables seamless cut-and-paste operations as presented in the work of Perez *et al.* [PGB03]. This work provides a series of manipulation techniques that range from replacement by, or mixing with, another source image region, to alterations of some aspects of the original image inside the selection, such as texture, illumination, or color (see examples in Figure 3.14). All these features have the Poisson equation at their heart. Solving this equation is equivalent to compute the function whose gradient is the closest, in the L_2-norm, to some prescribed vector field, called by the authors the guidance vector field, under given boundary conditions. The user usually specify such guidance vector by selecting a region from source images and the algorithm reconstructs the target regions to obtain a seamless manipulation.

Besides Diffusion Curves, the Poisson equation is also the basis of the real-time gradient domain painting [MP08], where colors are sparsely defined along curves drawn in raster images and interpolated everywhere else. Jeschke *et al.* [JCW09a] introduced a faster

Figure 3.14 *Images from Poisson Image Editing by [PGB03] showing application of the Poisson Equation. Left: Seamless cloning. Right: By importing seamlessly a piece of the background undesirable artifacts can easily be hidden.*

and more accurate solver by exploiting the fact that the constraints are very sparse. Also, triangulation-based solutions [FHL+09] are interesting alternatives to pixel-based diffusions. Although all these solutions are efficient, the diffusion is always uniform. Our work is strongly inspired by the aforementioned approaches, but we aim for a more flexible tool that provides the user with more control over the diffusion process.

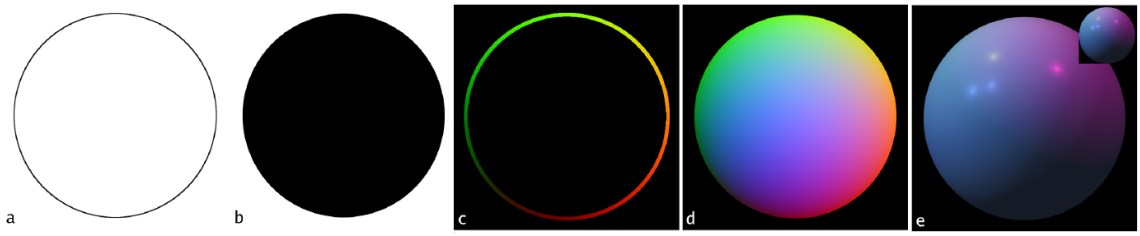


Figure 3.15 *Curve normal approximation, interpolation and character illumination from 'Lumo: Illumination from Cel Animation by [Joh02].*

Diffusion processes were also used by Johnston [Joh02] to interpolate normals derived from outlines of a 2D drawing, as shown in Figure 3.15. In his technique, 2D normals are calculated by computing the gradient along outlines (c) with the help of a mask (b) to constraint these normals to point outwards the curve. These normals are then interpolated and diffused to the interior of the circle (d) by solving a partial differential equation with a similar result as solving the Poisson equation. Once normals are computed, the 2D drawing can be relighted giving it a 3D appearance (e). Jeschke *et al.* [JCW09b] used a very similar diffusion curve technique to diffuse colors on 3D surfaces and create textures, displacement maps and geometry images (Figure 3.16). They present three algorithms that enable the real-time color diffusion on surfaces while retaining the vector character of the diffusion curve style. These approaches can benefit from the extensions proposed in this chapter as

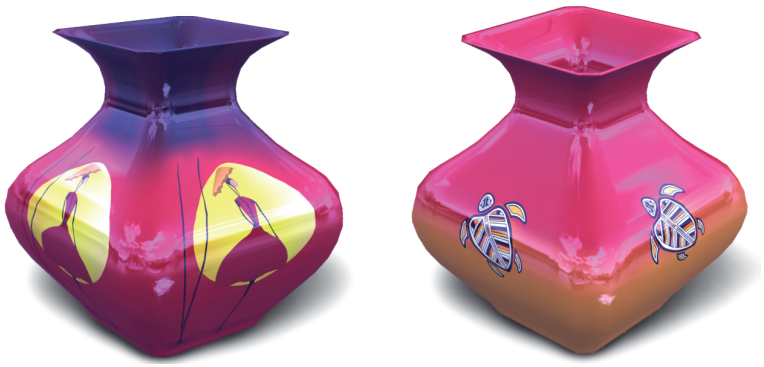


Figure 3.16 *A 3D vase rendered in different styles using diffusion curve. Image from 'Rendering Surface Details with Diffusion Curves' by [JCW09b].*

well.

Isenberg *et al.* [IBCSC06] also proposed a pipeline to enrich 2D vector illustrations by capturing line information from another data structure (as 3D meshes) into a line primitive called G-Stroke [IB06], an augmented line primitive used to store stroke properties along with geometrical and topological information used during the vector rendering process. This approach also proposes to capture shading information (as Gouraud shading) by using a library to redirect OpenGL calls into a 2D mesh structure, having, therefore, triangles to depict shading. There are a number of techniques that achieve resolution independence, of which reconstruction using diffusion is just one possibility. The Ardeco system [LL06b] simulates complex shading via local approximations using linear or quadratic gradients. Since it is an image conversion process, the results may contain a very large number of regions. Solutions for the derivation of vector information from implicit surfaces [SEH08] and meshes [EPD09, EWHS08] also exist, but the works focus on the extraction of shape primitives, not their visual appearance or texture.

3.6 Mathematical Background

Before presenting our algorithm, we will discuss interpolations based on the Poisson equation, which is the principle that underlies Diffusion Curves (Section 3.6.1). We reformulate the relationship into a constrained linear system that will provide the basis for our work (Section 3.6.2). We present how to modify the system to support diffusion barriers, curves that block the diffusion processes without emitting colors (Section 3.7.1). We show how to guide the diffusion process by orienting it according to a user-specified flow field (Section 3.7.2). We then add the possibility to control the strength of a color during the diffusion (Section 3.7.3) before addressing a non-local extension of the diffusion process to allow us, e.g., to transfer color from one part of the image to another (Section 3.8).

3.6.1 Diffusion Process

Our work builds upon the Poisson-equation framework previously applied in many contexts [TT99, PGB03, OBW+08]. Consider an image I with n pixels, $\{I_k \,|\, k \in 1 \ldots n\}$ (colors are addressed individually as I_k or as a grid simply as $I_{i,j}$). The goal is to derive an interpolant matching a set of constrained pixel colors $\{C_k \,|\, k \in \mathcal{I}\}$, where $\mathcal{I} \subseteq \{1 \ldots n\}$ is an index set, and having a gradient close (L_2-norm) to a given vector field $w = \{w_k \,|\, k \in 1 \ldots n\}$. The vector field values are also stored in pixels and addressed, just like for I with $w_k := (w_k^x, w_k^y)$.

The image I is defined implicitly using the Poisson equation:

$$\begin{aligned} \triangle I &= \operatorname{div} w, \\ \text{and } I_k &= C_k, \ \forall k \in \mathcal{I}, \end{aligned} \tag{3.1}$$

where $\triangle$ is the Laplace operator, and div is the divergence operator. The solution is usually found by solving a discretized version of Equation 3.1 for each color channel separately. A Gauss-Seidel solver could be used, but more efficient conjugate gradient or multigrid solvers (as in [OBW+08]) are an option. In the case of Gauss-Seidel iterations, a value $I_{i,j}$ needs to be updated by adding $(I_{i+1,j} + I_{i-1,j} + I_{i,j+1} + I_{i,j-1} + \operatorname{div} w_{i,j})/4$.

In the case of Diffusion Curves, colors are specified along each side of the curve and represent hard constraints. In addition, the vector field w is zero everywhere except across constraint curves. In other words, the solution will show a continuous, smooth change of colors except across hard constraints.

3.6.2 Reformulating the Diffusion Process

To facilitate the understanding of how to influence the diffusion process, we need to look a little closer at its properties. After solving Equation 3.1, the resulting image I is the solution to the following constrained minimization:

$$\begin{aligned} I \;=\; & \underset{\mathit{Image}\ J}{\operatorname{argmin}} \sum_{i=0}^{n} |\nabla J_i - w_i|^2, \\ & \text{subject to } J_k = C_k, \ \ \forall k \in \mathcal{I}, \end{aligned} \tag{3.2}$$

where ∇ is the gradient operator, C_k are the color constraints at the pixel positions $\mathcal{I}$, and $w_i = (w_i^x, w_i^y)$ is the vector field w at pixel position i. The solution is the result of a minimization process that searches for the image whose gradient is the best fit to a given vector field, while respecting the color constraints.

In our work, we want to guide the diffusion process in various ways. Consequently, we cannot always rely on the original Poisson equation. Instead we will set up a constraint system involving hard and soft constraints. Hard constraints ($I_k = C_k$) are the colors stored at pixel positions $\mathcal{I}$ and defined by the initial color curves chosen by the user. Hard constraints will be satisfied exactly. The soft constraints guide the diffusion in the image and

implicitly define the color of the remaining pixels. Soft constraints might not be satisfied exactly, but the solution best satisfies our system in the least square sense.

To illustrate the use of such an equation system, we start by writing the Poisson-equation as a soft constraint system:

$$\begin{pmatrix}\nabla_x\\ \nabla_y\end{pmatrix}\begin{pmatrix}I_1\\ \vdots\\ I_n\end{pmatrix} = \begin{pmatrix}w_1^x\\ \vdots\\ w_n^x\\ w_1^y\\ \vdots\\ w_n^y\end{pmatrix}, \tag{3.3}$$

where ∇_x is a simple matrix that encodes the derivative along the axis x, (w_k^x, w_k^y) is the vector field w's value at pixel $k \in 1 \ldots n$. Each line of ∇_x is of the form $(0, \ldots, 0, -1, 1, 0, \ldots, 0)$. ∇_y is defined accordingly. The matrix encodes the properties of the solution, which we are going to modify for our purposes.

In general, the Equation system 3.3 is over-constrained. To find the least-squares fit, we use the pseudo-inverse. For this, the equation needs to be multiplied by (∇_x^t, ∇_y^t). The result of doing so is:

$$\begin{aligned}(\nabla_x^t, \nabla_y^t)\begin{pmatrix}\nabla_x\\ \nabla_y\end{pmatrix}\begin{pmatrix}I_1\\ \vdots\\ I_n\end{pmatrix} &= (\nabla_x^t, \nabla_y^t)\begin{pmatrix}w_1^x\\ \vdots\\ w_n^x\\ w_1^y\\ \vdots\\ w_n^y\end{pmatrix}\\ (\nabla_x^t\nabla_x + \nabla_y^t\nabla_y)\begin{pmatrix}I_1\\ \vdots\\ I_n\end{pmatrix} &= \nabla_x^t\begin{pmatrix}w_1^x\\ \vdots\\ w_n^x\end{pmatrix} + \nabla_y^t\begin{pmatrix}w_1^y\\ \vdots\\ w_n^y\end{pmatrix},\end{aligned} \tag{3.4}$$

where n is the number of pixels in the image.

The operator $(\nabla_x^t\nabla_x + \nabla_y^t\nabla_y)$ is the discrete version of the Laplace operator and, similarly, $(\nabla_x^t w^x + \nabla_y^t w^y)$ is the divergence. In the Poisson-equation example, the lines in $\nabla_x^t\nabla_x$ have the form $(0, \ldots, 0, 1, -2, 1, 0, \ldots, 0)$ which corresponds to a discrete second derivative. The similar structure of $\nabla_y^t\nabla_y$ implies that the lines in matrix $(\nabla_x^t\nabla_x + \nabla_y^t\nabla_y)$ have the form:

$$4I_{i,j} - I_{i+1,j} - I_{i-1,j} - I_{i,j+1} - I_{i,j-1} = \operatorname{div} w_{i,j},$$

which readily corresponds to the discrete Poisson equation.

3.7 Diffusion Control via Constraint Systems

Figure 3.17 *For Diffusion Curves colors have to be defined along the curve. Here, the background was supposed to be dark and this color is dragged into the interior of the hat (red circle).*

We will now illustrate how to apply the previous formulation in order to improve upon the original standard definition.

3.7.1 Diffusion Barriers

An important issue addressed in this chapter relates to the inflexibility of the relation between curves and colors. For each Diffusion Curve, one needs to define color values for *both* sides of the curve. These double-sided constraints often oblige users to select colors at awkward locations, which produces unwanted side-effects as shown in Figure 3.17. In this example, the black line defined on the exterior side of the hat impacts the color of the ribbons at that location. In order to avoid such artifacts, the user must select colors along the intersection between hat and ribbons. Furthermore, the choice of the right colors is difficult, since they need to match the diffused region accordingly. A better solution would enable the diffusion process to determine some of the colors directly. In other words, the curve in question would, on one side, define colors in the interior of the hat, but, on the other side, simply be used as a barrier to prevent the ribbon and hat colors from mixing.

Being able to specify our constraint system allows us to define locally controllable diffusion behavior. Omitting constraints that relate two pixels breaks the connectivity between them and, therefore, blocks the diffusion process at that location. For a better understanding, Figure 3.18 illustrates such a scenario.

Each pixel represented in the image stores a soft constraint that indicates the direction in which color information is diffused (a). Thus, every pixel diffuses color to its 4-connected neighbors, except those pixels located along the two columns in the middle of the grid. In order to illustrate the impact of locally manipulating a soft constraint, such pixels relate to only two of their neighbors, therefore breaking the connectivity between pixels in these columns. When color information is defined at some pixel positions (b), the result is that

it will be diffused following the connectivity defined by the soft constraint (c-d). Because pixels in the middle of the grid do not relate, color information cannot cross them, and is therefore prevented from mixing (e).

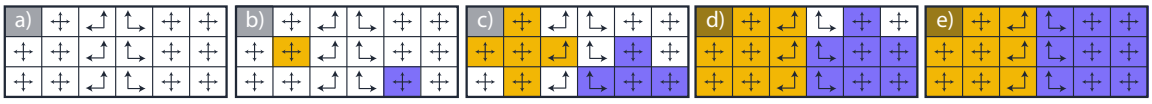


Figure 3.18 *a) Soft constraints indicating the diffusion behavior. b) Color is attributed to some pixels. c-d) Color diffusion illustration. e) Result after minimization. The missing soft constraints placed on the pixels in the middle of the image prevent colors from mixing.*

This simple operation enables us to define *diffusion barriers*: curves that do not actively emit colors but, instead, are responsible for blocking the diffusion process from crossing the pixels underneath it. This kind of curve is useful when the user desires to restrain the diffusion from reaching a certain region without having to actually define any color at that location.

Figure 3.19 shows a practical example of the use of such a curve. Blue and orange color pixels are placed on the image (a). If no other curve is added, the blue and orange pixels will be diffused and mix at certain locations (b). Nevertheless, if we place a circle curve as a barrier (c), the diffusion of the blue pixels will be restrained to its interior; analogously, the orange to its exterior (d). In practice, diffusion barriers are obtained by breaking the connectivity between pixels underneath the curve and those located on its left side (e).

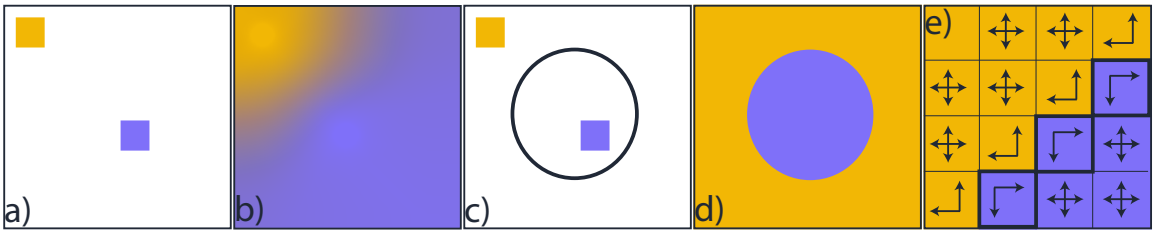


Figure 3.19 *a) Blue and orange strokes. b) Diffusion of blue and orange strokes. c) The circle defines a diffusion barrier. d) Diffusion blocked in the interior and exterior of the circle (diffusion barrier). e) Soft constraints breaking connectivity across the curve.*

It is also possible to only set a different behavior to one side of the curve. In this case, one side emits colors, whereas the other serves as a barrier to prevent colors from crossing the curve at that location. This is an important tool that lets the user avoid defining colors at awkward locations, as previously shown in Figure 3.17.

Figure 3.20 left indicates the lines where placing double-sided color constrained curves is problematic. To solve this problem, we transform these lines into diffusion barriers. For

this, we prevent the right side of the red line from emitting colors to the interior of the hat, but on its left side, we remove the constraints connecting it to the pixels underneath the curve. Analogously, the interior side of the green line will emit its original colors, while its exterior side will work as a barrier. The result of the diffusion is depicted in Figure 3.20 right. Notice that colors diffused from the ribbons are now nicely expanded to the boundaries of the hat without discontinuity artifacts.

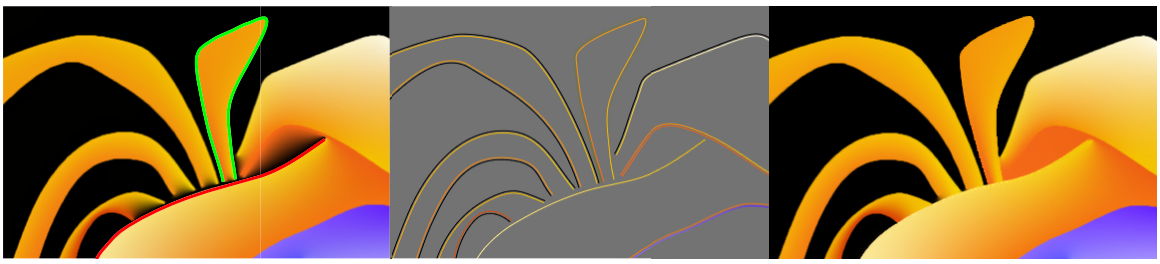


Figure 3.20 *Left: Lines along which problems occur. Center: Red and green line curves are transformed into barrier curves emitting colors from only one of its sides. Right: Result of the diffusion.*

3.7.2 Anisotropic Diffusion

We have seen in Section 3.6.2 that the diffusion process is guided by soft constraints on the derivative. Minimizing derivatives in all directions ensures the uniformity of the results. While this is of interest in many situations, it can be useful to give more control to this process. When drawing motion-blur-like streaks, flames, paint strokes and other phenomena, the color interpolation often has a privileged direction. In other words, continuity is enforced more strongly along one direction than another. Such behavior can be achieved via directional smoothness constraints, resulting in an anisotropic diffusion.

Let's look at a simple example. We have seen that each pixel has a row in the matrices ∇_x and ∇_y which enforces a smoothness along the corresponding axes. Leaving out the row in ∇_y would lead to a diffusion along the x-axis. This is a direct consequence of the fact that differences along the y-axis are no longer penalized. Figure 3.21 (left) shows the influence of this process and illustrates the resulting motion-blur-like streaks obtained with this solution.

For an arbitrary diffusion direction $\vec{d} := (\cos\theta, \sin\theta)^T$, the matrix row needs to be changed. The directional derivative along $\vec{d}$ is $(\nabla_x, \nabla_y)\vec{d}$. Correspondingly, the discretized constraint reads:

$$\cos\theta(I_{i+1,j} - I_{i,j}) + \sin\theta(I_{i,j+1} - I_{i,j}) = 0. \tag{3.5}$$

Replacing the original full-derivative constraint leads to a diffusion process only along

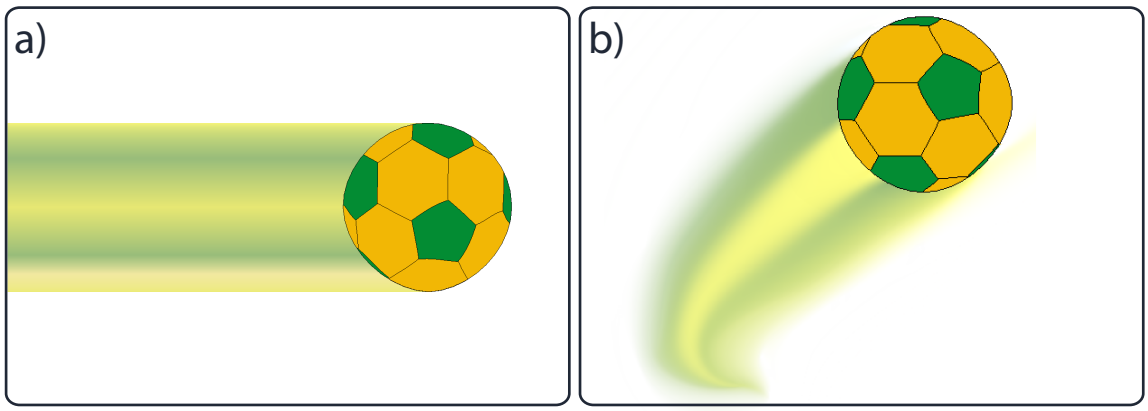


Figure 3.21 *Left: Horizontal diffusion; Right: Path-guided diffusion*

$\vec{d}$. Our goal is to use differing directional constraints to globally guide the diffusion (Figure 3.21, right).

In general, diffusion is rarely just following a single direction. Usually, a tradeoff between a privileged direction and its orthogonal counterpart is wanted. This implies the need for a similar smoothness constraint involving $\vec{d}^{\perp} := (-\sin\theta, \cos\theta)^T$. Adding both equations to the system would result again in a uniform diffusion process (it merely reflects a rotation of the basis vectors, meaning that $\nabla_{\vec{d}}$ and $\nabla_{\vec{d}^{\perp}}$ take the role of ∇_x and ∇_y respectively in Eq. 3.3). Nonetheless, an anisotropic result can be obtained by scaling the constraints differently, as this influences the least-square result of the equation system. We will show how to use this observation to control how strongly a given direction is respected during the color diffusion.

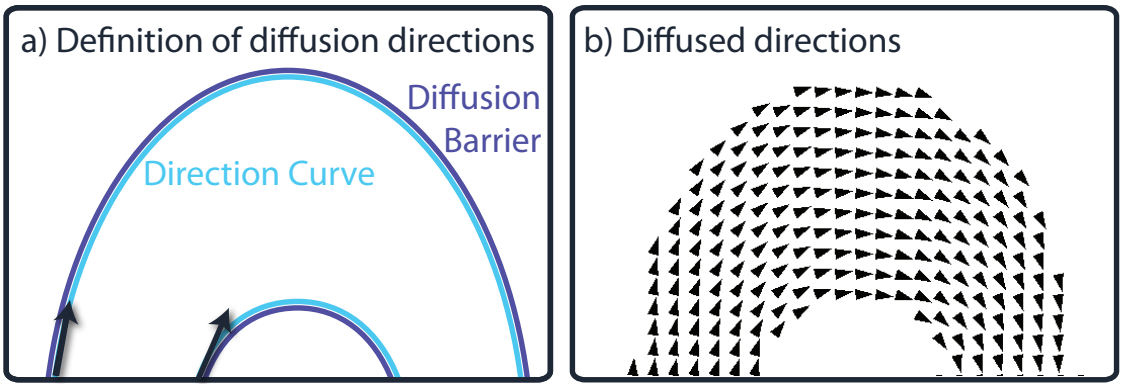


Figure 3.22 *To define per-pixel diffusion directions, the directions are themselves diffused.*

In order to specify a direction $\vec{d}$ and a tradeoff between standard and anisotropic diffusion, we suggest that the user draws *direction curves*. These curves contain a 2D vector, whose direction defines an orientation $\vec{d}$ and whose length results in a scaling factor to per-

form the tradeoff. The vectors defined by the directional curves are spread via a uniform diffusion, leading to a value in each pixel. In our interface, we let the user define vectors of length smaller than one, because a global scale does not affect the solution.

In the example of Figure 3.22 b, diffusion barriers were used to refrain the diffusion to the right (outer arc) and left (inner arc) sides of the curves. Adding color constraints, our diffusion process, according to Equation 3.5, leads to the result depicted in Figure 3.23. Here, color information defined along a curve (a) is dragged by the flow field resulting in an arc-shaped diffusion of colors creating a rainbow effect (b). To define a diffusion direction, the user only defined a single direction per curve, although more would have been possible. Just like colors, directions are interpolated along the curve, but follow the curve's tangent direction.

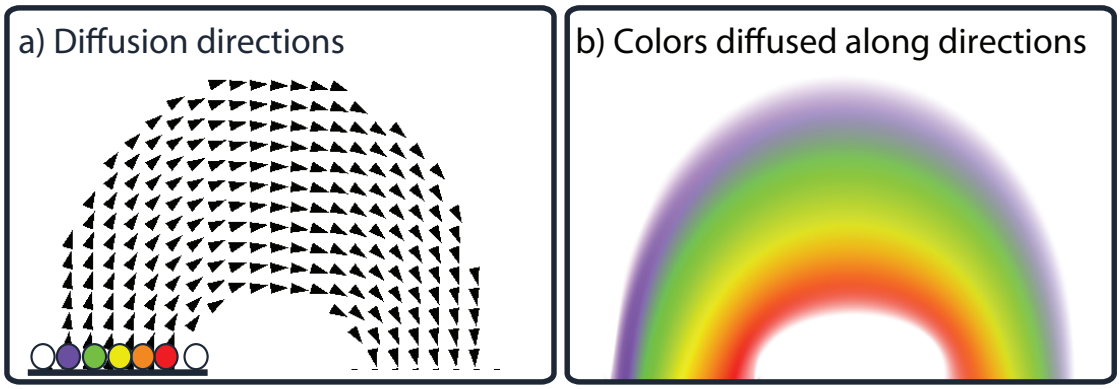


Figure 3.23 *The directions define constraints that ensure that the color is diffused accordingly.*

While in the previous example the directions were basically of constant length, directional constraints can vanish when opposing directions are merged during diffusion. In these areas, a privileged direction does not exist and a uniform diffusion should be applied. This need is compatible with our idea to use the length of $\vec{d}$ to define the tradeoff between standard and anisotropic diffusion. One possibility would be to use $1 - \|\vec{d}\|$ as a scaling factor on the equation according to $\vec{d}^{\perp}$. For $\|\vec{d}\| = 1$, only the diffusion along $\vec{d}$ is applied and for $\vec{d} = 0$ the uniform diffusion is reestablished.

In practice, we use a different solution with a threshold τ. If the vector is longer than the threshold, we renormalize it. Only if it is below τ, we use the length $\vec{d}/\tau$ as before. This threshold could also be diffused, but we found that a global value of 0.5 is usually a good choice.

Figure 3.24 illustrates the influence of anisotropic diffusion and the threshold. Standard diffusion curves lead to a very smooth result that is missing many of the vivid characteristics one would expect in the case of a fire illustration. With our solution, the contrast is improved because color follows the flow of the lines and the final result looks more detailed, although we only relied on the same color curves. Modifying the threshold allows us to obtain a more uniform body for the fire.

Figure 3.24 *A complex anisotropic diffusion defined with a small set of curves. The two examples use different threshold settings to tradeoff uniform and anisotropic diffusion.*

3.7.3 Color Strength

The previous section presented a way to control diffusion directions, but one limitation is that it does not allow us to influence the diffusion speed. In other words, independent of the direction, the diffusion between two colors will weight both colors in the same way. In this section, we will present a solution to attribute a strength to a color in order to define its dominance in the diffusion process.

Figure 3.25 *Left: Standard diffusion (equal strength), Middle: Orange stronger than blue. Right: Varying strength along curve.*

Figure 3.25 depicts a simple example with two color constraints, orange on the top and blue at the bottom. As expected, the diffusion process spreads these colors uniformly over the remaining image connecting both color constraints. In order to achieve fine-grained results, we introduce *color strengths*, a mechanism to control the region of influence of colors during the diffusion process. By manipulating the color strength, the artist can make the orange color become more dominant over the blue, thus pushing the diffusion in this

direction (Figure 3.25, middle). A variety of effects can be obtained if different values of color strength are defined along lines as, for example, the diagonal diffusion effect in Figure 3.25, right. Figure 3.26 shows an example of complex color gradients achieved by manipulating the color strength in the interior and exterior of the eye. Compared with the standard result, we show that the color strength extension can lead to interesting results with no additional curves.

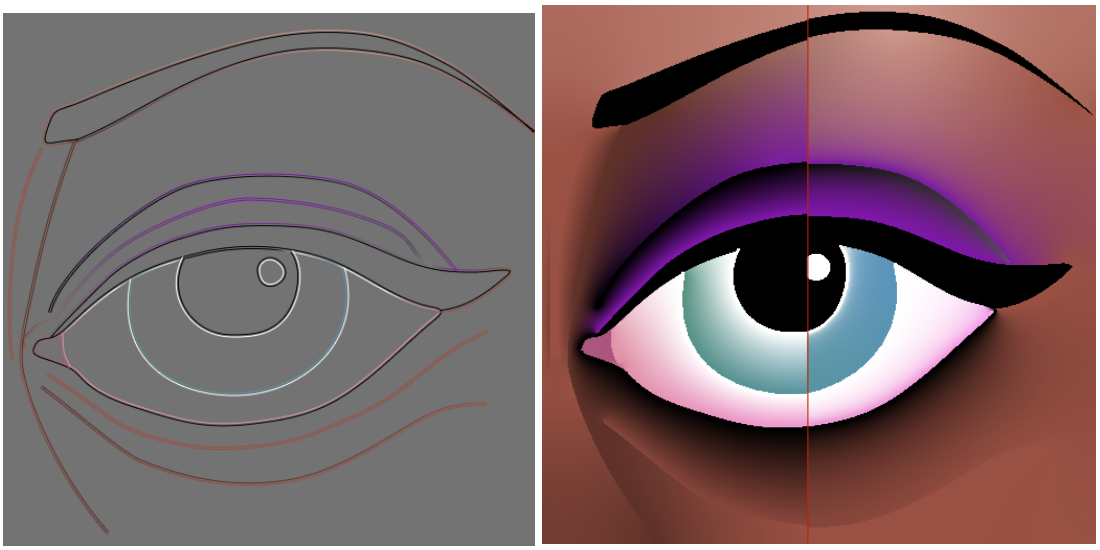


Figure 3.26 *The curves (left) define both parts of the image (right). The left part uses uniform weights, the right has varying weights.*

One way of controlling the strength of colors during diffusion is to formulate this problem as an interpolation process. Intuitively, if we have two colors c_1, c_2 with respective strengths a_1 and a_2, then we would like the interpolation T of the two to yield:

$$T\left((c_1, a_1), (c_2, a_2)\right) = \frac{a_1 c_1 + a_2 c_2}{a_1 + a_2}.$$

As we can see, the equation results in c_i for $a_i \to \infty$, and if $a_1 = 0$, the equation simplifies to c_2. Therefore, the values a_1, a_2 can be used to control the dominance of one color over the other. The result is a linear combination of the initial color values that naturally favors colors with a higher strength. This generalizes:

$$T\left((c_1, a_1), \ldots, (c_k, a_k)\right) = \frac{\sum c_i a_i}{\sum a_i}$$

In some sense, when thinking of a blending process, the strength value will indicate the mixing coefficients. Due to the normalization, such a weighted blending is non-linear and, hence, would not fit into the diffusion framework. Nevertheless, it is possible to linearize the computations using *homogenous colors*.

A homogenous color is defined by a RGB-tuple (r, g, b) and an alpha value $a \neq 0$. Algebraically they resemble homogenous coordinates, widely used in projective geometry calculations [Wil06]. Two homogenous colors (r_1, g_1, b_1, a_1) and (r_2, g_2, b_2, a_2) describe the same actual color when $a_2(r_1, g_1, b_1) = a_1(r_2, g_2, b_2)$. If a_i is not zero, the actual color is obtained via a projection mapping $P(r, g, b, a) = (r/a, g/a, b/a)$. It is easy to verify that the projection of the sum of homogenous colors corresponds to the weighted sum of the actual colors, as defined above.

The key idea of our extension is that the alpha value of a color will define the color strength. In the interface, the user only specifies a standard color $c = (r, g, b)$ and a color strength a. This input is then transformed into a homogenous color by mapping it to (ar, ag, ab, a). Each channel (including the alpha channel) is thus diffused separately, but all following the same diffusion behavior. At the end of the diffusion process, we perform the projection $(r/a, g/a, b/a)$ to obtain the final result. The correctness of this solution becomes clear when inspecting the way that the Gauss-Seidel iterations would update the values in the solver, where an average is computed in each step. The final projection then transforms the result into a weighted sum and the diffusion will reflect the weight.

We explicitly excluded the case when a_i equals zero. It makes the color's contribution to the weighted sum be zero as well. Therefore, it would be possible to use this special case to define diffusion barriers, but in practice, this can lead to small artifacts along the boundaries and care has to be taken to correct them. Such difficulties do not arise with the solution presented in Section 3.7.1.

3.8 Beyond Local Constraints

The final problem we will address relates to the fact that the diffusion is usually locally defined. In other words, a differently colored region will always block the diffusion on its boundary. In this section, we will present a solution to connect different areas of the image to ensure a continuous diffusion between them. To some extent this can serve as a color picking for colors that are only implicitly defined by the diffusion process.

Our solution to ensure the same color on two locations is to simply link them in the diffusion process with soft constraints via a similar condition as the one that usually exists between neighboring pixels. For two pixels I_k, I_j, this translates to a soft constraint of the form $I_k - I_j = 0$. Again, the importance of this similarity can be steered by multiplying the equation with a factor. It is, hence, possible to ensure that both pixels will receive similar values at the end of the diffusion. The definition of such a constraint is simple. We allow the user to link two curves and then define points of correspondence between them, by default, we match both curves via their parametrization uniformly. It is also possible to match one curve with several others if needed.

There are several applications that arise from this strategy. It is possible to address smaller occlusions without resorting to layers, by *channeling* colors from one side of an object to another, as illustrated in Figure 3.12. Here, color gradients are continued across

boundaries and small gaps. Another application is the creation of seamless textures. This is usually a complicated process because colors have to be matched correctly across boundaries. Solutions exist to create such textures in a postprocess [PGB03], but in this case the artist has little control over the appearance. By matching boundary pixels via non-local constraints, the diffusion can be wrapped around the domain and during the design process the result can already be visualized. It is also possible to move and repeat constraints, so that their displacement effectively drags the texture over the screen. An example can be found in Figure 3.27.

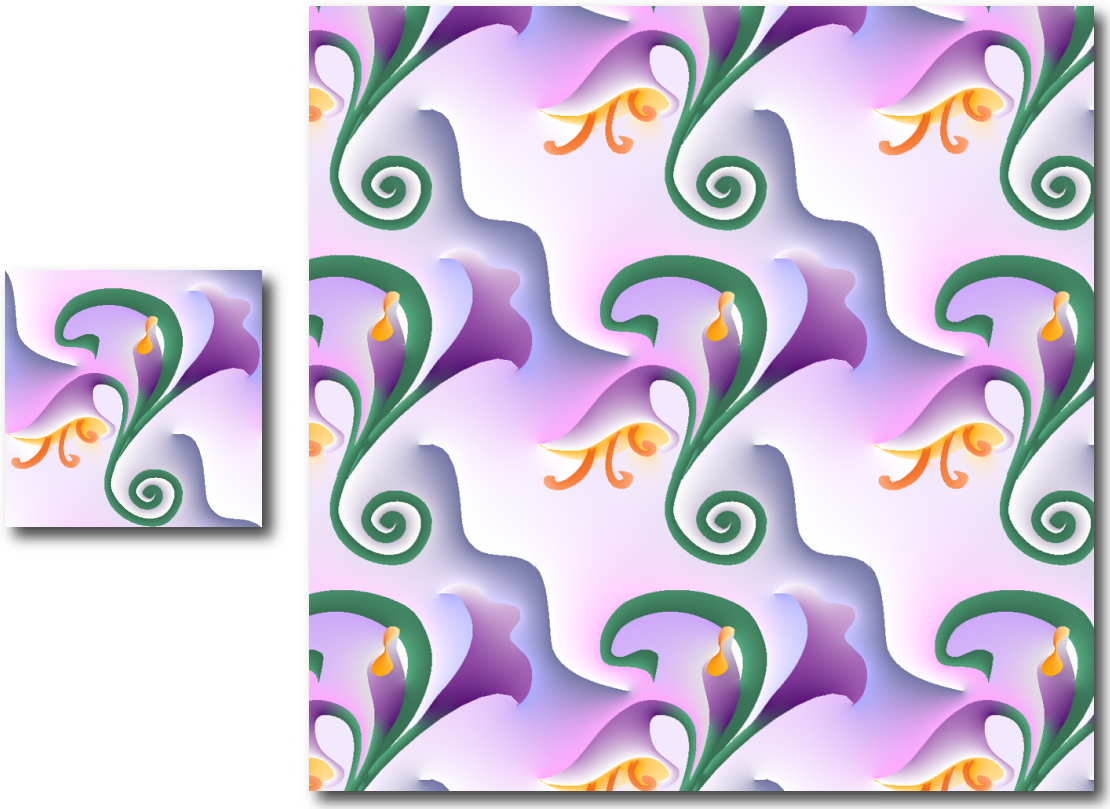


Figure 3.27 *Seamless diffusion textures via non-local constraints.*

3.9 Results

Our method was implemented using OpenGL on a NVIDIA GT285 graphics card. Besides the non-local constraints, all other extensions can be handled locally and thus can be integrated directly into the diffusion curve algorithm. Color strength comes at an added cost of

about 30%, because the diffusion uses four instead of three channels. Directional diffusion comes at roughly twice the cost, because we first diffuse directions, but it still leads to a real-time solution. Unfortunately, the quality of the final result suffers from the multi-grid solver, and, for the non-local constraints, a local diffusion model is no longer very efficient. Instead, we rely on a general global linear solver implemented in CUDA which, as a side benefit leads to more precise images. Unfortunately, due to its generality the solver is slower and our system then does no longer reach real-time performance. For the scene depicted in Figure 3.10, the computation took approximately $4 - 5$ seconds for a 512×512 image. Although the feedback is not instant, it is sufficiently fast to create convincing drawings in a small amount of time. If needed, one could imagine caching non-local constraint results to give an approximate, but faster feedback. In the same spirit, we could also maintain an inverted matrix to allow interactive color adaptation, but we keep such investigations for future work.

Figure 3.28 *Left: Image outlines. Center: Shading using color diffusion. Right: Result of relighting using a normal map created by normal vector diffusion.*

The advantage of our solution is that it is more flexible. Artifacts such as halos can be avoided if the artist desires. Diffusion barriers also make the color fill more intuitive and can be useful for diffusing normals. The folds on the skirts in Figure 3.28 and on the lion's mane in Figure 3.29 benefits from this solution and makes the lighting look more realistic. More details about the derivation of normal maps to enrich 2D drawings are discussed on Chapter 4. We will explore this idea in more details in Chapter 4. Color strength is a simple way of making adjustments to the illustration without increasing its complexity. The same holds for anisotropic diffusion that allows us to increase the richness in the illustrations drastically. A few directional curves can intuitively define a complex color diffusion. Finally, non-local diffusion allows easy color transfer and occlusion treatment.

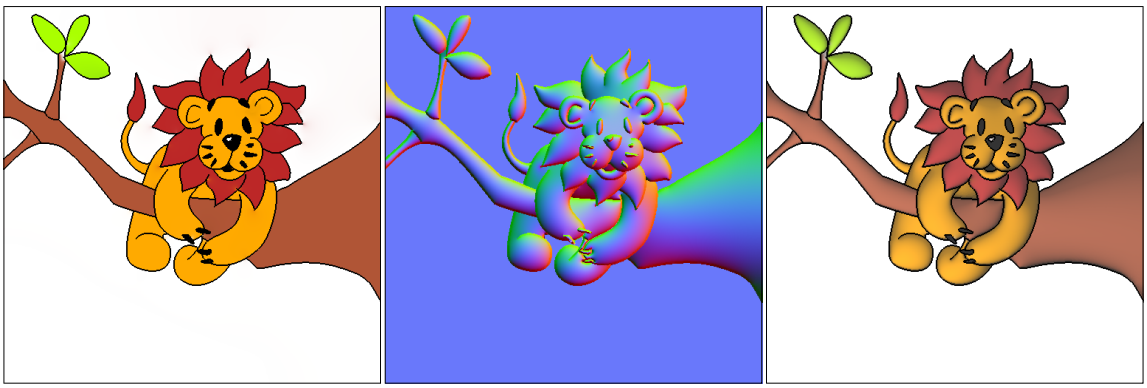


Figure 3.29 *Our approach can diffuse not only colors (left), but also normals (middle) for relighting purposes (right).*

3.10 Discussion

In this chapter, we presented several new methods to increase the flexibility of diffusion-based tools for artists. We illustrated several scenarios in which our solution can be of strong benefit for the user. Our work also enabled new designs that were previously not easily realizable with existing solutions.

Being compact, vector graphics representations are of great benefit for users. A large set of manipulation techniques are made easily due to its intrinsic concise mathematical formulation and controllability via a set of parameters. When adequate user interfaces are provided, artists are able to interact with these parameters in a total transparent way, making the creation process intuitive. Nevertheless, compact representations have also drawbacks. For being compact, it tries to represent areas by sparse information defined via mathematical formulation. Colored regions or curves are not as precisely defined as for raster representations, but implicitly. This compressed codification might be interpreted as a limitation since it might not translate exactly into the artist's wish. The more the manipulations are fine grained, the more these features will constitute a challenge for this representation since it will require an adequate vocabulary to represent it.

In the case of our technique to control diffusion processes, the definition of a directional field over regions were particularly challenging. Proposing to the user a smearing-like tool was an intuitive way to indicate how colors must be diffused across regions of the image while keeping the representation 'sparse enough' to be encoded into mathematical formulations and, therefore, stored as vectorial representations. This explains our choice of encoding these guidance fields as directional curves. Nevertheless, this approach leaves place for improvement. Brush-like tools, with the possibility of controlling radius and strength might also be a good alternative. The user could then 'sculpt' the vector field by modifying and overlapping vectors with the use of such brushes. This technique would be relatively easy to implement in raster images since each directional information needed at each pixel would be directly stored at that point. In order to keep the representation vector-based,

therefore as general as possible, this directional field would need to be encoded into primitives. The question goes back to 'what primitive from the available vocabulary would be suitable?'. Diffusion Curves represent a great step forward in helping bridge this gap. Although limitations exist, it can be considered as a seminal work to improve and enrich the vector graphics vocabulary in order to render the format more expressive. We believe this is a promising field where researches will face the great challenge of presenting controllable, but yet user-friendly solutions that enable expressiveness while easy to manipulate.

Orzan *et al.* [OBW+08] also proposed an automatic vectorization mechanism based on Diffusion Curves. From a raster input image, curves were extracted and colors information sampled along them. This information was diffused and the image reconstructed. We believe that our directional diffusion technique can improve this approach. We suggest the derivation of a flow field from the color gradients presented in the source image. Such flow field can then be used with curves and colors to help the reconstruction. We believe that this could lead to more accurate results and the need of less color curves. Nevertheless, adding the complexity of directional curves.

Another interesting point to be discussed about this Chapter concerns the set of tools that can be presented to the user. Many formulations presented here are subjects of handling by a set of arguments, or allow for great flexibility. For example, we have seen that the tradeoff between directional or uniform diffusion can be controlled by the user. We proposed here a very simple way to control this tradeoff (linearly), but several other approaches might be considered. We believe that strong connections between researchers and artists are the best way to conceive expressive and user-friendly tools and are of major importance, in one hand propose innovative graphic solutions and, on the other hand, to understand what better suits the digital art and illustration communities needs.

To conclude, in this chapter, we showed that diffusion processes are of great potential to enrich compact representations like vector graphics. The advent of Diffusion Curves showed that it was possible to take advantage of the well accepted concept of drawings based on line to create complex illustrations. We proposed a series of extensions that, not only improves expressiveness of these vector primitives, but also that are flexible and powerful enough to be used in many diffusion processes. For instance, our work could be applied in other domains. Flow fields are very versatile and one recent example is street modeling [CEW+08] where the proposed Diffusion Curves tools can be used. In the design process and provides the possibility to guide the diffusion of various information that could be a useful extension for city simulations. Another example concerns vectorial illustrations from 3D models. It is usually tedious, but an automatic transformation into layered vector graphics is possible [EPD09]. The resulting document is usually refined by an artist. If diffusion curves are involved, care must be taken to ensure color consistency across cuts that were introduced to allow a layer decomposition. Our diffusion constraints can ensure smoothness in the final illustration across layers. In the same way, we could also connect different images at the same time in order to produce not a single, but various matching texture tiles, that can then be employed as Wang Tiles [CSHD03] for modeling and rendering repeating patterns with variations.

4 Shape and Tone Depiction from 2D Drawings

Lines have been used in many ways throughout the history of art. Drawing lines on planar surfaces like walls or paper sheets has been one of the most common forms of expression. Drawings are manifestations of semiotics, that is, one way through which the function of assignment of meaning is expressed and constructed [Pia92]. Even in early stages of human development, children represent their surrounding three-dimensional world on flat sheets of paper, or are able to recognize 3D objects directly from their corresponding 2D illustration depicted in books or tv cartoons. Although these images are necessarily incomplete and abstract, they contain sufficient information to make assumptions about the shape. Interestingly, a silhouette alone provides many cues and allow us to identify objects, such as the horse in Figure 4.1 (left) by the English artist and book illustrator Walter Crane. The only information lies in this line bounding the object's shape. Despite the drawing's simplicity, we are able to create a mental volumetric representation from this very economic representation. Crane considered lines as a true language, a sensitive and vigorous speech of many dialects. In his book *Line & Form* [Cra00], he completed:

> *Outline*, one might say, is the Alpha and Omega of Art. It is the earliest mode of expression among primitive peoples, as it is with the individual child, and it has been cultivated for its power of characterization and expression, and as an ultimate test of draughtsmanship, by the most accomplished artists of all time.

Figure 4.1 *Left: Horse outlines illustrated by Walter Crane. Even such 'economic' line drawing allows us to recognize the depicted object. Right: Drawing by John Flaxman. Lines can be used to indicate many material properties, here, the softness of the tissues.*

Once familiar with this representation, most of us are capable to, if not draw, interpret such sets of planar lines. This phenomenon is difficult to explain because it is situated at

the interface between the physical and mental world. Biology explains the process by the stimulation of light on the rods and cones at the bottom of the eye, but cannot tell how this process leads to imaging. All we know is that, when we look at the set of lines illustrated in Figure 4.1 (right) by the draughtsman and sculptor John Flaxman, we immediately identify the two human figures and their inanimate canes. We are further capable of estimating the softness of the tissues composing their clothes. This indicates that drawing is not only a visual process, but instead, we rely on experience - of all our senses - to interpret the illustration [Nic90].

To derive a rich representation, scientists have found many different points of view to explore. Rather than a complete state of the art, the purpose of the following sections is to analyze which line and scene features are of importance and can be linked to computational algorithms.

If artists are capable of extracting lines from a mental representation of a scene, is a computer able to extract similar lines from three-dimensional objects? If so, which surface features can successfully generate lines? These are the questions discussed in Section 4.1, where we give a brief overview about computer-line-drawing approaches. Then, looking at line drawings from a different perspective, we see that these minimal skeletons of visual forms are capable of reproducing the illusion of an actual *seeing* of the object. What two-dimensional features trigger the three-dimensional perception? These are concerns raised by many researches in the computer graphics and vision communities. Section 4.2 reviews an illustrative number of computational mechanisms for the extraction and validation of how three-dimensional structures are perceived when looking at lines.

When an object is placed in an illuminated environment, its surface material interacts with the lighting conditions revealing its shape, position, and characteristics (color, light response, roughness). Section 4.3 shows that, inspired by artistic techniques, researchers have exploited the depiction of shading cues to propose approaches to render enhanced views of objects. Nevertheless, when dealing with two-dimensional pictures, surface information is not directly at hand. The section reviews approaches to retrieve surface information from shaded images, and show how such inferred information can be used to re-render the scene with different perspectives.

Section 4.4 discusses the interesting case of enriching two-dimensional line drawings with 3D information. Many traditional computer techniques for cartoon creation rely on user interventions to annotate drawings with shape information. Although approximate, this shape information can be successfully used to add shading, texture, and many other features to bring vivid colors into drawings. We will look into details of *Lumo: illumination for cel animation* by Scott F. Johnston [Joh02], a seminal image-based algorithm for creating normal map images from cartoon line drawings. This work has inspired many others, in particular, Winnemöller *et al.* [WOBT09] to add texture information on top. These two techniques are the main block upon which we sketch a series of improvements for the extraction of approximative surface normals from two-dimensional line drawings in Section 4.5. For discussion purposes, the reader will realize that we revisit many techniques mentioned along the chapter, trying to take advantages of such different perspectives. It is an exploration into the intriguing world of lines and the many interpretations made by humans, and computers. The ideas and preliminary here-presented results have not been

published.

4.1 Computer-Assisted Drawings

Many books teach drawing techniques [Nic90, Edw87]. They often discuss the use of some types of lines, like contours, as well as their meaning and physical interpretation. Nevertheless, it seems impossible to extract an exact algorithm from these sources because, according to many authors, drawing is a complex process where observation outstrips artifices or techniques [Nic90].

Although nobody seems to be able to characterize and encode all subtle and complex visual perceptual cues and composition, it is of interest to the scientific community to study and develop approaches to better understand it. Several computer algorithms have have been proposed in order to automatize the generation of drawings from many types of sources and in a broad range of styles. When the field of Non-photorealistic rendering (NPR) got a strong attention in Computer Graphics, seminal works were proposed to enhance the depiction of scenes by manipulating the appearance of geometric properties. The main goal of this new born field was to overcome the limitations imposed by photorealistic approaches and to produce images that effectively communicate a message and/or were visually close to those produced by hand [SS02, INC+06]. Since then, the creation of images and 3D scenes that mimic hand-made styles became a solid venue in computer graphics. Several approaches for different techniques been proposed as, for instance, pen-and-ink techniques [WWSS96, WS94, SFWS03, INC+06], watercolor painting [CAS+97, BKTS00, BNTS07], or algorithms for generating realistic drawing strokes that can take on the appearance of pastels, charcoals, or crayons [GTDS10, GTDS04, MTG06]. Many challenges remain open and new research continues to evolve from these works.

Automate Line Drawings

An important topic in the Computer Graphics community is to aim at the understanding and extraction of lines from 3D surface models. Many of these methods tried to identify which features presented on 3D meshes are likely to appear in hand-made drawings. In the following paragraphs, we will present the main mathematical models to extract such features and to render their line representations. We do not intend to be exhaustive, but rather give the reader a quick understanding that there are many different ways to generate lines from a variety of surface features.

Contours are probably the most simple and straightforward line to draw. When we move our viewpoint along these lines, it is comparable to 'touching' an object and we can almost feel the delimiting structure of the shape. Mathematically, these contours, known as silhouettes, are 3D curves characterized by points on the object where the view vector $v(p)$ is orthogonal to the normal $n(p)$ of the surface. Given a point p of a smooth closed surface S, p lies in a contour if $n(p) \cdot v(p) = 0$. The view vector $v(p)$ can be assumed to be $(0, 0, -1)$ for an orthographic camera. Extracting the corresponding contour lines from 3D objects is, at

least nowadays, a relatively efficient task. Olson and Zhang [OZ06], for instance, proposed an efficient silhouette extractor for triangle meshes based on 3D Hough transforms. Such transform allows the organization of the mesh data more effectively for silhouette computations than the traditional dual transform. Dual transforms had been previously used in the work of Hertzmann and Zorin [HZ00] to segment contour curves into smooth parts with constant visibility, as well as an automatic method for generating hatch marks in order to convey surface shape. Figure 4.2 shows a drawing generated with this approach. We realize that, although crucial to understanding, contour lines give a first coarse idea of the shape (left), but they do not convey information about the interior shape of the surface. We, nevertheless, see more details on the corresponding 3D shape when, for instance, shading is added to the model (right).

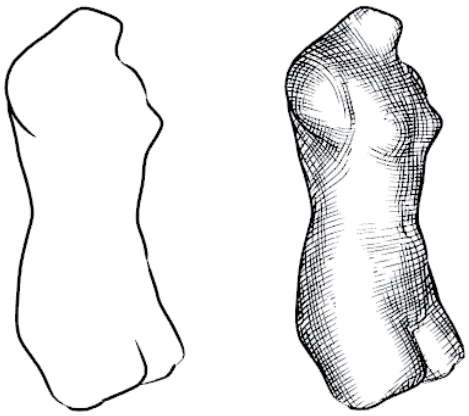


Figure 4.2 *Direction fields on the Venus. Left: Silhouettes alone do not convey the interior shape of the surface. Right: Shape perception is enhanced when shading is added to the drawing.*

To fill the void inside the contour curves, different types of lines can be used and can easily be found in many artistic drawings. This is the case of ridges which are long narrow upper sections, like the horizontal line formed by the juncture of two slopping planes (for example, the line formed by the surfaces at the top of roof). Valleys are analogous to the ridges. As for contours, algorithmic solutions were proposed to extract such geometric lines efficiently [IFP95, OBS04, YBS05].

Other lines, such as suggestive contours [DFRS03, DFR04] and apparent ridges [JDA07], have been introduced by the research community. Even though not present in the artistic vocabulary, studies have shown that these lines are indeed drawn on hand-made artworks [CGL+08].

It is sometimes argued that paintings have the appearance to be more dynamic than photographs, maybe because the artist has the freedom to look at the model from different, but yet close, points of views which are later combined into a single image. This is also the philosophy of suggestive contours in relation to the traditional occluding contours. By ex-

panding the point of view, suggestive contours (by DeCarlo *et al.* [DFRS03]) reveal those locations at which the surface *is almost in contour* from the original viewpoint - locations at which the dot product $n(p) \cdot v(p)$ is a positive local minimum rather than a zero. Geometrically, they are view-dependent curves along which the radial curvature is zero and where the surface bends away from the viewer. When combined with traditional contours, suggestive contours can increase the perception of the underlying surface as depicted on Figure 4.3.

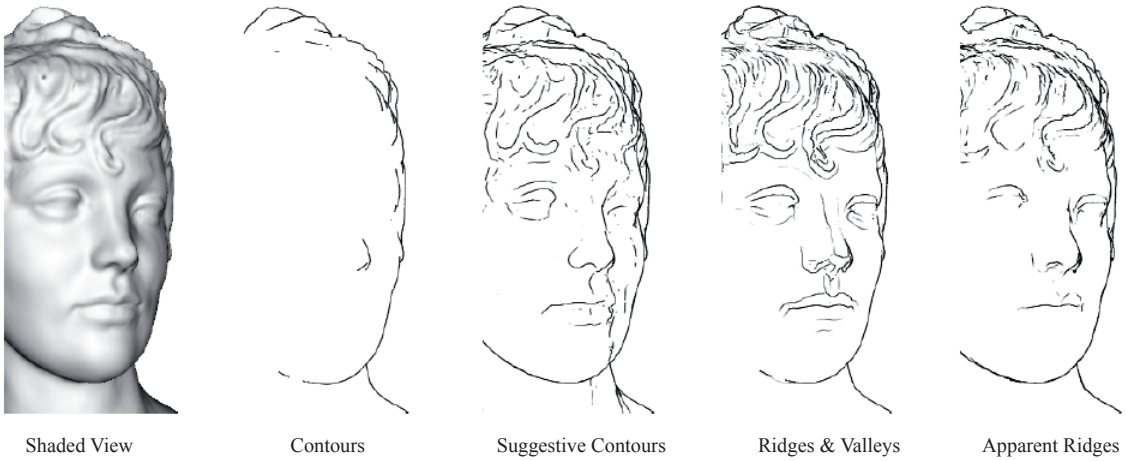


Figure 4.3 *The Bust model rendered with several different feature lines. From left to right: the 3D model, contours, suggestive contours, ridges and valleys, apparent ridges.*

Another example of recently proposed lines are the apparent ridges by Tilk Judd *et al.* [JDA07]. Apparent ridges are view-dependent lines that appear when the surface normal is changing at a locally maximal rate with respect to image position, like a derivative on the buffer containing the normal map. These lines are able to capture shape features that contours, ridges, valleys, or even suggestive contours are not able to, such as the nose depicted in Figure 4.3.

Although each of the aforementioned lines seem to get very close to traditional hand-made work, there is no such thing as a perfect type, a single line type that best reveals all the shape features encoded in human drawings. Several other line models have been proposed throughout the last years and each of them communicates shape aspects that were omitted by others. The more pronouncing demarcating curves [KST08], for instance, are lines that appear commonly on archeology drawings and typically separate ridges and valleys on 3D objects (hence the name demarcating). Many other mathematical models exist and can appear alone or in combination with other lines: view-dependent highlight lines [DR07], lines from shading features [LMLH07], or yet crease lines [KMM+02] drawn directly on 3D models, to cite a few.

Even though it seems well accepted that these mathematical lines are likely to appear in hand-made drawings, a comparative study comparing these propositions and human-made

drawings was needed. Assembling many of these models together, Cole *et al.* [CGL+08] have recently proposed a study to verify how well these algorithms can describe the human artists' lines. This study considered lines generated with an image-based approached, such as the Canny edge detector [Can86], and the aforementioned geometric ridges and valleys, suggestive contours, and apparent ridges. The result of this study reveals interesting facts. For example, all human drawings analyzed exhibited examples of multiple classes of lines. This reinforces the idea that no line model can be considered self-sufficient to represent every aspect of the drawing. Also, the study indicates that the large majority of hand-made lines are overlapped by these computer generated models. These are very good news which suggest that the research community is on the right track toward understanding the line drawing mechanism. Nonetheless, it is important to continue the investigation of line types in art, to propose efficient solutions to validate the mathematical models, as well as to look at the problem from different perspectives, such as, how lines influence the perception of shape.

4.2 Lines and the Perception of Shape

Accurate drawings resembling photography convey information in a form which can be understood by an observer because it conforms with what she would have seen. Nevertheless, even if a drawing is less accurate, or if it makes transgressions of standard physics, these limitations do not seem to interfere with the viewer's understanding of the scene. This can be observed in Picasso's skewed faces, in the colored shadows of the works by Matisse, or in the inaccurate lines used in children drawings. Cavanagh [Cav05] has shown that physical transgressions like impossible shadows, color, reflections or contours often pass unnoticed by the viewer and these facts suggest that the brain uses a simpler, reduced physical model to understand the world. This simplification relates to our ability of making abstractions. The author suggests that the real world is subject to an extensive set of physical constraints, but few of these seem to be checked by our vision. It is this smaller set of constraints that allows us to recognize objects and scenes even if they do not exist in our real world. For instance, we are capable of recognizing hands with only two or three fingers, or yet a 'hand' of a mouse or a woodpecker. We are capable of making conceptual interpretations, rather than a merely physical analysis of the world.

More than understanding incorrect physics or non-existent objects, the human brain also has the impressive power of constructing meaningful images from fragmented information. Line drawings demonstrate that minimal skeletons of visual forms are capable of reproducing the illusion of more. They make the eye recognize the content as being consistent with what is seen when looking at an actual view or object [Gil91]. This happens because we are able to decode lines, colors, and other means of representations into shapes and significations.

4.2.1 Shape Reconstruction from Labeling

Research in computer vision offers a number of computational mechanisms for extracting three-dimensional structures from two-dimensional line drawings [Sug86]. The main question studied by these approaches is to find what shape is represented by the primitives. Although it seems to be clear that many different shapes could project into the same line drawing when viewed from the correct viewpoint, many authors propose to characterize, in some useful and complete way, the constraints imposed upon this infinite set of scenes. In order to be able to characterize the scene, it is important to make use of a suitable categorization of the features encoding information: lines and regions. This process is called *labeling* and has been used by many authors as a language with which the sparse information is encoded. Huffman [Huf71] and Clowes [Clo71] set forth the first labeling scheme valid for polyhedra. The Huffman-Clowes labeling method classifies line segments into three categories: convex edges, concave edges, and occluding edges. The goal of this process is to establish a connection between the two-dimensional representations and their shape in three dimensions. Lines with different labels impose different types of constraints on the three-dimensional interpretation. Some researchers have attempted to determine a unique three-dimensional shape which supposedly is the one perceived by a human observer based on line constraints. Typically, these approaches use some criterion to choose one among the possible spatial interpretations. For example, Brady and Yuille [BY83] search for the most compact shape possible.

Kanade [Kan81], Sugihara [Sug86], and several others have extended the Huffman-Clowes labeling scheme based on junction libraries. Waltz [Wal75] extended this approach to categorize regions under lighting conditions. A different extension was proposed by Malik [Mal86] who augmented the labeling vocabulary for drawings of piecewise smooth surfaces. For a better understanding of how such labeling process can help with the interpretation of the perceived surface, we illustrate a few label constraints defined by this technique.

Figure 4.4 shows a labeling example for Malik's approach. A "+" label represents a convex edge: an orientation discontinuity such that the two surfaces meeting along the edge in the scene enclose a filled volume corresponding to a dihedral angle less than π. A "-"label represents a concave edge: an orientation discontinuity such that the two surfaces meeting along the edge in the scene enclose a filled volume corresponding to a dihedral angle greater than π. A "$\leftarrow$" or a "$\rightarrow$" represent an occluding convex edge. When viewed from the camera, both surface patches which meet along the edge project on the same side of the line, one in front of the other. As one moves in the direction of the arrow, these surfaces are to the right as seen from the camera. A "$\leftarrow\leftarrow$" or "$\rightarrow\rightarrow$" represents a contour. As one moves in the direction of the twin arrows, again, the surface lies to the right.

As it has been said before, such labels impose constraints on the shape they represent. For instance, the author suggests contour labels to impose constraints as follows. Considering an orthographic projection along the z-axis, let n be the unit surface normal, and I be the unit tangent vector at a point of the contour, hence, $n \cdot I = 0$. As a contour corresponds to points on the surface where the line of sight vector $\hat{z}$ lies in the tangent plane of the

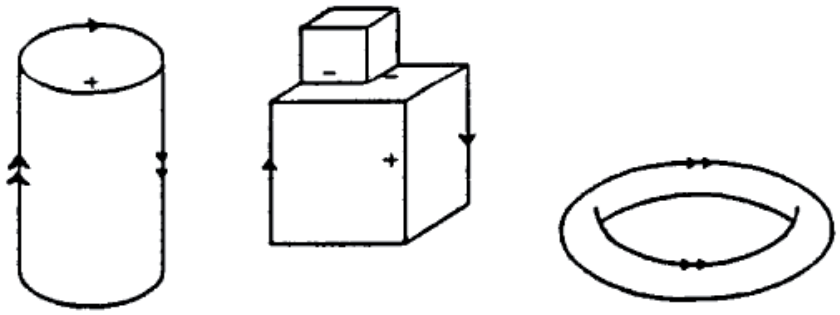


Figure 4.4 *Labeling scheme by [Mal86]. A "+" label represents a convex edged. A "-" label represents a concave edge. A "←" or a "→" represents an occluding convex edge. A "←←" or "→→" represents a contour.*

surface, it follows that $n \cdot \hat{z} = 0$ for points on the surface (equivalently $n_z = 0$). Therefore, in general, n lies in the image plane and can be constructed by drawing the outward-pointing unit vector perpendicular to the projection of the contour in the image plane.

In addition to lines, shape constraints are also defined inside areas. In the case of Malik's approach, each area in the image is assumed to be the projection of a connected smooth-surface part. Therefore, the functions that map each image point to its position and orientation must be smooth within a single area. Also, the surface normals at all the visible surface patches must have positive n components. It follows that, as the surface normal in a smooth patch can be written in terms of the partial derivatives of I with respect to x and y. Then, if a C^2 function $z(x, y)$ is specified, the orientation function $n(x, y)$ is automatically determined and smooth. Therefore, considering a dense labeling of a line drawing, a candidate solution to this set of constraints is obtained by specifying a piecewise smooth function $z(x, y)$ corresponding to the depth at each visible point in the scene.

Probably this labeling mechanism is not sufficient to describe the delicate process that is triggered by human perception, but such information is useful to constrain the computer solution and make it approach the perceived shape. Nonetheless, just basing the derivation on lines we consider useful, might prevent the computer from generating an appropriate result as well. To achieve the perceived shape one might actually need to modify the original input. From our point of view, this input might miss certain important lines, might exhibit superfluous lines, or even wrongly positioned ones. Ultimately, perception is more complex than what can be captured via a few labels.

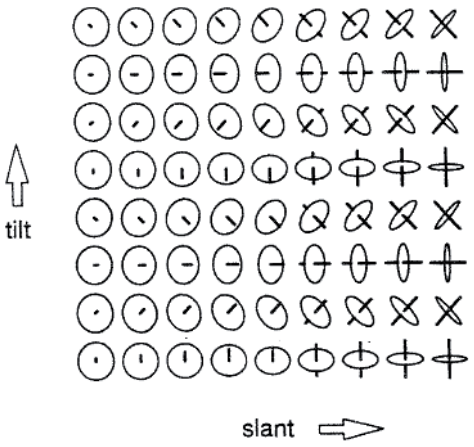


Figure 4.5 *A collection of gauge figures used in [KvK92]. Rows depict constant tilt, columns depict constant slant.*

4.2.2 Perceptual Experiments

Although drawings trigger the impression of an actual seeing of the object, such projection is vacuous. Koenderink *et al.* [KvK92] have proposed strategies for measuring the internal representation of three-dimensional surfaces generated via the inspection of two-dimensional portrayals, usually from photographic sources. More recently, Cole *et al.* [CSD+09] proposed a study to investigate the ability of sparse line drawings to depict 3D shape. Cole's approach is based on the gauge figure technique to reveal the perceived surface orientation at any arbitrary location in an image. In this context, a gauge figure is circular disk, pierced orthogonally through the center with a straight line (Figure 4.5). To indicate the perceived shape, one can adjust the gauge figure in such a way that the disk looks tangent to the surface of the depicted object by manipulating the slant and tilt of the projection of the gauge figure with the mouse, for example. Gauge figures had already been used previously [KvK92, KvKT01], nevertheless Cole *et al.* were the pioneers to use it in the context of line drawings. To obtain local estimates of the surface orientation at a large number of points spread over a picture.

In the aforementioned investigation, Cole *et al.* considered not only hand-made line drawings, but also computer-generated illustrations. As in previous research [CGL+08], this choice of including 'artificial' drawings was motivated by the need of investigating how well lines generated by compute algorithms are effective in conveying shape. Therefore, the input data considered human-made line drawings, as well as drawings created using algorithmic line models as suggestive contours, ridges and valleys, and apparent ridges. In order to confront the effectiveness of line drawings against shaded images, these renditions were also included in the study. Subjects were then asked to place gauge figures on the surface of these drawings to indicated the surface perceived. Figure 4.6 depict an example of such task. The result of this study suggested interesting facts concerning the

human perception of line drawings, as well as the effectiveness of computer generated line models. In general, the study suggested that interpretations across viewers were consistent regardless of the type of drawing. Also, different people interpret line drawings roughly as similarly as they interpret shaded images (with a slight advantage for the latter), but that, at times, even shading failed to communicate shape effectively. Concerning the computer generated lines, they demonstrated that not all line drawings are equally effective at depicting shape. For example, suggestive contours behaved more or less adequately depending on the characteristics of the model. Although the computer generated lines were found to be placed at similar locations of hand-made lines (as mentioned before), the study reveals that the computer is not ready to replace human skills: artist are superior in selecting the most appropriate lines in each case. This finding opens the space to future approaches to explore algorithms for selecting the best set of lines to depict a shape.

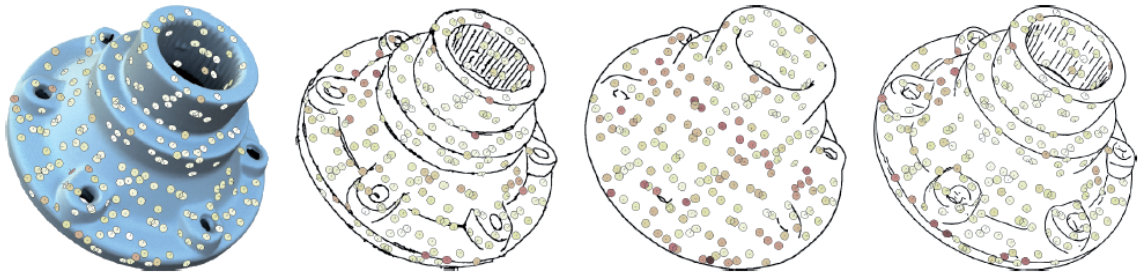


Figure 4.6 *Gauge figures placed on the surface of drawings to indicate the perceived shape in [CGL+08].*

4.3 Shading and the Perception of Shape

When an object is placed in an illuminated environment, its surface material interacts with the lighting conditions revealing its shape, position, and characteristics (color, light response, roughness). This is a complex effect present in the real world, which behavior is sought by many researches to be efficiently modeled in the computer. Although some level of flexibility is allowed by these algorithms, non-photorealistic techniques allow artists to arrange surfaces and lights to suit the message of the piece rather than the requirements of the physical world. The goal is not only to accurately imitate an existing scene, but also to communicate other characteristics of objects in a visually comprehensible manner [VPB+09]. For instance, in Archeology, fine surface characteristics are depicted in sharp relief as depicted in Figure 4.7.

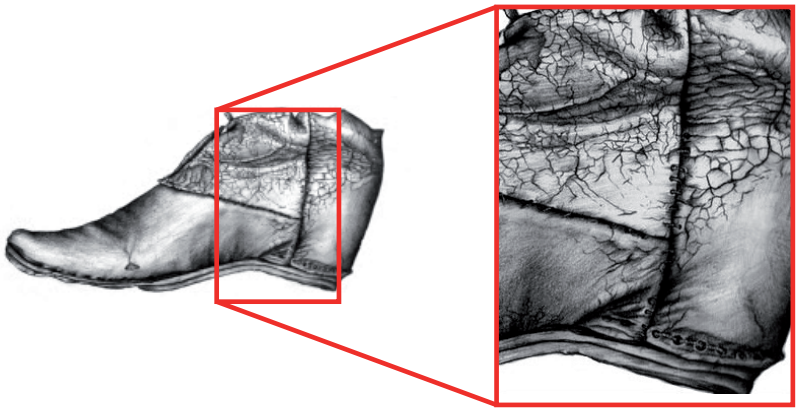


Figure 4.7 *Archaeological illustration, medieval shoe (leather), detail. Non-photorealistic techniques allow artists to arrange surfaces and lights to suit the message of the piece rather than the requirements of the physical world. Helena Michel©. Pencil on paper.*

4.3.1 Manipulating 3D Scene Depiction

Inspired by artistic techniques, researchers have exploited the depiction of shading cues to propose approaches to render enhanced views of objects. Gooch *et al.* [GGSC98] presented a lighting model for three-dimensional scenes that provides a better shape comprehension by mapping the change in surface orientation into variations of hue instead of brightness variations. Cignoni *et al.* [CST05] proposed a view-independent algorithm to enhance the depiction of high-frequency components of a model for an enhanced depiction of its surface details. The technique is based on a normal modification, i.e., a manipulation on the surface orientation of the model while leaving the silhouette, therefore its geometry, unchanged. Later on, the work of Rusinkiewicz *et al.* [RBD06] extended this idea to reveal relief surface details. Differently of Cignoni's, this approach does not perform changes on the surface orientation, but rather simulates the effect of illuminating parts of an object with different light positions that are, ultimately, combined into a single rendition. Although these methods can effectively enhance detail perception, their non-photorealistic look impairs the depiction of complex materials and illumination. To bridge this gap, Barla *et al.* [VPB+09] proposed a new light warping approach that stems from rendering techniques, but leads to an enhancement of important surface features.

4.3.2 Manipulating 2D Scene Depiction

Obtaining a surface normal from two-dimensional images has found interesting applications for content manipulation. A number of authors generated normal maps for a variety of applications. For example, once a normal map is created with the help of pen-based interface, Okabe *et al.* [OZM+06] proposes to change the illumination of objects in a real photographs (see Figure 4.8 top). In the work of Hui Fang and John C. Hart [FH04], a shape from shading technique was used to recover surface normals from a input photograph. The resulting normal information was then used to texture objects in photographed surface (see Figure 4.8 middle). Rusinkiewicz *et al.* [TFFR07] have demonstrated that normal enhanced images are a good alternative data structure lying between simple 2D images and full 3D models. Named *RGBN* images, this structure is the result of having the standard color image components (*RGB*) augmented with normal information (*N*). This information can be captured using a photometric stereo approach relying on several images (captured from a single camera position) of an object illuminated from different directions. The resulting data structure was used in the creation of complex non-photorealistic illustrations that emphasize certain features of the image (for instance, revealing surface relief as depicted on Figure 4.8, bottom). Similarly, a photogrammetric modeling method was proposed by Debevec *et al.* [DTM96] for the recovery of the basic geometry from a set of still photographs. The geometric information was then used to model and render the photograph's equivalent architecture scene. Jones *et al.* [JGB+06] also presented an image-based technique for relighting dynamic human performances under spatially varying illumination. Their approach is based on flashing light rapidly from different directions and therefore compute normals on the fly.

It is important to note that human vision is surprisingly tolerant of certain physical inaccuracies and therefore accurate surface information is not required in some cases. For instance, Kahn *et al.* [KRFB06] exploited the flexibility of human vision to reliably estimate illumination in their image-based material transformation technique. Using a similar approach, Lopez-Moreno *et al.* [LMJH+10] proposed a method to stylize photographs based on the extraction of approximate depth information from a single input image. Based on the the hypothesis that exact surface information is not a strong requirement in shading, in the next section we will investigate how approximate shape input can be used in order to enrich a more sparse representation: two-dimensional line drawings.

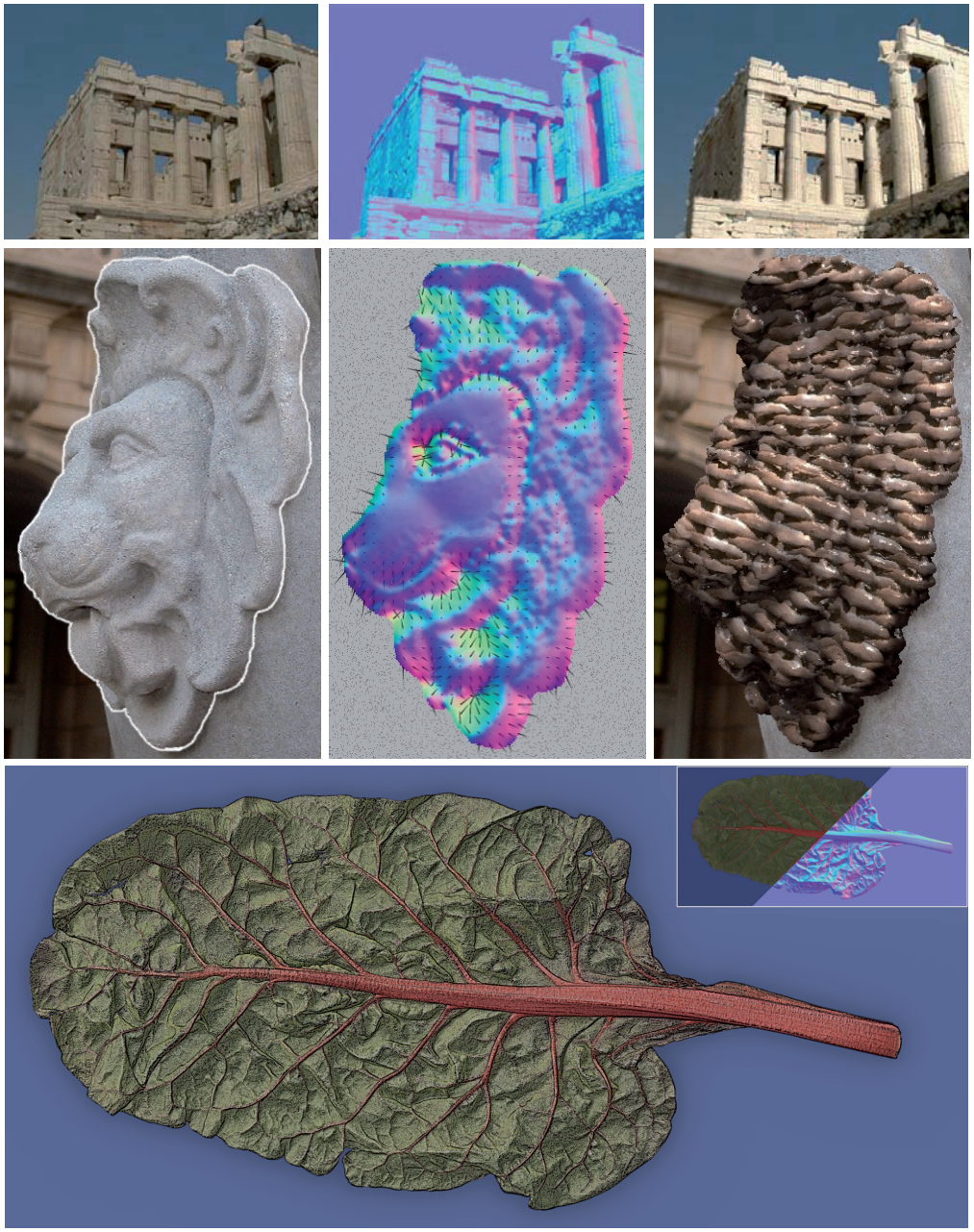


Figure 4.8 *Manipulating 2D Scene Depiction. Top: Relighting photographs in [OZM+06] using normal map created with the help of pen-based interface. Middle: Normal map created with the help of shape from shading techniques enabled textured objects in photographed surfaces in [FH04]. Bottom: Surface detail is revealed thanks to RGBN images created from a photometric approach in [TFFR07].*

Shape from Shading Techniques

Shape from Shading (SFS) is a classical problem in computer vision that aims to derive 3D scene description given one or more 2D shaded images. A very didactic survey about the many existing techniques can be found in the work by Ruo Zhang *et al.* [ZTCS99]. The shape recovered from these techniques can be expressed in several ways: depth $z(x, y)$, surface normal (n_x, n_y, n_z), surface gradient (p, q), and surface slant, ϕ, and tilt, θ. Considering a simple model of image formation, like the Lambertian model, the gray level of a point on a surface depends on the angle between the light source and the surface normal directions. Even with such a simplified material model like the Lambertian, and a given light source directions, this problem is still not simple. This is because each surface point has at least two unknowns (for instance, (p, q) for the surface gradient) and each pixel in the image provides one single gray value. We have thus an underdetermined system. Finding a unique solution is difficult; it requires additional constraints. For instance, the approach by Leclerc and Bobicks [LB91] obtains the solution based on two additional constraints: the brightness constraint and the smoothness constraint. The brightness constraint requires that the reconstructed shape produce the same brightness as the input image at each surface point, while the smoothness constraint ensures a smooth surface reconstruction. The system is solved directly for depth by using a discrete formulation and employing a conjugate gradient technique. To ensure convergence, a stereo depth map was used as an initial estimate. Other kinds of approaches derive shape based on the assumption of surface type. This is the case of Pentland's local approach [Pen84] to recover shape information from the intensity, and its first and second derivatives. He used the assumption that the surface is locally spherical at each point. Recent Shape From Shading approaches are able to reconstruct scenes which are considered acceptable by most people, nevertheless, studies indicate that the human visual system relies on different mechanisms to solve the shape from shading than computer vision normally employs [ZTCS99]

.

4.4 Shading from Line Drawings

Although objects can be well represented on a flat surface by means of accurately drawn outlines, in some cases, it is impossible to express the surface character without the addition of light and shade [Spa00, CGL+08]. For instance, a circular plane and a sphere may each result in a circular outline for certain views. Under such conditions, the difference between the bodies can only be expressed by shading. This situation is indeed extreme, nevertheless, mere outline can never adequately render the delicate variations of the surface of flowers and fruits, for example.

Indeed, we have seen in the previous section how much information is encoded in shading. In fact, it could help differentiating a sphere from a circle. Nevertheless, when dealing with line drawings, surface recovery might be an even more difficult problem. This is due to

a series of facts. First, line drawings are an economic means to represent scenes. One characteristic is that the information in the drawing is very often sparse, therefore insufficient for shape reconstruction. Second, artist's drawings are really two-dimensional projections of 3D characters as visualized in the animator's mind [DFVR02]. It means that the artist can take shortcuts by representing cues more economically, and arranging surfaces and lights to suit the message of the piece rather than the requirements of the physical world. Consequently, it is not clear that an actual three-dimensional scenes could be reconstructed from an artistic depiction, at least not following physical models.

Although a difficult problem, the computer graphics community has found ways to overcome these limitations. For instance, if the information on the drawing is too sparse to allow for surface recovery, one alternative is to rely on the user to 'complete' or suggest the missing information. The challenge is to identify *how* this user intervention will take place.

One approach in this direction is the work of Wu *et al.* [WTBS07]. A 3D shape is defined by interpolating sparse normal information provided by the user via so-called *shape palettes*. The process is illustrated on Figure 4.9 and described as follows. After drawing a simple 2D primitive (curves, dots, etc), the user specifies its 3D orientation by drawing a corresponding primitive over the surface of some familiar shape, as, for instance, a sphere (b). The normal information along the corresponding primitive is then adapted and transferred to the drawing at the primitive location. The familiar shape is therefore used as a *palette* of *shape* information. A dense normal field is then created from the sparse normal information by minimizing an energy function. This surface information can now be used for many purposes as for 3D shape recovery or rendering (c). The drawback of this technique is the need of a large and variate set of shape palettes in order to allow expressiveness. Different approaches rely on hand-held devices as pen-tablets to allow a more intuitive and flexible user intervention. This is the case of the previously mentioned technique of Okabe *et al.* [OZM+06] to relight photographs. Here, using pen-tablet technologies, the user assigns sparse surface normals to a photograph by tilting the pen. The use of such technology was inspired by the free surface modeling approach by Igarashi *et al.* [IMT06].

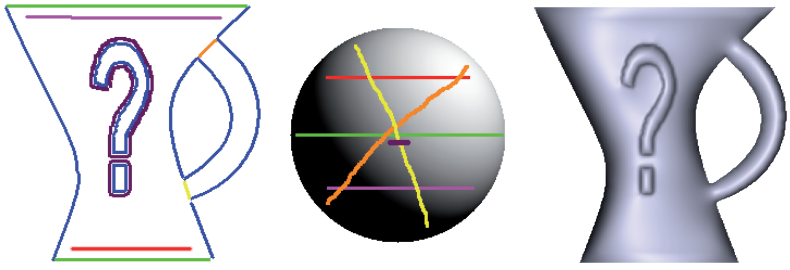


Figure 4.9 *Shape Palettes by [WTBS07]. A surface is created by linking 2D primitives drawn in the freehand view to their corresponding 3D shape on the 'shape palette' (the sphere). Specifying this relationship will generate a cylinder-like structure.*

4.4.1 Enriching Cartoon Drawings

The possibility to extrapolate normals from two-dimensional curves is especially useful in the field of hand-drawn animation. A big problem when creating the cel images is to illuminate the 2D character as it were a 3D object living in a 3D illuminated environment: the shading problem. S. F. Johnston [Joh02] discusses the shading problem very clearly, indicating that the components involved in the illumination of a point on a surface (position and surface normal) are incomplete in hand-drawn artwork - the surface normal is unknown and the position information lacks depth. Current research on computer assistance for traditional animation focuses on two main lines: geometry-based methods and image-based methods. In the first, geometric objects are created in order to support the animation process. The latter uses image-based techniques to render the 2D character without involving actual 3D geometry.

The shading problem in computer-assisted 2D animation is a recent concern. To overcome the lack of proper shape information, many authors, like Di Fiore *et al.* [DFVR02], discussed the incorporation of approximate 3D models into the traditional computer animation process. According to the authors, approximate models can be of great help to provide features, such as frame-to-frame coherence, rapid inking and shading, or the creation of different extreme poses while retaining the proportions of the object (avoiding redrawing the object). In their technique, 3D polygonal objects are created by revolving sketches of 2D circular and rounded forms (as for free surface modeling techniques). Other approaches, nevertheless, exist. For instance, Correa *et al.* [CJTF98] presented an approach to distort crude 3D models created by artists in such a way that the it conforms to the hand-drawn art with the help of two techniques: a silhouette detection scheme and a depth preserving warp. Once the textured 3D model is efficiently warped into the 2D shape, it can then replace the flat colors that would be used in the ink-and-paint stage of traditional cel animation (Figure 4.10 left).

Figure 4.10 *Three-dimensional information improving traditional computer animation process. Left: Vivid textures applied to cel animation by [CJTF98]. Right: Semi-automatic shadows for cel animation by [PFWF00]*

Making use of rich representations as 3D models has also inspired researches for the creation of shadows in cel animation. In the work of Petrović *et al.* [PFWF00], line art created by artists are converted into *character mattes*: bitmaps that define regions of the

image covered by the character. Simply put, these layers are then inflated to become three-dimensional by borrowing techniques, again, from free surface modeling systems (Igarashi *et al.* [IMT06]), while preserving their image-space silhouettes. With such approximative 3D figures, user-defined light positions and background objects at hand, the system then finally computes self, contact, and cast-shadowing (Figure 4.10 right). The inflate metaphor common in shape sketch interfaces has also been used by Joshi and Carr [JC08] to enhance 2D drawings and photographs with 3D geometry. An initial surface mesh is generated by triangulating the area bounded by user-defined curves. A 'thin-plate spline' [Wah90] is then generated to inflate the initial mesh.

The great advantage of the geometry-based methods is the possibility of supporting both the frame-to-frame coherence and the shading process. However, these methods cannot cope with more fluid animations, where few strokes would change the implied geometry drastically [Joh02]. In addition to this, 2D cartoon drawings can be very 'unrealistic' in the sense that it might not correspond to any three-dimensional surface. Image-based methods work directly with the vivid drawings made by the artist, but it often requires a large number of images.

4.4.2 Image-based Methods

Johnston [Joh02] proposed a seminal image-based approach to approximate a normal field from two-dimensional line drawings. The technique estimates normals wherever possible and uses a sparse interpolation to approximate them over the remaining image. In the following we will review this technique, to later (Section 4.5) propose a discussion about its contributions and possible directions for improvements.

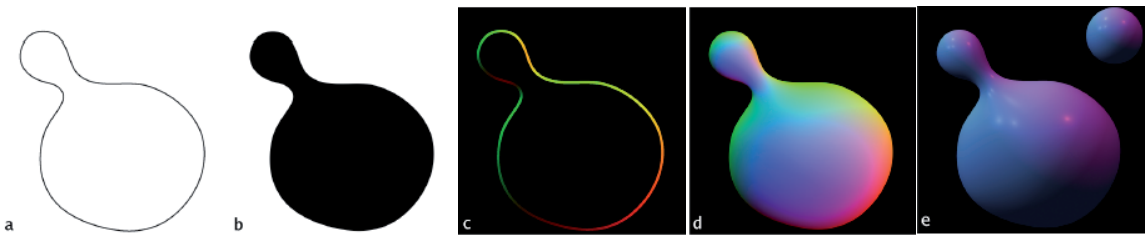


Figure 4.11 *Normal fields from two-dimensional drawings by [Joh02]. (a) Outlines. (b) Image matte. (d) Normals along outilines. (d) Normal interpolation over image matte. (e) Illumination.*

Figure 4.11 illustrates this principle. In cel animation, exterior silhouette and interior folds (a) represent locations where the surface normal is perpendicular to the eye vector (orthogonal to the paper plane). By computing gradient information across an image matte (b), normal vectors can be obtained along these silhouettes and folds (c). These normal are then interpolated across the image matte to generate a normal field (d). The approximate

normal map can finally be used to depict shading (e). Normal vectors computed from silhouettes result in values for N_x and N_y, but N_z is equal to zero along the curve. To give the impression of volume, (N_x, N_y) are linearly interpolate across the field, with N_z recomputed from the result to maintain unit-length.

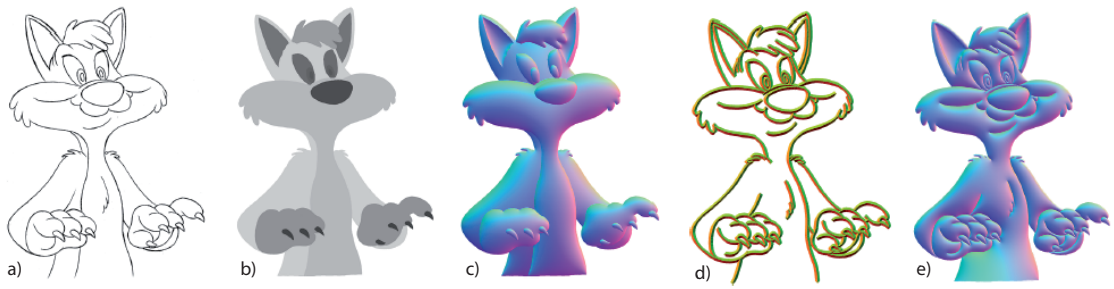


Figure 4.12 *Normal derivation by [Joh02]. (a) Outlines. (c) Region matte. (d) Blobbed normals. (e) Two-sided normals. (f) Quilted normals.*

Figure 4.12 shows that, when this algorithm is applied to a more complex shape, the result seems to have very little shape detail. Using mattes to separate regions (b) and applying the technique on each of them, creates a more accurate representation of the character's normals (c). However, the result still lacks much of the detail contained in the original image (a). When the gradient is computed across ink lines, rather than matte edges, opposing normals are generated on each side of the lines (d). When these normals are interpolated, much more detail is revealed, but the image appears quilted because the internal folds should only affect the normals on one side of the edge, not both (e). The solution proposed by the author is to blend these blobby and quilted results into a single image that conveys a more harmonic shape. We will discuss the approach in the following.

For Johnston, a drawn line can be interpreted as a boundary separating two regions. In many circumstances, one of these regions overlaps the other. Tagging edges with white on the "over" side and black on the "under" side produces an illustration that better defines the layering within the image (Figure 4.13 a). When the over/under edge illustration is interpolated, it forms a grayscale matte (Figure 4.13 b). The quilted normal image is a more accurate representation of the normals on the "over" side of the edges where the matte is white; the normals on the darker "under" side of an edge are less certain. These mattes can be viewed as a measure of confidence in the known normals. The normal on the under side of an edge should interpolate the shape of the underlying object as if that edge did not exist. By blending the quilted and blobby normals using the over/under confidence matte as a key, a more adapted normal image is formed (Figure 4.13 c). Johnston used this normal map to shade cartoon characters in order to insert them into live scenes, as depicted on (d). Given the impression of a volumetric body, the illustrated cat has a more vivid appearance and matches harmonically the live scene.

Inspired by Johnston's technique to create normal surfaces from two-dimensional curves

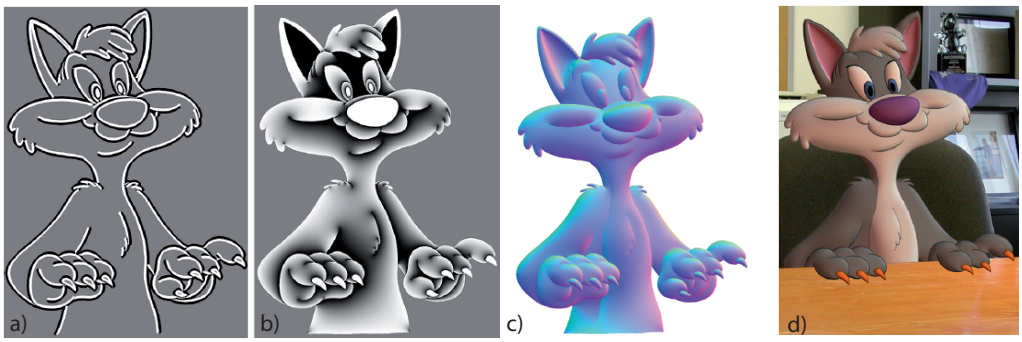


Figure 4.13 *Normal blending technique by [Joh02]. (a) "Under/over" assignment. (b) Confidence Matte. (d) Blended normals. (e) Shading result.*

and their previous work on diffusion, Winnemöller *et al.* [WOBT09] introduced an approach to design and manipulate textures in 2D images. The more interesting characteristic of this work was to show that diffusion processes can be suitable to spread many kinds of information. Previously introduced as a color diffusion, the authors extended this idea to diffuse more complex information, such as texture coordinates and normal vectors. The contribution of this approach over the interpolation process presented by Johnston is that now, due to this vector-based primitive, the entire process of drawing and shading can take all advantages of a vector-based approach. After drawing the character's outlines, the user specifies normal constraints on the curves by the definition of normal control points, in the same spirit as for color diffusion. The normal values defined by these control points are then interpolated and normalized along the curve. The sparse normal information is spread over the remaining image by the same diffusion process proposed for Diffusion Curves [OBW+08]. Thus, each component N_x, N_y, and N_z is interpolated separately. Inspired by the inflation technique of Joshi and Carr [JC08] Winnemöller's approach allows the user to manipulate normal values at the location of curves. Thus, different values of (N_x, N_y, N_z), in particular those with $N_z \neq 0$ can be sent to the solver, allowing the definition of a broad range of curve types. Figure 4.14 shows an example of a vector-based image created with this approach. Drawing with colors (a), to the normal map (b), texture coordinates map (c), and texture pattern (d - top), everything is created using the Diffusion-Curve metaphor. When used in combination, these features allow very complex effects to be created while keeping the result vector-based (d). Diffusion Curves can therefore be a versatile primitive for the creation of vector-based normal fields. In the next section, we propose a discussion over a series of improvements on this technique to allow for the creation of a broader range of surface normal types.

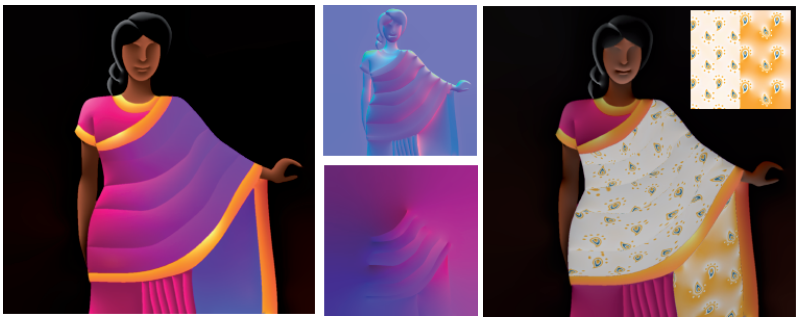


Figure 4.14 *Diffusion Curve metaphore used to design and manipulate textures in 2D images. From the support drawing with colors (a), to the normal map (b), texture coordinates map (c), and texture pattern (d - top), everything is created using the Diffusion Curve metaphor.*

4.5 Ongoing Work on Shading from Curves

Image-based approaches are indeed closer to the two-dimensional pipeline to which artist of traditional drawing and animation are used to. They add to the scene a touch of three-dimensionality without losing its fluid appearance. Nevertheless, in order to provide expressive tools much needs to be improved. In this section, we will discuss a set of ongoing ideas to help improving the expressiveness of creating shading effects from line drawings. Although many of them could not yet be validated, we believe that their discussion will help the conception of a flexible and user-friendly tool for enriching 2D drawings. We will borrow ideas and findings from solutions previously presented in this document and discuss how they could contribute to such a system.

4.5.1 “Under”, “Over”

Johnston’s under/over tag system was proposed in order to create a layering system of regions that form the drawing. This layering system did make sense in the case of his illustrations because we observe that they are composed of one basic line type: contours. Because contours can be understood as boundaries that delineate a shape and separate it from its background, they represent depth discontinuities in the scene. Consequently, this kind of lines indicates a layering system where the region falling on one side of the curve (and delimited by it) occludes the region on the other side. Thus, Johnston’s under/over tags indicates to the system which side of each curve delimits the object, and which belongs to its background.

Later on, when this line tag system is interpolated, it creates a confidence matte: a map that indicates how confident the system can be about the shape at certain locations. Locations tagged now as "over" received the normals from the quilted normal map, therefore revealing more of its shape. On the other hand, the "under" areas received the normals presented in the blobbed normal map because these surfaces are less detailed. This technique is based on an interesting fact about contours. Although these lines tell us something about the shape it delimits on one side of the line, nothing can be said about the surface on the other side. Therefore, locations nearby the curve and falling on the side delimiting the object (tagged as "over") are more likely to be accurately defined by the geometry of the curve. Thus, Johnston's approach attributes the detailed quilted normals to these locations. Analogous, the geometry of a contour curve can say nothing about the shape occluded underneath it. Thus, the shape at those locations are predicted with low confidence. Consequently, less detailed surface as in the blobbed shape is preferable to be defined there.

The work of Winnemöller *et al.* [WOBT09] obtained a similar normal map results without directly using Johnston's blending approach. Based on the metaphor of the original Diffusion Curves, every curve is constrained on both sides and provide the corresponding normal information. To mimic the blending result obtained by Johnston using Diffusion Curves, one might define silhouette normals on one side of the curve, and low detail normal vectors as $(0, 0, 0)$ on its other side (so that low detailed shape information is diffused at that location). Although this technique can mimic the under/over blending system to some extend, we observe that a more efficient approach can be obtained by the use of our previously introduced Diffusion Barriers 3.7.1.

4.5.1.1 Diffusion Barriers

Our more flexible primitive, Diffusion Barrier, allows for an easy representation of many types of curves as, for instance, contours. This curve can diffuse normal information at one side, while simply blocking the diffusion at the other side. This representation is therefore more appropriate to the problem since there is no need to define normal constraints on the occluding side of the curve. Shape information will very likely be derived from the diffusion of other curves nearby. This solution is compatible with Johnston's affirmation that the area should be interpolated without considering such a silhouette.

In Lumo approach, tools to tag the edges make an initial best-guess for each line segment by inspecting a relative-depth list of paint regions. To derive normal information at the object side of a contour curve, we propose to automatically extract it from its geometry. It is a well known fact that a normal vector at a point p of a smooth surface is defined by the cross product of two tangent vectors on the surface. In the contour case, one of these tangents is known to be aligned with the view direction. Considering an orthographic camera, it follows that the normal of the surface is given by the cross product of the 2D curve's tangent and the view direction, i.e., $(n_y, -n_x, 0) \times (0, 0, -1) = (n_x, n_y, 0) = n(p)$. This means that contours represent points on the surface where the normal vector $n(p)$ corresponds to the normal of the 2D curve. Although there are two choices for the sign

of the normal, we suggest that a good approximation can be performed by relying on the tangent direction of the curve. It assumes the surface to lie on the left of the curve, which allows us to chose the vector pointing outwards the interior of the shape. When using such line drawing system to generate normal maps, contour lines can be efficiently generated by an automatic initialization of Diffusion Barriers. With such approach, solely normal constraints along one side of the curve are calculated, the interior of the shape. The other side remains unconstrained. Figure 4.15 shows the result of a prototype system based on Diffusion Barriers to illustrate our presented approach. Our solution generates normal maps with more shape details than Lumo's blobbed solution, while avoiding the quilted look. In addition, we believe that our previously introduced non-local diffusion presented on Chapter 3 can be used to improve the shape retrieved from partially occluded areas.

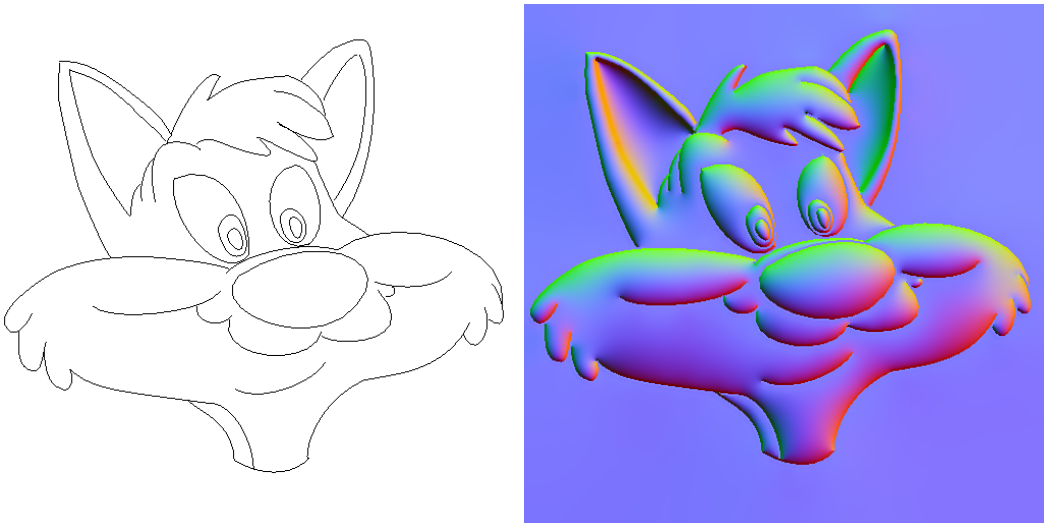


Figure 4.15 *One-sided curves. Left: Outlines. Right: Normal interpolation from one-sided normal curves.*

4.5.2 Mathematical Lines

In the case of Johnston's approach, we realize that one can generate convincing shapes because he was able to point out the characteristics of the surface represented by contours. This suggests that, if these properties are known for other types of lines, as for those discussed on Section 4.1, it would be possible to generate different types of surfaces. For instance, when drawing suggestive contours with Diffusion-Curve primitives, the system can set the normals along the line to lay on the image plane, but to slightly bend towards the viewer when reaching the contour prolongations. The same holds for ridge and valley lines. These lines characterize for an abrupt change in the normal surface direction on its left and

right side. When this kind of lines are drawn, predefined normals can be suggested on both sides of the curve, and a draft solution obtained. If, as for Winnemöller *et al.* [WOBT09], the user has the possibility to manipulate the normals on these lines via normal control points, a broader variety of surfaces could be easily drawn. In this way, we propose to extend Lumo's set of lines with others already well defined in computer-generated line drawings.

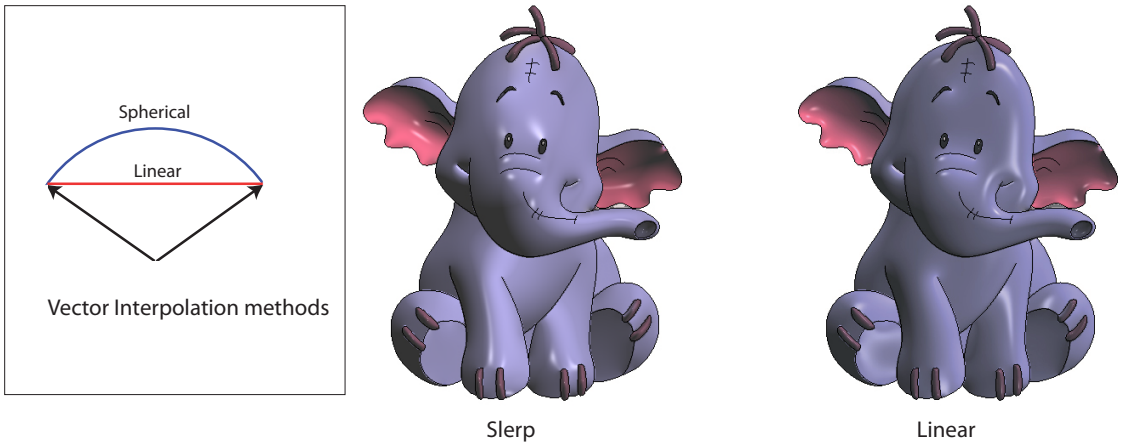


Figure 4.16 *Different interpolation methods. Left: Spherical versus Linear interpolations. Middle: Shading result using SLERP interpolation. Right: Shading result using linear interpolation (with n_z recomputation).*

4.5.3 Surface Reconstruction from Line Properties

In a certain way, using the surface properties of lines to reconstruct surfaces shares similarities to the labeling systems presented in Section 4.2.1. We remind the reader that, once lines were labeled, their geometric characteristics were crucial in imposing constraints on the surface, helping the system to retrieve corresponding shapes. Approaches as discussed in Malik's work [Mal86] to extract smooth surfaces based on line constraints might also be of help in interpolating normal vectors across regions. In the case of interpolation methods as for Lumo, such function relates to the interpolation scheme employed. This fact suggests that the interpolation function plays a major role in the definition of the surface once normal constraints are specified along curves.

Because only sparse information is available, it is difficult to propose a general interpolation technique capable of defining any arbitrary shape, but such interpolation schemes, do result in surfaces with different characteristics. To illustrate its fact, we show an example generated by interpolating normals using a spherical linear interpolation technique (SLERP). Spherical interpolation between two vectors does not interpolate each coordinate

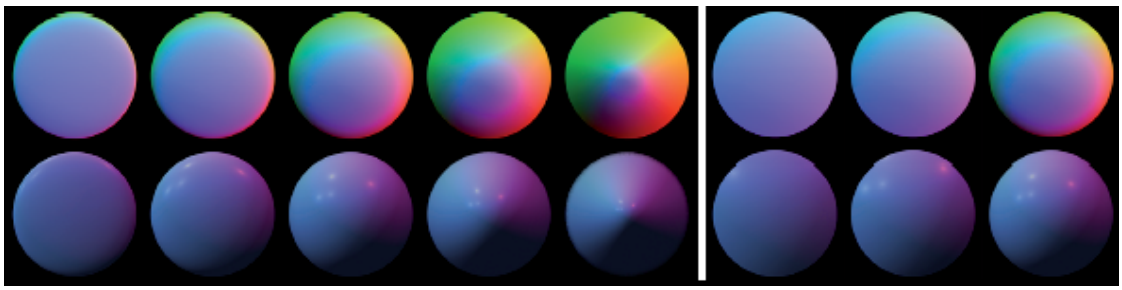
individually like linear interpolation does. Instead, the direction and lengths are interpolated individually (see Figure 4.16 left). The spherical interpolation of two normalized vectors (p_0, p_1) is calculated using the formula:

$$\mathrm{Slerp}(\mathrm{p}_0, \mathrm{p}_1, \mathrm{t}) = \frac{\sin[(1-\mathrm{t})\Omega]\mathrm{p}_0 + \sin[\mathrm{t}\Omega]\mathrm{p}_1}{\sin\Omega}$$

where Ω is the angle between (p_0, p_1). The result of using this interpolation to generate normal maps from lines is that the spherical shape of objects appears more pronounced when compared to Johnston's approach. The modified interpolation will shift more normals away from the viewer, hereby enhancing the shape. This fact can be observed in Figure 4.16 middle and right. In particular, realize how the region in the center of the elephant's face is more pronounced when using the SLERP method. As a side effect, the SLERP interpolation also has the advantage that n_z no longer needs a special treatment as was the case in Johnston's approach. Generally, we want to indicate that it would be interesting to exploit these different interpolation schemes in order to control the generation of a broader range of shapes.

Adjusting Shape Curvature with Lumo

For both Johnston's and Winnemöller's approach, the normal vector in a pixel p is calculated by minimizing differences within a certain neighborhood. For these approaches, n_x and n_y are interpolated across the region, and n_z recomputed to define a unit vector. The result of this algorithm is that normals are interpolated to create the profile of a hemisphere. To enable the manipulation of the generated shape, local and global scaling approaches were suggested by Johnston. In the first one, the blending of quilted and blobbed normals can be manipulated by adjusting the edge confidence intensity nearby lines. This is similar to adjusting the normal values along lines in the diffusion curve approach. Nevertheless, care must be taken when recomputing the n_z coordinate. Two other algorithms were proposed to control the *puffiness* of a region. They can be seen as a post-process once the spherical normal map is obtained via traditional diffusion. The first idea is to replace a normal with the normal of a sphere that has been scaled in z; the second replaces hemispherical normals by the normals of a visible unit-sphere slice of thickness S: n_x and n_y are linearly scaled by $\sqrt{S \cdot (2 - S)}$ and n_z is recomputed to maintain unit length. This method can indeed generate interesting results from shallow to pointed surfaces as illustrated in the Figure bellow. On the left, z-scaled sphere normals with values $S = (0.25, 0.5, 1.0, 2.0, 4.0)$. On the right, uniformly scaled normals with values $S = (0.125, 0.25, 1.0)$



4.5.4 Anisotropic Diffusion

Control over the diffusion process itself can also bring benefits to depiction of different shapes. For instance, making use of our anisotropic diffusion presented on section 3.7.2. Once normals are able to follow the direction of a given vector field, a variety of shapes become possible and many tedious definitions are simplified. In addition, getting inspiration from shape from shading approaches, we believe that the expressiveness of the system can yet be improved if such anisotropic diffusion is extended in order to reflect changes also in gradient magnitudes. Here again, as for colors, solutions for adapted interfaces to define such gradients must be derived. It is important to remember that keeping the solution vector-based can be a great contribution to an overall production pipeline.

4.5.5 Animated System

Another interesting challenge is to study the possibility to work on animated systems. For this to work, care must be taken especially when deforming objects. This is a major problem even for current animation software solutions like, for example, Toon Boom Animation© [1]. In particular, when applying texture with this solution on top of animated two-dimensional drawings, the result does not look convincing because the texture seems to slide on the surface [2]. When using a Diffusion-Curve approach to create normal fields, care must be taken when curves are deformed. In this case, normal constraints along curves must be adapted in order to reflect such shape deformations. Temporal-coherence must also be taken into account. While curve-based definitions could be directly transferred from one frame to the other if the curve relations are indicated by the artist, we could further make use of non-local constraints over time in order to attach values from different frames to each other. This assures a coherent animation with little effort. In addition, the problem of animated scenes requires real-time-oriented solutions. Therefore, optimizations in hardware, taking advantage of the latest features of graphic cards, must be considered to derive an easy-to use solution that produces real-time results. Our current attempts went in the direction of sparse GPU solvers, but in the future, we plan to investigate more specified solutions, that exploit the locality of the diffusion process.

[1] http://www.toonboom.com/products/animate/

[2] an illustration of this limitation, taken from the Toon Boom Galery can be found in http://artis.inrialpes.fr/Members/Hedlena.Bezerra/resources/chasse_galerie_sample_03.mov

4.6 Discussion

There are many perspectives through which one can look at line drawings. We have seen in this chapter that, when someone is drawing, lines are representations for the mind of three-dimensional entities. Further, even if such drawings are created by someone else, we are able of re-projecting such primitives and recognize their signification to a large extent. In fact, much could be interpreted by using just lines, but shade cues and colors can add harmony to the depiction. Further, shading information can ease the process of perceiving volumetric shapes.

However, as a general note, automatically inferring information from a given two-dimensional line drawing is a very complex task. Finding a good set of controls often implies that one should abstract the problem and evaluate it from different perspectives. It can be useful, for instance, to adapt a solution for controlling the generation of cartoon normal fields from computer vision techniques or from understanding of psychology. In fact, bringing knowledge from different fields related to the understanding of perception in art is an ongoing effort. This is a trend that can increasingly be observed in the scientific community of the field (examples include events such as NPAR 2010 [DS10]). This chapter follows this direction and discusses a variety of methods that, in the context of 2D image shading, could contribute to a more general, yet simple surface definition machinery.

5 Conclusion

In this dissertation, we investigated the process of producing expressive imagery with the aid of computers. In particular, we have seen that algorithms and mathematical tools can provide a substantial basis to provide tools that are easy to use and can deliver high-quality illustrations. Even novice users can produce convincing results. Unfortunately, such constructs often restrain true artists and reduce their freedom. One direction would have been to derive complex solutions to reestablish the generality. Nonetheless, when looking at the history of computer graphics and even art, complex techniques are often doomed to fail because they block out the novice user and can be very nonintuitive.

Our thesis is that it is possible to combine simplicity and artistic freedom. One key insight is that both goals can be reached in a very simple manner. We found that it is important to maintain the actual creation process simple, so that even an unexperienced user understands the principle of the approach and can quickly produce convincing artworks. On the other hand, we believe that it is crucial to enable a higher control over the simple process to regain the artistic freedom, hereby, combining the best properties of both worlds. Similar concepts can be found in standard software systems, where default values might remain untouched by beginners, but are tweaked by professional users.

Nonetheless, there is one more aspect in our observation, the application itself must exhibit a certain simplicity, so that people understand immediately the effect of their manipulations. This is best achieved by building the creation process upon a simple task (flooding colors, grouping elements, inflating drawings...), that are particularly easy to perform by a machine, well understood by a user, but would demand a lot of effort when done manually. Although, it should be pointed out that simplicity does not necessarily imply a naive process. In many cases, we showed that only a sound mathematical foundation allows us to achieve a convincing behavior, rather we claim that the process should be simple in the sense that its outcome is predictable and intuitive to the observer.

Interestingly, such simple processes (not unlike the well-known *Game of life*) do not necessarily lead to simple outcomes. In fact, even complex results can be achieved by simple processes, and to navigate appropriately in this vast space of potential results, one needs the possibilities to guide the outcome in an intuitive way. Usually, such controls can themselves be simple (following the previous example, e.g., an initial configuration for the *Game of Life*), but need to exhibit the needed generality.

Our thesis is that by allowing for a general control over such a simple process, we can combine simplicity of usage with general artistic freedom. Such a claim is hard to prove, because we would need to address all tools, potentially even algorithms that have not yet been conceived. Instead, we show that this principle is a very useful and powerful concept by illustrating our findings with various examples.

In this dissertation, we illustrated that grouping information is a crucial element for the stylization of a scene, can improve understanding of spatial relationships and even convey semantic information. Furthermore, we show that while the grouping itself is a tedious process that can be performed by a machine and can follow a simple process, an artist would like to control this behavior in order to fulfill the artistic vision. Our principle enables complex stylization that takes the dynamic scene configuration into account without the need to

resort to tedious manipulations. Hereby, we show that convincing results can be obtained easily without restricting the artistic freedom and that scene structure is an important component in the stylization process.

A second example is our work on Diffusion Curves. Vector illustrations have many advantages, but producing gradients is tedious because the few existing techniques do not provide enough generality to produce convincing illustrations. Building upon Diffusion Curves, we showed that by providing intuitive higher level controls, it is possible to produce complex gradients, and reestablish generality without resorting to nonintuitive solutions. Many difficult tasks are simplified by our solution and properties, such as diffusion barriers, appear as natural extensions. Our approach generalizes many aspects of Diffusion curves and delivers a useful algorithm to produce complex vector illustrations.

A third example illustrates how cartoon drawing illumination can be guided by following the same principles. Initially, we show that a simple inflation scheme can provide the user with an initial guess of the surface shape. While this form might be acceptable for a novice user, more freedom is needed when manipulating surface shape more accurately. Via higher level controls (e.g., tweaked normal definitions, inflation techniques or feature curves) it is possible to generalize the approach without removing its simplicity. It becomes easy to define normal fields over surfaces that allow efficient cartoon illumination. In particular, such an approach avoids the tedious redrawing of elements when changing the illumination direction, which would be the case when working with shading directly instead of the surface geometry.

Our findings indicate that it is useful to augment simple processes with higher-level control in order to allow simple, yet general expressive content creation. Ultimately, we cannot prove the optimality of our solutions, but we believe they are convincing enough to defend our principle. Furthermore, each of our contribution is, as itself, a useful contribution to the computer graphics community, as our algorithms provide a deeper understanding of the processes that link art and algorithmics. Furthermore, our work delivers solutions that are a gain for the artistic toolbox and provide new ways of abstraction and stylization that help to better convey the artistic intentions. In this sense, we believe that we reached our goal to better understand how to combine artistic freedom with simplicity of usage. In the end, art is for everybody, not only from a passive point of view, where someone observes an artwork, but also from an active point of view, where art is created. All we need is guidance... or *control*.

Bibliography

[BD07] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Asia-Pacific Symposium on Visualisation 2007*, 2007.

[BEDT08] Hedlena Bezerra, Elmar Eisemann, Xavier Décoret, and Joëlle Thollot. 3d dynamic grouping for guided stylization. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 89–95, New York, NY, USA, 2008. ACM.

[BEDT10] Hedlena Bezerra, Elmar Eisemann, Doug DeCarlo, and Joëlle Thollot. Diffusion constraints for vector graphics. In *NPAR '10: Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 35–42, New York, NY, USA, 2010. ACM.

[BGH03] J.A. Bangham, S.E. Gibson, and R. Harvey. The art of scale-space. In *Proceedings of British Machine Vision Conference 2003*, pages 569–578, 2003.

[BKTS00] Adrien Bousseau, Matt Kaplan, Joëlle Thollot, and François X. Sillion. Interactive watercolor rendering with temporal coherence and abstraction. In *Proceedings of NPAR 2006*, pages 141–149, 2000.

[BNTS07] Adrien Bousseau, Fabrice Neyret, Joëlle Thollot, and David Salesin. Video watercolorization using bidirectional texture advection. *ACM Transaction on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3), 2007.

[BTM06] P. Barla, J. Thollot, and L. Markosian. X-toon: An extended toon shader. In *Proceedings of NPAR 2006*, pages 127–132, 2006.

[BY83] Michael Brady and Alan Yuille. An extremum principle for shape from contour. In *IJCAI'83: Proceedings of the Eighth international joint conference on Artificial intelligence*, pages 969–972, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

[Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[Car88] Stefan Carlsson. Sketch based coding of grey level images. *Signal Process.*, 15(1):57–83, 1988.

[CAS+97] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 421–430, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Cav05] Patrick Cavanagh. The artist as neuroscientist. *Nature*, 434:301–307, March 2005.

[CDF+06] F. Cole, D. DeCarlo, A. Finkelstein, K. Kin, K. Morley, and A. Santella. Directing gaze in 3d models with stylized focus. In *Proceedings of Eurographics Symposium on Rendering 2006*, pages 377–387, 2006.

[CEW+08] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. *ACM Trans. Graph.*, 27(3), 2008.

[CGL+08] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–11, New York, NY, USA, 2008. ACM.

[CJTF98] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture mapping for cel animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 435–446, New York, NY, USA, 1998. ACM.

[Clo71] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79 – 116, 1971.

[CM02] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[Coo86] R. L. Cook. Stochastic sampling in computer graphics. In *Proc. of SIGGRAPH*, pages 51–72. ACM Press, 1986.

[Cra00] Walter Crane. *Line and Form*. Project Gutenberg EBook, 1900.

[CS68] Charles Csuri and James Shaffer. Art, computers and mathematics. In *AFIPS '68 (Fall, part II): Proceedings of the December 9-11, 1968, fall joint computer conference, part II*, pages 1293–1298, New York, NY, USA, 1968. ACM.

[CSD+09] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–9, New York, NY, USA, 2009. ACM.

[CSHD03] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 287–294, New York, NY, USA, 2003. ACM.

[CST05] Paolo Cignoni, Roberto Scopigno, and Marco Tarini. A simple normal enhancement technique for interactive non-photorealistic renderings. *Computer & Graphics*, 29(1):125–133, feb 2005.

[DFR04] Doug DeCarlo, Adam Finkelstein, and Szymon Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 15–145, New York, NY, USA, 2004. ACM.

[DFRS03] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 848–855, New York, NY, USA, 2003. ACM.

[DFVR02] F. Di Fiore and F. Van Reeth. Employing approximate 3d models to enrich traditional computer assisted animation. In *Proceedings of Computer Animation*, pages 183–190, 2002.

[DR07] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 63–70, New York, NY, USA, 2007. ACM.

[DS02] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. In *Proceedings of SIGGRAPH 2002*, pages 769–776, 2002.

[DS10] Doug DeCarlo and Matthew Stone. Visual explanations. In *NPAR '10: Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 173–178, New York, NY, USA, 2010. ACM.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1996. ACM.

[Dur02] Fredo Durand. Limitations of the medium and pictorial techniques, July 2002. Perceptual and Artistic Principles for Effective Computer Depiction, Course Notes for ACM SIGGRAPH 2002.

[Edw87] Betty Edwards. *Drawing on the Artist Within: An Inspirational and Practical Guide to Increasing Your Creative Powers*. Fireside, April 1987.

[EG01] James H. Elder and Richard M. Goldberg. Image editing in the contour domain. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):291–296, 2001.

[Eld99] James H. Elder. Are edges incomplete? *International Journal of Computer Vision*, 34(2-3):97–122, 1999.

[EPD09] Elmar Eisemann, Sylvain Paris, and Frédo Durand. A visibility algorithm for converting 3D meshes into editable 2D vector graphics. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[EWHS08] Elmar Eisemann, Holger Winnemöller, John C. Hart, and David Salesin. Stylized vector art from 3d models with region support. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008)*, 27(4), June 2008.

[EZ96] James H. Elder and Steven W. Zucker. Space scale localization, blur, and contour-based image coding. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR 1996)*, pages 00–27, 1996.

[EZ98] James H. Elder and Steven W. Zucker. Local scale control for edge detection and blur estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(7):699–716, 1998.

[Fel03] Jacob Feldman. Perceptual grouping by selection of a logically minimal model. *Int. J. Comput. Vision*, 55(1):5–25, 2003.

[FH04] Hui Fang and John C. Hart. Textureshop: Texture synthesis as a photograph editing tool. *ACM Transactions on Graphics*, 23:2004, 2004.

[FHL+09] Zeev Farbman, Gil Hoffer, Yaron Lipman, Daniel Cohen-Or, and Dani Lischinski. Coordinates for instant image cloning. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–9, New York, NY, USA, 2009. ACM.

[GGSC98] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452, New York, NY, USA, 1998. ACM.

[Gil91] Robert W. Gill. *Basic Rendering: Effective Drawing for Designers, Artists and Illustrators*. Thames & Hudson, 1991.

[Gom61] Ernst Hans Gombrich. *The Story Of Art*. Phaidon Press, 1961.

[GTDS04] Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François Sillion. Programmable style for npr line drawing. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)*. ACM Press, june 2004.

[GTDS10] Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François X. Sillion. Programmable rendering of line drawing from 3d scenes. *ACM Trans. Graph.*, 29(2):1–20, 2010.

[Huf71] D. A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[HZ00] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[IB06] Tobias Isenberg and Angela Brennecke. G-strokes: A concept for simplifying line stylization. *Computers & Graphics*, 30(5):754–766, October 2006.

[IBCSC06] Tobias Isenberg, Angela Brennecke, Mario Costa Sousa, and Sheelagh Carpendale. Beyond pixels: Illustration with vector graphics. Technical report, University of Calgary, February 2006.

[IFP95] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 52, Washington, DC, USA, 1995. IEEE Computer Society.

[IMT06] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 11, New York, NY, USA, 2006. ACM.

[INC+06] Tobias Isenberg, Petra Neumann, Sheelagh Carpendale, Mario Costa Sousa, and Joaquim A. Jorge. Non-photorealistic rendering in context: an observational study. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 115–126, New York, NY, USA, 2006. ACM.

[JC08] P. Joshi and N. Carr. Repoussé: Automatic inflation of 2d art. In *Proceeding of Eurographics Workshop on Sketch-Based Modeling*, 2008.

[JCW09a] Stefan Jeschke, David Cline, and Peter Wonka. A GPU laplacian solver for diffusion curves and poisson image editing. In *Proceedings of SIGGRAPH Asia 2009*, pages 1–8, 2009.

[JCW09b] Stefan Jeschke, David Cline, and Peter Wonka. Rendering surface details with diffusion curves. In *Proceedings of SIGGRAPH Asia 2009*, pages 1–8, 2009.

[JDA07] Tilke Judd, Frédo Durand, and Edward Adelson. Apparent ridges for line drawing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 19, New York, NY, USA, 2007. ACM.

[JGB+06] Andrew Jones, Andrew Gardner, Mark Bolas, Ian McDowall, and Paul Debevec. Simulating spatially varying lighting on a live performance. In *Proceedings of 3rd European Conference on Visual Media Production (CVMP 2006)*, 2006.

[Joh02] Scott F. Johnston. Lumo: illumination for cel animation. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 45–ff, New York, NY, USA, 2002. ACM.

[Kan81] Takeo Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409 – 460, 1981.

[KHRO01] M. A. Kowalski, J. F. Hughes, C.B. Rubin, and J. Ohya. User-guided composition effects for art-based rendering. In *Proceedings of Symposium on Interactive 3D Graphics 2001*, pages 99–102, 2001.

[KMM+02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. Wysiwyg npr: drawing strokes directly on 3d models. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 755–762, New York, NY, USA, 2002. ACM.

[KRFB06] Erum Arif Khan, Erik Reinhard, Roland W. Fleming, and Heinrich H. Bülthoff. Image-based material editing. *ACM Trans. Graph.*, 25(3):654–663, 2006.

[KST08] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal. Demarcating curves for shape illustration. *ACM Trans. Graph.*, 27(5):1–9, 2008.

[KvK92] J.J. Koenderink, A.J. vanDoorn, and A.M. Kappers. Surface perception in pictures. *Percept Psychophys*, 52(5):487–496, November 1992.

[KvKT01] J.J. Koenderink, A.J. vanDoorn, A.M. Kappers, and J.T. Todd. Ambiguity and the 'mental eye' in pictorial relief. *Perception*, 30(4):431–438, 2001.

[KWH06] A. Kolliopoulos, J. M. Wang, and A. Hertzmann. Segmentation-based 3d artistic rendering. In *Proceedings of Eurographics Symposium on Rendering 2006*, pages 361–370, 2006.

[LB91] Yvan G. Leclerc and Aaron F. Bobick. The direct computation of height from shading. In *In Conference on Computer Vision and Pattern Recognition*, pages 552–558, 1991.

[LD04] T. Luft and O. Deussen. Watercolor illustrations of plants using a blurred depth test. In *Proceedings of NPAR 2004*, pages 11–20, 2004.

[LD06] Ares Lagae and Philip Dutré. Poisson sphere distributions. In *Vision, Modeling, and Visualization*. IEEE Computer Society, 2006. Accepted.

[LHM09] Yu-Kun Lai, Shi-Min Hu, and Ralph R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transaction on Graphics*, 28(3):1–8, 2009.

[LL06a] G. Lecot and B. Levy. Ardeco: Automatic region detection and conversion. In *Proceedings of Eurographics Symposium on Rendering 2006*, pages 349–360, 2006.

[LL06b] Gregory Lecot and Bruno Levy. Ardeco: Automatic Region DEtection and COnversion. In *Proceedings of the 17th Eurographics Symposium on Rendering (EGSR 2006)*, pages 349–360, 2006.

[LMJH+10] Jorge Lopez-Moreno, Jorge Jimenez, Sunil Hadap, Erik Reinhard, Ken Anjyo, and Diego Gutierrez. Stylized depiction of images based on depth perception. In *NPAR '10: Proceedings of the 8th international symposium on Non-photorrealistic animation and rendering*. ACM, 2010.

[LMLH07] Yunjin Lee, Lee Markosian, Seungyong Lee, and John F. Hughes. Line drawings via abstracted shading. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 18, New York, NY, USA, 2007. ACM.

[LRC+03] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan-Kaufmann, Inc., 2003.

[Mal86] Jitendra Malik. *Interpreting line drawings of curved objects*. PhD thesis, Stanford, CA, USA, 1986.

[MMK+00] L. Markosian, B. Meier, M. A. Kowalski, L. S. Holden, J. D. Northrup, and J. F. Hughes. Art-based rendering with continuous level of details. In *Proceedings of NPAR 2000*, pages 59–66, 2000.

[MP08] James McCann and Nancy S. Pollard. Real-time gradient-domain painting. In *Proceedings of SIGGRAPH 2008*, 2008.

[MTG06] Kyoko Murakami, Reiji Tsuruno, and Etsuo Genda. Natural-looking strokes for drawing applications. *Vis. Comput.*, 22(6):415–423, 2006.

[Nic90] Kimon Nicolaïdes. *The Natural Way to Draw: A Working Plan for Art Study*. Mariner Books, February 1990.

[Nol66] A. Michael Noll. Human or machine: A subjective comparison of piet mondrian's 'composition with lines' and a computer-generated picture. *The Psychological Record*, 16(1):1–10, January 1966.

[Nol67] A. Michael Noll. The digital computer as a creative medium. *IEEE Spectrum*, 4(10):89–95, October 1967.

[OBBT07] Alexandrina Orzan, Adrien Bousseau, Pascal Barla, and Joëlle Thollot. Structure-preserving manipulation of photographs. In *Proceedings of NPAR 2007*, pages 103–110, aug 2007.

[OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3):609–612, 2004.

[OBW+08] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation for smooth-shaded images. In *Proceedings of SIGGRAPH 2008*, volume 27, 2008.

[OZ06] Matt Olson and Hao Zhang. Silhouette extraction in hough space. In *Proceedings of EUROGRAPHICS 2006*, 2006.

[OZM+06] Makoto Okabe, Gang Zeng, Yasuyuki Matsushita, Takeo Igarashi, Long Quan, and Heung yeung Shum. Single-view relighting with normal map painting. In *In Proceedings of Pacific Graphics 2006*, pages 27–34, 2006.

[PD07] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *Proceedings of IEEE conference on Computer Vision and Pattern Recognition 2007*, 2007.

[Pen84] A. P. Pentland. Local shading analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):170–187, March 1984.

[PFWF00] Lena Petrović, Brian Fujito, Lance Williams, and Adam Finkelstein. Shadows for cel animation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 511–516, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Proceedings of SIGGRAPH 2003*, pages 313–318, 2003.

[Pia92] Jean Piaget. *La formation du symbole chez l'enfant: Imitation, jeu et rêve, image et représentation*, page 310. Delachaux & Niestle, 8th edition, July 1992.

[RBD06] Szymon Rusinkiewicz, Michael Burns, and Doug DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Trans. Graph.*, 25(3):1199–1205, 2006.

[SEH08] Matei Stroila, Elmar Eisemann, and John Hart. Clip art rendering of smooth isosurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):135–145, 2008.

[SFWS03] M. C. Sousa, K. Foster, B. Wyvill, and F. Samavati. Precise ink drawing of 3d models. *Computer Graphics Forum (Proceedings of Eurographics 2003)*, 22(3):369–379, September 2003.

[SLWS07] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Transaction on Graphics*, 26(3):11, 2007.

[Spa00] W. E. Sparkes. *How to shade from models, common objects, and casts of ornament*. Cassel, 1900.

[Spa98] Anne Morgan Spalter. *The Computer in the Visual Arts*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[SS02] Thomas Strothotte and Stefan Schlechtweg. *Non Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann, 2002.

[Sto10] Elizabeth Stover. What is a medium in art? Internet, July 2010.

[Sug86] Kokichi Sugihara. *Machine interpretation of line drawings*. MIT Press, Cambridge, MA, USA, 1986.

[Ter03] B. M. Ter Haar Romeny. *Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, Written in Mathematica*. Springer, 2003.

[TFFR07] Corey Toler-Franklin, Adam Finkelstein, and Szymon Rusinkiewicz. Illustration of complex real-world objects using images with normals. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, August 2007.

[TT99] Jack Tumblin and Greg Turk. Lcis: a boundary hierarchy for detail-preserving contrast reduction. In *Proceedings of SIGGRAPH 1999*, pages 83–90, 1999.

[VPB+09] Romain Vergne, Romain Pacanowski, Pascal Barla, Xavier Granier, and Christophe Schlick. Light warping for enhanced surface depiction. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[Wah90] G. Wahba. *Spline models for observational data*, volume 59. SIAM, 1990.

[Wal75] D. Waltz. Understanding line drawings of scenes with shadows. In Patrick Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.

[Wik10] Wikipedia. United states census, August 2010.

[Wil06] Philip Willis. Projective alpha colour. In *Proceeding of Eurographics 2006*, 2006.

[WJ95] M. P. Wand and M. Jones. *Kernel Smoothing*. Chapman and Hall, 1995.

[WOBT09] Holger Winnemöller, Alexandrina Orzan, Laurence Boissieux, and Joëlle Thollot. Texture design and draping in 2d images. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2009)*, 28(4):1091–1099, 2009.

[WOG06] Holger Winnemoller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. In *Proceedings of SIGGRAPH 2006*, pages 1221–1226, 2006.

[WQL+06] F. Wen, Q.Luan, L. Liang, Y-Q. Xu, and H-Y. Shum. Color sketch generation. In *Proceedings of NPAR 2006*, pages 47–54, 2006.

[WS94] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100, New York, NY, USA, 1994. ACM.

[WTBS07] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. Shapepalettes: interactive normal transfer via sketching. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 44, New York, NY, USA, 2007. ACM.

[WTXC04] J. Wang, B. Thiesson, Y. Xu, and M. F. Cohen. Image and video segmentation by anisotropic mean shift. In *Proceedings of European Conference on Computer Vision 2004*, pages 238–249, 2004.

[WWSS96] Georges Winkenbach, Georges Winkenbach, David H. Salesin, and David H. Salesin. Rendering parametric surfaces in pen and ink. In *SIGGRAPH 96 Conference Proceedings, Annual Conference Series*, pages 469–476. ACM SIGGRAPH, Addison Wesley, 1996.

[WXSC04] J. Wang, Y. Xu, H-Y. Shum, and M. F. Cohen. Video tooning. In *Proceedings of SIGGRAPH 2004*, pages 574–583, 2004.

[YBS05] Shin Yoshizawa, Alexander Belyaev, and Hans-Peter Seidel. Fast and robust detection of crest lines on meshes. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 227–232, New York, NY, USA, 2005. ACM.

[ZTCS99] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *Pattern Analysis and Machine Intelligence*, 21(8):690–706, August 1999.