# Content-Based Media Processing

A dissertation submitted to the University of Dublin
for the degree of Doctor of Philosophy

**Deirdre O'Regan**
University of Dublin, Trinity College, April 2010

Signal Processing and Media Applications
Department of Electronic and Electrical Engineering
Trinity College Dublin

*To my family and friends.*

# Declaration

I hereby declare that this thesis has not been submitted as an exercise for a degree at this or any other University and that it is entirely my own work.

I agree that the Library may lend or copy this thesis upon request.

Signed,

Deirdre O'Regan

April 12, 2010.

# Abstract

This thesis is concerned with Content-Based Media Processing (CBMP). CBMP involves the detection or exploitation of the salient content in digital media for applications in digital media processing. Salient content is defined as any attribute, object or "event" that is meaningful, recognizable or important to the user and/or application. The usefulness of CBMP is explored in three different projects in audio, image and video processing respectively.

The first audio-based project proposes a new algorithm for example-based Sound Texture Synthesis (STS). STS is the synthesis of a long body of sound texture from a short training example clip. The goal is that the resulting sound texture is perceptually similar to the training example, but does not sound like the training example repetitively tiled. Applications include damaged audio repair, and the generation of ambient sound for computer games, installations and movies. Example-based STS is content-based in that it involves estimating the statistics of the sound texture by measuring directly from the training data. A multi-resolution approach is taken here, and employment of the Dual-Tree Complex Wavelet Transform (DT-CWT) is found to be useful for complexity reduction. Content-based analysis is used for estimating the values of parameters inherent to the algorithm. This example-based STS algorithm performs well compared to another prominent algorithm in the field, and the resulting sound textures are of good quality and long duration, temporally varied and perceptually similar to the training example clips tested.

The second image-based project is concerned with Implicit Spatial Inference (ISI) with sparse local features for Face Detection. Face Detection is an extremely important tool in the field of CBMP, since the faces of people constitute much of semantic content in home movies and digital images, cinema and camera mobile phone clips. General Object Detection is also useful, and the use of sparse local features such as SIFT (Scale Invariant Feature Transform) is a popular approach in the field. Linking these sparse features together to properly localize objects, however, is a difficult problem. ISI is a novel technique for leveraging the implicit geometric context of these sparse features in a Bayesian framework. A likelihood is determined from the classification output of a machine learning algorithm trained on these features, and a Markov Random Field (MRF) prior is used to inject contextual geometrical information. The MRF is imposed on a graph obtained by Delaunay triangulation of the sparse features. Promising detection, segmentation and invariance results are obtained in a Face Detection task.

The third project video-based project is concerned with content-based video stylization, and this concept is explored in a novel framework for the Skin-Aware Stylization of Video Portraits. Video portraits are the animated head shots of people captured in the image sequences extracted from home movies, cinema, TV and mobile phone clips. A novel spatio-temporal Skin Detection algorithm is used to identify the skin-colored image regions in these sequences, allowing them to be highlighted while abstracting the less important background regions. The framework merges ideas in Non-Photorealistic Rendering (NPR) - such as edge-based cartoonization and motion expression - with the technique of Stroke-Based Rendering (SBR) for painterly stylization. Several novel techniques in SBR are proposed, including a probabilistic algorithm for brush stroke anchor point distribution, the manipulation of brush strokes in emphasizing motion, spatio-temporal color-sampling, and occlusion detection for mitigating the well-known problems of gaps and redundancy in motion-compensated brush stroke animation. The framework is used to stylize a number of video portraits of people with interesting and artistic results.

# Acknowledgments

First and foremost I would like to thank my supervisor Prof. Anil Kokaram, for all of his help, encouragement and support over the last four years. Many thanks also to Dr. David Corrigan and Dr. Francois Pitié for their immediate, generous and invaluable help and advice, especially during the last crucial months.

Thanks to all past and present members of the Sigmedia research group, and staff of the Electronic and Electrical Engineering Department. I particularly wish to thank Dr. Linda Doyle, Dr. Rozenn Dahyot, Dr. Naomi Harte, Prof. Frank Boland and Liam Dowling for their support and kindness over the years. Thanks also to Daire, Hugh, Riccardo, John, Gary and Darren for good memories.

Best of luck to the three other members of the thesis-writing support group! Dan Ring, here's to eight great years of friendship, adventures, and Italo! Gav Kearney, you're the best friend that anybody could wish for - rock on! Damien Kelly, you helped get me through with the supportive chats and empathy!

My research has been financed by the Irish Research Council for Science, Engineering and Technology (IRCSET) under the Embark Initiative, and I am very grateful for this funding. Some of this work was also financed by, and carried out at Adobe Systems Incorporated. Thanks to everybody at Adobe for a great summer spent interning in Seattle, most especially Dave Simons and Dr. Andy Moorer.

Finally I would like to thank my family for their continuing support and encouragement during my postgraduate years. To Marian for her unflagging optimism and sympathetic ear, and to Jerry without whom my many years of education would not have been possible! To my lovely sister Aoife who can brighten my dullest of days! And finally, to Aidan for being there for me through it all... thanks from the bottom of my heart.

# Contents

# List of Acronyms

**2D** 2-Dimensional

**3D** 3-Dimensional

**B-G** Bi-Gaussian

**CBMP** Content-Based Media Processing

**DT-CWT** Dual-Tree Complex Wavelet Transform

**DWT** Discrete Wavelet Transform

**EER** Equal Error Rate

**ICM** Iterated Conditional Modes

**IIR** Infinite Impulse Response

**ISI** Implicit Spatial Inference

**ITS** Image Texture Synthesis

**MRF** Markov Random Field

**NPR** Non-Photorealistic Rendering

**OCD** Object Class Detection

**PDD** Poisson Disk Distribution

**PDF** Probability Distribution Function

**PDS** Poisson Disk Sampling

**SBR** Stroke-Based Rendering

**SIFT** Scale Invariant Feature Transform

**STFT** Short-Term Fourier Transform

| | |
|---|---|
| **STS** | Sound Texture Synthesis |
| **SSD** | Sum of Square Differences |
| **SVM** | Support Vector Machine |

# 1 Introduction

With the rise of new digital media formats and ever-evolving digital capture devices, increasingly vast quantities of digital media are being produced by everybody from the ordinary digital camera owner to professionals in the broadcasting, cinema, and music industries. This digital revolution has created some very interesting challenges for the contributors and guardians of our technological world.

The problems of finding bandwidth for transmission, and space for storage of both new digital media, and digitized archive material, has fueled the continuous development of increasingly *content-based* compression techniques and summarization schemes. A specific example of such work is the recently popular Seam Carving for Content-Aware Image Resizing algorithm of (Avidan & Shamir, 2007), which is a clever scheme for reducing the size of a digital image by continuously removing "seams" of pixels that snake through the image avoiding the *salient* or important image regions. Saliency here is defined by regions of high local contrast or some other measure of importance. A more general example is the huge body of research concerned with the event-based summarization of sporting events for digital TV broadcast, including the research of (Kokaram et al., 2006; Kokaram et al., 2005) and (Denman et al., 2003). It is likely that the producers, broadcasters and owners of digital media will rely increasingly on content-based compression and summarization techniques such as these in the future.

Meanwhile, tools for the manipulation of digital media are extremely popular with both ordinary consumers, and industry professionals. Once based on simple sample-wise, pixel-wise, or frame-wise audio, image, or video processing filters, there is an increasing demand for smarter media processing tools. Tools that are capable of recognizing, and exploiting knowledge of the *meaningful* content in digital media, therefore bridging the so-called *semantic gap* between user and machine. The detection and localization of human faces in visual media for example, has become a hot topic in the field, most probably due to the fact that some of the most important content in home movies, TV broadcasts, and cinema material are captures of people, especially head shots and facial close-ups. Face Detection capability has even found its way into the software bundled with digital cameras and camera mobile phones where it is used to influence functions of the camera such as Auto-Focus (AF).

With the rise of media upload forums such as YouTube[1], MySpace[2], and Flickr[3], anybody can be an audio and/or visual artist. Tools for creative tasks such as image or video stylization and sound editing are becoming as omni-present in ordinary desktop software as they are in professional media processing software packages such as Adobe Creative Suite[4]. Time and effort, however, remain precious human commodities and any automation of media processing tasks is invaluable to ordinary consumers, professional media artists, computer game designers, and cinema post-production houses alike.

## 1.1 Content-Based Media Processing

Content-Based Media Processing (CBMP) involves the detection, analysis or modeling of the *salient* content in digital media in order to make decisions based on, or exploit the content in processing the media. In digital media, salient content can be thought of as any attribute, object or "event" that is semantic, important, recognizable or stand-alone to the user and/or application. There are many different levels of salient or semantic content, some examples of which are listed below:

**Low-Level:** Local extrema, various local or global statistics (e.g. correlations, entropy), pixelwise color values, motion vectors

**Mid-Level:** Periodicities (e.g. beats in music), features (e.g. spatial "patches" of image intensity or gradients), relative spatial geometry, properties of a motion field (e.g. occlusion, trajectories)

**High-Level:** Characteristic color classes (e.g. skin color), semantic events (e.g. goals scored in soccer broadcasts) and objects (e.g. faces)

[1]YouTube: `http//www.youtube.com`
[2]MySpace: `http//www.myspace.com`
[3]Flickr: `http//www.flickr.com`
[4]Adobe CS: `http://www.adobe.com/products/creativesuite/`

## 1.2 Thesis Outline

This thesis explores the concept of CBMP in three distinct projects involving audio, image and video processing respectively. Each of these projects hinges on the detection or exploitation of some aspect of salient content in a particular topic in media processing. In digital media, salient content can be thought of as any attribute, object or "event" that is semantic, important, recognizable or stand-alone to the user and/or application. Chapter 2 describes the audio-based project of example-based Sound Texture Synthesis (STS), Chapter 3 focuses on the image-based project of Implicit Spatial Inference (ISI) with sparse local features for Face Detection, Chapter 4 presents a review of the state-of-the-art in the stylization of visual media, Chapter 5 describes a novel algorithm for Skin Detection in images, and Chapter 6 presents the video-based project entitled Skin-Aware Stylization of Video Portraits. The following is a brief summary of each Chapter.

### Chapter 2: Example-Based Sound Texture Synthesis

This Chapter presents a novel example-based Sound Texture Synthesis (STS) algorithm. It begins with an explanation of how the algorithm was inspired by a well known example-based Image Texture Synthesis (ITS) algorithm and its multi-resolution extension. This is followed by a description of the algorithm, including a wavelet-based technique for complexity reduction. Next follows a subjective comparison of its sound texture results to those of another prominent wavelet-based STS algorithm. There is a discussion of the meaning and influence of some of the user-defined parameters inherent to the algorithm, along with some suggestions for content-based parameter estimation. Some more interesting sound texture results are then generated using these content-based parameter estimation techniques.

### Chapter 3: Implicit Spatial Inference with Sparse Local Features for Face Detection

This Chapter presents a novel algorithm for leveraging the implicit geometry of sparse local feature points for invariant feature-based Face Detection. A brief discussion of the state-of-art in the fields of Object and Face Detection is followed by a description of the algorithm, which has been named Implicit Spatial Inference (ISI). The Bayesian framework underlying this algorithm is discussed, including a technique for obtaining a likelihood from the confidence output of a typical point-wise feature-based machine learning classifier, and a method of imposing a Markov Random Field (MRF) on a sparse set of feature points using Delaunay triangulation. A Face Detection task is used to test the algorithm for detection, localization and segmentation accuracy and scale and rotation invariance.

### Chapter 4: On the Stylization of Visual Media

This Chapter presents a review of state-of-the-art algorithms for the artistic stylization of visual media (i.e. images and videos). This includes a description of sub-topics in the field of Non-Photorealistic Rendering (NPR) such as cartoonization, Stroke-Based Rendering (SBR), semantic stylization, motion summarization and expression.

### Chapter 5: Graph Cut-Based Skin Detection

This Chapter presents a brief review of techniques in the field of Skin Detection, followed by the description of a novel algorithm for Skin Detection in images. The underlying Bayesian framework of this novel skin detector encompasses probabilistic skin color modeling in the RGB color space and Graph Cut-based spatial smoothing.

### Chapter 6: Skin-Aware Stylization of Video Portraits

This Chapter presents a novel Non-Photorealistic/Stroke-based Rendering (NPR/SBR) framework for the skin-aware stylization of video portraits of people. The framework combines elements of cartoonization, motion depiction, spatio-temporal Skin and Edge Detection and content-based stylization. A description of the framework encompasses novel techniques in SBR for brush stroke anchor point distribution and motion expression, spatio-temporal color-sampling, and for the exploitation of occlusion detection in dealing with the well-known issues of gaps and redundancy in motion-compensated brush stroke animation. The framework is used to stylize a number of sequences containing head shots of people, and the visual results are discussed.

### Chapter 7: Conclusion

The final Chapter assesses the contributions of this thesis, discusses some issues in CBMP and directions for future work.

## 1.3 Contributions of this Thesis

The novel aspects of this thesis are summarized as follows

- A new example-based STS algorithm.
- The idea of estimating the parameters inherent to this - and possibly other example-based STS algorithms - through audio content analysis.
- A new invariant Face Detection algorithm. The proposed face detector is fully invariant to in-plane rotation, relative feature geometry and partial occlusion, and partially invariant to scale, illumination and pose (i.e. out-of-plane rotation).

- A probabilistic technique for inferring implicit geometric context in sparse local feature-based classification for the purpose of Object/Face Detection.
- A technique for obtaining a probabilistic likelihood from the real-valued confidence output of a typical feature-based machine learning classifier.
- The implementation of ISI on a sparse set of feature points by imposing a MRF via Delaunay triangulation.
- Ideas for producing a rough object segmentation from the final result of sparse local feature-based classification.
- A new probabilistic Skin Detection algorithm with Graph Cut-based spatial smoothing.
- A new skin-aware framework for the semantic stylization of video portraits.
- The merging of content-based stylization, cartoonization, SBR, motion summarization and other NPR effects in one video stylization framework.
- Several novel techniques in SBR including a new probabilistic algorithm for brush stroke anchor point distribution, spatio-temporal color sampling, and the detection of video occlusion for the mitigation of some well-known problems in motion-compensated brush stroke animation.

## 1.4 Publications

Portions of the work described in this thesis have appeared in the following publications

- "Wavelet-Based High Resolution Sound Texture Synthesis", by Deirdre O'Regan and Anil Kokaram, in *Proceedings of the 31st International Conference of the Audio Engineering Society (AES: Hi-Res Audio '07): New Directions in High Resolution Audio*, paper number 17, Queen Mary University, London, UK, June 2007.
- "Multi-Resolution Sound Texture Synthesis using the Dual-Tree Complex Wavelet Transform", by Deirdre O'Regan and Anil Kokaram, in *Proceedings of the 15th European Signal Processing Conference (EUSIPCO '07)*, pages 350-354, Poznan, Poland, September 2007.
- "Implicit Spatial Inference with Sparse Local Features", by Deirdre O'Regan and Anil Kokaram, in *Proceedings of the 15th IEEE International Conference on Image Processing (ICIP '08)*, pages 2388-2391, San Diego, CA, USA, October 2008.

- "Skin-Aware Stylization of Video Portraits", by Deirdre O'Regan and Anil Kokaram, in *Proceedings of the 6th European IEEE Conference on Visual Media Production (CVMP '09)*, London, UK, November 2009.

# 2
# Example-Based Sound Texture Synthesis

Sound Texture Synthesis (STS) is useful for synthesizing ambient sounds for installations, computer games and home movies, damaged audio repair, compression and storage. This Chapter presents a novel example-based STS algorithm. A brief examination of some of the concepts in STS and the closely related field of Image Texture Synthesis (ITS) is followed by a description of the new algorithm. Inspired by a well known 2D example-based ITS algorithm and its and multi-resolution adaptation, this 1D audio interpretation is used to synthesize long, perceptually and statistically similar sound textures from much shorter real-world training examples including audio clips of crowd noise, a baby crying, speech and music. The process employs the Dual-Tree Complex Wavelet Transform (DT-CWT) to reduce computational burden without sacrificing spectral coherency in the synthesized audio. This example-based approach to STS produces plausible and interesting sound textures that are comparable to the results of another well-known wavelet-based STS in the field. Content-based analysis of the short training example is used to estimate some of the parameters inherent to the algorithm. Specifically, Beat Detection, and Shannon entropy analysis are used for this purpose.

## 2.1 Sound Texture Synthesis

Sound Texture Synthesis (STS) is the automatic synthesis of a long, dynamic *sound texture* that is perceptually similar to a shorter audio *training example*, as can be seen in Figure 2.1. The biggest challenge of STS is the achievement of an acoustically plausible sound

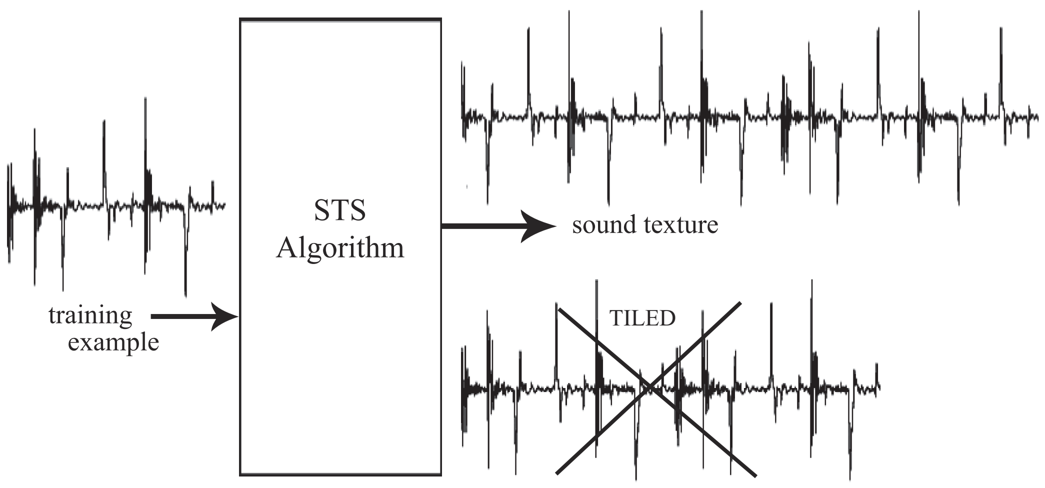


**Figure 2.1:** *The process of STS; the ideal algorithm takes a short audio training example input and produces a non-tiled sound texture output*

texture with an unpredictable temporal evolution. The latter property will be referred to hereafter as *variation*. Simple end-to-end repetition of the training example, or *tiling*, is easily detectable acoustically and should be avoided.

A variety of sound samples can be transformed into sound textures; natural (e.g. babbling water, crickets chirping), human (e.g. baby crying, speech snippets), musical (e.g. piano), and mechanical (e.g. road traffic). Some natural sounds could be considered *stochastic* in nature (e.g. heavy rainfall), whereas human speech and polyphonic music have specific, complex structures, and can be thought of as *quasi-periodic*. The process of STS is not as easily generalizable as Figure 2.1, since the spectral characteristics of each training example present a unique challenge.

Applications of STS include audio compression, ambient sound or music synthesis for computer games, installations and movies, re-synthesis of rare sounds (e.g. a rare bird call), and error correction or "hole-filling" in existing, damaged audio tracks.

Research in STS is growing in popularity alongside the related fields of Image Texture Synthesis (ITS) and Video Texture. Following the pixel-wise synthesis techniques reported often the field of ITS (e.g. (Efros & Leung, 1999) and (Wei & Levoy, 2000)), STS is taken to mean the *sample-wise* synthesis of a novel sound track that is much longer than the training example clip. This is subtly different to the idea of Audio Texture (AT) as defined in (Lu et al., 2004), which is concerned with analysis of the training example for location of *transition points* that divide the signal into a number of perceptually correlated segments. Audio Texture is then created by means of a continuous randomized playback of these segments. AT is like a *patch-based* alternative to the unit-based STS. The AT approach is common to the algorithms of (Lu et al., 2004; Hoskinson & Pai, 2007) and (Jehan, 2004).

STS is also distinct from pure computer music synthesis (Moorer, 1995), which involves the computer-generated composition of virtual, (usually) musical soundwaves without training from any real world audio example clip. Synthesis of complex real-world sounds like human speech is difficult using this technique, but it is useful for creating other-worldly and machine-like sounds such as the THX logo theme, 'Deep Note', composed by Dr. J. A. Moorer. For the interested reader, (Strobl et al., 2006) present a short review of a few different methods of sound and music synthesis, including STS, AT and pure computer music synthesis techniques.

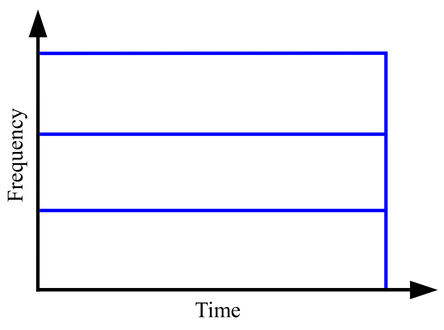
With regard to this body of research, the chosen approach of sample-wise STS has been inspired by the interesting results and challenges emerging from the field of ITS in recent years (Efros & Leung, 1999; Wei & Levoy, 2000), with particular regard to multi-resolution or wavelet-optimized ITS algorithms (Wei & Levoy, 2000; Gallagher & Kokaram, 2005). The STS algorithm of (Dubnov et al., 2002) has also been inspired by a multi-resolution ITS algorithm, and it is therefore considered most similar to the STS work presented in this Chapter. This algorithm of (Dubnov et al., 2002), and some ITS algorithms will be discussed later in Sections 2.3 and 2.4 respectively. To fully appreciate this discussion, however, it is necessary to understand the fundamentals of wavelet analysis, and a brief explanation of this topic will now be presented.

## 2.2 Wavelet Analysis

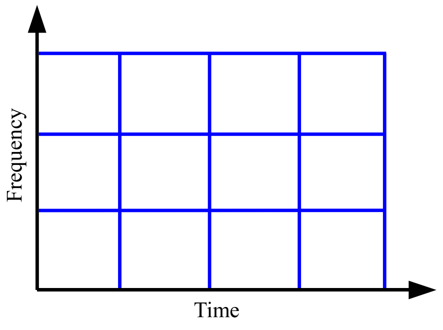
Wavelet analysis involves the decomposition of a signal into a number of different sub-signals representing the different levels of spectral resolution or *scales* present in the input signal over time, where scale is inversely proportional to frequency. It is distinct from the Fourier Transform and Short Term Fourier Transform (STFT) in that it results in a non-uniform decomposition of the time-frequency spectrum, as can be seen in Figure 2.2.

Wavelet analysis usually involves filtering the input signal with with a finite energy basis or *mother* wavelet function under various translations and dilations. The output *wavelet decomposition* is a useful time-scale breakdown of the signal into spectral *octaves*. For this reason, wavelet filtering is thought to bear similarity to that of the Human Auditory System (see (Kudumakis & Sander, 1993)), and it is conducive to the spectral analysis of non-stationary, real-world signals.

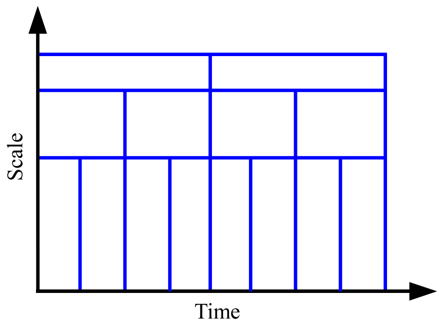
The concept of scale is similar to that of *granularity*. Small scale wavelet analysis reveals finer spectral granules in the signal, and vice versa. Notice that the high frequency (i.e. low scale) parts of the signal in Figure 2.2 (c) are represented with a finer time resolution and vice versa. This makes sense from an auditory perspective, since spectral activity in the high frequency bands of a signal is greater, by definition. Therefore, a finer time resolution is useful to capture this high frequency activity (i.e. smaller granules). A coarser time resolution is sufficient to represent the more more long term characteristics (i.e larger



(a) Fourier analysis



(b) STFT analysis



(c) Wavelet analysis

**Figure 2.2:** *Comparing the spectral decomposition of (a) the Fourier Transform, (b) the Short-term Fourier Transform (STFT), and (c) a typical wavelet transform.*

granules) of the low frequency bands, which evolve more slowly.

Wavelet transforms are increasingly popular in the field of Texture Synthesis. The DWT is used in (Hoskinson & Pai, 2007) for the location of transition points in AT (see Section 2.1 for explanation). (Hoskinson & Pai, 2007) describe an online demo applet[1] that can be used to experiment with different wavelets basis functions in the creation of a few ambient audio textures such as crickets chirping. The STS and ITS algorithms of (Dubnov et al., 2002) and (Gallagher & Kokaram, 2005) also make use of wavelet transforms for signal analysis, and these will now be discussed in the following two Sections.

## 2.3 The Wavelet-Based STS of Dubnov et al.

(Dubnov et al., 2002) assume that the short-term time-frequency characteristics of a typical stochastic sound texture can be learned statistically. In similarity to the well-known multi-resolution tree-structured ITS algorithm of (Wei & Levoy, 2000), (Dubnov et al., 2002) perform multi-resolution analysis of the example training clip with the Discrete Wavelet Transform (DWT), and the wavelet decomposition is used to build a tree-like statistical

[1](Hoskinson & Pai, 2007): `http://www.cs.ubc.ca/labs/lci/naturalgrains/`

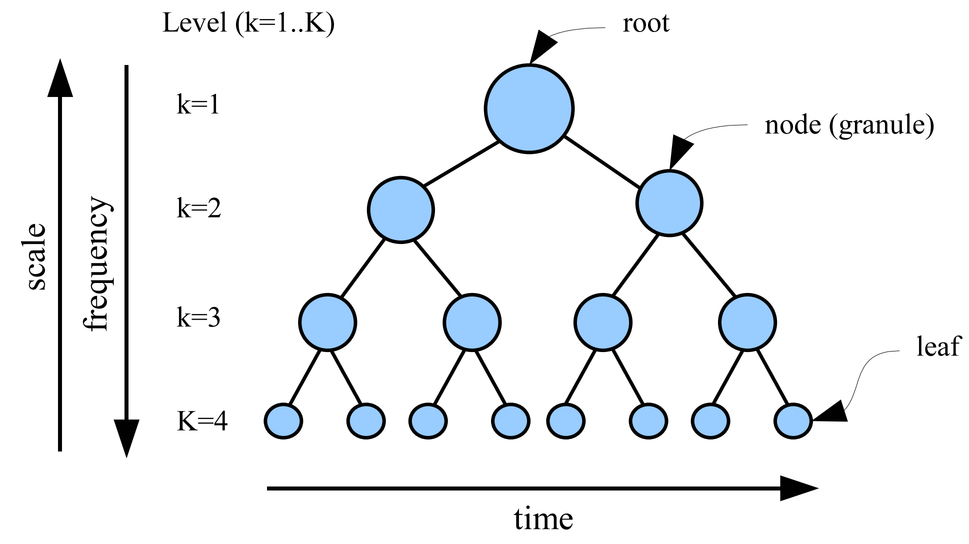


**Figure 2.3:** *The structure of the nodal wavelet tree used in the STS algorithm of (Dubnov et al., 2002).*

model of the training example clip, from which new samples of texture can be continuously drawn.

After wavelet analysis, the training example signal has been broken down into *dyadic* set of multi-resolution pieces or *granules*. These granules are arranged in an inverted tree-like structure with the largest scale (i.e. lowest frequency) granule at the root of the inverted tree, and granules of decreasing scale (i.e. increasing frequency) at descending levels all the way down to the node leaves, which are at the bottom of the tree. As can be seen in Figure 2.3, the structure is arranged such that neighboring nodes at the same tree level are temporally adjacent at the scale represented by that level, whereas parent and child nodes are related in scale-space. The wavelet tree, therefore, models the joint *time-scale* (i.e. time-frequency) dependencies of the training signal. In (Dubnov et al., 2002), a node's parents are referred to as its *ancestors*, whereas its same-level causal neighbors from the past are referred to as *predecessors*.

A novel sound texture is created by the synthesis of a new tree of depth, $K$, in a breath-first manner. The tree is first initialized with a root, and the same level $k = 1$ children as that of the training tree. In the synthesis of level $k = 2$, candidates for a new node are chosen by finding nodes in the training tree whose ancestors are similar to the ancestors of the node to be synthesized. This candidate set is then reduced to nodes whose five predecessors are similar to those of the node to be synthesized, and a new node is chosen randomly from it. For optimization purposes, the set of ancestors obtained for each candidate node can be inherited from parent to child to reduce complexity in the synthesis of the next level.

A variety of real-world audio training examples - including traffic noise, babbling water and a baby crying - are re-synthesized, and the resulting sound textures are quite interesting. These results are discussed in further detail in Section 2.8, but a general observation is that the algorithm seems limited to producing sound textures that are of the same, or of even shorter duration than the short example clips used for training. This could be attributed to the fact that the synthesized wavelet trees have only one root node, and are always synthesized to the same depth, $K$, as that of the training tree. (Dubnov et al., 2002) state that it is a trivial matter to extend the sound texture by means of producing a deeper wavelet tree in synthesis, but this has not been implemented. Both the training examples, and resulting sound texture files are obtainable at the URL associated with (Dubnov et al., 2002)[2].

## 2.4 On the Synthesis of Image Texture

Most ITS algorithms attempt to generate samples of synthesized *image texture* by modeling $p(I(\mathbf{X})|\underline{I}\Theta)$, or the likelihood of image intensity, $I(\mathbf{X})$, given some model parameters, $\underline{I}\Theta$. The well-known algorithm of (Efros & Leung, 1999) was the first in ITS to measure this likelihood directly from the image content, assuming a Markov Random Field (MRF) with a discrete sampling window.

This concept is known as *example-based* synthesis, and the idea is derived from a statistical technique first used by Shannon to generate English-like text letter by letter. Using a large example of training text, Shannon modeled language as a generalized Markov Chain, enabling an estimation of the Probability Distribution Function (PDF) for synthesizing new letters by measuring from the existing data. As discussed in (Efros & Leung, 1999), image texture can be pixel-wise synthesized using this technique adapted to image space. The concept of MRF is explained in (Efros & Leung, 1999) as the assumption that

> *The probability distribution of brightness values for a pixel given the brightness [i.e. intensity] values of its spatial neighborhood is [assumed to be] independent of the rest of the image.*

Unfortunately the window-matching form of this algorithm requires exhaustive searching that is prohibitively slow and computationally inefficient. (Gallagher & Kokaram, 2005) uses wavelet analysis to optimize the ITS algorithm of (Efros & Leung, 1999), reducing the computation burden involved. The Dual-Tree Complex Wavelet Transform (DT-CWT) (Kingsbury, 2001) is used for this task, and the texture synthesis occurs in wavelet space, as a *multi-resolution* process. The resulting multi-resolution example-based ITS algorithm produces image textures that are as plausible as those associated with (Efros & Leung, 1999), but with a fraction of the computational burden. Some of the image texture results

[2](Dubnov et al., 2002): `http://www.cs.huji.ac.il/labs/cglab/papers/texsyn/sound/`

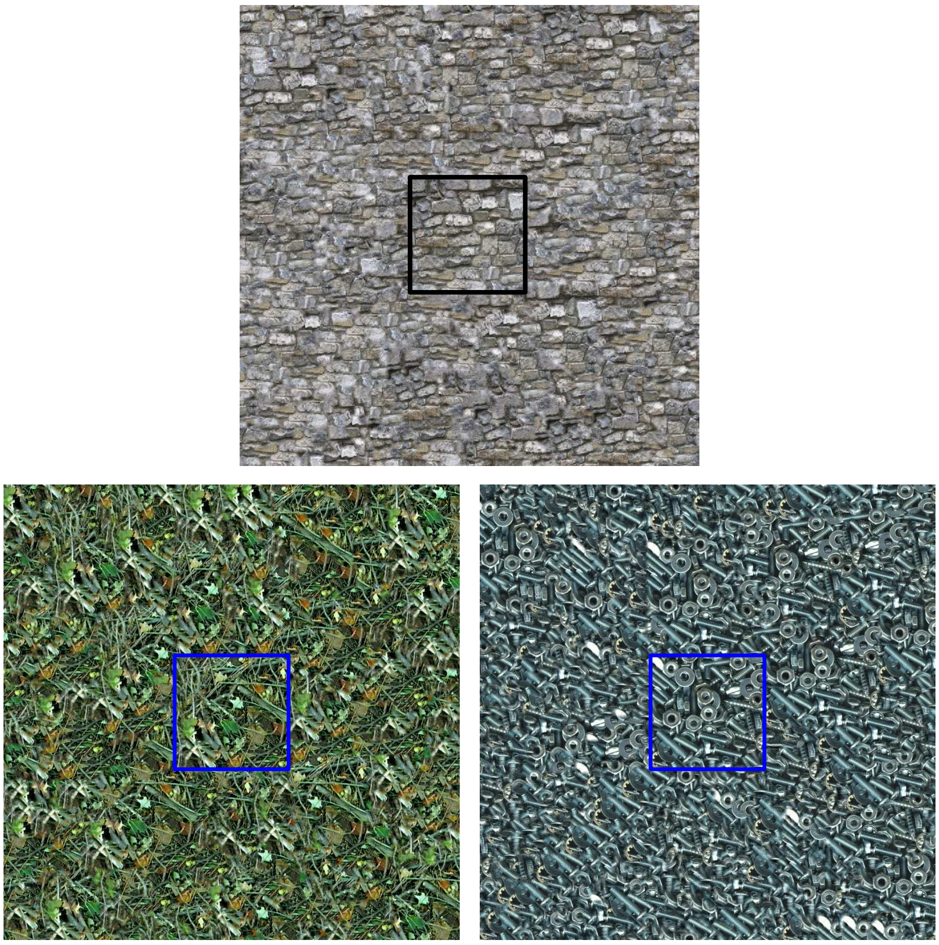


**Figure 2.4:** *Examples of image textures generated by the multi-resolution example-based ITS algorithm of (Gallagher & Kokaram, 2005). Images courtesy of (Gallagher, 2006).*

of ITS algorithm of (Gallagher & Kokaram, 2005) can be seen in Figure 2.4. Here, the training example has been placed in the box middle, and the texture is grown outwards from it. The explicit details of this algorithm will not be discussed here, since it is the inspiration behind the example-based STS work that will be described later in this Chapter.

### 2.4.1 Thoughts on a Multi-Resolution Example-Based STS

Although image and audio are presented and perceived quite differently, it is proposed that example-based STS can be thought of as the 1D version of 2D example-based ITS, and that multi-resolution optimization can be implemented in both. A wavelet-optimized scheme in STS is certainly useful considering the high sampling rate of sound files (e.g. $44kHz$ or $44,100$ samples per second). Sample-wise synthesis in the sound domain, therefore, is even more impractical than in image space.

Hence the example-based STS algorithm that has been developed here is like a version of

(Gallagher & Kokaram, 2005) example-based ITS, adapted specifically for the paradigm of audio. The multi-resolution form of this STS algorithm will be discussed later in Section 2.7, in terms of wavelet analysis with the DT-CWT. It is first necessary, however, to understand the 1D case of (Efros & Leung, 1999) example-based texture synthesis algorithm in single resolution.

## 2.5 Single Resolution Example-Based Sound Texture Synthesis

Suppose that our unit of synthesis, $y_s$, is a *single sample* of a body of sound texture. Let $\mathbf{Y}_s$ be the entire sound texture with $N$ samples to be synthesized from $\mathbf{Y}_e$, where $\mathbf{Y}_e$ is a shorter audio training example of $n$ samples. It is assumed that $\mathbf{Y}_e$ is long enough to approximate the statistical distribution of the underlying, infinite sound texture from which both $\mathbf{Y}_e$ and $\mathbf{Y}_s$ are derived. The empty container for $\mathbf{Y}_s$ is initialized by copying a short series of samples, or *seed*, from $\mathbf{Y}_e$ to a region in $\mathbf{Y}_s$. To interpret the 2D ITS algorithm of (Efros & Leung, 1999) in terms of 1D STS literately, this region would be placed at the mid-point of $\mathbf{Y}_s$. This interpretation seems counter-intuitive for audio, however, and so the seed could easily be placed at the beginning of $\mathbf{Y}_s$, and therefore the audio will be *grown* from this seed along the temporal axis in the same direction as time.

Figure 2.5 demonstrates the sample-wise synthesis process. Proceeding from the boundary of the seed onwards in time, let $y_s \in \mathbf{Y}_s$ be the next sample to be synthesized, and let $w(y_s)$ denote the *neighborhood* of samples, encapsulated by a sampling window of temporal extent, $W_s$, centered on $y_s$. In Figure 2.5, $w(y_s)$ is shown as a rectangular window with heavy black outline.

To be able to synthesize $y_s$, it is necessary to create an approximation to the conditional probability distribution, $p(y_s|w(y_s))$, determining the likelihood of the amplitude of $y_s$ given the state of its neighboring samples. In Figure 2.5, the distribution that must be estimated is labeled as PDF for Probability Distribution Function. To create this PDF, a search is conducted to identify all of the neighborhoods in the training example clip, $\mathbf{Y}_e$, that are similar in appearance to that defined by $w(y_s)$.

Let $d(w(y_s), w(y_e))$ represent the *perceptual distance* between $w(y_s)$ and some $w(y_e)$, where $w(y_e)$ is a neighborhood of extent $W_s$ centered on some sample site $y_e$ in $\mathbf{Y}_e$. Suppose that the particular $w(y_e)$ that is *most similar* to $w(y_s)$ corresponds to $w_m = \mathrm{MIN}_{y_e \in \mathbf{Y}_e}(d(w(y_s), w(y_e))$. The candidate set, $\Omega(y_s)$, is constructed such that

$$\Omega(y_s) = \left\{ \begin{array}{l} y_e \in \mathbf{Y}_e, d(y_s) = d(w(y_s), w(y_e)) \\ : d(y_s) \leq (1+\epsilon) d(w(y_s), w_m) \end{array} \right\} \tag{2.1}$$

where the value of $\epsilon$ is some error metric that determines the number of candidates chosen to

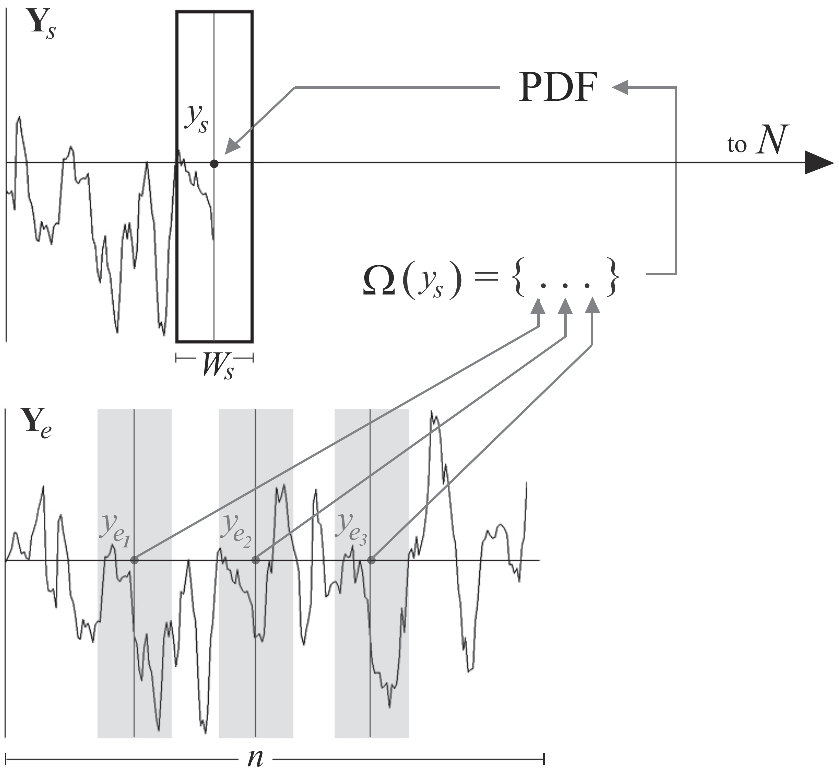


**Figure 2.5:** *The process of 1D sample-based STS, adapted from the 2D pixel-based ITS algorithm of (Efros & Leung, 1999). Estimating the PDF for $p(y_s|w(y_s))$.*

create the histogram estimation of the PDF of $p(y_s|w(y_s))$. According to (Efros & Leung, 1999), increasing the value of $\epsilon$ is likely to increase the element of variation in the final sound texture, $Y_s$. A value of $\epsilon > 0$ should be chosen to prevent the tiling and repetition of large chunks of $\mathbf{Y}_e$ in $\mathbf{Y}_s$. Choosing a value of $\epsilon = 0$ will mostly result in the process of sample-wise copying of $\mathbf{Y}_s$ from $\mathbf{Y}_e$, except when the value of $y_s$ has been sampled from the boundary of $\mathbf{Y}_e$, in which case variation might be introduced.

In Figure 2.5 depicting this process, $\Omega(y_s)$ is made up of just three candidates for $y_s$, denoted $y_{e_{1..3}}$ from $\mathbf{Y}_e$, and their perceptual distances from $w(y_s)$ are $d_{1..3}$. In keeping with (Efros & Leung, 1999), the distance $d(w(y_s), w(y_e))$ is defined as the Sum of Square Differences (SSD) as follows

$$d(w(y_s), w(y_e)) = \sum_{i=0}^{W_s} \frac{\mathbf{G}_i \mathbf{V}_i \sqrt{[w_i(y_s) - w_i(y_e)]^2}}{\sum\limits_{i=0}^{W_s} \mathbf{G}_i \mathbf{V}_i} \tag{2.2}$$

where $\mathbf{G}$ is a 1D Gaussian kernel of length $W_s$ and variance $\sigma = W_s/6.4$ and $\mathbf{V}$ is a binary vector that is non-zero where $y_s \in \mathbf{Y}_s$ has already been filled. The purposes of $\mathbf{G}$ is to emphasize local temporal coherency, and that of $\mathbf{V}$ is to enable "hole-filling" synthesis (i.e. to allow texture to be synthesized where there might be a small hole in the signal). When expressed as a histogram, $\Omega(y_s)$ is an approximation to the PDF representing $p(y_s|w(y_s))$. This histogram can be sampled randomly, yielding $y_s$. This probabilistic sample-wise synthesis process can be continued along the axis of time until all of the samples in $Y_s$ have been synthesized.

The extent of the sampling window, $W_s$, is a user-chosen parameter that defines the neighborhood of the implicit MRF, and it is assumed that the PDF is only valid if the extent of $W_s$ is sufficient to capture the underlying statistics of $\mathbf{Y}_e$. For best results, therefore, the user must choose a value of $W_s$ to capture the duration of the *longest repetitive temporal feature* in $\mathbf{Y}_e$. There is a corresponding, spatial case of this assumption in ITS (Efros & Leung, 1999). The unit of an audio sample is analogous to that of a pixel for image in the 2D case. Note that the length of the initializing seed must be at least ceil($W_s/2$) to ensure a valid synthesis of the first case of $y_s$.

Essentially an exhaustive window-matching algorithm, the computational burden of this process - whether it is used for STS or ITS - is very high. Furthermore, since digital audio is usually sampled at a much higher rate than digital images, the process of STS is extremely computationally inefficient. Drawing inspiration from the multi-resolution example-based ITS of (Gallagher & Kokaram, 2005), a multi-resolution version of this example-based STS has also been developed with the aim of reducing complexity. A brief discussion of Dual-Tree Complex Wavelet Transform (DT-CWT) is now needed, since it is used for wavelet analysis in the multi-resolution extension of example-based STS.

## 2.6 The Dual-Tree Complex Wavelet Transform

The Dual-Tree Complex Wavelet Transform (DT-CWT) uses a dual tree of wavelet filters to decompose the signal into multi-level wavelet coefficients. The structure of the wavelet decomposition is a *coarse-to-fine* representation of the spectrum granularity that is most useful for signal analysis and synthesis. At each level, $k$, the DT-CWT produces a band-pass complex signal of *detail* coefficients and a low-pass real signal that is passed on to the next level, $k+1$, for further decomposition. This process continues until as many wavelet levels as desired, $K$, are obtained.

The term *complex* refers to the fact that the DT-CWT produces complex wavelet coefficients. That is to say that the coefficients have two parts; a *real* and *imaginary* part, in the form of a complex number. It is both the dual-tree nature of the transform, and the Q-Shift structure of the wavelet filter banks that produce its complex coefficients. The internal filtering mechanisms of the DT-CWT is beyond the scope of this thesis, but the interested reader will find a more detailed explanation in (Kingsbury, 2001), and related publications. It is worth knowing, however, that the DT-CWT has the useful property of *perfect reconstruction*, meaning that no part of the original signal is lost in the inverse wavelet transform.

In contrast to the nodal wavelet tree described in Section 2.3, and the schematic of a wavelet transform seen in Figure 2.2 (c) of Section 2.2, the multi-resolution wavelet decomposition of the DT-CWT is conceptualized as an *inverse pyramidal structure* in this work, with large-scale coefficients at the bottom, the scales decreasing moving up the levels of the pyramid. Similarly to other wavelet transforms such as the DWT, however, the DT-CWT splits the signal into octaves. It is also *decimated*, and dyadic in structure such that each complex parent coefficient is associated with two complex children at the next wavelet level.

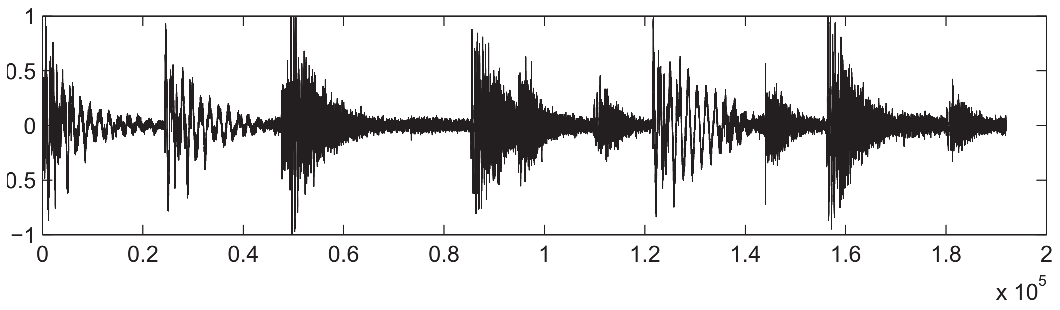
Unlike the DWT, the DT-CWT is *shift invariant.* Shift invariance means that if the sound samples are shifted within the audio signal undergoing wavelet decomposition, the complex wavelet coefficients undergo the equivalent translation in wavelet space. This property is due to the complex form of the coefficients.

Both the *magnitude* and *phase* of the complex coefficients are important metrics in the DT-CWT. Figure 2.6 (a), shows a short drum loop audio sample. After DT-CWT decomposition to level $K = 9$, Figure 2.6 (b) shows the $K = 9$ low-pass signal that is representative of the low-frequency information in the signal at that wavelet level. Figure 2.6 (c) is the magnitude (i.e. absolute) of the complex detail signal at level $K = 9$. In contrast, this captures useful high-frequency information, roughly localizing the onset of the *beats* in the drum loop. Figure 2.6 (d) shows the phase of the $K = 9$ complex signal, which is calculated as the inverse tangent of the imaginary divided by real parts of the coefficients of the complex detail signal, resulting in a range of $[-\pi, \pi]$ radians. It is important to note that pairs of complex wavelet coefficients can have similar magnitudes, but differing phases, and vice versa. This property is key to the shift invariance, which can be leveraged to promote temporal variation in the wavelet-based STS.

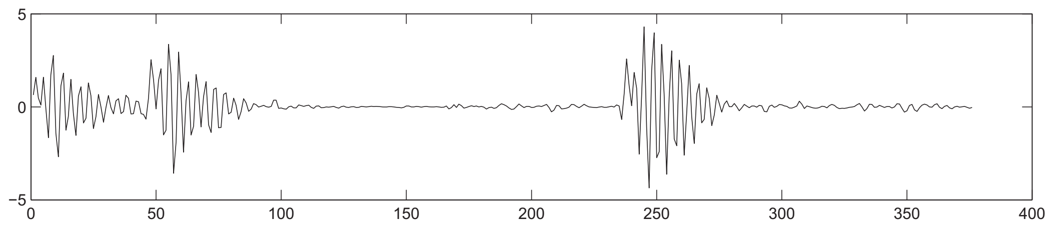
A reduced-complexity multi-resolution version of the example-based STS algorithm described in Section 2.5 can be formulated by means of a wavelet-optimized synthesis with the DT-CWT. Essentially, sound texture is first synthesized at the largest scale, which represents the *coarsest* level of detail in the signal. Other levels are textured by means of multi-resolution wavelet coefficient transfer. This technique will be discussed in the next Section.

## 2.7 Multi-Resolution Example-Based Sound Texture Synthesis

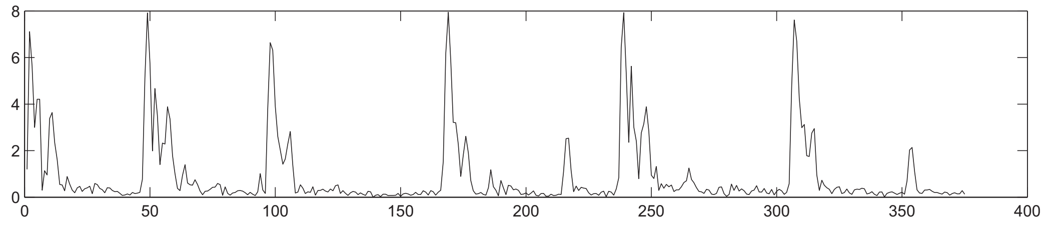
The process of a reduced-complexity, example-based STS is depicted in Figure 2.7, and proceeds as follows:



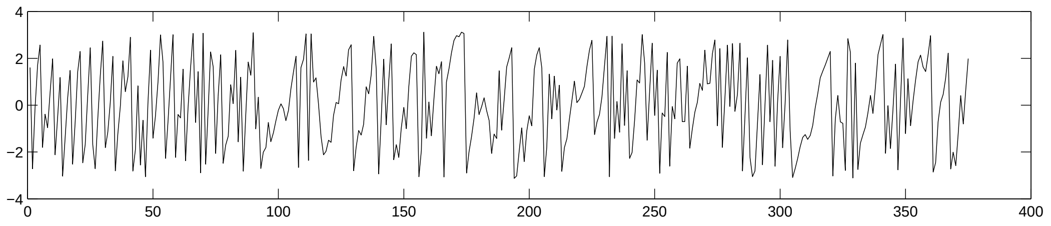
(a) Original drum loop



(b) Wavelet level $K = 9$ low-pass signal



(c) Magnitude of the level $K = 9$ complex detail signal



(d) Phase of the level $K = 9$ complex detail signal

**Figure 2.6:** *The DT-CWT of a drum loop to level $K = 9$; (a) original training example, (b) the final $K = 9$ low-pass signal, (c) magnitude of the final $K = 9$ complex detail signal, and (d) phase of the $K = 9$ complex detail signal. The horizontal axis represents sample number $[n]$. The DT-CWT is decimated so there are fewer samples at wavelet level $K = 9$ in the case of (b-d).*

- A $K$-level DT-CWT is performed on the short training example clip, $\mathbf{Y}_e$, resulting in an inverted pyramid of complex wavelet coefficients, $c_e^k \in \mathbf{C}_e^k$ with levels $k = 1..K$, where $\mathbf{C}_e^K$ is the final, coarsest detail signal. A final low-pass residue signal is also produced.

- A multi-level vector space, $\mathbf{C}_s^k$, is initialized to hold the DT-CWT space of the sound texture to be synthesized, $\mathbf{Y}_s$. If the target number of texture samples, $N$, is known in advance, then the extent of these containing vectors can be initialized prior to the synthesis stage. Note that the synthesis, however, is a continuous sample-wise process, such that samples of sound texture can be synthesized ad infinitum if desired. In an application where that would be required, the multi-level containment vectors could simply grow to hold each newly synthesized coefficient at run-time.

- A *seed* of samples from the DT-CWT decomposition of $\mathbf{Y}_e$ is placed in the containing vectors, $\mathbf{C}_s^k$, at each level. This operation is true to the dyadic parent-child structure described earlier in Section 2.6, such that the seed placed in $\mathbf{C}_s^{k-1}$ is twice the temporal extent of that placed in $\mathbf{C}_s^k$, and symmetrically above it. To adhere to (Efros & Leung, 1999) strictly, the multi-level seed can be placed in the center of any containing vectors, or it may be placed at the beginning of the containers at each level, as can be seen in Figure 2.7. The final level-$K$ low-pass signal array is also seeded appropriately.

- The sound texture coefficients, $c_s^k \in \mathbf{C}_s^k$, must now be synthesized. At the coarsest scale (i.e. level $k = K$) the single-resolution STS algorithm described in Section 2.5 is used to synthesize the detail coefficients, $c_s^K \in \mathbf{C}_s^K$. Again, the PDF of each $c_s^K$ is estimated, and a sample drawn from the candidate set of $\Omega(c_s^K)$. Since building this candidate set now involves dealing with complex numbers, Equation 2.2 is modified to the form

$$d(w(c_s^K), w(c_e^K)) = \sum_{i=0}^{W_s} \frac{\mathbf{G}_i \mathbf{V}_i \sqrt{[Re\{w_i(c_s^K) - w_i(c_e^K)\}]^2 + [Im\{w_i(c_s^K) - w_i(c_e^K)\}]^2}}{\sum_{i=0}^{W_s} \mathbf{G}_i \mathbf{V}_i} \tag{2.3}$$

where are $c_e^K$ and $c_s^K$ are the complex wavelet coefficients from the final levels, $\mathbf{C}_e^K$ and $\mathbf{C}_s^K$, of the wavelet decompositions of $\mathbf{Y}_e$ and $\mathbf{Y}_s$ respectively, and $W_s$, $\mathbf{G}$ and $\mathbf{V}$ are as previously described. In this case, however, $W_s$ is chosen to reflect the longest repetitive temporal feature of the level-$K$ complex detail signal, $\mathbf{C}_e^K$, on which the complex SSD matching process occurs. Here, the real and imaginary parts of the complex signal are separated as $Re$ and $Im$ respectively.

- When a $c_s^K$ match is placed in $\mathbf{C}_s^K$, the levels $k = K - 1..1$ are updated in keeping with the parent-child relationship. In other words, the two child coefficients of $c_s^K$ are obtained from those of $c_e^K$, and so on up the levels of the inverse pyramid. Therefore, when $\mathbf{C}_s^K$ is fully synthesized, so are all levels $k = K - 1..1$ above.

- Finally, the synthesized structure is inverse-transformed yielding the longer sound texture, $\mathbf{Y}_s$.

Equation 2.3 is similar to the SSD Equation 2.2 used for the case of single-resolution example-based STS (see Section 2.5), except that it has been modified so as to ensure that it is not simply the magnitude of the complex wavelet coefficients that are matched in the SSD, but the real and imaginary parts of the these complex coefficients. This is necessary to preserve the shift invariance property of the DT-CWT in synthesis, and it can have an effect on the statistical variation of the resulting sound texture.

It is also interesting to note that wavelet coefficient matching is not performed at levels above $k = K$. Instead, the multi-level $k = K - 1..1$ children of the $c_e^K$ match for each $c_s^K$ are transferred across to the wavelet space of $\mathbf{Y}_s$ where they are placed above $c_s^K$ in the same dyadic pattern as they appear in the wavelet space of $\mathbf{Y}_e$. To draw a parallel with the multi-resolution example-based ITS described in (Gallagher & Kokaram, 2005) and (Gallagher, 2006), this technique is known as the *Copy variant* of the algorithm, because the multi-resolution wavelet coefficients are "copied" to multiple, higher wavelet levels following complex SSD matching at the coarsest level of detail. Other variants of multi-resolution ITS described in (Gallagher, 2006) involve *coarse-to-fine* complex SSD matching at decreasing scale levels in wavelet space, and an optimization technique to reduce the complexity of a fully exhaustive search at each level. It was concluded by (Gallagher, 2006), however, that the Copy variant produces results that are comparable to the other multi-resolution synthesis techniques, but with far less computational complexity. A 1D version of the Copy variant is adopted for multi-resolution example-based STS here.

Again, $W_s$ is a user-defined parameter that is chosen to reflect the duration of the longest repetitive temporal feature, but this time with regard to the level-$K$ complex detail signal, $\mathbf{C}_e^K$, in the wavelet space of training example $\mathbf{Y}_e$. Therefore, the level of DT-CWT decomposition, $K$, and MRF window size $W_s$ are related. They are complementary variables in that smaller $W_s$ is needed to capture the longest repetitive temporal feature in $\mathbf{C}_e^K$ for larger $K$.

This multi-resolution example-based STS has a huge computational advantage over the single resolution version of the algorithm described in Section 2.5. The complexity reduction depends on the depth of DT-CWT decomposition, $K$, and corresponding reduction of the sampling neighborhood, $W_s$. Disregarding the complexity-reducing effect of a shorter $W_s$, the number of template-matching operations needed to create a sound texture is already reduced by $2^K$ (i.e. two to the power of $K$). Note that one template-matching (i.e.

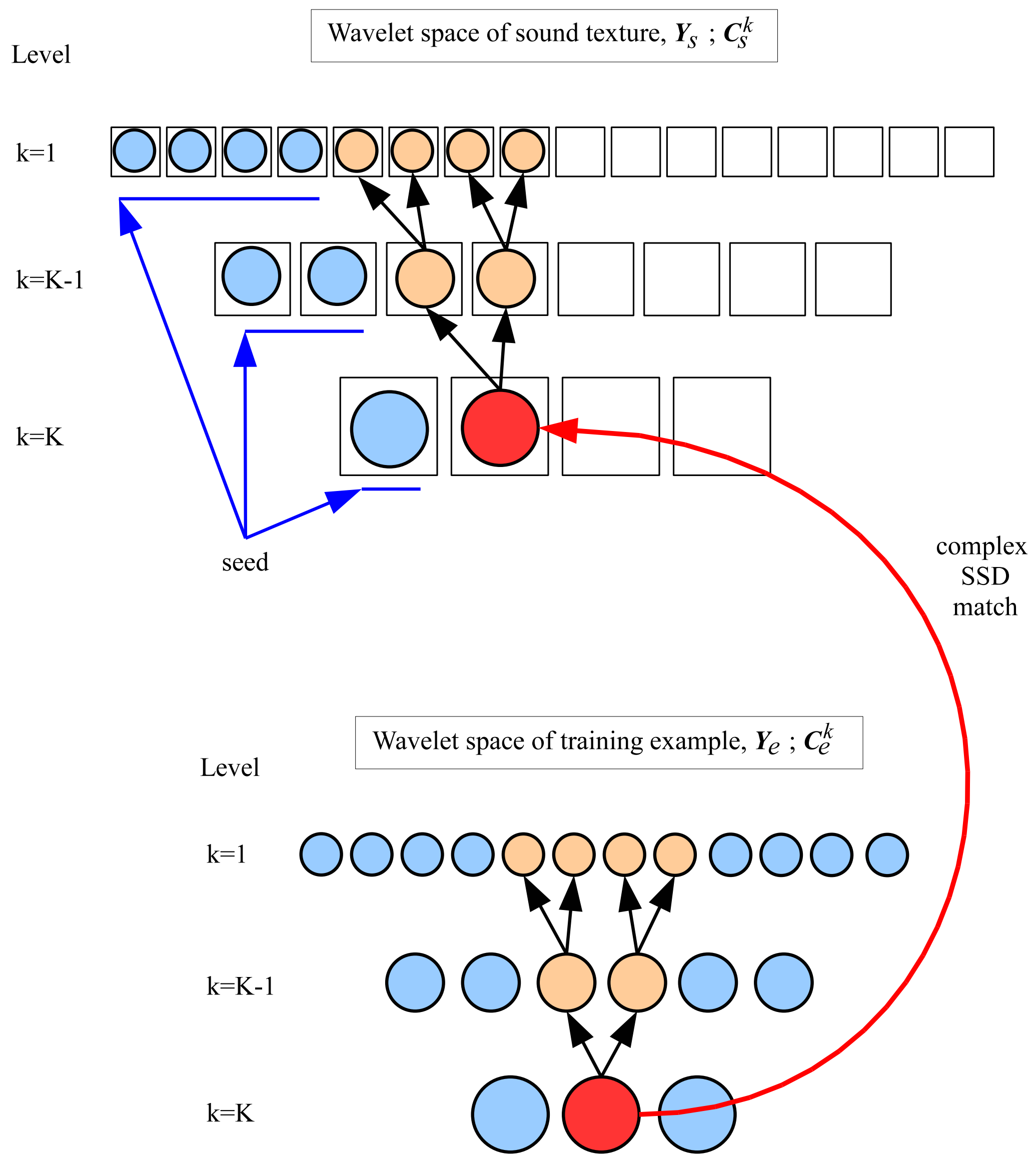

Wavelet space of sound texture, $Y_s$ ; $C_s^k$
Level
k=1
k=K-1
k=K
seed
complex SSD match
Wavelet space of training example, $Y_e$ ; $C_e^k$
Level
k=1
k=K-1
k=K

**Figure 2.7:** *Multi-resolution example-based STS; sound texture is grown temporally onwards from a seed placed in signal containers, $\mathbf{C}_s^k$, initialized to form the wavelet space of the resulting sound texture, $\mathbf{Y}_s$. When a complex wavelet coefficient from the coarsest level, $\mathbf{C}_e^K$, of the training example clip, $\mathbf{Y}_e$, is chosen by a probabilistic sampling process it is transfered to $\mathbf{C}_s^K$, along with its complex children from levels $k = K - 1..1$*

Euclidean distance) computation for each existing sample of the sound texture (including the seed) is needed to produce each new sequentially synthesized sample. With $44k$ samples per second, and the fact that the existing body of sound texture is always increasing by one sample for each sequential computation, this complexity reduction is quite significant especially in the production of a very long sound texture.

| | Training Example | $f_s[Hz]$ | $t_1[s]$ | Description | $t_{2_d}[s]$ |
|---|---|---|---|---|---|
| 1 | *drum loop* | 22$k$ | 3 | Near-periodic; percussion with occasional cymbal. | 5 |
| 2 | *baby crying* | 11$k$ | 13 | Quasi-periodic; baby crying, annoying. | 11 |
| 3 | *traffic jam* | 11$k$ | 22 | Event-on-background; traffic noise with "yelling" and "honking" events. | 35 |
| 4 | *shore splashing* | 11$k$ | 18 | Event-on-background; seawater ebbing with "splashing" event. | 11 |
| 5 | *formula 1* | 11$k$ | 16 | Event-on-background; F1 cars accelerating with "gear-shifting" event. | 23 |

**Figure 2.8:** *Description of training examples used, and resulting sound texture durations, $t_{2_d}$, achieved by the wavelet-based STS of (Dubnov et al., 2002). Here $t_1$ are the time durations of the original training example clips. The training set and results are obtainable at the URL associated with (Dubnov et al., 2002), and also included on the DVD accompanying this thesis.*

This wavelet-based STS algorithm is distinct from that of (Dubnov et al., 2002) in that only temporal predecessors are involved in wavelet coefficient matching, and not hierarchical ancestors (see Section 2.3 for explanation). It is therefore interesting to compare the sound texture results of this multi-resolution STS algorithm with those of (Dubnov et al., 2002) for benchmarking purposes. This comparison will be described in the next Section.

## 2.8 Dubnov et al. Training Examples and Sound Textures

As discussed in Section 2.3, a number of sound textures were synthesized by the wavelet-based STS of (Dubnov et al., 2002) from a selection of real-world training example clips. These training example clips, their sampling rates, $f_s$, original time durations, $t_1$, and the durations of the sound textures created by (Dubnov et al., 2002), $t_{2_d}$, are listed in Table 2.8. It is clear that some of the sound textures are actually *shorter* in duration than their corresponding training example clips, which is strange considering the goal of creating a sound texture of extended duration. Other goals in STS are the avoidance or undesirable tiling of the training clip, looping, and uncomfortable glitch-like artifacts known as *clicks*.

The (Dubnov et al., 2002) textures of training examples 1, *drum loop*, and 2, *baby crying*, are interesting, but they contain a clicks, glitches and somewhat erratic variation. The texture of 3, *traffic jam*, has short, repetitive loops, separated with abrupt silences. The latter effect could occur because a period of recorded silence at the start of the training example clip. Although generally plausible, the domineering presence of long-term car-horn "honking" is not reflected well in the synthesized texture.

(Dubnov et al., 2002) note that their algorithm cannot achieve a plausible re-synthesis of sound signals with both long-term patterns, and sporadic short-term *events*. These sound phenomena will be referred to as *event-on-background* signals hereafter. Example 4, *shore splashing*, is in this category. It sounds like the relaxed tempo of water ebbing on a shore, punctuated with stochastic-sounding "splashing" events. (Dubnov et al., 2002) acknowledge an unrealistic "nervous splashing activity" in their resulting sound texture. Example 5, *formula 1*, presents a similar problem. (Dubnov et al., 2002) state that their sound texture does not reflect a good balance between the so-called "long sound phenomena" of gradual engine acceleration and short-term "gear-shifting activity" in this training example. Perhaps this effect is due to the limiting technique of matching a fixed number of multi-level predecessors (i.e. five) when sampling wavelet coefficients from the training example for synthesis (see Section 2.3 for explanation).

Both the training example set and sound texture results produced by this algorithm are obtainable at the URL associated with (Dubnov et al., 2002)[3], and also included on the DVD accompanying this thesis.

### 2.8.1 Comparative Results

Table 2.9 summarizes an experiment in which the example-based STS algorithm presented in Section 2.7 of this Chapter has been used to produce sound textures from the training example clips used by (Dubnov et al., 2002). These training examples are labeled as $\mathbf{Y}_e$ hereafter in keeping with the notation of Section 2.7. The values for parameters $K$, $W_s$, and $\epsilon$ that produced the best sound texture results are listed, along with the durations, $t_{2_o}$, of the best sound textures produced. Both the training example files, and sound texture results are included on the accompanying DVD. Again, it is worth noting that there is no theoretical limit to the number of sound texture samples that can be produced by this algorithm, and the duration of the resultant sound texture is therefore arbitrarily chosen. In this case, the values of $t_{2_o}$ have been chosen to be many times longer than the durations, $t_{2_d}$ of the sound textures produced (Dubnov et al., 2002) (see Table 2.8).

Extensive trial-and-error experimentation was carried out until the best parameter combinations were found, which is a fairly time-consuming and arduous task. The third column in Table 2.9 states whether the multi-level seed was placed at the start or in the center of the signal containers initialized for the resulting sound texture, as well as the exact samples of the training clip, and time duration constituting the seed. Note that the number of samples, $[n^K]$, is actually the number of wavelet coefficients that were placed in the level-$K$ complex detail signal, $\mathbf{C}_s^K$, in the wavelet space of the sound texture, $\mathbf{Y}_s$. The number of wavelet samples placed in $\mathbf{C}_s^{K-1..k}$ above are true to the dyadic relationship of the DT-CWT, as previously discussed in Section 2.6. The duration in seconds, $[s]$, however, reflects the approximate time duration of the multi-level seed in the final sound texture, $\mathbf{Y}_s$.

[3](Dubnov et al., 2002): `http://www.cs.huji.ac.il/labs/cglab/papers/texsyn/sound/`

| $\mathbf{Y}_e$ | Training Example | Seed (samples$[n^K]$, duration$[s]$) | $K$ | $W_s$ | $\epsilon$ | $t_{2_o}[s]$ |
|---|---|---|---|---|---|---|
| 1 | *drum loop* | placed at start $(1..125, 1.5)$ | 8 | 41 | 0.3 | 63 |
| 2 | *baby crying* | placed at start $(100..170, 0.4)$ | 6 | 23 | 0.1 | 73 |
| 3 | *traffic jam* | placed in center $(300..311, 0.3)$ | 8 | 5 | 0.01 | 70 |
| 4 | *shore splashing* | placed at start $(400..550, 3.5)$ | 8 | 51 | 0.1 | 74 |
| 5 | *formula 1* | placed start $(200..300, 2.3)$ | 8 | 21 | 0.1 | 75 |

**Figure 2.9:** *Parameter values and time durations, $t_{2_o}$, of the best sound textures achieved by testing the new example-based STS algorithm on the training example clips used in the similar work of (Dubnov et al., 2002). The Seed column records the placement of the seed, number of samples, $[n^K]$, placed at level $K$ in wavelet space, and the equivalent time duration in ordinary sample space, $[s]$. Recall that there is no theoretical constraint on the duration of sound texture that can be produced by this algorithm, and therefore the values for $t_{2_o}$ have been arbitrarily chosen here. The resulting sound texture files are included on the DVD accompanying this thesis.*

It is interesting to note that the depth of DT-CWT wavelet decomposition is fairly high when compared with a value of $K = 3$ that is always used in the example-based ITS algorithm of (Gallagher, 2006). This is intuitive, however, since the sampling rate of digital audio is relatively much higher than that of digital images. The sound textures that were produced in this experiment are generally plausible, longer in duration than those of (Dubnov et al., 2002), and sound smooth, varied, and not tiled.

The best sound texture of $\mathbf{Y}_e$ 1, *drum loop*, is varied and interesting, with cymbals appearing pseudo-randomly in time. The waveforms of the original training example and $11s$ of the best sound texture are compared in Figures 2.10 (top right) and (bottom) respectively. Visually, the temporal envelope of the sound texture is reminiscent of that of the training example, but it is of much longer duration and well varied. Interestingly, the best performing value for $W_s$ appears to roughly correspond to the *tempo* of the piece at $K = 8$ of the DT-CWT, as can be seen in Figure 2.10 (top left).

A spectrogram (i.e. STFT) of the $13s$ training example, $\mathbf{Y}_e$ 2, *baby crying*, is shown in Figure 2.11 (top left), and the $11s$ sound texture of (Dubnov et al., 2002) is shown in Figure 2.11 (top right). The first $30s$ of the best sound texture generated by the new example-based STS algorithm presented here is shown in Figure 2.11 (bottom). The spectral energy of the latter looks natural with regard to the training example clip, and demonstrates good variation. The texture sounds plausible and varied, with very few audible clicks. It is clearly longer, and its spectral envelope looks somewhat more similar to the training example than the sound texture of (Dubnov et al., 2002). All of the spectrograms in Figure 2.11 were created with identical parameters.

A slight modification was necessary in creating plausible sound textures from $\mathbf{Y}_e$ 3, $\mathbf{Y}_e$

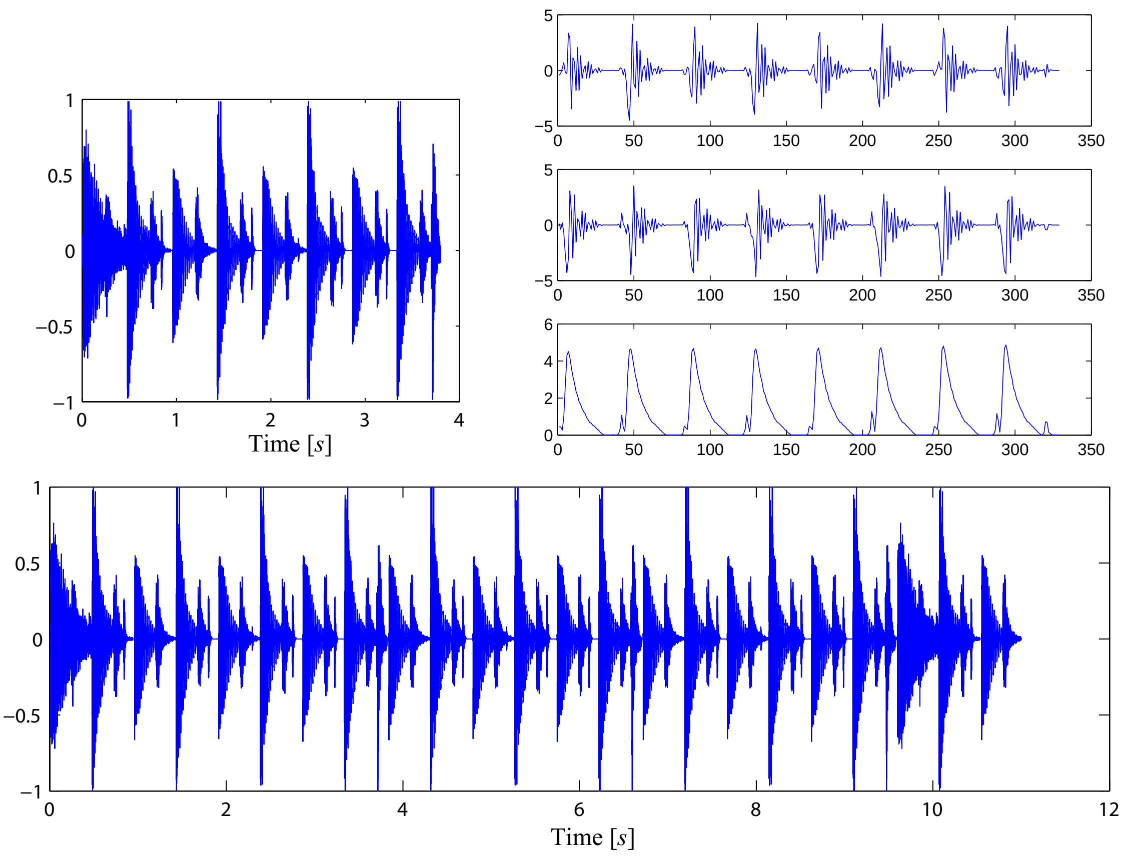


**Figure 2.10:** *Synthesized drum loop texture; (top left) the 3$s$ training example,* $\mathbf{Y}_e$ 1*, (top right) the real, imaginary, and absolute (t-b) values of the detail coefficients at* $K = 8$ *of the DT-CWT of* $\mathbf{Y}_e$ 1*, and (bottom)* 11$s$ *of the sound texture generated with a* 1.5$s$ *seed,* $K = 8$*,* $W_s = 41$ *and* $\epsilon = 0.1$*.*

4 and $\mathbf{Y}_e$ 5. Specifically, the beginnings and/or ends of these training example clips needed to be cropped to remove small boundaries of recorded silence in the files. It was found that if the silence boundaries are left in $\mathbf{Y}_e$, their presence seems to trigger end-to-end tiling of the training example clip in the process of example-based STS if the value of the error metric, $\epsilon$, is not large enough. However, by increasing the value of $\epsilon$, quality is sacrificed in the rest of the sound texture. Some example sound textures resulting from leaving the clips unmodified (i.e. not cropping the silence at the beginning and end) are included on the DVD accompanying this thesis, their parameters listed in Appendix A.

The sound texture results from the modified training examples are satisfactory, however, in that they are both representative of the general ambiance of their training example clips, and also seem to reflect the quasi-periodic nature of event-on-background signals. Figure 2.12 (top) shows training example $\mathbf{Y}_e$ 4, *shore splashing*. Figure 2.12 (center) is the sound texture result of (Dubnov et al., 2002), which was described in their paper as overly

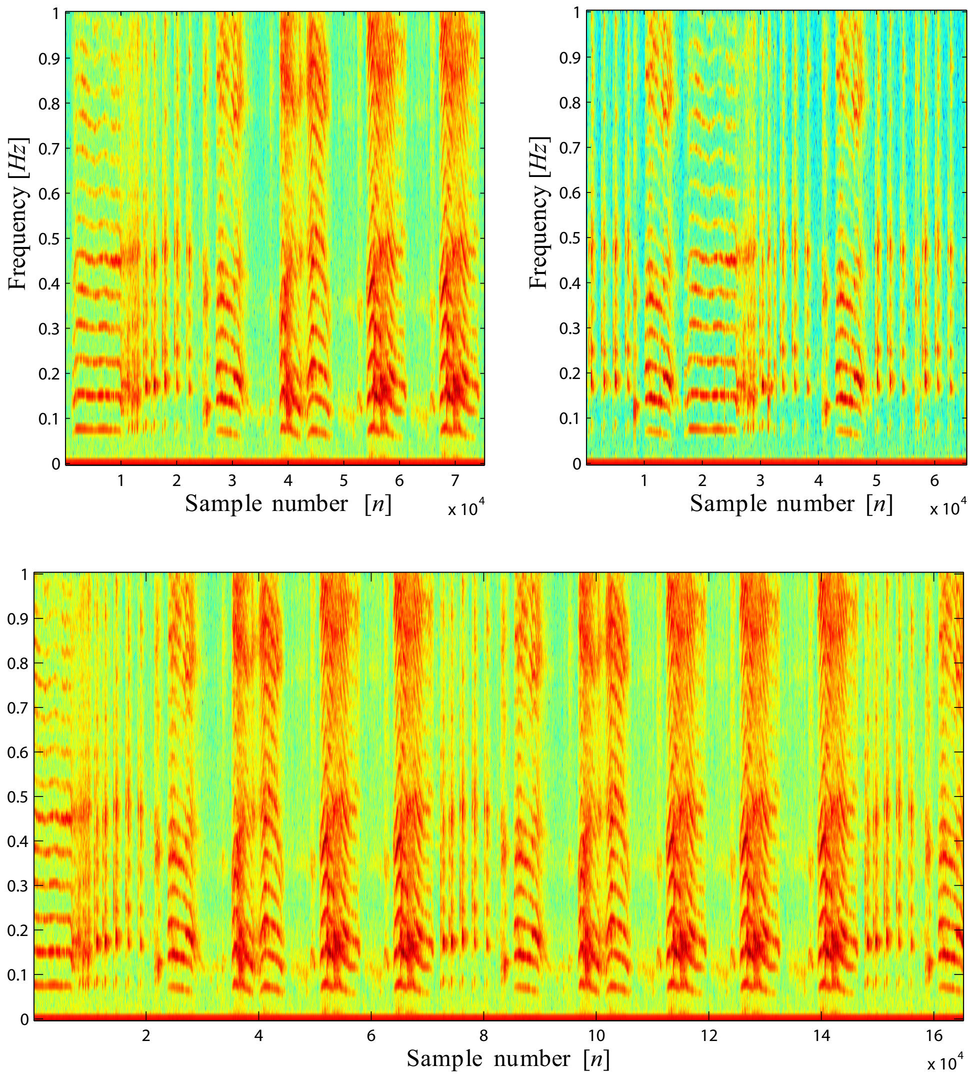


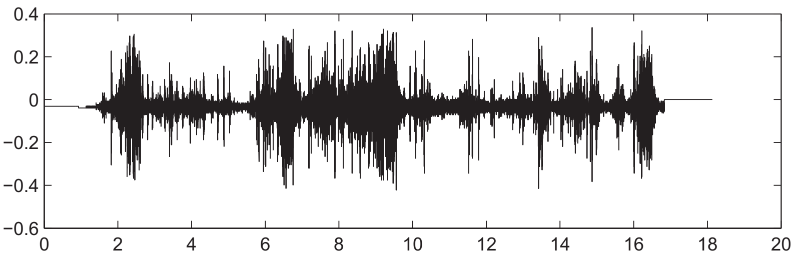
**Figure 2.11:** *Synthesized baby crying texture; (top left) spectrogram of the* $13s$ *training example,* $\mathbf{Y}_e$ 2, *(top right) the* $11s$ *sound texture of (Dubnov et al., 2002), and (bottom)* $30s$ *of the sound texture generated by the new example-based STS algorithm with a* $0.4s$ *seed,* $K = 6$, $W_s = 23$ *and* $\epsilon = 0.1$.

---

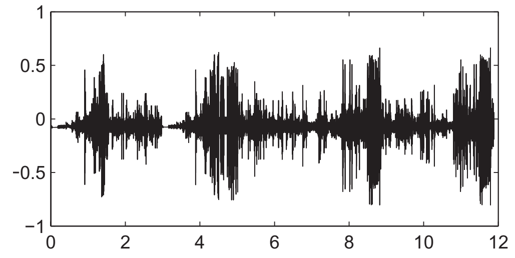
“nervous”. Figure 2.12 (bottom) shows $25s$ of the best sound texture produced by the new example-based STS algorithm, which seems more reflective of the ambiance of the training example clip than the sound texture of (Dubnov et al., 2002).

### 2.8.2 Subjective Listening Test

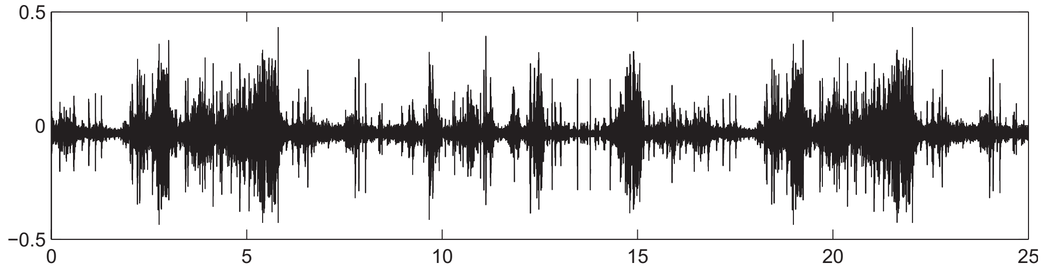
A simple set of perceptual experiments were undertaken in order to compare the new example-based STS algorithm presented here with that of (Dubnov et al., 2002). Five participants were firstly briefed on the concept of a sound texture. It was explained that



(a) Training example $\mathbf{Y}_e$ 4, *shore splashing*



(b) Sound texture generated by (Dubnov et al., 2002)



(c) Sound texture generated by multi-resolution STS

**Figure 2.12:** *Synthesized shore texture; (a) training example* $\mathbf{Y}_e$ 4*, (b) the sound texture generated by the algorithm of (Dubnov et al., 2002), and (c) the sound texture generated by the new example-based STS algorithm with a* $3.5s$ *seed,* $K = 8$*,* $W_s = 51$ *and* $\epsilon = 0.1$*. Time is measured on the x-axis in seconds* $[s]$*.*

*Sound Texture Synthesis (STS) involves the synthesis of a sound texture from a short audio training example clip, such that it sounds natural and varied with regard to, and may be of longer duration than the training example.*

Audio clips were played over Genelec 1029A stereo speakers arranged to face the participants. It was explained to participants that they would hear a short training example clip, followed by two distinct sound textures **A** and **B**. These textures were the results of the two STS algorithms under test. Participants were asked to choose one of the two sound textures with regard to each of the following questions

- **Q1** Which is more *natural* with regard to the training example? A or B?
- **Q2** Which is more *varied*? A or B?

| Training example | **A/B** | **Q1: X/Z** | **Q2: X/Z** | **Q3: X/Z** |
|---|---|---|---|---|
| *shore splashing* | X/Z | 0/5 | 0/5 | 0/5 |
| *baby crying* | Z/X | 0/5 | 1/4 | 1/4 |
| *drum loop* | Z/X | 0/5 | 1/4 | 0/5 |
| *traffic jam* | Z/X | 0/5 | 1/4 | 0/5 |
| *formula 1* | X/Z | 0/4 | 2/4 | 0/5 |

**Figure 2.13:** *The results of a subjective listening test comparing the STS algorithm of (Dubnov et al., 2002),* **X**, *to the new example-based STS algorithm presented here,* **Z**, *by asking five participants three questions* **Q1**, **Q2** *and* **Q3** *gaging which of the two sound textures is preferable. Here, the second column,* **A/B**, *lists the order in which the tracks from each algorithm were presented to the listeners (i.e. first played, slash, second played), but the other columns always list the votes for algorithm* **X** *versus (i.e. slash) algorithm* **Z** *in that order. The sum of the votes in some columns of the final row do not add up to five. This anomaly is explained in Section 2.8.2.*

- **Q3** Which has better *sound quality*? A or B?

In these kinds of listening tests the challenge is to guide the listeners to pay attention to the particular sound qualities being examined. Therefore, the meaning of the colloquial terms in the questions were expanded as follows:

- **Natural**: Sounds like, or has the same "ambiance" and fullness of the spectrum with regard to the training example, contains both long-term and short-term sound "events".
- **Varied**: Does not sound like the training example clip on a loop or *tiled*.
- **Sound quality**: There are few "clicks" or "glitches".

An example of a miscellaneous drum loop training example was played on a loop to demonstrate the concept of *tiled*. Both the order of the five training example clips, and the assignment of A or B to the sound textures of the two STS algorithms were randomized. The participants were allowed to request to hear the training example, sound textures A and/or B as often as required to make a decision, but were not allowed to discuss their deliberations or choices with other participants.

The results of this test are listed in Table 2.13. Each row is representative of a different training example clip and its resultant sound textures (e.g. the sound texture results of *shore splashing* are evaluated in the first row). The sound texture result of (Dubnov et al., 2002) is labeled as **X**, and that of the new example-based STS algorithm presented here is labeled as **Z**.

The play order of the two sound textures being compared are separated by a slash (i.e. A/B). In the first row, for example, the column with heading A/B lists the play order as X/Z indicating that sound texture X was played before sound texture Z in this case. The numbers in each column count the number of participants who chose the respective sound texture in response to a particular question, where Q(1-3) are the three questions previously defined. For the results of *shore splashing* in the first row, in response to Q1, “Which is more natural with regard to the training example?”, 0 listeners preferred sound texture X while 5 preferred sound texture Z.

Overall, participants chose the new example-based STS algorithm presented here over the previous work of (Dubnov et al., 2002). One of the responses to Q1 in row five is not registered because one participant circled both A *and* B, presumably because it was felt that neither sound texture was preferable with regard to Q1. One particular participant often answered differently in Q2, but in discussions after the test it transpired that this listener was unsure about the explanation of *variation* as given at the start of the test, deciding that variation meant randomness literally at the sample-level; in other words a very chaotic signal. This outlier serves to highlight how difficult it is to quantitatively assess audio results in terms of subjective phenomena. This is because it is ultimately not possible to equate listening properties in detail between different listeners.

In post-test discussions, participants also revealed that they had been somewhat confused by the obvious differences in duration between the two sound textures of each training example. As previously mentioned, some of the sound textures produced by the algorithm of (Dubnov et al., 2002) are so short that they are actually shorter than their corresponding training example clips, whereas those produced by this algorithm are of much longer duration. In their paper, (Dubnov et al., 2002) do not define a sound texture as something that is of longer duration than a training example clip, but given the well known applications of sound texture discussed at the beginning of this Chapter, it is intuitive that extended duration should be a goal in STS. This is another example, however, of the difficulty of quantitatively assessing results and concepts in audio processing.

However, the test performed here does give good evidence that the sound textures produced by this algorithm are plausible and varied with regard to the training example clips, of good quality, and ultimately usable in a media applications. Furthermore, it seems that this STS algorithm is capable of generating natural sound textures from all kinds of training examples with a variety of different spectral profiles. It is capable of dealing with both quasi-periodic and event-on-background training examples which have both long-term, noise-like and short-term, high frequency characteristics in their spectral profiles.

The best sound textures produced by the wavelet-based STS algorithm of (Dubnov et al., 2002), and also those produced by the new STS algorithm presented in this Chapter, are included on the accompanying DVD. Some alternative sound textures of $\mathbf{Y}_e$ $(1-5)$ produced by the latter - not the best results, but still interesting - are also included on the accompanying DVD, their parameters listed in Appendix A.

### 2.8.3 Observations Concerning Parameters

As previously mentioned, the best parameters for each sound texture were found by virtue of trial-and-error. It soon becomes obvious that choosing too small a value for the extent of the sampling window, $W_s$, results in the looping of particular small sections of the training example within the sound texture, whereas choosing too large a value causes the phenomenon of tiling. This is because $W_s$ must be chosen carefully to ensure a valid PDF for each sample with regard to the multi-resolution characteristics of the training example. Choosing too large a value of DT-CWT decomposition level, $K$, can also result in looping or tiling because the final complex wavelet band $\mathbf{C}_e^K$ becomes too coarse for a valid PDF. Furthermore, the values of $W_s$ and $K$ are correlated in that the deeper the level of wavelet decomposition, $K$, the smaller a value of $W_s$ needed to produce an adequate sound texture. Finally, after choosing values of $W_s$ and $K$, increasing the value of the error metric, $\epsilon$, past a particular "sweet point" merely results in a clicking and garbled sound texture, again because the PDF becomes invalid.

It was suggested earlier that the sampling window size, $W_s$, could be related to the tempo of the training example at DT-CWT decomposition level, $K$. The best sound texture of training example $\mathbf{Y}_e$ 1, *drum loop*, for example, was produced with $W_s = 41$ at level $K = 8$. This window size seems to roughly correspond with the spacing of the peaks seen in the level $K = 8$ wavelet decomposition of the signal in Figure 2.10 (b). These peaks are representative of the *beat characteristics* of the signal at that wavelet level. This observation motivates a scheme for estimating the values of certain parameters within the multi-resolution STS framework through content-based analysis of the training example signal, $\mathbf{Y}_e$.

## 2.9 Content-Based Parameter Estimation

Since the extent of sampling window, $W_s$, is completely dependent on the value of DT-CWT wavelet decomposition level, $K$, the latter must be established before the former can be estimated. Recall from Section 2.7 that it is only the complex coefficients from the coarsest scale (i.e. level $K$) of the wavelet space of the training example, $\mathbf{C}_e^{k..K}$, that are involved in the window-matching process of complex SSD matching with the wavelet space of the sound texture, $\mathbf{C}_s^{k..K}$. Hence the depth of DT-CWT decomposition should be chosen to be great enough that the computational burden of this exhaustive window-based sampling is significantly reduced, while still insuring that the complex wavelet coefficients at the coarsest level of the decomposition, $K$, encodes enough useful *information* - such as the beat characteristics of the signal - for a valid statistical synthesis. Hence, the choice of

$K$ should result in a fairly coarse complex detail signal at level $K$, that still retains some characteristic periodicity and *salient* detail features.

### 2.9.1 Choosing the DT-CWT Level

Shannon entropy is a measure of the information content of a signal. The entropy, $H$, of a signal $\mathbf{X}$ is defined as

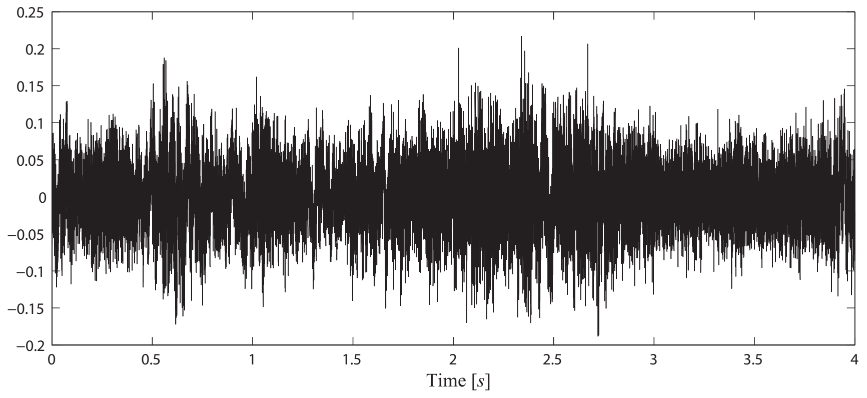
$$H(\mathbf{X}) = -\sum_{i=1}^{m} p(x_i) \log_2 p(x_i) \tag{2.4}$$

for random variable $\mathbf{X}$ with $m$ outcomes $\{x_i : i = 1, .., m\}$, and $p(x_i)$ is the Probability Distribution Function (PDF) of these $x_i$. In its simplest incarnation, the PDF can be expressed as a normalized $m$-bin histogram of the $x_i$. A low value of $H^k$ implies a near-random signal with little periodicity and few informative or salient detail features. It is this observation that leads to a method of choosing a good value for the depth of DT-CWT decomposition, $K$, with regard to the training example $\mathbf{Y}_e$.

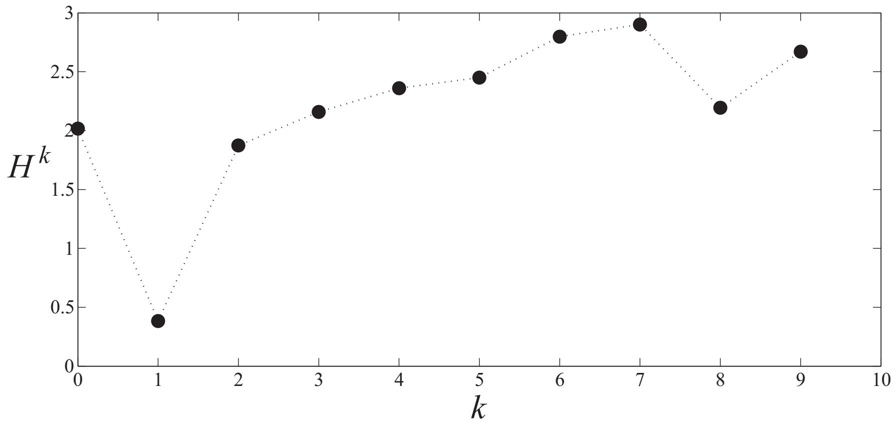
Figure 2.15 (a), is $4s$ of a typical event-on-background training example clip of *crowd chatter*. This signal contains both noise-like characteristics from the general chat of the crowd watching a game at a baseball stadium, overlaid with event-like bursts of speech from one particular male speaker who can be heard over the background noise. Figure 2.14 (b) plots the Shannon entropy values, $H^k = H(\mathbf{C}_e^k)$, of the band-pass complex detail signals, $\mathbf{C}_e^k$, from levels $k = 1..10$ of the DT-CWT decomposition of this sound sample. The PDF in this measurement (see Equation 2.9.1) is estimated by a normalized histogram of the absolute values of the complex wavelet coefficients in $\mathbf{C}_e^k$ from each level $k$.

The value at zero in this graph is the Shannon entropy of the $4s$ audio signal in ordinary sample space. Note that the entropy, $H^{k=1}$, of the $k = 1$ detail signal is lower than that of the original training example marked at $H^{k=0}$. This is to be expected, since $\mathbf{C}_e^{k=1}$ is merely one level of granularity within the original signal. The entropy values, $H^{k=1..10}$, of the complex detail wavelet levels $\mathbf{C}_e^{k=1..10}$ seem to increase steadily to a high point at level $k = 7$ before suddenly dipping at level $k = 8$. This seems intuitive since each further level of the wavelet decomposition should reveal salient granules at increasingly coarse wavelet levels. However, the resolution should become too coarse at a certain depth of DT-CWT decomposition to retain the sharp, event-like detail features - such as those constituting the man talking event in the *crowd chatter* training example seen in Figure 2.14 (a) - and this explains the dip in entropy seen Figure 2.14 (b).

One possible scheme to determine the level of DT-CWT decomposition, $K$, therefore, is an entropy check at each level, $k$, of the first few seconds of the training example, $\mathbf{Y}_e$, such as



(a) Event-on-background crowd chatter



(b) $H^k$ measured at each level of the DT-CWT

**Figure 2.14:** *Choosing the DT-CWT level, $K$; (a) example training clip of event-on-background* crowd chatter*, and (b) the process of choosing $K$ by analyzing the Shannon entropy of the absolute of the $k$-level complex detail signals. A $K = 7$ DT-CWT decomposition is deemed acceptable with regard to the criterion presented in Equation 2.5.*

$$H^{k+1} > H^k \Rightarrow k = k + 1 \ , \ H^{k+1} \leq H^k \Rightarrow k = K \tag{2.5}$$

The entropy in Figure 2.14 (b) dips in wavelet space for the first time at $k = 8$. Therefore, using this scheme, level $K = 7$ would be chosen as the depth of DT-CWT for multi-resolution STS with the *crowd chatter* training example analyzed in Figure 2.14.

Since the goal is to reduce the computational burden of the algorithm, while still ensure a valid multi-resolution STS, this criterion can be useful for choosing the depth of DT-CWT decomposition, $K$. Usually only the first few seconds of the training example need to be analyzed, provided they are representative of the general characteristics of $\mathbf{Y}_e$. This test is only treated as a guideline, however. Often a value of $K$ one more or less than the prediction from Equation 2.5 is found to perform well.

Furthermore, note that the entropy in Figure 2.14 (b) rises again after $k = 8$. There are also points in this plot at which Shannon entropy is ill-defined, such as at $k = 10$. Equation 2.5 may not be the optimal criterion for entropy analysis, but it is certainly useful for an initial guess that can be refined if an adequate sound texture is not produced. Other options for entropy analysis include choosing $K$ as the deepest $k > k_d$ with $H^k > t$, where $k_d$ is the DT-CWT level at which $H^k$ dips for the first time (e.g. $k = 8$ in the case of Figure 2.14 (b)), and $t$ is a threshold on the minimum entropy allowed. A visual inspection of the behavior of $H^k$ for multiple levels of DT-CWT decomposition can also be useful in choosing the value of $K$.

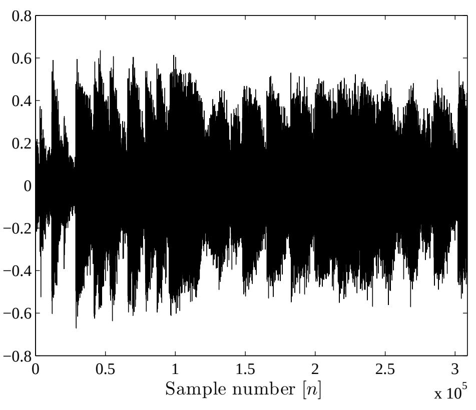
### 2.9.2 Choosing the Temporal Extent of the Sampling Window

Recall from Section 2.7 that the temporal extent of the sampling window, $W_s$, should be chosen to reflect the longest repetitive temporal feature in $\mathbf{C}_e^K$. Figure 2.15 (b) shows $C_e^K$ from the 10-level DT-CWT of a training example of *polyphonic music* seen in Figure 2.15 (a). It is clear to see that the salient quasi-periods of the music appear to be encoded in the *peaks* of $\mathbf{C}_e^K$. In the case of a musical signal, these peaks usually represent *beats*, such as the ones played by the drummer in a band. The *tempo* of the piece is defined by the spacing of the beats, or the *beat period*. If it can be assumed that many quasi-periodic sound examples have some sort of tempo - albeit erratic in the case of natural sounds such as human laughter, water splashing, or event-on-background signals - then perhaps the most suitable value of $W_s$ could be estimated as a multiple of the dominant beat period. This is certainly plausible in the case of musical signals such as the *polyphonic music* example seen in Figure 2.15 (a), or the *drum loop* training example, $\mathbf{Y}_e$ 1, shown in Figure 2.10 of Section 2.8.

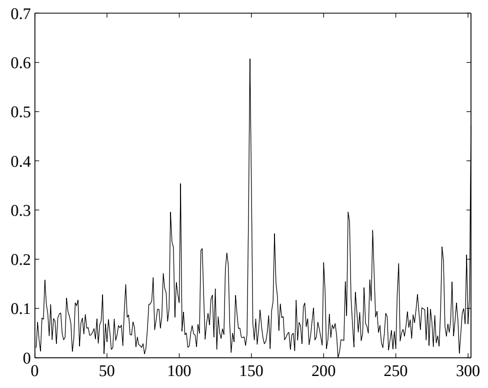
Beat Detection is a means of determining the tempo of percussive signals automatically. The most notorious Beat Detection algorithm is probably that of (Scheirer, 1998), which estimates the tempo of a piece of an audio signal by correlating it with a series of *comb* filters. Each comb filter consists of a train of impulses at a fixed period, with a different period for each filter. The tempo of the piece is determined as that of the most correlated comb filter. Since only a rough estimation of tempo is needed for choosing $W_s$, however, a much simpler beat finding technique is used here.

Beat Detection is carried out on the coarsest detail signal, $\mathbf{C}_e^K$, produced by DT-CWT decomposition of the training example, $\mathbf{Y}_e$, to level $K$. The steps involved in this process are as follows:

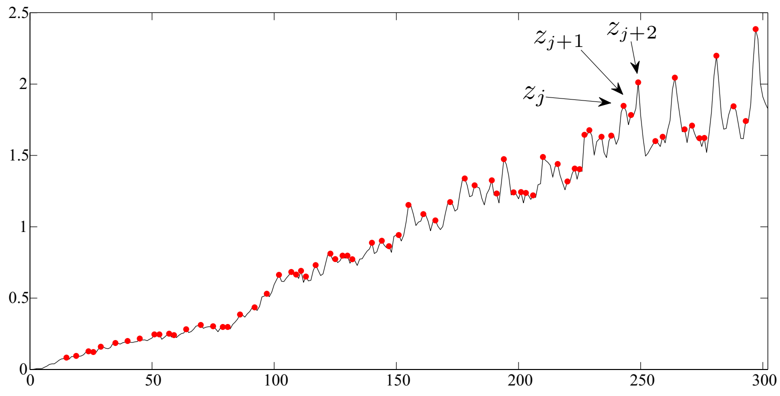
- Taking the absolute of $\mathbf{C}_e^K$, the signal is optionally smoothed to reduce the effects of noise. Smoothing is usually only necessary for very noisy signals, and may be performed with a 5-tap averaging filter if needed.



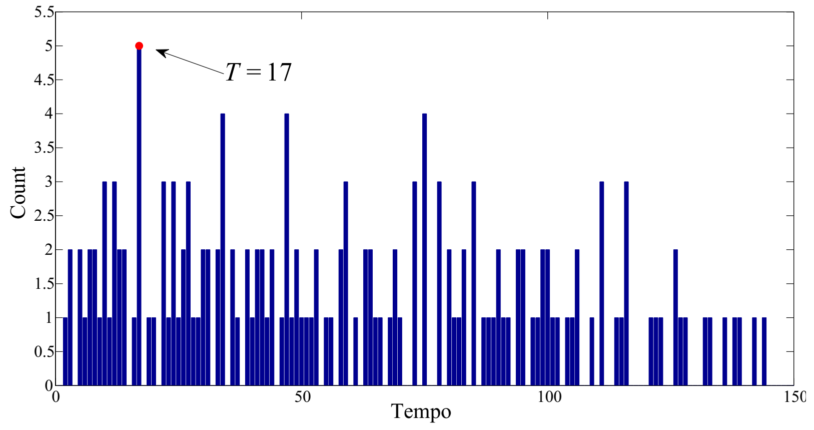
(a) Polyphonic music



(b) Complex detail signal, $\mathbf{C}_e^K$



(c) Autocorrelation, $\mathbf{R}^K$, of $\mathbf{C}_e^K$



(d) Beat histogram

**Figure 2.15:** *Beat detection to estimate $W_s$; (a) 7s of* polyphonic music, *(b) the coarsest wavelet detail signal, $\mathbf{C}_e^K$, from the DT-CWT to level $K = 10$, (c) the salient peaks in the Autocorrelation, $\mathbf{R}^k$, of $\mathbf{C}_e^K$ - three are labeled as $z_{j..j+2}$, and (d) the beat histogram with dominant beat period of $T = 17$ marked. The x-axis of (a), and (b-d) are measured in samples, $[n]$, or $[n^K]$ in the case of the latter three which are wavelet signals.*

- The Autocorrelation (i.e. self-correlation), $\mathbf{R}^K$, of $\mathbf{C}_e^K$ is calculated. $\mathbf{R}^K$ for Figure 2.15 (b) is shown in Figure 2.15 (c)[4].
- The next step is to locate the salient peaks in $\mathbf{R}^K$. A peak, $z_j$, is detected as a local maximum in the signal if the following condition is met:

$$(r_i - r_{i-1}) > \delta_p \ , \ (r_i - r_{i+1}) > \delta_p \tag{2.6}$$

  where $r_i \in \mathbf{R}^K$ and $\delta_p$ is a user-defined threshold. The peaks have been marked in Figure 2.15 (c) with $\delta_p = 0.001$, and three of them are labeled as $z_{j..j+2}$.
- The intervals (i.e. in samples, not time in seconds) between all peaks detected in $\mathbf{R}^K$ to all other peaks is measured. For example, the sampling interval from $z_j$ to $z_{j+1}$ is measured, as is that from $z_j$ to $z_{j+2}$, and so on. The intervals are used to construct a *beat histogram* of possible dominant beat periods. Figure 2.15 (d) shows the histogram resulting from calculating all possible peak intervals in Figure 2.15 (c).
- The dominant beat period, $T$, is estimated as the maximum of this histogram. It is clearly $T = 17$ in the case of Figure 2.15 (d). The value of $T$ is assumed to be related to the tempo of the training example clip, and it can be used as a rough estimate for the sampling window size, $W_s$, in multi-resolution STS.

The final estimate of tempo, $T$, is obviously highly dependent on the extent of any smoothing on $\mathbf{C}_e^K$, and also the value of $\delta_p$ chosen. In the case of the $\mathbf{C}_e^K$ shown in Figure 2.15 (b), no smoothing was performed, and $T = 17$ was found to be a (if not the) dominant beat period for a wide range of $\delta_p$. Only the left channel of the stereo file was used to form the beat histogram seen in Figure 2.15, but the $T = 17$ was also estimated for the right channel independently. This is interesting, since a beat period of $T = 17$ at level $K = 10$ of the DT-CWT corresponds to roughly $\{44100/(17 \times 2^{10})\} \times 60 = 152$bpm (i.e. beats per minute) in ordinary sample space, which is close to the human-measured bpm of 156bpm for this *polyphonic music* training example seen in Figure 2.15 (a). The $7s$ of content used for Beat Detection was taken from $(0:09)$ to $(0:16)$ in this song, a segment that is fairly representative of the tempo of the entire tune.

Due to the complexity of both the Human Auditory System, and of the way that sound and music is processed in the brain, the computer-measured dominant tempo of a piece of music could be very different from that which is audible to the human. This is especially true in the case of quasi-periodic signals like music or natural rhythms, which might contain many beat layers of varying tempo. Recall that the extent of $W_s$ should capture a long

[4]Autocorrelation can be performed in either Fourier transform, or ordinary sample space. Both methods are used interchangeably in this work, and produce similar results. The latter technique was used in the formation of this Figure.

| $\mathbf{Y}_e$ | Training Example | $f_s[Hz]$ | $nQ$ | $t_1[s]$ | Description |
|---|---|---|---|---|---|
| 6 | *crowd chatter* | $22k$ | 1 | 13 | Event-on-background; baseball game chatter with vendor shouting event. |
| 7 | *crowd chatter 2* | $44k$ | 2 | 6 | Event-on-background; restaurant chatter with lady shouting event. |
| 8 | *drum loop 2* | $96k$ | 2 | 2 | Near-periodic; 120bpm funk fusion drum loop. |
| 9 | *baby laughing* | $44k$ | 2 | 10 | Quasi-periodic; baby laughing, four phrases of laughter. |
| 10 | *polyphonic music* | $44k$ | 2 | 20 | Quasi-periodic; the song "Classical Gas" by M. Williams ((1 : 09) to (1 : 29) in song). |
| 11 | *german speech* | $22k$ | 1 | 12 | Quasi-periodic; male speaking the poem "Das gemeinsame Schicksal" by F. Schiller. |
| 12 | *english speech bg music* | $22k$ | 1 | 10 | Quasi periodic; male quoting the Bible over background music. |
| 13 | *piano phrases* | $22k$ | 1 | 26 | Quasi-periodic; Two phrases of piano music. |

**Figure 2.16:** *Training examples used in further experiments, their sampling rates, $f_s$, number of channels, time durations, $t_1[s]$, and a description of content. The training example files are included on the DVD accompanying this thesis.*

repetitive feature in $\mathbf{C}_e^K$, but $W_s = T$ is merely an initial rough estimate for the best value of $W_s$. If the initial estimate of $W_s = T$ does not produce an adequate sound texture in multi-resolution STS, other strong maxima in the beat histogram are also tested. Sometimes the best value for $W_s$ is actually a fraction or multiple of $T$.

## 2.10 Further Experiments

The content-based parameter estimation techniques discussed in the previous Section make it easier, less time-consuming, and less frustrating to carry out multi-resolution example-based STS experimentation with a much wider variety of audio training examples. Therefore, a further selection of training example clips are chosen to test the algorithm. Table 2.16 lists these samples, $\mathbf{Y}_e$, their sampling rates, $f_s$, the number of channels in the audio files, $nQ$, their durations, $t_1$, and a description of their content.

Notably, some of these training examples are *stereo* in that they have two channels (i.e. $nQ = 2$) as opposed to *mono* (i.e. $nQ = 1$). Multi-resolution STS of stereo training examples is carried out one of the three options listed here:

**A** Only the left channel is involved in the complex SSD matching of wavelet coefficients at the coarsest level, $K$, of the DT-CWT (see Section 2.7 for background), but the sampling pattern is used for the synthesis of both channels simultaneously.

| $\mathbf{Y}_e$ | Training Example | Stereo | Seed (samples$[nK]$, duration$[s]$) | $K$ | $W_s$ | $\epsilon$ | $t_2[s]$ |
|---|---|---|---|---|---|---|---|
| 6 | *crowd chatter* | N/A | placed at start $(100..130, 0.2)$ | 7 | 11 | 0.1 | 71 |
| 7 | *crowd chatter 2* | B | placed at start $(1..58, 0.7)$ | 9 | 55 | 0.7 | 66 |
| 8 | *drum loop 2* | B | placed at start $(1..50, 0.3)$ | 9 | 47 | 0.5 | 62 |
| 9 | *baby laughing* | B | placed at start $(1..80, 0.5)$ | 8 | 77 | 0.1 | 70 |
| 10 | *polyphonic music* | B | placed at start $(1..20, 0.5)$ | 10 | 17 | 0.3 | 90 |
| 11 | *german speech* | N/A | placed in center $(200..400, 0.6)$ | 6 | 51 | 0.001 | 28 |
| 12 | *english speech bg music* | N/A | placed in center $(400..550, 1.7)$ | 8 | 201 | 0.1 | 26 |
| 13 | *piano phrases* | N/A | placed start $(100..200, 1.2)$ | 8 | 51 | 0.3 | 116 |

**Figure 2.17:** *Parameter values and time durations, $t_2$, of the best sound textures achieved by testing the new example-based STS algorithm on a wide variety of training example clips. The Seed column records the placement of the seed, number of samples, $[n^K]$, placed at level $K$ in wavelet space, and the approximate time duration in seconds, $[s]$, in ordinary sample space. These sound texture files are included on the DVD accompanying this thesis.*

**B** Only the right channel is involved in the matching process, but the sampling pattern is used to drive the synthesis of both channels simultaneously

**C** Both channels are involved in the matching process. This is achieved by a straightforward expansion of Equation 2.3 to include both channels.

**D** The channels are synthesized independently in separate multi-resolution STS processes. This method is not an effective one for stereo synthesis, but is included here for comparative purposes.

The values for parameters $K$, $W_s$, and $\epsilon$, and durations, $t_2$, of the best sound textures produced by multi-resolution STS are listed in Table 2.17. The sound texture files are included on the DVD accompanying this thesis. Also listed in the Stereo column of Table 2.17 is the synthesis option used if the training example file is stereo. Although it appears that $B$ is the only option used for stereo synthesis, options $A$ and $C$ have also produced some interesting sound texture results, and these are included on the accompanying DVD, their parameters listed in Appendix A.

### 2.10.1 Discussion

Sound textures of the diverse training examples $\mathbf{Y}_e$ $(6 - 13)$ were created with the aid of the content-based parameter estimation techniques described in Section 2.9.

Shannon entropy analysis was used to choose the depth of DT-CWT, $K$, in most cases. Quite often (i.e. for approx one out of every two signals analyzed), the criterion presented in Equation 2.5 indicated the value of $K$ which produced the best sound texture (e.g.

$K = 7$ for $\mathbf{Y}_e$ 6, *crowd chatter*, as seen in Figure 2.14). As previously described in Section 2.9.1, however, the value of $K$ producing the best sound texture was occasionally found to be one more or less that indicated by Equation 2.5, or a visual analysis of $H^k$ was needed to manually choose an intuitive value. In stereo training examples, interestingly, Shannon entropy analysis of both channels independently was always indicative of the same value for $K$. Overall, Shannon entropy analysis was found to be a very useful tool in the content-based estimation of the parameter $K$ in multi-resolution STS.

Beat Detection was also found to be very useful for estimating the extent of the sampling window, $W_s$, in the coarsest complex detail signal, $\mathbf{C}_e^K$, of the training example. Much of the time the extent of $W_s$ was estimated correctly by the *most* dominant beat period, $T$, in the training example (e.g. $W_s = 17$ for $\mathbf{Y}_e$ 10, *polyphonic music*, as seen in Figure 2.15). In the majority of cases the best sound texture was produced with a value of $W_s$ set equal to *one of the* dominant beat periods determined by Beat Detection of the training example. Surprisingly, this technique was found to work even in the case of the event-on-background training signals such as $\mathbf{Y}_e$ 6, *crowd chatter 2* in which $W_s = 55$ was estimated from a dominant beat period detected in both of the stereo channels individually, with smoothing and $\delta_p = 0.01$. As previously discussed in Section 2.9.1, smoothing was often used in the case of noisy signals, but not for those with definite periodicity such as $\mathbf{Y}_e$ 8, *drum loop 2*, and $\mathbf{Y}_e$ 10, *polyphonic music*. Values of $\delta_p = 0.01$, $\delta_p = 0.05$, $\delta_p = 0.001$ were interchanged for peak detection depending on the granularity of the signal, but the most dominant beat periods were often detected for a wide range of $\delta_p$, and often in both channels of stereo training examples individually.

Example-based STS of the stereo training examples (i.e $\mathbf{Y}_e$ $8 - 10$) was often most effective with stereo option $B$, although options $A$ and $C$ also produced good results which are listed in Appendix A. Stereo option $D$, however, was found to be an ineffective method of stereo synthesis. Figure 2.18 demonstrates that example-based STS cannot be carried out on stereo channels independently. Figure 2.18 (top) shows the stereo channels of $\mathbf{Y}_e$ 8, and Figure 2.18 (bottom) shows the best sound texture result with stereo option $B$. Figure 2.18 (center) shows the sound texture that would result from performing example-based STS on both channels independently (i.e. stereo option $D$). It is clear to see in the latter that the two channels fall out of sync during synthesis, and therefore stereophonic coherency is lost in the sound file. The reason for this is, obviously, that the two channels in a stereo audio file are usually correlated, and therefore cannot be synthesized independently of one another. Examples of some bad sound textures created created with stereo option $D$ are included on the accompanying DVD, their parameters listed in Appendix A.

The sound textures of $\mathbf{Y}_e$ 6 *crowd chatter*, and $\mathbf{Y}_e$ 7 *crowd chatter 2* - both event-on-background type signals - are reproduced fairly well in multi-resolution STS. These sound textures do not evoke tiling due to the fairly random emergence of the vendor, and lady shouting events respectively, over the continuous background noise. A slight glitch can

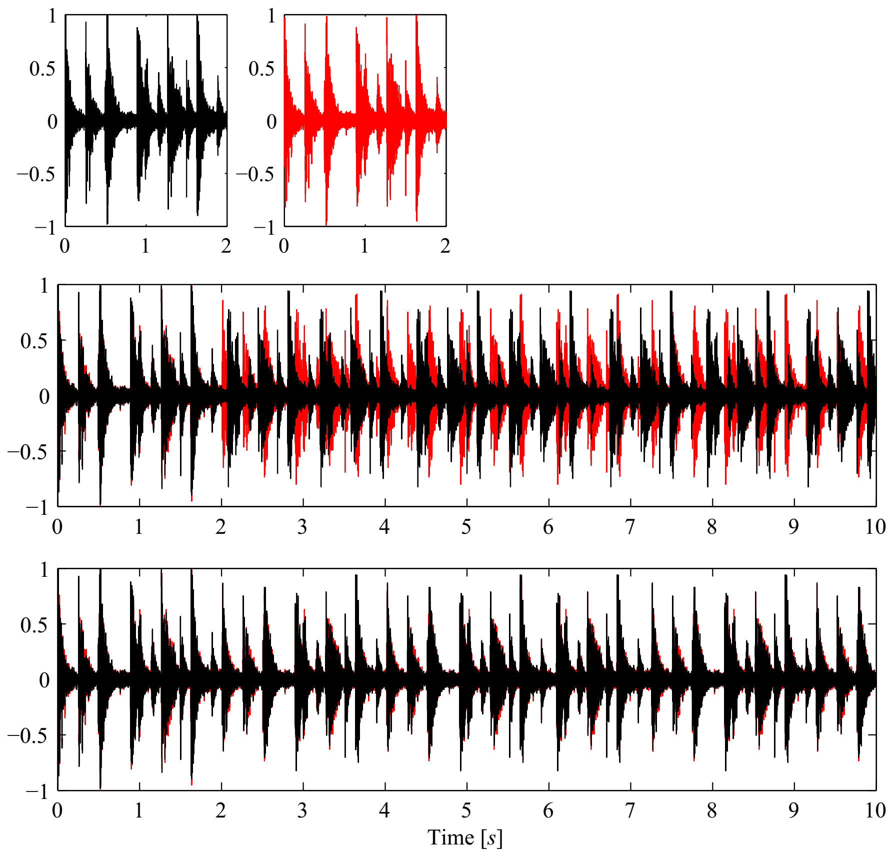
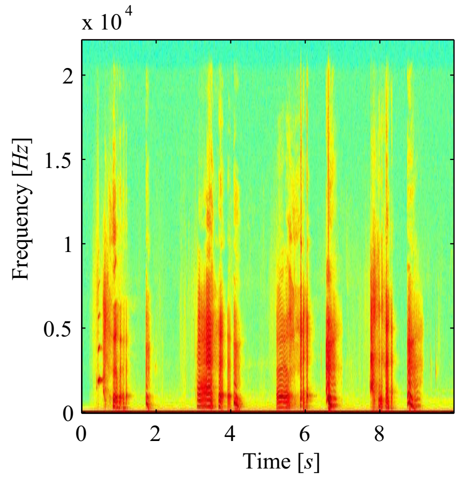


**Figure 2.18:** *Issues with stereo multi-resolution STS; (top left and right) stereo training example* $\mathbf{Y}_e$ 8*, with the left and right channels shown in black and red respectively, (center)* $10s$ *of the sound texture that would result if both channels were synthesized independently (i.e. stereo option D), and (bottom) stereo multi-resolution STS using stereo option A. Other parameters are* $K = 9$*,* $W_s = 47$*,* $\epsilon = 0.5$*, and a seed of* $0.3s$

sometimes be heard in the latter after the vendor shouts “nuts”, but this does not happen in every instance of the event.

The ambiance of $\mathbf{Y}_e$ 8, *drum loop 2*, is well represented in the resulting sound texture which is varied and interesting. The best sound texture of $\mathbf{Y}_e$ 9, *baby laughing*, also has adequate variation and does not sound looped or tiled, as can be seen in Figure 2.19. Variation is not as common in the sound texture of $\mathbf{Y}_e$ 10, *polyphonic music*, which sounds exactly like the training example for about $20s$, until the sound texture finally takes a novel direction. It is hypothesized that this spectrally rich, big-band musical arrangement does not lend itself easily to example-based STS because of its complex, multi-layered harmony of dozens of musical instruments in unison, and the fact that the piece is very lively with few *cadences* (i.e. resting points). Variation in the sound textures created by example-based STS often occurs at points of low amplitude or at a cadence, and so large portions of the



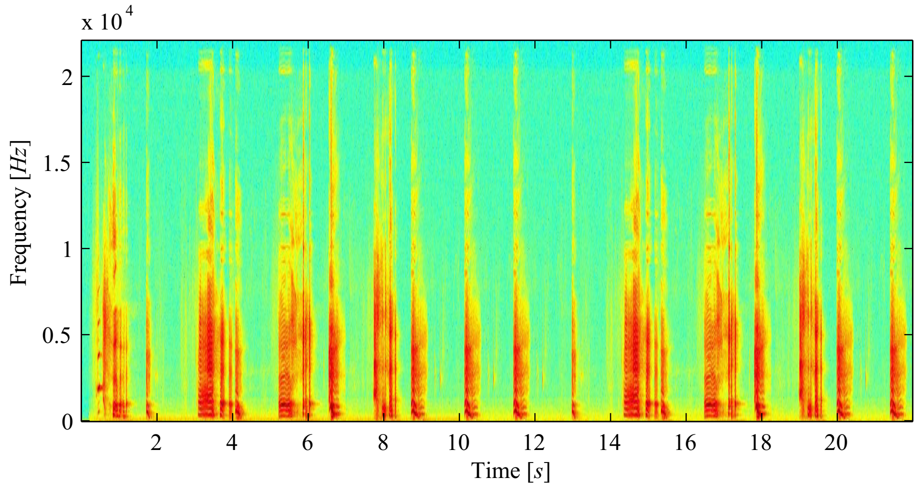


**Figure 2.19:** *Synthesized baby laughing texture; (top) spectrogram of the* $10s$ *training example,* $\mathbf{Y}_e$ 9*, and (bottom)* $20s$ *of the sound texture, generated with* $K = 8$*,* $W_s = 77$*,* $\epsilon = 0.1$*, a* $0.5s$ *seed, and stereo option* $B$*.*

---

training example of $\mathbf{Y}_e$ 10 are tiled up to rare points of cadence in the sound texture, and then a cycle of looping often occurs between these points.

The training examples of $\mathbf{Y}_e$ 11, *german speech*, and $\mathbf{Y}_e$ 12, *english speech bg music*, produced interesting sound textures. The best sound texture of the former is plausible and varied, with some small pulsing artifacts, while the best sound texture of the latter has some looping, but good variation overall. During experimentation with these training examples, it was found that by setting the value of DT-CWT depth $K$ very high (e.g. $K > 12$), and that of the sampling window size $W_s$ very low (e.g. $W_s = 3$), a breakdown of the structured language phrases to a *phoneme-like* level seemed to occur in the sound textures of $\mathbf{Y}_e$ 11 and $\mathbf{Y}_e$ 12. Some of these interesting examples are also included on the accompanying DVD, their parameters listed in Appendix A. Figure 2.20 (top) shows the

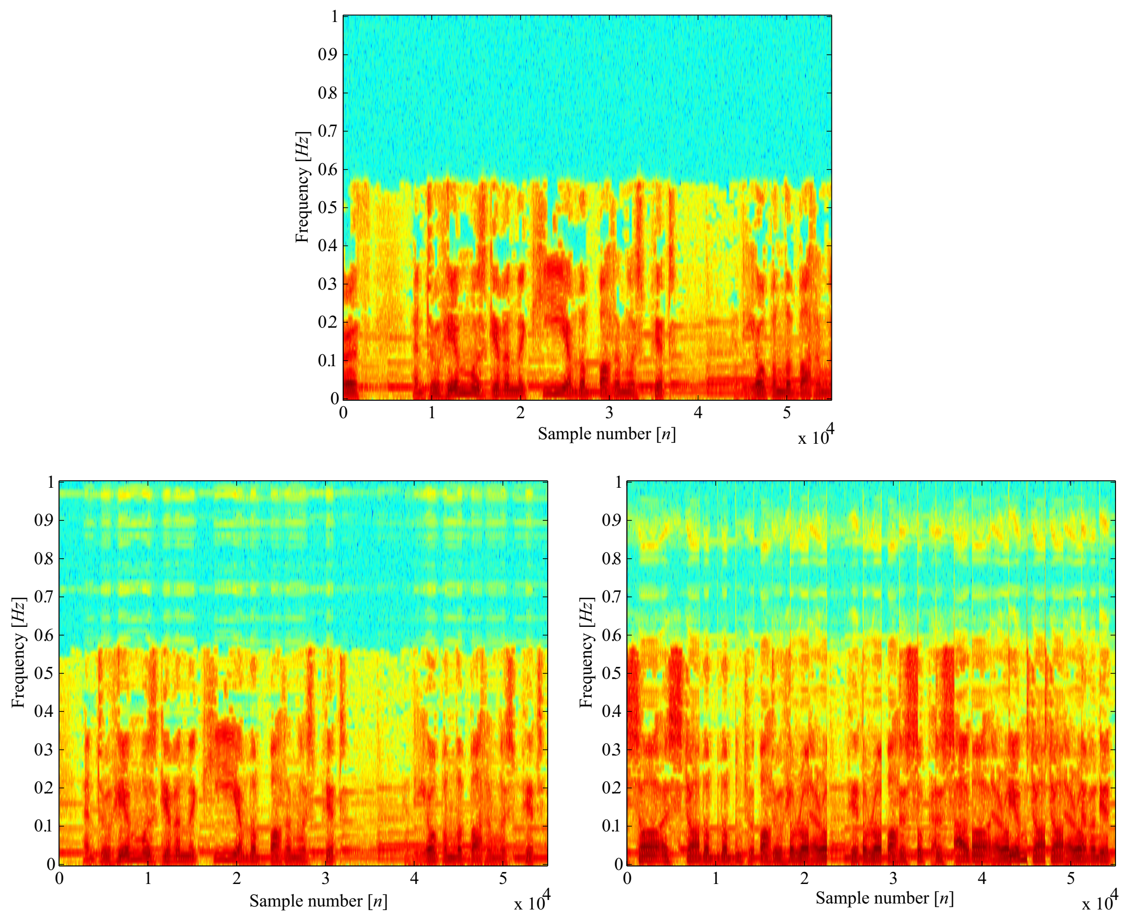
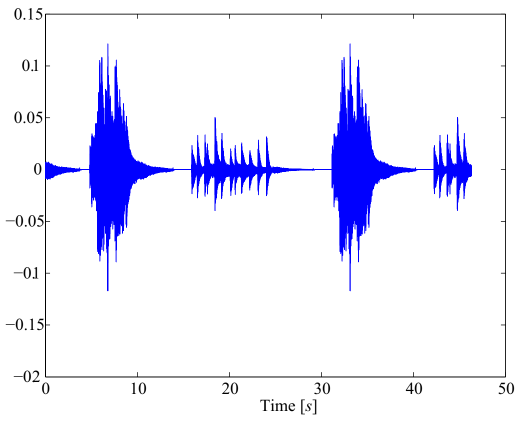
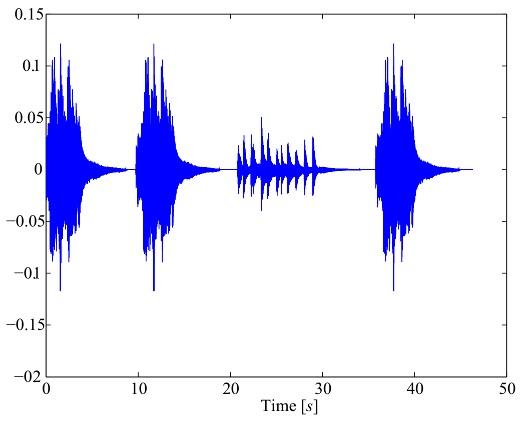


**Figure 2.20:** *Synthesized speech on background music texture; (top)* 2.5*s of training example* $\mathbf{Y}_e$ 11, *(bottom left)* 2.5*s of the best sound texture with* $K = 8$, $W_s = 201$, *and* $\epsilon = 0.1$ *with a* 1.7*s seed, and (bottom right)* 2.5*s of an alternative sound texture with* $K = 12$, $W_s = 3$, $\epsilon = 0$ *and a* 0.9*s seed. Here, the seeds are of different durations for the purpose of experimentation, and for ensuring a statistically valid seed according to a particular level of DT-CWT decomposition, $K$. Note that the* 2.5*s texture samples, (bottom left and right), are taken from somewhere in the middle of the bodies of sound texture and not from the beginning where the seeds are placed.*

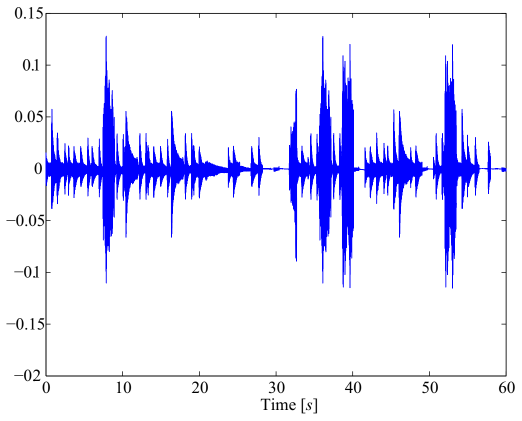
waveform and spectrogram of approximately 2.5$s$ of training example $Y_e$ 11, *english speech bg music*, and Figure 2.20 (bottom left) demonstrates plausible sounding texture with good variation. The high frequency inconsistencies seen in the spectrogram are the small pulsing artifacts, but these are only slightly audible. It is thought that the presence of background music contributes to this effect. Figure 2.20 (bottom right) shows 2.5$s$ of an alternative sound texture with $K = 13$, $W_s = 3$ and $\epsilon = 0$. This particular sound texture contains short periods of distortion, but it is still interesting because words that are not present in the training example are almost decipherable in the phoneme-like re-synthesis of the speech



(a) Piano phrases



(b) Random ordering



(c) Note-like ordering

**Figure 2.21:** *Synthesized piano texture; (a) visible tiling of the training example* $\mathbf{Y}_e$ 13 *in a bad sound texture, (b) random ordering of piano phrases in the best sound texture with* $K = 8$, $W_s = 51$, $\epsilon = 0.3$ *and a* $1.2s$ *seed, and (c) decomposition of the phrases to almost note-level with* $K = 15$, $w = 3$, $\epsilon = 0.001$ *and a* $5.9s$ *seed.*

in the sound texture.

Similarly interesting phenomena are observable in example-based STS with $\mathbf{Y}_e$ 7, *piano phrases*. By choosing too large a value for $W_s$, the two phrases heard in the training example are simply tiled in the sound texture, as can be seen in Figure 2.21 (a). The best sound texture produced for this training example results in a random ordering of the two phrases, as can be seen in Figure 2.21 (b). This is because the best values of $K$ and $W_s$ - chosen by Shannon entropy analysis and Beat Detection respectively - are conducive to allowing randomization at the points of cadence between the two phrases. At DT-CWT levels $K > 10$ and with smaller $W_s$, however, these phrases can be decomposed into much smaller units. Figure 2.21 (c) demonstrates the breakdown of the original phrases to almost *single notes* with $K = 15$, $w = 3$, and $\epsilon = 0.01$. The resulting sound texture is like like short, unpredictable bursts of piano played erratically.

## 2.11 Future Work

As discussed in Section 2.9, some content-based parameter estimation techniques have been established to save time and frustration in searching for the unique parameters for the multi-resolution example-based STS of each training example. Shannon entropy analysis and Beat Detection (see Section 2.9) were found to be useful in determining the values of DT-CWT decomposition depth, $K$, and sampling window, $W_s$, parameters, much of the time. Although these parameter estimation techniques are more reliable for some genres of training example than others, they still constitute an overall vast improvement on the trial-and-error method of parameter estimation required by the ITS algorithms of (Efros & Leung, 1999) and (Gallagher & Kokaram, 2005) that inspired this STS algorithm. Perhaps these techniques will also be useful for determining similar parameters in other example-based, multi-resolution or wavelet-based STS algorithms. Future work, however, would concentrate on honing these techniques to better reflect the class of sound texture being synthesized.

The ideal STS system would attempt to resolve these parameters fully automatically through complete analysis of the content of the training example clip, $\mathbf{Y}_e$. This idea is based on the assumption that these parameters are always related to, or influenced by certain salient features in the audio. An obvious example of this idea is that the tempo of a musical signal could be used to determine the sampling window extent, $W_s$, as previously discussed in Section 2.9. This however, is a music-specific technique that may not be appropriate for the case of an event-on-background training clip, for example. It would be useful, therefore, to have some kind of initial classification of the training example clip so that the system could then select which techniques or features to be used to estimate the parameters for each genre. (McKinney & Breebaart, 2003) classify audio files as classical music, popular music, speech, noise or crowd noise using a combination of features based on low-level signal properties, perceptual models of hearing and Mel-Frequency Cepstral Coefficients (MFCCs). After genre classification, the next step would be to discover the appropriate features that could be used to estimate the parameters of the multi-resolution example-based STS algorithm for each genre. Beat Detection, harmonic analysis and music-specific psychoacoustic features might be be useful for estimating the values of $K$ or $W_s$ in the synthesis of musical signals, whereas phonetic analysis, MFCCs and speech-specific perceptual models might be more appropriate in the case of speech signals. Perhaps this proposed STS system will not be completely automatic, and some user interaction might be involved in choosing these parameters for example clips of the event-on-background genre, for example.

The significance of the parameter $\epsilon$ (see Section 2.7 for explanation) has not fully been explored in this work. According to (Efros & Leung, 1999), the likelihood of variation emerging in the texture is related to the value of $\epsilon$, but experiments with multi-resolution STS indicate that each training example usually has one particular "sweet" value for $\epsilon$. Increasing it past this point seems to result in glitches, clicks and distortion in the resulting sound textures, although this observation may be influenced by individuals' perceptions of sound cues and definitions of sound texture.

As previously mentioned in Section 2.10.1, the multi-resolution STS algorithm has a tendency to favor points of cadence or low-amplitude troughs - including periods of silence - as points for variation within the sound texture. This tendency echoes the explicit objectives of locating low-energy transition points in many patch-based AT algorithms (Lu et al., 2004; Hoskinson & Pai, 2007; Jehan, 2004). Perhaps $\epsilon$ could be tuned on the fly as amplitude troughs emerge or dissipate in the sound texture, or vary with the tempo of a near-periodic percussive signal. The value of $\epsilon$ could also decrease on early detection of glitches, clicks or distortion in the sound texture to curb the propagation of errors.

It might also be interesting to consider the non-decimated levels of the DT-CWT decomposition of the training example clips as an image training example, and attempting to synthesize sound texture by performing example-based ITS - such as that of (Gallagher & Kokaram, 2005) - on the wavelet space arranged as a 2D image. This is a possible direction for future work.

## 2.12 Conclusion

A novel example-based Sound Texture Synthesis (STS) algorithm has been presented in this Chapter. Taking inspiration from a well-known example-based Image Texture Synthesis (ITS) algorithm and its multi-resolution extension, the Dual-Tree Complex Wavelet Transform (DT-CWT) has been employed for a wavelet-based STS with the goal of reducing the computational burden of an otherwise heavy exhaustive window matching-based sample-wise synthesis. To quantify this complexity reduction, the number of template-matching operations needed to create a sound texture is at least reduced by $2^K$ (i.e. two to the power of $K$), where $K$ is the level of DT-CWT decomposition used in creating the sound texture. The complexity is then further reduced by a corresponding reduction in template-matching widow extent, $W_s$. The resulting sound textures compare favorably with those produced by the other well-known wavelet-based STS algorithm in the field. In further experiments with a more diverse set of real-world training example audio clips, the sound textures synthesized are long, spectrally coherent and randomly varied rather than tiled or looped. They are mostly representative of the general ambiance of the training example clips. Some techniques for content-based parameter estimation - Shannon entropy analysis and Beat Detection specifically - have been discussed, and used to reduce the time and frustration inherent to the original method of user-based trial-and-error parameter estimation. Although the algorithm is not yet fully automated, this exploration into content-based analysis for parameter estimation is considered an interesting problem for future work.

# 3

# Implicit Spatial Inference with Sparse Local Features for Face Detection

Object, and particularly Face Detection and localization, are useful for content-based media filtering, since head shots of people constitute much of the content in home movies, digital TV and images, cinema and camera mobile phone clips. This Chapter introduces a novel way to leverage the implicit geometry of sparse local features for the purposes of Face and/or Object Detection. A brief discussion of the state-of-the-art in both of these topics is presented, including the types of classification algorithms used, an explanation of sparse local features, geometry models for invariant Object Detection and some of the motivations for improvement in these fields. A novel algorithm for Implicit Spatial Inference (ISI) is then presented. A two-class Bayesian scheme is used as a framework, and the likelihood is derived from the real-valued feature-wise classification of the machine learning algorithm Gentle AdaBoost, whose output is formulated as a probabilistic distribution with a Bi-Gaussian (B-G) model. The main contribution, however, is a novel scheme for the injection of prior contextual spatial information. This Markov Random Field (MRF) prior is imposed on a graph formed by Delaunay triangulation of the sparse local feature points. A Face Detection task is used to show that this framework may be useful for Face and Object Detection with rotation, scale, pose, illumination and occlusion invariance capabilities. The framework is also shown to be useful for localization and it is capable of producing a rough segmentation mask that could be used to seed a finer one.

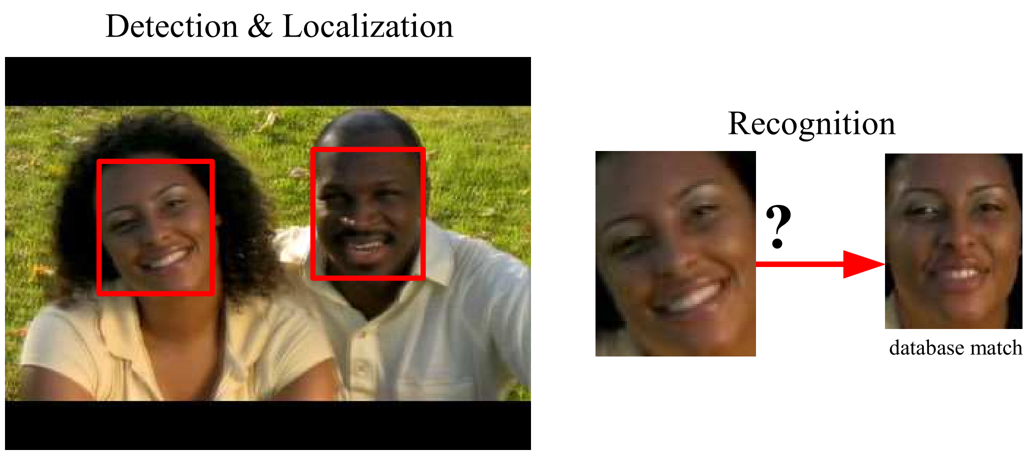


**Figure 3.1:** *Face Detection and Localization (left) is distinct from Face Recognition (right)*

## 3.1 Face Detection Explained

Face Detection is the automatic task of *localization* (i.e. segmentation of the region(s) occupied by) the human face(s) in an image. Face Recognition is the automatic identification of a particular face. The difference between these two concepts is demonstrated in Figure 3.1. This work discussed in this Chapter is concerned with Face Detection, and not Face Recognition. Face Detection is useful because faces are arguably the most important pieces of content in visual material containing people. Therefore content-based media applications can usually benefit from a Face Detection algorithm. In media adaptation, for example, special treatment - such as better compression or stylization - can be localized to the image region in which the face was detected. Face Detection functionality can be found in some modern digital cameras (e.g. Olympus E-450 D-SLR) and camera mobile phones (e.g. Samsung G800) where it is used to influence the camera's automatic focus or optimize exposure and/or flash output.

## 3.2 Face Detection Algorithms

Face Detection is an extremely popular, broad and varied research topic, and a few algorithms of relevance to this work will now be discussed. Face Detection algorithms are often categorized and described with regard to the terms in Table 3.2. For the interested reader, the survey of (Yang et al., 2002) is a more detailed review of much of the literature.

Face detectors that are scale, but not pose or rotation invariant are often called *frontal*, and the non-rotation invariant are often called *upright* face detectors. A *fully invariant* face detector is one that is scale, rotation, pose, illumination and occlusion invariant. The success of a particular face detector is usually quantified by comparing *true positive* with

| | |
|---|---|
| *color-based* | Faces detected involving the use of color features (e.g. skin color heuristics). |
| *grayscale* | Faces detected in grayscale images, or without the use of color values. |
| *scale invariant* | Faces detected at any *scale* within the image. |
| *rotation invariant* | Faces detected at any arbitrary *in-plane* rotation. |
| *pose invariant* | Faces detected at any arbitrary rotation, tilt or yaw (a.k.a. *out-of-plane* rotation). |
| *occlusion invariant* | Faces detected regardless of occlusion. |
| *illumination invariant* | Faces detected in all illumination conditions. |

**Figure 3.2:** *Popular terminology for categorizing Face Detection algorithms. See (Yang et al., 2002) for more information on the field.*

*false positive* results in a series of tests. A true positive is defined as a correctly detected face, whereas a false positive is a non-face region (e.g. background) incorrectly detected as a face. These tests may be performed on a popular Face Detection test database (e.g. MIT+CMU (Rowley et al., 1998b) or Caltech (Fergus et al., 2003)), and may involve systematically scaling, rotating or warping the test images, depending on the goals and invariance capabilities of the face detector.

### 3.2.1 The Viola-Jones Face Detector

The most notorious Face Detection algorithm of recent times is that of (Viola & Jones, 2004). In its well-known form, it is a fast frontal, upright, grayscale face detector. Tested on a particular grayscale test set - MIT+CMU (put together by (Rowley et al., 1998b)) - it is scale, but not rotation invariant. It is also somewhat pose invariant, but not explicitly designed to be so. The Viola-Jones algorithm is a typical example of a *feature-based* detector. Certain features are defined, and then the algorithm is *trained* by extracting these features from thousands of small, cropped, frontal face examples - specifically 4916 tiny images of $24 \times 24$ pixel resolution. These are known as the *positive* training examples. The detector is also trained with a *negative* training set, which is made up of thousands of background images. In the testing stage, the face detector can then classify features extracted from the test image as belonging to either the positive or negative class.

Three types of simple rectangular features are used in this algorithm for training and testing. Two of these features are the 1D and 2D Haar basis functions, and the third is not really a Haar basis function, but it is similar to one. Therefore together these features are referred to as *Haar-like*. Haar-like features appear to be globally representative of faces, in that a single positively classified feature is detected as a face. The detector learns to classify these features using a machine learning algorithm known as AdaBoost (Freund & Schapire, 1995). AdaBoost is an interesting classification-boosting algorithm that combines the moderate results of a number of weak classifiers by taking a weighted combination of their voting contributions. The result, in the case of the Viola-Jones face detector, is a stronger classifier that can identify unseen Haar-like features as belonging to a face or not with some degree of certainty, called the *margin*. A threshold is taken to binarize the classification.

The Haar-like features can be computed at multiple scales. Scale invariance for the face detector is implemented, therefore, by computing and classifying Haar-like features at twelve discrete scales. The detector is *window-based*, meaning that a spatial window is scanned along the image at each scale and with a step sizes of a couple of pixels, and the Haar-like features in each *sub-window* are evaluated and classified. The bounding boxes of overlapping, multi-scale positive sub-windows are merged spatially to create a bounding box for the face. In previous Face Detection algorithms, scale invariance is achieved by applying the detector to a pyramid of successively sub-sampled versions of the original test image (Osuna et al., 1997; Sung & Poggio, 1998; Rowley et al., 1998a; Rowley et al., 1998b; Roth et al., 2000). In the algorithm of (Viola & Jones, 2004), however, the creation of a test image pyramid is unnecessary due to the scalable optimization scheme for computing the Haar-like features.

To reduce the computational burden of exhaustive searching, this algorithm classifies the positive Haar-like features in a tree-like *cascade* of classification tests, using only a couple of Haar-like features to reject the majority of features in the first tier, and so on until a positive feature is detected or not. Each layer of the cascade is trained by AdaBoost, and parameters can be adjusted to trade off efficiency with accuracy. A two-cascade network of symbiotic detectors, and a boot-strapping training technique is employed to further improve the detector's efficacy. This classification scheme, together with an optimized technique for computing the Haar-like features makes this a fast face detector with acceptable accuracy. It was the world's first real-time face detector, according to (Viola & Jones, 2004).

In comparison to the performance of a number of previously prominent Face Detection algorithms (Sung & Poggio, 1998; Rowley et al., 1998b; Osuna et al., 1997; Schneiderman & Kanade, 2000; Roth et al., 2000) on the same or comparable test dataset, the authors cite that this detector outperforms in the tradition of comparing true positive with false positive results. The test database used - the original MIT+CMU (Rowley et al., 1998b), but excluding the new rotation invariance test set - consists of images of multi-scale faces and some pose variety, but they are of very low resolution. Since the Viola-Jones faced detector is trained on tiny, low-resolution training images, it is conducive to success with these test images.

The Viola-Jones algorithm has been widely adopted and reimplemented (e.g. within Microsoft Research and Intel) and it has spawned a renewed interest in the adoption of

AdaBoost and other classification-boosting algorithms in the fields of Face and Object Detection.

### 3.2.2 Rotation and Pose Invariance

The point is made in (Rowley et al., 1998b) that any frontal, upright face detector could become rotation invariant by applying it normally to a selection of successively rotated versions of the test image. This concept is similar to the use of a test image pyramid for scale invariance, although it is more computationally expensive, and non-ideal for real-time or point-and-click camera-based applications.

(Rowley et al., 1998a) presents a frontal, upright, and illumination invariant face detector that classifies sub-windows of a test image as face or not using a Neural Network trained on thousands of small pre-processed frontal face and non-face templates. Unlike the previously discussed algorithm of (Viola & Jones, 2004), this face detector is not feature-based, and scale invariance is achieved by applying it to a test image pyramid and merging the results over scales. This is achieved by clustering the centroids of positively classified sub-windows over multiple scales, eliminating those that do not have significant cluster centers *and* overlap with a positive detection.

A rotation invariant extension to this algorithm is proposed in (Rowley et al., 1998b). The *orientation* of each sub-window extracted from the test image pyramid is estimated by evaluating the spatial distribution of blobs of low intensity constituting the eyes, nose and mouth. The sub-window is then "de-rotated" to upright, and the normal upright network-based detector is applied. Overlapping detections are merged using a statistical process in which space, scale and orientation are parameters.

Like (Viola & Jones, 2004), both of these detectors are tested on the low resolution MIT+CMU face database created by the authors, Rowley et al., and the latter reports impressive rotation invariance when tested on a specially created new subset of the database containing faces under rotation. The former is also tested on a subset of the FERET database (Phillips et al., 1996). FERET consists of higher resolution images in a variety of upright poses, but with only one face per image, in the center of a uniform clutter-free background. (Rowley et al., 1998a) use FERET to explore the sensitivity of a slight pose invariance in their system. An extension to full pose invariance is proposed in (Rowley et al., 1998b), but not tested. According to the authors, this would involve the training of separate detector networks for different views of the face.

Later rotation and/or pose invariant face detectors seem to extend or adapt the concept of a boosted-classification algorithm with Haar-like features as made famous by (Viola & Jones, 2004). The algorithm of (Xiao et al., 2004), for example, presents some extensions to a Viola-Jones-style detector. The resulting system is also trained for in-plane and out-of-plane facial rotation (i.e. pose variations) in the range $[-45°, 45°]$ in both cases. First, the detector evaluates the in-plane rotation of each sub-window using a technique that appears to be an optimization of the orientation estimator of (Rowley et al., 1998b), and then the rotation-compensated sub-window is presented to a Viola-Jones-style upright face detector that has been trained with a positive set of thousands of small, cropped faces with pose variations in the aforementioned range. The algorithm employs some new Haar basis functions and cascade-network optimization techniques that are suited to the pose invariant nature of the detector. Again, the detector is tested on the low-resolution MIT+CMU database (Rowley et al., 1998b), but also on the CMU PIE (Pose Illumination and Expression) set. The latter is a database of single faces in a variety of upright poses with a uniform uncluttered background. Then tested on these sets, this face detector appears to be fast and fairly robust, with rotation and pose invariance for a limited degree of pose variation in PIE.

The face detector of (Huang et al., 2007) is trained with $30,000$ frontal, $25,000$ half-profile, and $20,000$ full-profile cropped faces (all $24 \times 24$ pixels) with a limited range of pose variations. Four different feature-based, AdaBoost-like detectors are networked to deal with four different limited ranges of rotation and pose, and positive responses are clustered according to space, scale, and pose. Interestingly, the feature space is based on a set of sparse, scale invariant features which the authors claim to be more computationally efficient than the rigid, Haar-like features introduced in (Viola & Jones, 2004). According to the authors, the detector appears to outperform others in terms of speed and accuracy when tested on the rotation, and also on a newly added profile faces test set of the ever-growing MIT+CMU database. The system appears to cope well with faces in difficult poses such as full-profile.

## 3.3 Observations on Face Detection

A few Face Detection algorithms have been discussed, and some general observations on the state-of-the-art are as follows:

- Taking into account the normal pose of the majority of faces in images, a frontal, upright, scale-invariant face detector is quite useful.
- Rotation and pose invariance are becoming increasingly important, but pose and multi-invariant Face Detection in real-world, high-resolution images is still an open problem.
- Face detectors are often trained with tens of thousands of tiny face and non-face examples, and the training process can take weeks (see (Viola & Jones, 2004)).
- Feature-based algorithms seem to dominate the literature (Schneiderman & Kanade, 2000; Roth et al., 2000; Viola & Jones, 2004; Xiao et al., 2004; Huang et al., 2007). Boosted machine learning algorithms are popular (Viola & Jones, 2004; Xiao et al.,

2004; Huang et al., 2007). Multi-scale window-based scans are often carried out for feature evaluation (Rowley et al., 1998a; Rowley et al., 1998b; Viola & Jones, 2004; Xiao et al., 2004; Huang et al., 2007).

- Haar-like features are popular features, but the use of alternative invariant features is interesting. Wavelet features are explored in (Schneiderman & Kanade, 2000), sparse Winnow features in (Roth et al., 2000), and sparse granular features in (Huang et al., 2007).
- Determining the facial boundary by grouping the detected features in scale-space seems to be a necessary post-process in window/feature-based face detectors.

Although not discussed in depth here, it is worth noting that Skin Detection (see (Vezhnevets et al., 2003) for a review) as a pre-process, or operations based on skin color heuristics can be introduced to the problem of Face Detection in color images. This makes sense since faces are normally observed within skin-colored regions in color photographs. The Face Detection algorithm of (Terrillon et al., 2000) is based entirely on modeling the spatial distribution of skin color values of faces in images, with Support Vector Machine (SVM)-based classification. In (Xiao et al., 2004), a post-process evaluates a skin color likelihood for the proposed positive face detections, and those which fall below a threshold as discarded as false positives. The argument in support of purely grayscale Face Detection, however, is that it will still work in the absence of color values when needed (e.g. detecting faces in images scanned from newspapers or artistic black-and-white photography).

## 3.4 Object Detection

In recent years, the topic of Face Detection has been largely overshadowed by the more general field of Object Detection. In Object Detection, the goal is to detect *instances* of a particular *class* of objects - such as cars or bicycles - in images and videos. Object Class Detection (OCD) seems a more suitable title for this field, but the literature attributes OCD to the task of verifying the presence of an object class *somewhere* in an image, without any localization of the object in the image. In contrast, Object Detection involves object localization. Face Detection has now become a sub-topic of Object Detection, because faces in general can be viewed as a class of objects. Similarly to face detectors, object detectors are usually trained with a number of *positive* (i.e. object) and *negative* (i.e. non-object/background) training examples, and machine learning algorithms are used to classify unseen test images. State-of-the-art Object Detection is largely feature-based, but sparse local features - also known as *interest points* or *keypoints* - dominate the literature. One of the most popular of these local features is Lowe's Scale Invariant Feature Transform (SIFT) (Lowe, 2004).

### 3.4.1 SIFT Features

SIFT features (Lowe, 2004) are sparse, local, and based on the appearance of the object at particular interest points. They are invariant to image scale and rotation. They are also robust to changing illumination conditions, noise, and minor changes in pose or viewpoint. Because the features are local, some of the SIFT features associated with a particular object may still be detected despite its partial occlusion.

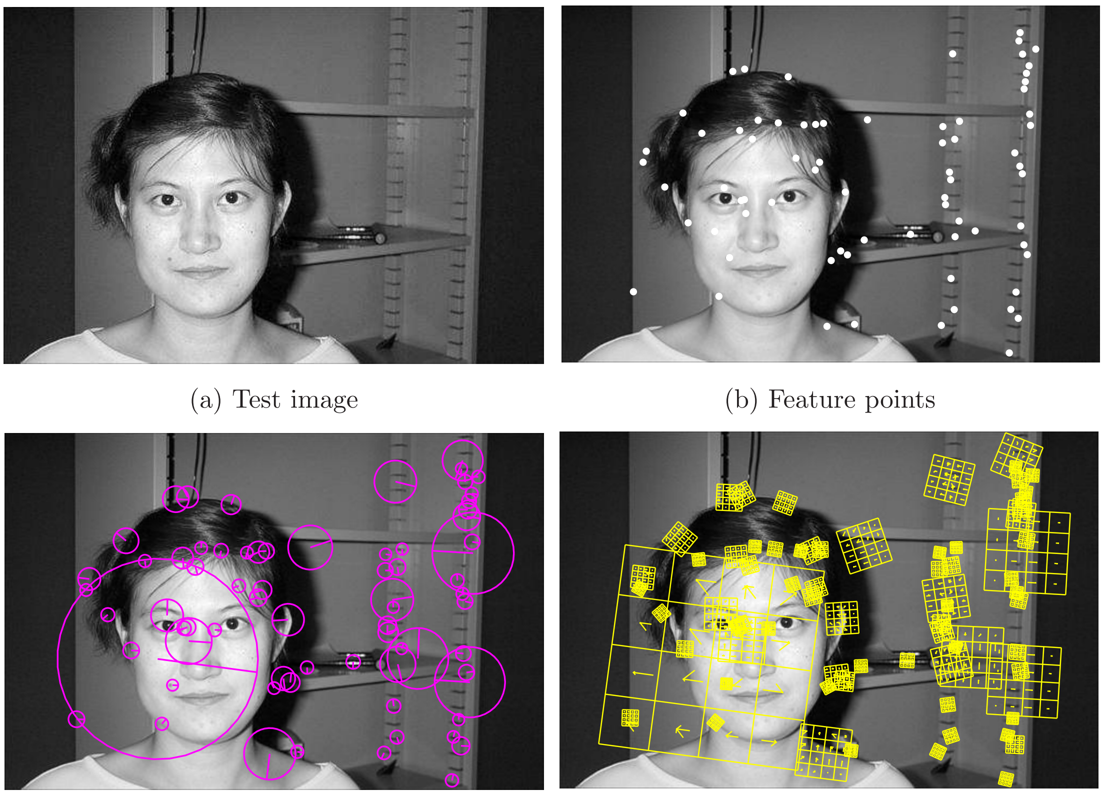
SIFT *feature points* are extracted from an image by convolving it with a series of Gaussian filters at different *scales*. Next, the difference of successive Gaussian-blurred images are taken. This operation is known as Difference of Gaussian (DoG). SIFT feature points are established at local scale-space extrema of the DoG pyramid. After some post-processing to remove the non-robust or *unstable* extrema, the dominant *orientation* of the features that remain are determined through analysis of the local image gradients. The next stage of the SIFT transform is the formation of a *descriptor* that captures the appearance of the local region surrounding the feature point. This takes the form of an array of sixteen eight-bin histograms recording the magnitude and direction of local gradients at the same scale as, and aligned according to, the scale and orientation of the feature point respectively. The final 128-element SIFT features vectors contain information on both *appearance* (i.e. descriptor of the local region) and *geometry* (i.e spatial location, scale and orientation).

An example of some SIFT feature points, their scales and orientations can be visualized in Figure 3.3 (b) and (c). The form of the descriptors can be visualized in Figure 3.3 (d). Over 700 SIFT features were attributed to this test image using default parameters for SIFT (code by (Vedaldi, 2006)), but only about a tenth of those are shown here for clarity.

Since they are invariant and robust, most of SIFT features found in one image should be detectable in another image capturing the same scene, at any scale or rotation, or even from a slightly different viewpoint. This property is called *repeatability*. The repeatability of SIFT features is high, but not 100%, especially in the case of major viewpoint changes. A more detailed explanation of SIFT can be found in (Lowe, 2004).

In order to use sparse features such as SIFT for detecting a class of objects such as faces, it is usual to employ a machine learning algorithm for classification. This is similar to many of the Face Detection algorithms discussed in Section 3.2. Usually, the object detector is presented with a selection of *positive* and *negative* training images; one set containing instances of the object class and the other set consisting of background images. Feature points and descriptors are normally extracted from these images, and the appearance descriptors used to train the classifier. The classifier should then be able to classify previously unseen SIFT descriptors as either positive or negative (i.e. belonging to an instance of the object or not).

Although SIFT is the most popular for this task, other types of interest point-plus-descriptor combinations exist, including Speeded-Up Robust Features (SURF) (Bay et al.,

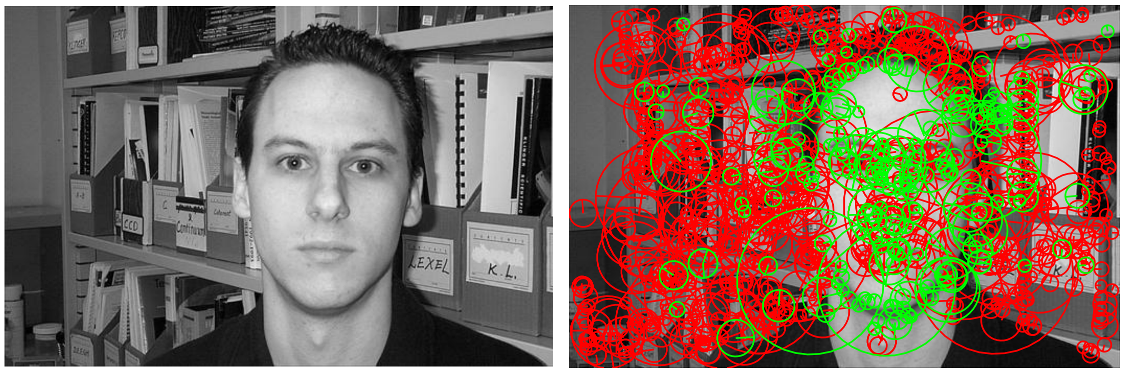


(a) Test image (b) Feature points

(c) Scales and Orientations (d) Descriptors

**Figure 3.3:** *SIFT features; (a) grayscale test image, (b) SIFT feature points, (c) scale and orientation denoted by a circle and a line receptively, and (d) descriptor orientation histograms. Only one tenth of the SIFT features generated for this image are shown here. The code used for SIFT is courtesy of (Vedaldi, 2006).*

2008), Kadir-Brady Saliency Features(Kadir & Brady, 2001), and Harris-Affine (Mikolajcyk & Schmid, 2002). The latter two examples are particularly interesting in that they are designed to be *affine invariant* - that is to say that they are repeatable under any arbitrary affine transformation of the image scenery (i.e. viewpoint change or object pose). The common attributes of all of these feature types including SIFT, however, are that they are locally descriptive, sparse, repeatable, and invariant to a number of geometric transformations with regard to image scenery. Another common attribute is that they are normally established over multiple image scales, and therefore low-resolution images - such as those found in the MIT+CMU Face Detection test set as described in Section 3.2 - will not produce many useful invariant features.



(a) Test image (b) SIFT features classified

**Figure 3.4:** *Classification of facial SIFT features using Gentle AdaBoost (Vezhnevets & Vezhnevets, 2005); (a) grayscale test image, (b) SIFT features classified as face (green) or background (red). Here, Gentle AdaBoost has been trained with* 23,080 *positive,* 36,903 *negative SIFT features and* 200 *iterations of boosting (see (Vezhnevets & Vezhnevets, 2005) for explanation). The SIFT code is from (Vedaldi, 2006).*

### 3.4.2 Geometric Contexts

Many sparse local feature-based algorithms concentrate on Object Class Detection (OCD) using a *bag-of-features* approach (Nowak et al., 2006), in which the presence of a bunch of local features positively classified in an arbitrary test image constituted the *presence* of the object somewhere, but OCD does not require object localization. Since SIFT features are local and sparse, object localization can be difficult, because some spurious mis-classifications are likely to occur using any point-wise feature classification algorithm. This problem can be visualized in Figure 3.4, where the SIFT features have been classified by Gentle AdaBoost (Vezhnevets & Vezhnevets, 2005) trained to classify these features as face (green) or background (red).

In Figure 3.4, there are many mis-classified features, but there is a spatial correlation between the correctly classified face features. This is useful if the object is partially occluded, but local features have still been detected on the part of the object that is in sight.

It is clear to see that the geometric relationships between sparse local features are important, and that articulating these features in terms of both appearance (i.e. the descriptor histogram of the surrounding local region in the case of SIFT) and relative geometry (i.e. spatial location and perhaps scale and orientation) is a good idea. Recent work in the field used sparse local feature points in the tasks of Object Detection and/or more sophisticated OCD by leveraging the relative geometric contexts of features using a variety of *geometric models*. This work and these models will now be discussed following a brief note on benchmarking in the field.

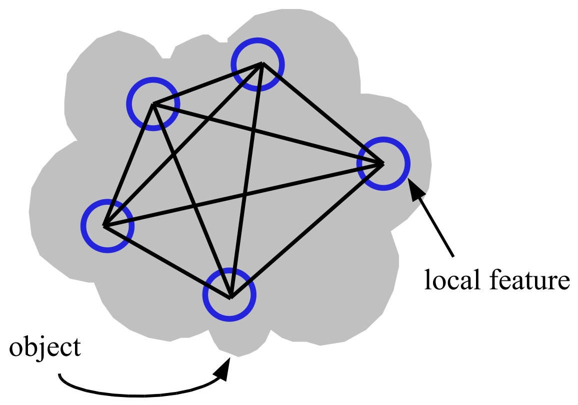


**Figure 3.5:** *Fully connected constellation for modeling the relative geometric contexts of sparse local feature points.*

### Benchmarking Metrics

Benchmarking in Object Detection and OCD is often concerned with *Recall* and *Precision*. Essentially, Recall is defined as the number of true positives over total positives in some sense, and Precision is the number of true positives over the sum of false positives and true positives. The words "positive" and "negative" however, are very ambiguous in this field. They could be attached to a *bounding box* detecting an object or not, how well this bounding box localizes the object in the image, verifying the presence of an object class somewhere in the image in an OCD test, or the classification of individual sparse local features such as SIFT with regard to OCD and/or Object Detection tasks. The Equal Error Rate (EER) is often defined as the performance of the algorithm when parameters are adjusted such that there is a Recall-Precision equal error in testing. EER is an interesting metric, but its significance is situational, just like Precision and Recall. Many applications in the field of Information Retrieval, for example, are concerned with maximizing information Recall at the expense of Precision, and so EER is less important.

### Constellation Model

A typical structure for relating the geometric contexts of sparse feature points on an object is the Constellation Model. This technique models the object or object class in question as a fully-connected *constellation* of features, where the interaction between all pairs of features are taken into account in both training, and classification with, the model. This idea can be visualized in Figure 3.5. It is usually quite a computationally expensive model to train (see (Fergus et al., 2003) for an example training process).

(Fergus et al., 2003) presents a probabilistic part-based OCD algorithm where one or more sparse local features are clustered to form *parts*. The local features are Kadir-Brady Saliency Features (Kadir & Brady, 2001) which capture appearance as a scale-adjusted radial histogram of pixel intensities surrounding the feature's center point. The interaction between parts is modeled using using a joint Gaussian-density function with parameters relating to both the appearance (i.e. histograms), and relative geometrical contexts (i.e. scales and locations) of parts on a Constellation Model. Variance in these parameters is modeled by Gaussian distributions, and occlusion is also modeled statistically. This fully-connected part-based model is theoretically scale, occlusion, illumination and somewhat pose invariant, but rotation invariance is not possible since the orientation of features is not taken into account as a parameter in the model. The authors acknowledge the computational complexity and resulting difficulties in training this complex model. It is both trained and tested on the faces, motorbikes, airplanes and cars datasets from the newly introduced Caltech database created by (Fergus et al., 2003). Only a few hundred images per class are used to train the object class detector, and it is tested on the same amount of images per class. The authors cite Precision/Recall rates of over 90% at EER in each catagory, but these refer to classification and not object detection or localization results. Interestingly, the authors note some limitations on the scale invariance due to the detector performing badly on very small images. This is probably due to the fact that sparse local features are not easily detectable in low-resolution images, and that the geometric model involved in this object class detector dependends highly on picking up useful features in the test images.

### Star Graph Models

The Star Graph Model relates the relative geometric contexts of sparse local features to the object centroid, or some other (usually central) landmark on the object, (usually) in both training and testing. This concept is visualized in Figure 3.6. The central landmark omits the need to model the pairwise geometric relations between all of sparse features, as in Constellation Models. The *star graph*, therefore, is a more popular model in Object Detection and OCD.

(Moreno et al., 2007) models the relative geometric contexts of SIFT (Lowe, 2004) and other local features - clustered into parts - on a star graph in both training and testing. Both the appearance (i.e. the SIFT descriptor) of the parts, and their spatial locations relative to a central landmark location are modeled probabilistically with a degree of variance. The feature descriptors are classified as positive or negative using either AdaBoost (Freund & Schapire, 1995) or a SVM, and a Markov Random Field (MRF) is used as a prior on the spatial distribution of parts. OCD experiments are performed on a few different object categories in the Caltech database (Fergus et al., 2003). In a small Face Detection experiment, only 10 positive images are used for training, and 90 positive and 100 negative images for testing. The output of the detector appears to be a rectangular bounding box that can be compared to the official Caltech-101 ground truth mask for each image.

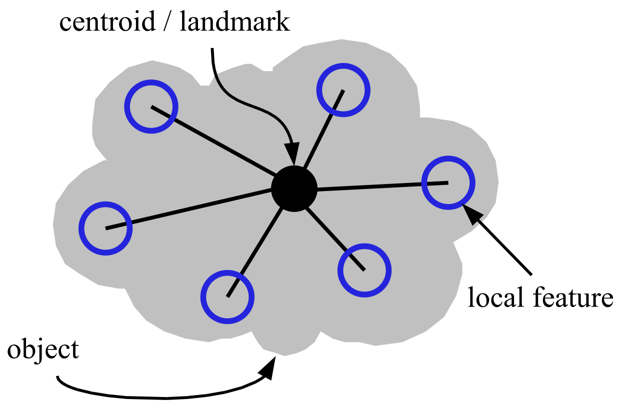


**Figure 3.6:** *Star graph for modeling the relative geometric contexts of sparse local feature points.*

Precision/Recall rates upwards of 80% at EER are reported for this small test, but it is rather unclear as to what these metrics of Precision and Recall are actually measuring with regard to the output bounding box. Scale and rotation invariance are not explicitely tested for, and perhaps not implemented.

A fairly invariant star graph-based object class detector is presented by (Mikolajczyk et al., 2006), and it is used to detect four object classes, including pedestrians. The sparse local features are Canny-detected edge points combined with Laplacian scale detection and a SIFT descriptor (Lowe, 2004). A star graph is employed as the model, and features from multiple object classes are clustered in both training and testing according to local appearance and geometric context in terms of spatial location, scale and orientation relative to the central landmark. Rotation invariance is embedded by using polar rather than cartesian coordinates for orientation. In training, a single hierarchical *codebook* tree is formed capturing the joint probabilities of appearance clusters and their geometric contexts for multiple classes. A similar tree is formed in testing an unseen test image, and an object class in the tree structure is detected when it is matched to a particular class in the codebook using a fast matching strategy, which the authors use to reduce the computational complexity of exhaustive searching with this model. The Pascal database (Everingham et al., 2005), which includes the Caltech-101 database (Fei-Fei et al., 2004), is used in training and testing most of the object classes. For each object class, only a few hundred images are used for positive and negative training examples, and test images. Although object detection and bounding box localization is inherent to this algorithm, it is only the object classification results that are evaluated, and not localization accuracy. Objects from different classes are classified in images simultaneously with a Precision/Recall performance of between $72 - 89\%$ at EER.

The Boundary-Fragment-Model (BFM) of (Opelt et al., 2006b), and the related multifeature Combined-Model (CM) of (Opelt et al., 2006a) are other interesting algorithms with star graph model-based training and testing, featuring AdaBoost classification with experiments on the Caltech/Caltech-101 (Fergus et al., 2003; Fei-Fei et al., 2004) or subset of the Pascal databases (Everingham et al., 2005). The performance of these object class detectors is good, but training their models is computationally expensive, and so a process of optimized learning is employed.

#### Other Models

The Pairwise Spatial Relations (PSR) approach of (Zhang et al., 2005) attempts to capture the relative spatial geometry of sparse local feature points in a radial histogram centered on the object in both training and testing. The scales and orientations of feature points are not taken into account in the model, but the *shape context* histogram is still scale and rotation invariant, and its log-polar distribution of bins is intended to imbue some degree of pose invariance to the geometric model. The radial histogram is strongly dependent on feature point repeatability and sensitive to occlusion. AdaBoost is utilized for classification within the framework, and OCD testing is carried out on the Caltech (Fergus et al., 2003) database. A Pascal database (Everingham et al., 2005) subset of multi-view bicycles is also used to test the pose invariance of the model, and EER rates of over 80% are reported.

## 3.5 Observations on Object Detection

A few Object Detection algorithms have been discussed, and some general observations on the state-of-the-art are as follows:

- Invariant Object Detection is an open, and interesting problem. Classification results of Object Class Detection (OCD) tasks are more readily measured and reported, even though many of the algorithms in the field effectively perform object bounding-box localization within their frameworks.

- SIFT features (Lowe, 2004), or variants of SIFT seem to be the most popular sparse local features (Zhang et al., 2005; Mikolajczyk et al., 2006; Opelt et al., 2006b; Opelt et al., 2006a; Moreno et al., 2007). Boosted machine learning algorithms are popular (Zhang et al., 2005; Opelt et al., 2006b; Opelt et al., 2006a).

- There are far fewer images in the training and test databases used for Object Detection than those used for the previous generation of Face Detection algorithms (see Section 3.2), but the new databases consist of much higher resolution images.

- State-of-the art research is concerned with establishing the relative geometry of features. Geometry is often posed as a graphical structure with underlying probabilistic modeling in both training and testing.

- Sometimes the rigidity of these models seems to constrain certain aspects of invariance. Features must often be clustered into groups or parts for use with these models. The training of these structures can be computationally expensive.
- The Caltech/Caltech-101 databases (Fergus et al., 2003; Fei-Fei et al., 2004) are popular, but it is well known that the images in these databases contain little pose, orientation, scale or illumination variation, almost no occlusion and little background clutter. The objects are also frequently located in the center of the images. Therefore object detectors based on rigid geometric models are more likely to perform favorably when trained and tested on these limited databases. (Ponce et al., 2006) examines some of these issues.

## 3.6 Motivations for Improvement in Face and Object Detection

The idea of using sparse local features such as SIFT (Lowe, 2004) for Face Detection is interesting. The invariance inherent to these features might lend itself to invariance in the resulting feature-based face detector. As discussed previously in Section 3.2, existing Face Detection algorithms tend to use many thousands of example images in training, techniques involving test image manipulation to enable scale and rotation invariance, or separate pose-enabled detectors to allow for pose invariance. By using sparse SIFT features which are themselves rotation and scale invariant, the volume of training material and necessity for test image manipulation could be reduced.

As discussed in Section 3.4.2, the use of *explicit* graphs, clustered parts, and shapes to model geometry is computationally intensive in training, and could be a potential barrier to full invariance in many of the local feature-based Object Detection algorithms. This is a motive to consider the idea of leveraging the *implicit* geometric distribution of sparse local features, after they have been point-wise classified with a typical machine learning classifier trained only on feature appearance, and *not* geometry. That is to say that geometry is only considered at the detection and *not in the training stage*, avoiding the need to cluster features into codebook parts or constrain their geometric parameters (i.e. location, scale and orientation) to rigid graphical models when training. It would also be useful to produce and evaluate a good object localization and/or segmentation from such an algorithm, since localization, and especially segmentation accuracy is not often measured in the field of Object Detection.

The following Section presents a technique for incorporating geometry to local feature-based Object Detection in this way, using a traditional Bayesian framework in which a Markov Random Field (MRF) is proposed on the sparse set of feature points. It is the MRF that imposes a weak geometry constraint after an initial point-wise feature classification. Novel aspects of the algorithm include realizing the MRF on a graph created by the Delaunay triangulation of the spatial locations of the features, defining a likelihood based on the output of a typical machine learning classification algorithm, and producing a object localization and rough object segmentation. Used in conjunction with SIFT features, this technique - which has been named Implicit Spatial Inference (ISI) - has resulted in a face detector with rotation, scale, partial occlusion and illumination invariance capabilities, and rough segmentation ability, as expected.

## 3.7 Implicit Spatial Inference

Implicit Spatial Inference (ISI) is a probabilistic technique that is capable of “injecting” geometric context after an initial point-wise classification of the *appearance* of sparse local features (e.g. SIFT (Lowe, 2004)) in a test image. The motivation for ISI is that the features of objects (e.g. faces), are spatially correlated in images, but not rigidly so. Therefore, the machine learning classifier need only be trained on the appearance of features, and their relative geometry can be introduced in the object detector as ISI afterwards.

As can be seen in Figure 3.7, ISI is capable of classifying a tight network of local features on the object, and it is therefore conducive to a rough segmentation that could be useful for content-based filtering of images and videos. In Figures 3.8 (b) and (c) for example, the rough segmentation seen in Figures 3.8 (a) is used as a basis for stylization and face scrambling respectively.

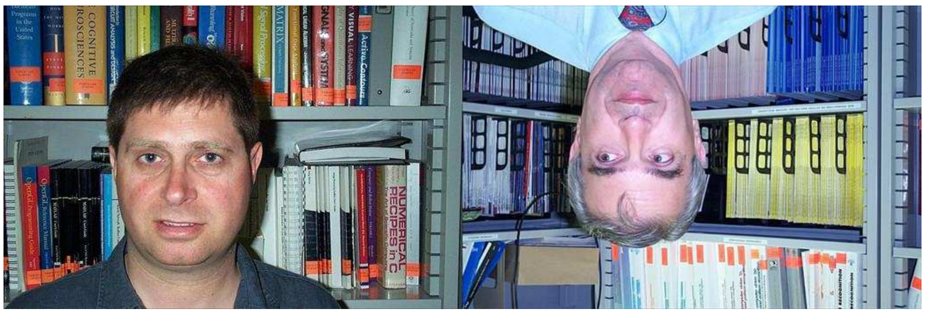
The probabilistic form of this technique is only possible if the initial feature appearance classification is non-binary. This is true of any probabilistic classification where a likelihood is obtained by proposing a parametric model in feature appearance space, such as a Gaussian distribution for each class. It is also true however, of many modern machine learning algorithms such as state-of-the-art versions of SVMs, Neural Networks, and many variants of boosting algorithm such as Modest and Gentle AdaBoost (Vezhnevets & Vezhnevets, 2005), the latter being the classification algorithm used in this work.

### 3.7.1 The Bayesian Framework

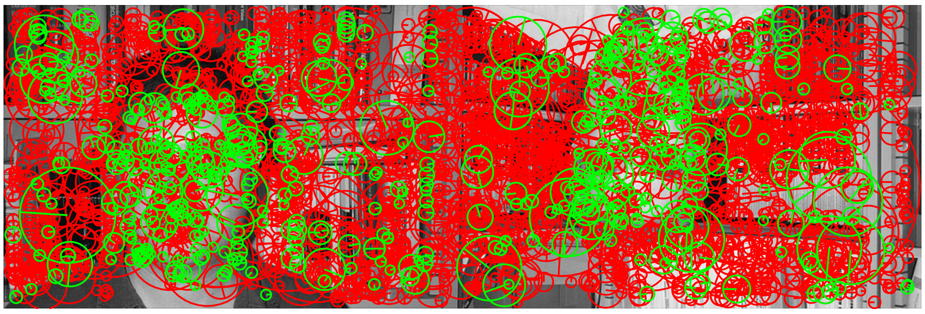
Consider that an unseen test image produces $K$ sparse local features (e.g. SIFT interest points and descriptors), and the goal is to classify these features, $\mathbf{f}_k(\mathbf{x}_k) = \mathbf{f}_k$, centered at image coordinates $\mathbf{x}_k = [x_k, y_k]$ as positive (i.e. belonging to object such as a face), or negative (i.e. belonging to background). Let $l_k = l(\mathbf{x}_k)$ represent this binary label field

$$l_k = \begin{cases} 1 & \text{if } \mathbf{f}_k \text{ is a positive feature} \\ 0 & \text{if } \mathbf{f}_k \text{ is a negative feature} \end{cases} \tag{3.1}$$

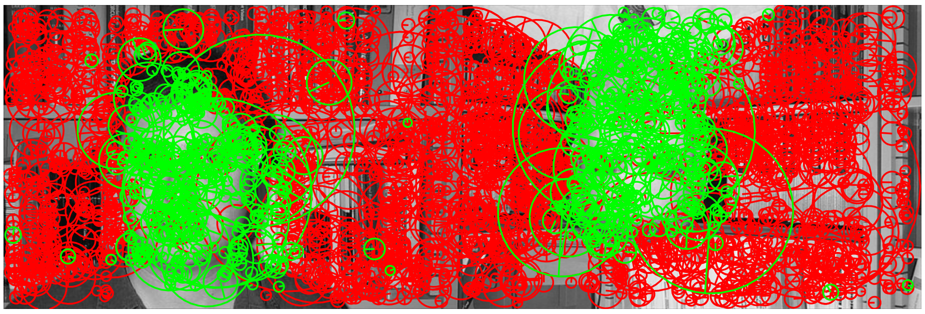
According to Bayes’ rule, the probability that $\mathbf{f}_k$ belongs to either class is



(a) Test image



(b) Initial appearance-based feature classification

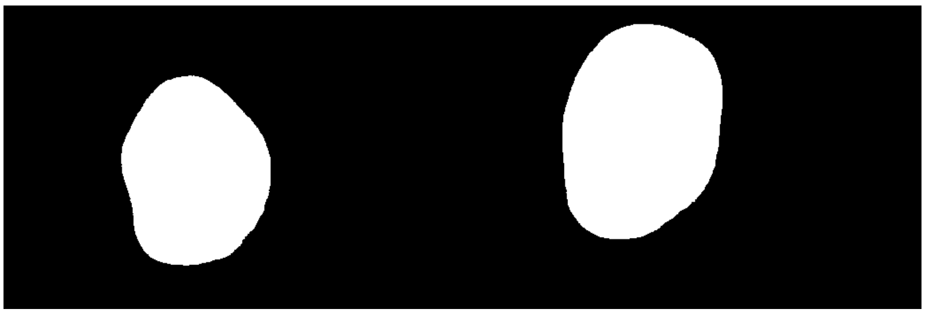


(c) ISI-improved feature classification

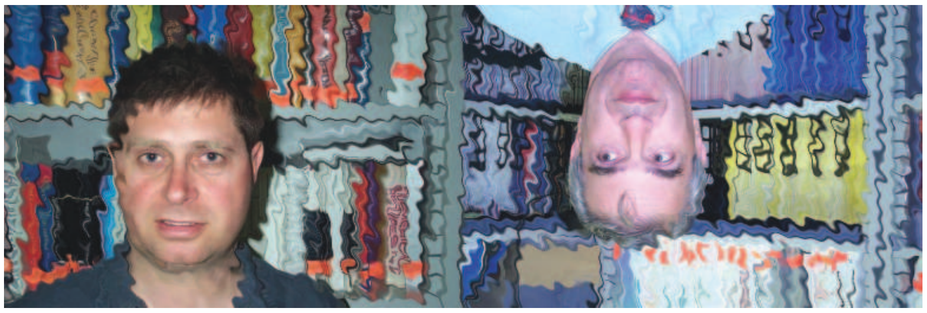
**Figure 3.7:** *ISI in feature classification; (a) test image - a hybrid of two images taken from the Caltech-101 (Fei-Fei et al., 2004) database (one flipped), (b) Gentle AdaBoost (Vezhnevets & Vezhnevets, 2005) appearance-based SIFT feature classification and (c) geometrical context injected with ISI. The SIFT code is from (Vedaldi, 2006).*

$$\underbrace{p(l_k|\mathbf{f}_k)}_{posterior} \propto \underbrace{p(\mathbf{f}_k|l_k)}_{likelihood}\,\underbrace{p(l_k|\mathbf{L}_k)}_{prior} \tag{3.2}$$

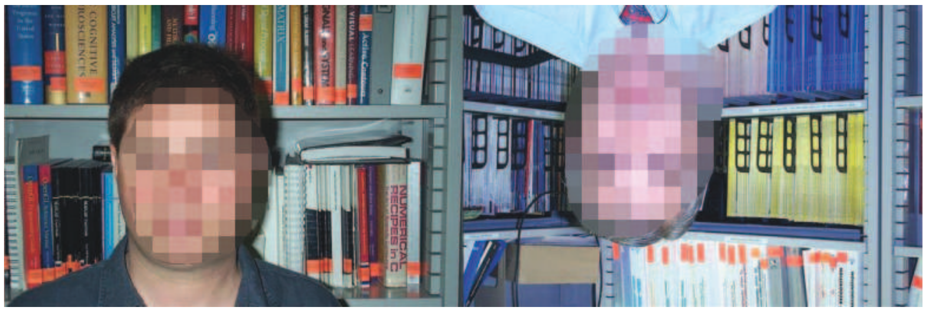
where $\mathbf{L}_k{=}\mathbf{L}(\mathbf{x}_k)$ is the spatial *neighborhood* of labels around $\mathbf{x}_k$. The likelihood represents a connection between the observed feature *appearance* and the label that is assigned, while the prior quantifies knowledge about the label field before observation. It by means of the prior that contextual spatial information is introduced to the solution. A MRF is used as



(a) ISI-generated rough segmentation mask



(b) Stylization with the segmentation mask



(c) Identity hiding with the segmentation mask

**Figure 3.8:** *Content-based media filtering; (a) rough segmentation mask generated by ISI in a Face Detection task, (b) background stylization with Adobe Photoshop's "ripple" filter using the rough segmentation as a mask, and (c) face scrambling with the mask.*

---

the prior in the usual way, although it is defined on a uniquely designed Delaunay graph which will be discussed later in Section 3.7.3.

### 3.7.2 Obtaining the Likelihood

As previously discussed, a likelihood might be obtained by modeling the Probability Distribution Function (PDF) of the appearance of positive and negative features in Gaussian distributions. However, given the success of the recent Face and Object Detection schemes based on point-wise classification algorithms (Schneiderman & Kanade, 2000; Roth et al., 2000; Viola & Jones, 2004; Rowley et al., 1998a; Rowley et al., 1998b; Xiao et al., 2004; Zhang et al., 2005; Opelt et al., 2006b; Opelt et al., 2006a; Huang et al., 2007; Moreno et al., 2007), it is proposed to utilize the output of such a classifier to determine this likelihood.

This might seem strange, but consider the output of a real-valued AdaBoost classifier, such as Gentle AdaBoost (also known as, and hereafter GentleBoost (Vezhnevets & Vezhnevets, 2005)) which is used in this work. It is a point-wise boosted classifier which yields a real-valued measure of *confidence*, $c_k \in \Re$, that is assumed to be related to the likelihood of $l_k = 0$ or $l_k = 1$ at each site. Specifically, GentleBoost can be trained to label the *appearance* of a feature $\mathbf{f}_k$ according to the rule

$$l_k = \begin{cases} 1 & \text{for } c_k > h \\ 0 & \text{for } c_k \leq h \end{cases} \tag{3.3}$$

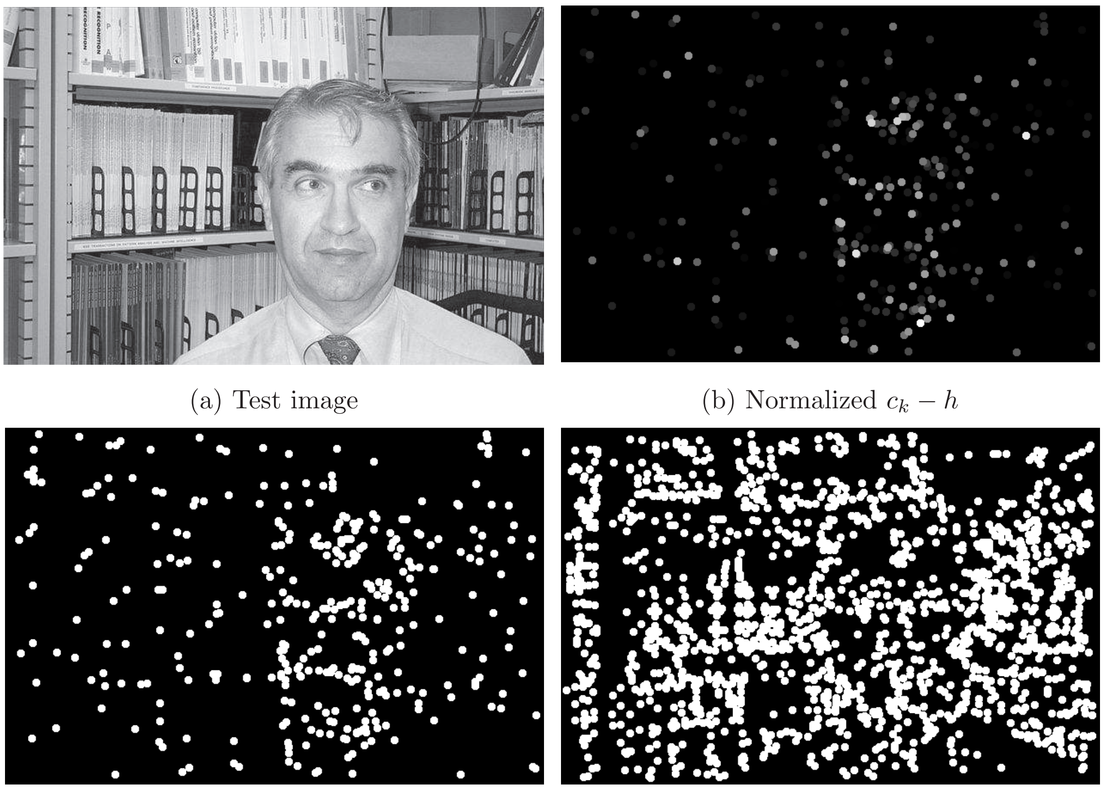
where $h$ is a user-defined threshold integral to the GentleBoost algorithm, enabling a rough binary classification of the feature. The confidence values given by GentleBoost to the appearance of the SIFT features, $\mathbf{f}_k$, detected in Figure 3.9 (a) can be visualized in Figure 3.9 (b), and the binary labeling of $\mathbf{f}_k$ as positive or negative in Figures 3.9 (c) and (d) respectively.

As can be seen in Figures 3.9 (c) and (d), the binary classification by $h$ is not very accurate, but the actual values of $c_k$ are interesting. The *margin*, $|z_k| = |c_k - h|$, is related to the confidence of the binary labeling, $l_k$, with regard to $h$, and therefore the metric $z_k = (c_k - h)$ is useful for visualization purposes. In Figure 3.9, the feature points, $\mathbf{f}_k$, are dilated for clarity and colored with the normalized values of $z_k$. Increasingly confident positive features are seen as brighter, whereas increasingly confident negative features are darker. The actual values of $c_k$, therefore, can be used as likelihood in the proposed Bayesian framework.

For use as a likelihood, the PDF of $c_{k..K}$ must be determined, and a Bi-Gaussian (B-G hereafter) model is used to determine the two-class likelihood distribution. Using this model, a Gaussian distribution is attributed to each class as follows

$$p_g(\mathbf{f}_k|l_k) = \begin{cases} \frac{1}{(\sigma_p)\sqrt{2\pi}} e^{-(c_k - c_{\mu_p})} & \text{for } l_k = 1 \\ \frac{1}{(\sigma_n)\sqrt{2\pi}} e^{-(c_k - c_{\mu_n})} & \text{for } l_k = 0 \end{cases} \tag{3.4}$$

where $c_{\mu_p}$ and $c_{\mu_n}$, $\sigma_p$ and $\sigma_n$ are the means and standard deviations obtained by GentleBoost classification of thousands of SIFT feature descriptor histograms from the positive (i.e. face) and negative (i.e. background) feature training sets used to experiment with this algorithm, as will be discussed in more detail in Section 3.8. Figure 3.10 demonstrates this model. The top two graphs show histograms of the GentleBoost $c_k$ output values in classification of the positive (top) and negative (center) SIFT descriptor (i.e. appearance


(a) Test image (b) Normalized $c_k - h$

(c) Positive labels (d) Negative labels

**Figure 3.9:** *GentleBoost feature classification; (a) test image, (b) normalized confidence attributed to the appearance of feature points,* $\mathbf{f}_k$*, (c) binary labeling of positive, and (d) of negative features with the user-defined* $h = -1.5$*.*

only, no contextual geometric information) training sets, and the bottom graph shows these $c_k$ values modeled as Gaussian PDFs using the B-G model, where the positive and negative training sets are colored green and red respectively.

### 3.7.3 The Prior

Modeling the spatial arrangement of local features with MRFs is interesting since they are sparse and spatially non-uniform. A novel solution is to triangulate the *spatial coordinates* of the features. Delaunay triangulation has been chosen for this task. In computational geometry, the Delaunay triangulation of a give set of points is one that ensures that no point in the set is within the circumcircle of any triangle in the triangulation (Delaunay, 1934). Delaunay triangulation is concerned with maximizing the global minimum of all the angles in the triangulation, with the goal of avoiding slim triangles, or “slivers”.

The Delaunay triangulation of a set of SIFT feature points obtained from a test image can be seen in Figure 3.11. Points in the graph that are connected with lines will be known

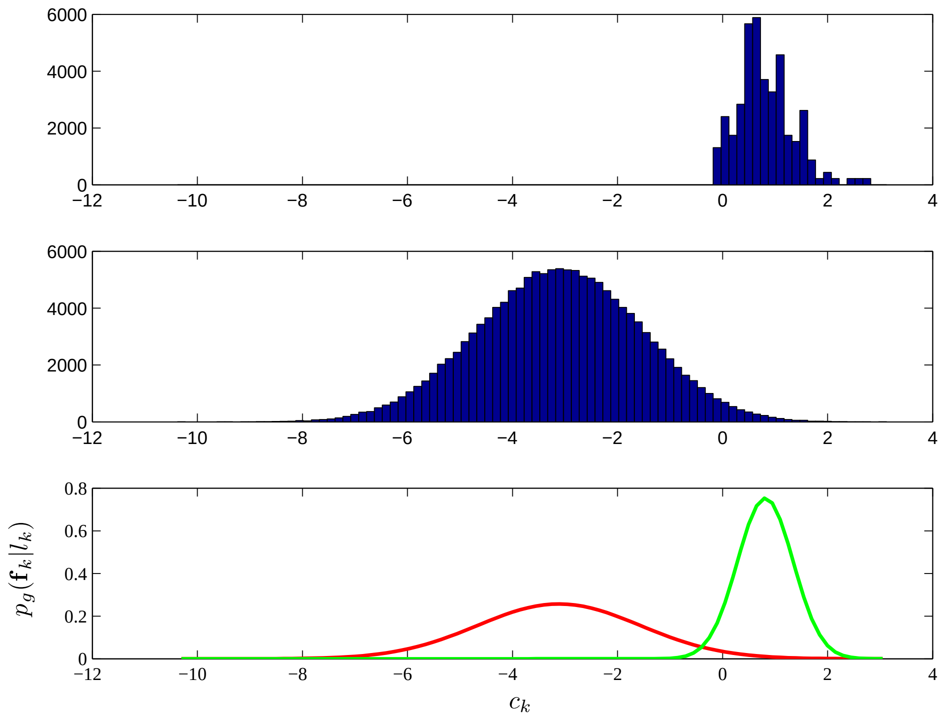


**Figure 3.10:** *The Bi-Gaussian (B-G) likelihood model; (top) histogram of GentleBoost $c_k$ values for positive and (center) negative training SIFT feature appearance sets (i.e. no contextual geometric information), and (bottom) the B-G likelihood PDFs computed using Equation 3.4. Green is positive, and red is negative.*

---

hereafter as *Delaunay neighbors*, and the lines will be known as *Delaunay connections*. A feature points Delaunay neighbors of degree $n_d$ can be computed from this graph. Figures 3.12 (a) and (b) demonstrate the concept of an $n_d = 1$ and $n_d = 2$ Delaunay neighborhoods respectively. Note that this idea of local spatial connectivity is distinct from the global shape concept of the geometric models discussed in Section 3.4.2 respectively. It is a less rigid, implicit model of connectivity between feature points. a MRF can be defined on a Delaunay graph in terms of Delaunay neighborhoods of some $n_d$. Given this, the prior probability can be modeled as

$$p_q(l_k|\mathbf{L}_k) \propto \exp -\Big\{\sum_{s\in S} \lambda_{sk}|l_s \neq l_k|\Big\} \tag{3.5}$$

where $l_s$ are the labels of all sparse feature points linked to $\mathbf{x}_k$ of $\mathbf{f}_k$ by Delaunay connections in its Delaunay neighborhood of degree $n_d$. Each term is weighted by $\lambda_{sk} = (1 + 1/d_{sk})$, where $d_{sk}$ is the Euclidean distance in pixel units from each feature point's centroid coordinate, $\mathbf{x}_k$, to each of the other feature point coordinates, $\mathbf{x}_s$, in its Delaunay neighborhood. It is tempting to introduce a suppression term for the case of Delaunay neighbors whose

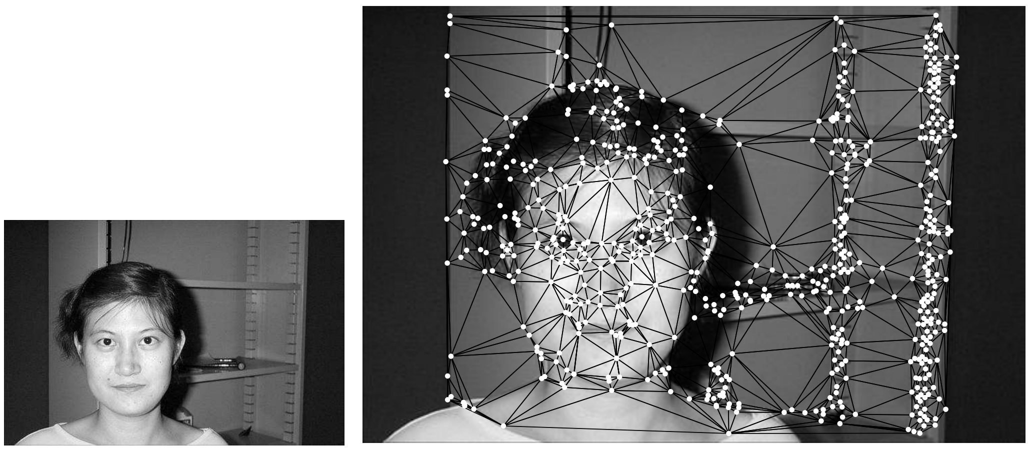


**Figure 3.11:** *Test image (left), and a Delaunay triangulation of all of the SIFT features (right) obtained from the image.*

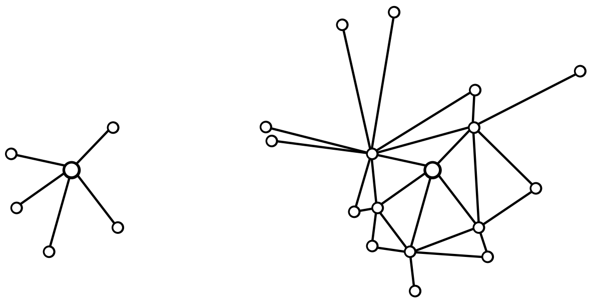


**Figure 3.12:** *Delaunay-triangulated MRF neighborhoods; (a) neighborhoods of degree* $n_d = 1$*, and (b)* $n_d = 2$*, where the feature point being evaluated is a the center of the graph.*

connections cross edges in the test image, but this is avoided since objects can have some internal edges (e.g. faces have nose, mouth and eyes).

### 3.7.4 Computing the Posterior Energy

At the detection stage, the goal is to maximize the posterior probability associated with each feature point, $\mathbf{f}_k$. This is equivalent to minimizing the energy, $E_k$, since $E(p) = -\log(p(.))$. Equation 3.2 expressed in terms of a log energy minimization problem becomes

$$E_k(l_k = 1) = \Lambda_p\{E_k(p_g) + \alpha_d E_k(p_q)\} \quad (3.6)$$

$$E_k(l_k = 0) = \Lambda_n\{E_k(p_g) + \alpha_d E_k(p_q)\} \quad (3.7)$$

where $E_k(p_g)$ us the likelihood energy modeled by the B-G model of the appearance of features, and $E_k(p_q)$ is the prior energy used to inject geometric context to the framework. $\Lambda_p$ and $\Lambda_n$ reflect the ratios of total energy associated with the positive and negative classes respectively, and $\alpha_d$ signifies the relative influence of prior to observed knowledge. The posterior is computed using an iterative local energy minimization process known as Iterated Conditional Modes (ICM) (Besag, 1986) in this work. ICM works by minimizing the energy at each feature site, iteratively converging to a local minimum over all sites. After this process, all of the sparse local features are labeled as positive (i.e. belonging to the object) or negative (i.e. background).

The influence of the prior energy inferring implicit geometric context can be clearly visualized when juxtaposed with the likelihood energies in Figures 3.13 (a-f) in which an example image being evaluated at the detection stage. It is obvious that the final labeling, seen in Figures 3.13 (e) and (f), has been positively influenced by inclusion of the prior. It is also apparent in Figure 3.13 (e) and Figure 3.14 (b) that the Delaunay-based ISI activates a tight network of final positive feature points, $\mathbf{f}_p$, on the object.

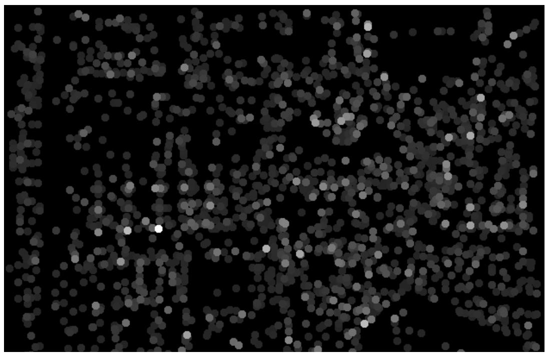
### 3.7.5 Final Rough Segmentation

Following the ISI process, a rough object segmentation can be generated by compositing a normalized Gaussian mask, or "hump" at the center coordinate, $\mathbf{x}_p$, of each feature that has been activated by ISI as positive (i.e. belonging to a face), $\mathbf{f}_p$. Each Gaussian has variance, $\upsilon_p$, proportional to the scale, $\omega_p$, of its $\mathbf{f}_p$. Assuming ISI has been successful, the combined presence of these Gaussian humps will be strong around the facial region in the image, as can be seen in Figure 3.14 (d). Applying a global threshold, $t_p$, yields a rough object segmentation, $M_p$, as can be seen in Figure 3.14 (e). A circle centered on, and with area equivalent to $M_p$ is useful for visualizing the localization, as can be seen in Figure 3.14 (c) and (f). Figure 3.14 (f) compares $M_p$ to an elliptical ground truth mask, $M_{gt}$, that might be used for evaluation in testing. Here the region of agreement between the two masks is colored white, whereas the under and overlap is colored gray.

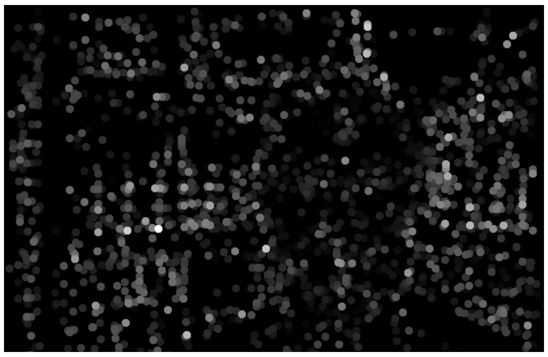
## 3.8 Testing the ISI Framework

The proposed ISI framework is put to the test with a SIFT feature-based (Lowe, 2004) Face Detection task, and *not* an Object Class Detection (OCD) task. Therefore only images with faces are used to test the algorithm, and the goal is to localize and segment the faces in the images. It is worth noting that the goal here is not the achievement of world-beating Face or Object Detection results, but rather to determine if any part of the framework has a useful contribution to these fields.

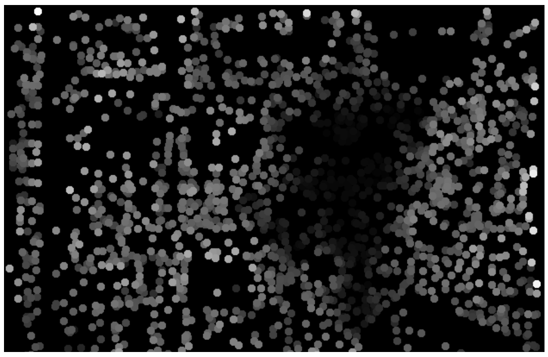
In keeping with the literature with regard to Object Detection with sparse local features



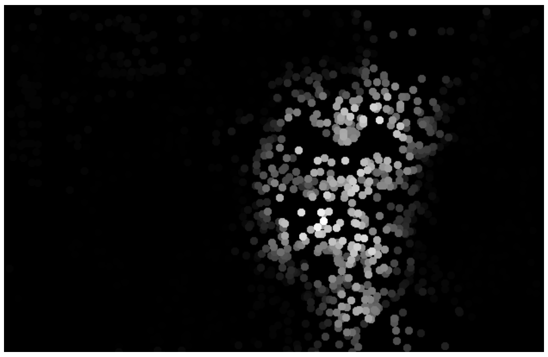
(a) Negative B-G energy



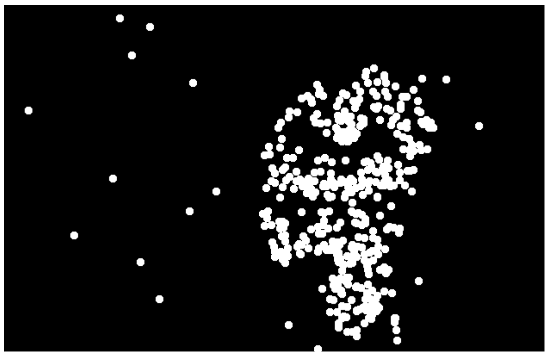
(b) Positive B-G energy



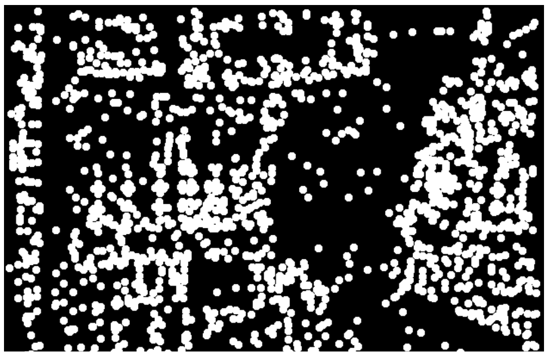
(c) Negative prior energy



(d) Positive prior energy

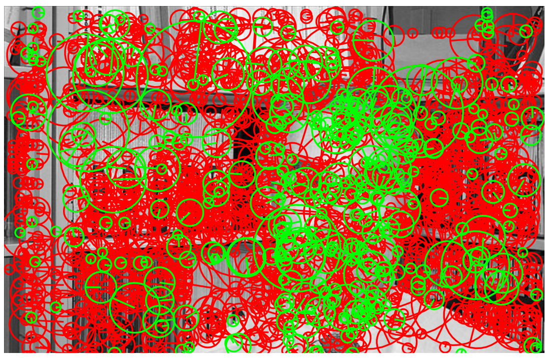


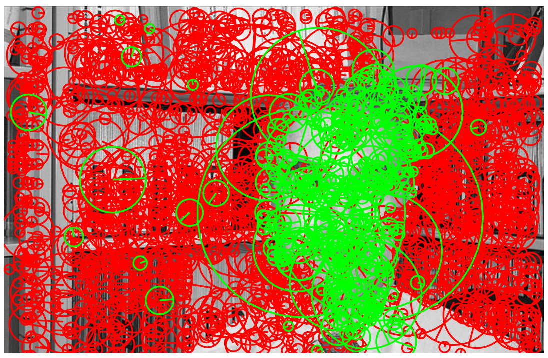
(e) Final positive labels

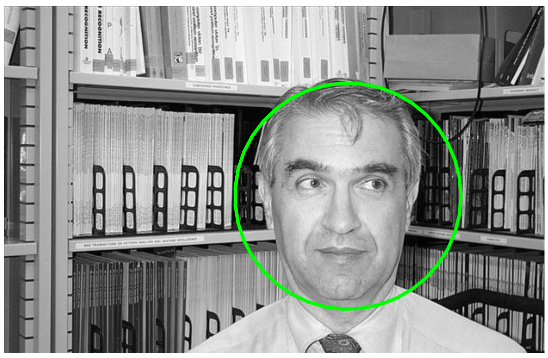


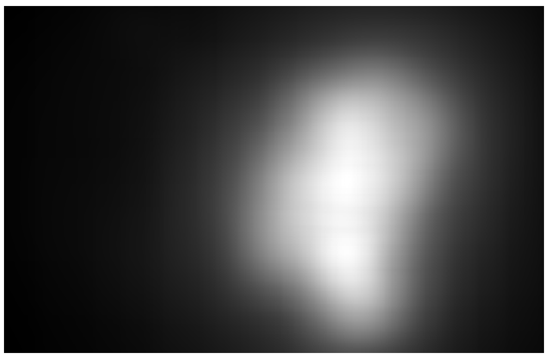
(f) Final negative labels

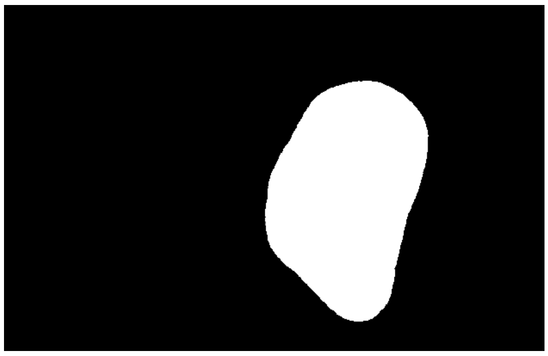
**Figure 3.13:** *Minimizing the posterior energy; (a) negative, and (b) positive B-G likelihood energy, $E_k(p_g)$, (c) negative, and (d) positive prior energy, $E_k(p_q)$, (e) final positive, $E_k(l = 1)$, and (f) negative, $E_k(l = 0)$, binary feature labeling. Here $h = -1.5$, $\Lambda_p = 1$ and $\Lambda_n = 2$, $n_d = 4$, and $\alpha_d = 20$. The test image is seen at the top.*


(a) GentleBoost classification


(b) Final ISI classification


(c) Circled detection


(d) Gaussian segmentation


(e) Segmentation mask, $M_p$

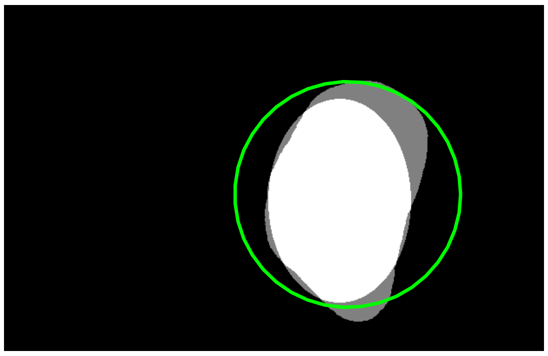

(f) Comparing $M_p$ to $M_{gt}$

**Figure 3.14:** *Detection and segmentation; (a) initial GentleBoost classification based on feature appearance only, and (b) post-ISI classification (face features are green, background is red), (c) circled detection, (d) Gaussian segmentation mask, and (e) thresholded to create a binary rough segmentation mask, $M_p$, and (f) $M_p$ being compared to a ground truth mask, $M_{gt}$ (the elliptical region). Here* $h = -1.5$, $\Lambda_p = 1$ *and* $\Lambda_n = 2$, $n_d = 4$, $\alpha_d = 20$, *and* $t_p = 0.6$.

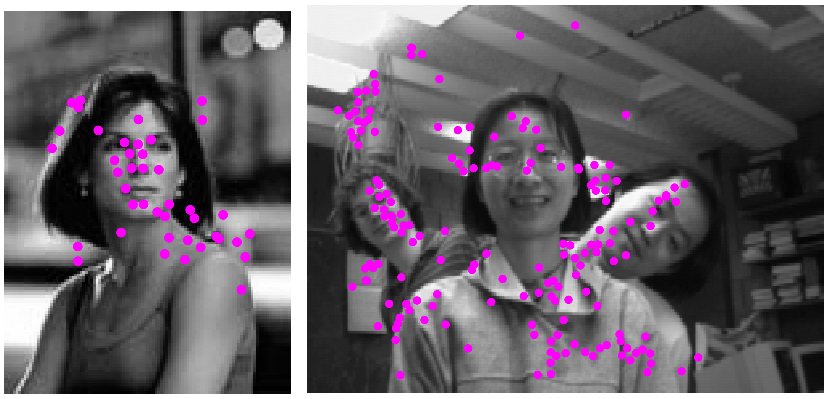

(a) MIT/CMU image (b) MIT/CMU rotated

**Figure 3.15:** *Images from the low-resolution MIT/CMU Face Detection test sets created by (Rowley et al., 1998b); (a) this* $108 \times 144$ *pixel,* 96*dpi image produced* 63 *SIFT features, and (b) this* $320 \times 240$*, pixel* 96*dpi (i.e. dots per inch) example from the rotation invariance test set (added by (Rowley et al., 1998b)) produced* 246 *SIFT features. Default parameters for the SIFT code of (Vedaldi, 2006) were used to obtain the feature points, all of which are marked here in purple (some overlap).*

(see Section 3.4.2), the face set from the Caltech-101 database (Fei-Fei et al., 2004) is deemed most suitable for both training and testing[1]. The test images in MIT/CMU (Rowley et al., 1998b) cannot be used here since they are of too low resolution to generate enough useful SIFT feature points for evaluation, as can be seen in Figure 3.15 (a) and (b). Besides these two face database sets, there are not many other useful options.

Caltech-101 faces is comprised of slightly multi-scale images varying in size from $418 \times 276$ to $628 \times 415$ pixels at 96dpi (i.e. dots per inch) capturing the head and shoulders of subjects in varying lighting, including subjects with beards, glasses, some occlusion and cartoons. Although this database is not thought to be very challenging, it is certainly sufficient to highlight some of the strengths and weaknesses of the proposed algorithm. To properly test for scale and rotation invariance, however, the Caltech-101 images must be systematically scaled and rotated in testing.

### 3.8.1 Training

The set of 435 Caltech-101 face images is split into two separate training and test sets. The face are manually cropped out of the 218 training images for use as the positive set for training the GentleBoost (Vezhnevets & Vezhnevets, 2005) classifier, while 550 unmodified images from the Caltech-101 background database are used as the negative training set. A

[1]Since both the content, and Internet location of this database has changed since this work was carried out, the exact databases that were used for training and testing here are included on the DVD accompanying to this thesis.

(a) Positive set

(b) Negative set

**Figure 3.16:** *Example training images; (a) positive set of cropped faces, and (b) negative set of background images.*

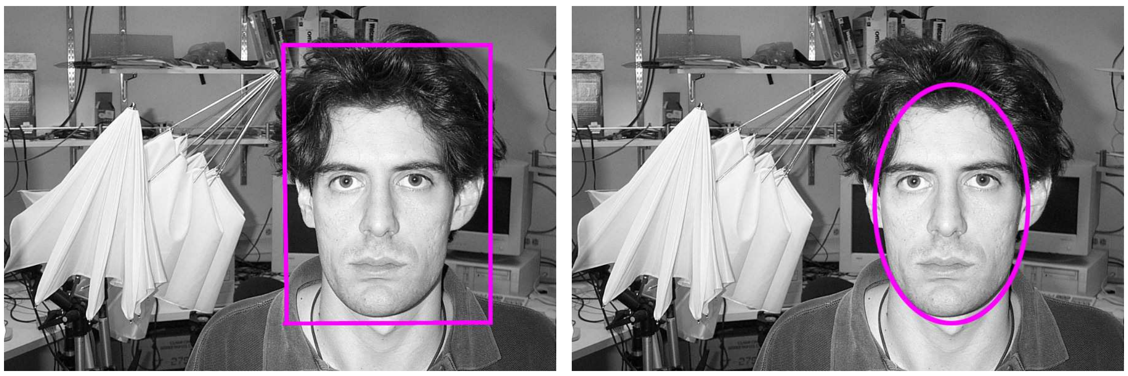
few examples of images from the positive and negative training sets used can be seen in Figure 3.16. Note - for the purpose of visualization - that Figure 3.16 does not truly convey the relative size of positive to negative training images. The small cropped face images range from to $132 \times 212$ to $252 \times 171$ pixels at 96dpi, and the various background images range from $223 \times 147$ at 96dpi to $500 \times 331$ pixels at 192dpi.

The default parameters supplied with the SIFT code of (Vedaldi, 2006) are used to obtain around $40,000$ positive and $142,000$ negative SIFT features from these training sets. GentleBoost is then trained with these features using 400 iterations of boosting. It is worth mentioning once more that only the appearance (i.e. descriptor histograms) of the SIFT features, and not their geometry parameters (i.e. spatial location, scale and orientation) are given to GentleBoost for learning. A detailed explanation of the particulars of GentleBoost is beyond the scope of this thesis, but more information can be found in (Vezhnevets & Vezhnevets, 2005).

### 3.8.2 Testing

At the detection stage, the appearance of the SIFT features, $\mathbf{f}_{k..K}$, obtained from each of 217 face test images are presented to GentleBoost, and the output confidence distribution, $c_{k..K}$, is passed onto the Bayesian framework - along with the geometry parameters of the features - for classification with ISI. The final output of the algorithm is a face localization and rough segmentation for each test image.

A series of ground truth masks have been manually created by the author for testing the algorithm. Each binary mask, $M_{gt}$, encapsulates the outline of the face with a much tighter elliptical mask that the official rectangular Caltech-101 ones, which are thought to



(a) Caltech-101 mask (b) New bespoke mask

**Figure 3.17:** *Comparing ground truth masks; (a) the official Caltech-101 (Fei-Fei et al., 2004) ground truth, and (b) bespoke mask that was created for, and is more suited to this work.*

be large and forgiving (Ponce et al., 2006). The difference between the the official and new bespoke ground truth masks can be seen in Figures 3.17 (a) and (b) respectively, where the purple boundary is encapsulating the facial region. As can be seen in Figures 3.17 (b), the new masks are more conducive to benchmarking a proper face segmentation, which the ISI-based Face Detection algorithm has been designed to achieve. These bespoke ground truth masks are included on the DVD accompanying to this thesis.

After ISI is complete, the rough segmentation mask, $M_p$ and ground truth mask, $M_{gt}$ are compared for each image, and metrics are computed to determine if the face was detected and/or and segmented accurately. As discussed previously, Figure 3.14 (b) shows the comparison of $M_{gt}$ with the $M_p$ calculated for the test image shown in to Figure 3.14 (a). White pixels indicate true positive regions, and gray pixels reveal false positive or false negative regions, outside or within $M_{gt}$ respectively.

### 3.8.3 Test Metrics

Some metrics are needed for evaluating the detection and segmentation results, and these will now be defined. Since the Caltech-101 database is used in experiments, the benchmarking paradigms of the Object Detection research community are adopted over those of the Face Detection community. However, most of Object Detection algorithms that were discussed in Section 3.4.2 end up comparing the predicted *bounding box* of a detected object to the bounding box of the official Caltech-101 ground truth mask (as can be seen in Figure 3.17 (b)). Since the ISI framework produces a much tighter segmentation output, $M_p$, and a new strict ground truth mask, $M_{gt}$, has been designed especially, these metrics have to be interpreted in terms of $M_p$ and $M_{gt}$.

Interpreting, therefore, a metric that is used in testing algorithms of (Mikolajczyk et al., 2006; Opelt et al., 2006b) and (Opelt et al., 2006a), a *True Detection*, $TD$ is defined as follows

$$\frac{\text{area}(M_p \cap M_{gt})}{\text{area}(M_p \cup M_{gt})} > 0.5 \Rightarrow TD \tag{3.8}$$

and *Percent True Detection*, $\%TD$, is defined as

$$\%TD = \frac{\#TD}{N_T} \tag{3.9}$$

as in the percentage of times this criterion is met in an experiment with the set of $N_T = 217$ test images. Since it is not conducive here to measure a *False Detection* in terms of a false positive bounding box as in other Object Detection algorithms, the metric of *False Detection Area*, $FDA$, has been created especially for evaluating the ISI algorithm, and is defined as

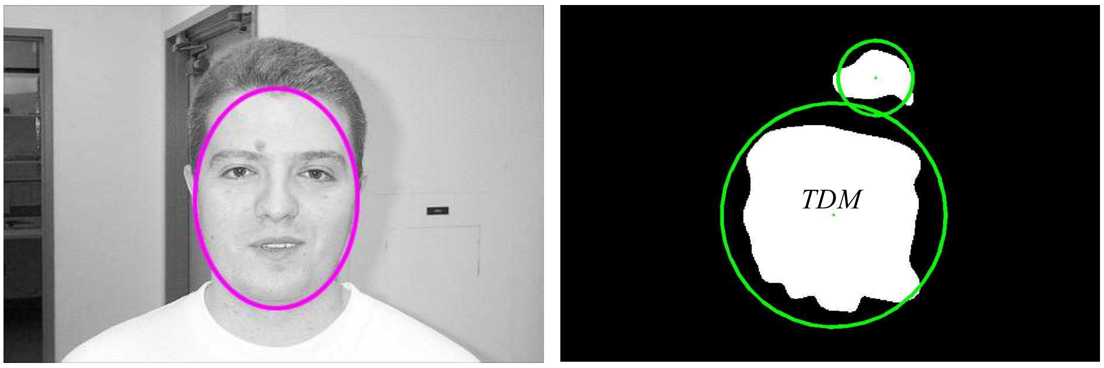
$$FDA = \text{area}\{M_p \cap (M_{gt} \cup TDM)^c\} \tag{3.10}$$

where $TDM$ is the *True Detection Mask* of any $TD$ recorded for the image. The meaning of $TDM$ is that it is any connected region within the binary segmentation mask, $M_p$, that has contributed to a $TD$ as defined with the ground truth mask $M_{gt}$ in Equation 3.8. Following from this, *Percent False Detection Area*, $\%FDA$, can be evalatued over the entire set of $N_T = 217$ test images as

$$\%FDA = \frac{\sum_j^{N_T} FDA_j}{\sum_j^{N_T} \text{area}\{(M_{gt_j} \cup TDM_j)^c\}} \tag{3.11}$$

where the area of (i.e. number of pixels in) any other connected region in the $M_p$ generated for test image $j$ that is not part of $TDM_j$ is added to $FDA_j$. The idea can be seen in the less-than-perfect segmentation mask of Figure 3.18 (b), where the $TDM$ is the connected binary region that has been counted as part of a $TD$ with regard to the ground truth mask, $M_{gt}$, seen in Figure 3.18 (a). The other binary connected region in Figure 3.18 (b) is counted as an error, and its area added to the measure of $FDA$ for this test image.

This last metric is a good way to quantify detection errors in the ISI algorithm. It is quite a harsh metric in comparison with the error metrics used in the Object Detection community, since Object Detection algorithms tend to record a $FD$ - for classification, or more rarely detection purposes - in terms of a false positive bounding boxes, but not in terms of single false positive features or pixels (with the exception of bag-of-features OCD algorithms (Nowak et al., 2006) that are not related to this work). In Object Detection, *Percent False Detection* is then sometimes measured as the percentage of false positive



(a) Ground truth mask (b) Segmentation mask

**Figure 3.18:** *The concepts of $TDM$ and $FDA$; (a) the bespoke ground truth mask, $M_{gt}$, and (b) segmentation mask, $M_p$, with the binary connected $TDM$ labeled. The other binary connected region is counted as $FDA$.*

bounding boxes out of the total number of bounding boxes predicted. The author finds this error metric lacking, since an object detector could potentially encapsulate the entire image with one false positive bounding box, and this error would only be given as much weight as an error in which only one pixel in the image was marked as a false positive. $\%FDA$ as defined here is therefore a more reresentative metric since it is formulated as the percentage of false positive segmented *pixels* out of the total number of pixels in the image that are not contained within any $TDM_j$, produced by the algorithm in testing.

Segmentation results are evaluated by interpreting the metrics of Recall, $R$, and Precision, $P$, that were previously mentioned in Section 3.4.2. Since this work is concerned with the actual localization of objects (and not just OCD), and since it is not a bounding box-producing algorithm like those discussed in Section 3.4.2, these metrics must be completely redefined as pixel-wise measurements of segmentation accuracy and error with regard to the very strict ground truth masks, $M_{gt}$ as follows

$$R = \frac{\sum_j^{N_T} area(M_{p_j} \cap M_{gt_j})}{\sum_j^{N_T} area(M_{gt_j})} \tag{3.12}$$

$$P = \frac{\sum_j^{N_T} area(M_{p_j} \cap M_{gt_j})}{\sum_j^{N_T} area(M_{p_j})} \tag{3.13}$$

These metrics are not quite comparable, therefore, to the measures of Precision and Recall defined for many of the bounding box-producing object class detectors that were discussed in Section 3.4.2, nor are they conducive to measuring Equal Error Rate (EER). The author is not concerned with OCD, or with beating state-of-the-art Face or Object Detection results here. These metrics of $R$ and $P$ have been specially defined for object localization, and to investigate whether the ISI framework can improve on rough segmentation results that would achievable by a local feature-based point-wise machine learning classifier alone.

### 3.8.4 Choosing Parameters

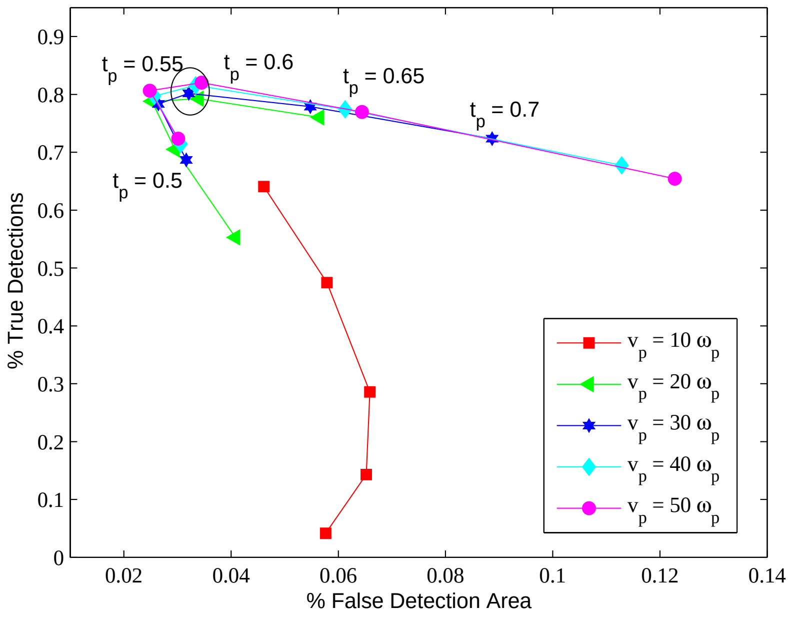
Since the ISI framework has many parameters, some initial trials are needed to choose suitable values for $t_p$, $v_p$, $h$, $\Lambda_p$ and $\Lambda_n$ (see Section 3.7 for definitions) before defining experiments for detection accuracy and invariance. In order to do this, the multi-parameter space is explored. Figure 3.19 demonstrates one such trial in which values of $t_p$ are varied with $v_p$ at the point where $h = -1.5$, $\Lambda_p = 1$ and $\Lambda_n = 2$, $n_d = 3$, $\alpha_d = 20$. Figure 3.19 (top) shows $\%TD$ versus $\%FDA$, and Figure 3.19 (bottom) plots $R$ versus $1 - P$ measured over all test images. At the circled points in the graphs, $t_p = 0.6$ with $v_p = 30\omega_p$ clearly represents a good trade-off between $\%TD$ and $\%FDA$, $R$ and $P$ overall. This particular trial was preceeded by two others varying parameter $h$ with $\lambda = \Lambda_n/\Lambda_p$ and vice versa. Similar graphs comparing $\%TD$ versus $\%FDA$, and $R$ versus $1 - P$ for these two trails are included in Appendix B.

### 3.8.5 Experiment 1

To test the ISI framework for detection and segmentation accuracy, the Face Detection task is performed on the all of the Caltech-101 test images while varying the degree of the Delaunay neighborhood, from $n_d = 1$ to $n_d = 5$ (see Section 3.7.3) in increments of one, with the prior energy weight varying from $\alpha_d = 0$ to $\alpha_d = 100$ (see Equations 3.6 and 3.7) in increments of 20. Varying $n_d$ and $\alpha_d$ in this way is an investigation of influence of the spatial prior (see Equation 3.2) in the Bayesian framework underlying the algorithm. Note that when $\alpha_d = 0$, the prior energy (i.e. ISI) is absent from the solution, and results are based purely only the likelihood obtained from the GentleBoost classifier. Therefore this experiment can be used to determine if ISI constitutes an improvement on the detection and segmentation results that could be gleaned from a point-wise appearance-based feature classifier alone. The results of this test can be visualized in plots of $\%TD$ versus $\%FDA$, and $R$ versus $1 - P$, over all test images in the database. These are shown in Figure 3.20 (top) and (bottom) respectively, with $n_d$ varying with $\alpha_d$.

#### Discussion

The circled point in Figure 3.20 (top) represents the best result of Experiment 1 in terms of detection and segmentation accuracy. Very good detection results of $\%TD = 87.6\%$ versus $\%FDA = 0.03\%$ are observed at the point where $n_d = 3$ and $\alpha_d = 20$. In Figure 3.20 (top) it is also clear to see that when the spatial prior energy has no influence (i.e. $\alpha_d = 0$), reasonable rates of $\%TD$ and $\%FDA$ can be achieved. When $\alpha_d > 0$ however, the



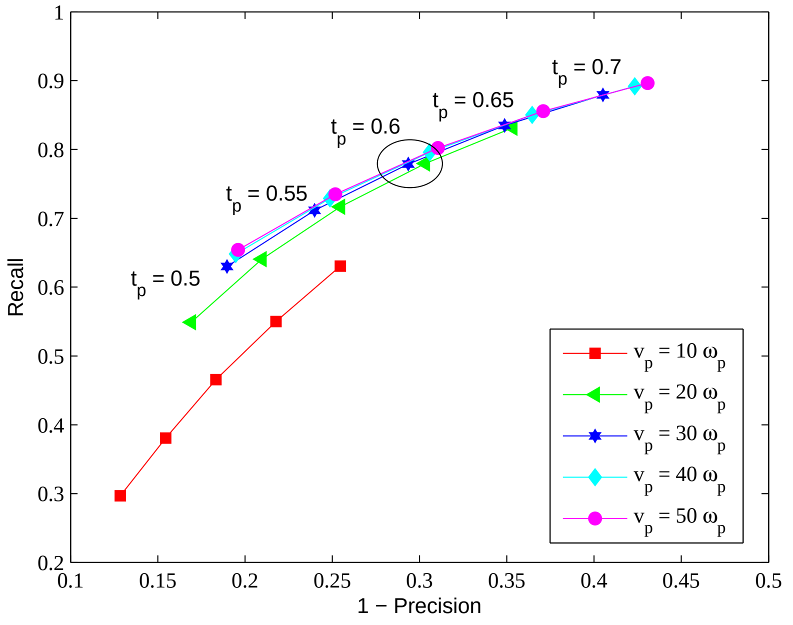


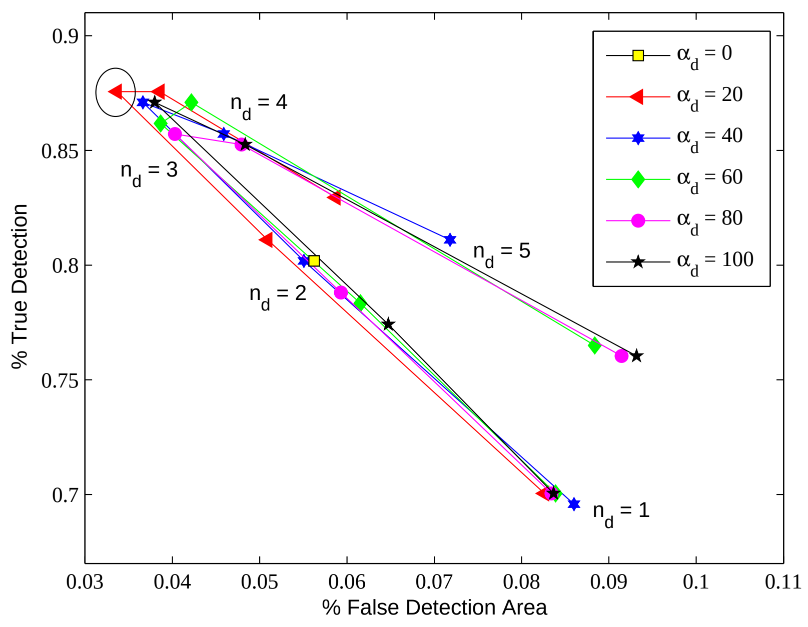
**Figure 3.19:** *Choosing suitable values for the threshold, $t_p$, and variance of the Gaussian projections, $v_p$, in creating the rough face segmentation masks; (top) $\%TD$ versus $\%FDA$, and (bottom), $R$ versus $1-P$. Other parameters are set at $h=-1.5$, $\Lambda_p=1$ and $\Lambda_n=2$, $n_d=3$, $\alpha_d=20$.*

degree of $n_d$ can influence a degradation ($n_d < 2$) or real improvement ($n_d > 2$) in these results, whereas the actual value of $\alpha_d$ is less critical. ISI, therefore, is important in the task of detection, and injection of contextual geometric information via the spatial prior can improve on segmentation results that would be possible with a point-wise machine learning classifier alone.

Interestingly, $\%TD$ plummets while $\%FDA$ soars for $n_d > 4$ in Figure 3.20 (top). This is probably due to the fact that the bespoke ground truth masks, $M_{gt}$, created especially for this algorithm, encapsulate the faces more tightly and are much stricter than those included with the Caltech-101 database (Fergus et al., 2003) (see Figure 3.17). What could be happening with $n_d > 4$ is that the area of the final region masked by the segmentation, area($M_p$), becomes too large to be counted as a $TD$ with regard to $M_{gt}$ (see Equation 3.8). These large, missed detections are then accumulated in $\%FDA$. An explanation for this could be that SIFT features are often detected at the boundary of facial regions in training, and perhaps ISI over larger Delaunay neighborhoods with $n_d > 4$ is activating more of these boundary features and classifying them as positive features, $\mathbf{f}_p$, in testing. Then the Gaussian humps composited on their center coordinates, $\mathbf{x}_p$, (see Section 3.7.4 for explanation) might contribute to the segmentation mask, $M_p$, overshooting the edges of the strict ground truth mask, $M_{gt}$, especially if the scales of those features are large.

Two good results in terms of segmentation accuracy are shown circled in Figure 3.20 (bottom). While $P = 0.67$ versus $R = 0.885$ are best with $n_d = 4$ and $\alpha_d = 20$, a smaller Delaunay neighborhood of $n_d = 3$ works almost as well, producing $P = 0.67$ versus a still acceptable $R = 0.866$. Figure 3.20 (bottom), also shows that with prior energy and ISI are involved (i.e. $\alpha_d > 0$), segmentation Recall, $R$, improves as Precision, $P$, takes an increasing hit. This effect is probably down the same problem of positive features, $\mathbf{f}_p$, being classified at the very boundaries of objects, coupled with the fact that the ground truth masks, $M_{gt}$, are very strict, as discussed in the previous paragraph. Since the goal here is to achieve only a rough segmentation, the $R$ versus $P$ trade-off with these masks is probably quite acceptable for some applications, like face blurring or abstract content-based stylization, as can be seen in Figure 3.8.

The output of this ISI algorithm is a detection and approximate localization of face objects in images containing faces, and not a detection of object class over many object categories. Therefore, unlike the other Object Detection and OCD algorithms that were discussed in 3.4.2, this algorithm does not produce a large detection bounding box, but a specific object segmentation. Hence it is difficult to compare the Recall, $R$, versus Precision, $P$, with corresponding versions of $R$ and $P$ formulated for these other Object Detection algorithms since the definition of these metrics is completely situational, and here they are defined in with regard to a different - and stricter - ground truth mask. With regard to Face Detection benchmarking, the algorithm of (Viola & Jones, 2004) quotes a *Correct Detection Rate* versus *Number of False Positives* in experiments where parameters of the



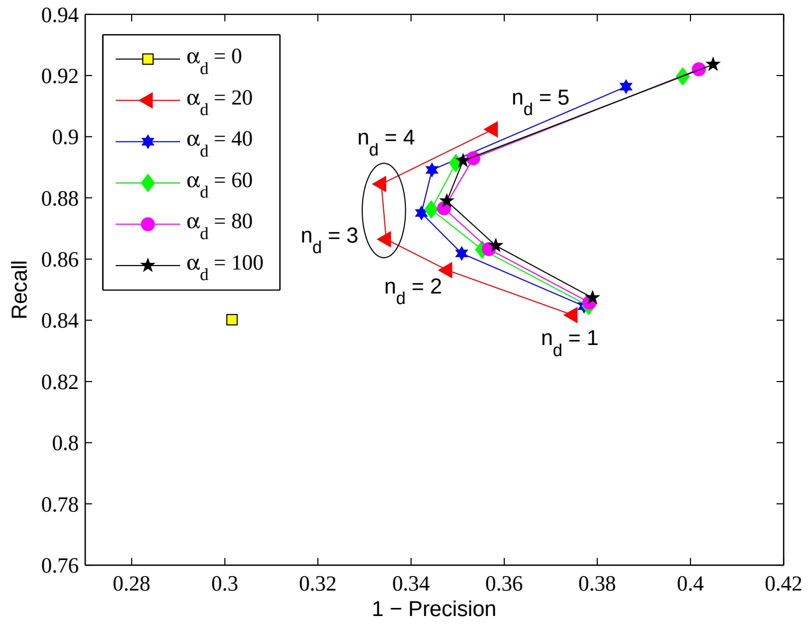


**Figure 3.20:** *Experiment 1 varying Delaunay neighborhood degree, $n_d$, with the relative influence of spatial prior energy, $\alpha_d$; (top) $\%TD$ versus $\%FDA$, and (bottom), $R$ versus $1-P$. Other parameters are set at $h = -1.5$, $\Lambda_p = 1$ and $\Lambda_n = 2$, $t_p = 0.6$, $v_p = 30\omega_p$. Here $\alpha_d = 0$ represents a single trial in which no spatial prior energy - and hence no Delaunay neighborhood computation - was included in the tasks of Face Detection and segmentation.*

system are varied, and these metrics are in terms of numbers of sub-windows evaluated. Two slightly different variations of the algorithm described in (Viola & Jones, 2004) are cited as detecting between 81.1% and 93.9% of true positive sub-windows, for between 10 and 422 false positive sub-windows, in experiments using the MIT/CMU face database (Rowley et al., 1998b). It is hard to say if the face detector presented here can be compared directly with the metrics of Percent True Detection, $\%TD$, versus False Detection Area, $\%FDA$. If it can, the high point of $\%TD = 87.6\%$ versus $\%FDA = 0.03\%$ when $n_d = 3$ and $\alpha_d = 20$ seems quite acceptable.

### 3.8.6 Experiment 2

The invariance of the algorithm is investigated by fixing the values of $n_d = 3$ and $\alpha_d = 20$, and applying the detector to systematically rotated and scaled versions of the Caltech-101 test image set - specifically six in-plane rotations from $0°$ to $300°$ in increments of $60°$, and five scalings from 0.6 to 1.4 times the original spatial dimensions of the test images in increments of 0.2. Note that these transforms are only for the purpose of testing the invariance of the framework, and not a necessary stage in the ISI framework. This action is necessary due to the nature of the Caltech-101 test images, which are all upright and nearly uniform in scale. The results of carrying out this experiment are shown in Table 3.21. The Table lists the values of $\%TD$ versus $\%FDA$, and $R$ versus $1 - P$ that were obtained by performing tests on all of the $N_T = 217$ test images over each of the six rotations for each image scaling. Also listed are the mean value of these metrics over all rotations for each image scaling.

#### Discussion

Some good results of $\%TD$ versus $\%FDA$ and $R$ versus $1 - P$ for Experiment 2 are highlighted in bold in Table 3.21. Table 3.21 also demonstrates that the ISI framework is almost perfectly rotation invariant since similar values of $\%TD$, $\%FDA$, $R$ and $P$ are observed over all test image rotations at a fixed scale. It is also quite robust to test image scaling within a certain range. Good results are observed for scalings in the range of almost 0.8 to 1.4 times original image dimensions, with better results recorded for image upscaling than downscaling. The problem with downscaling could be due to the fact that the quantity of SIFT features needed to support all aspects of the ISI framework are harder to obtain in low-resolution images, as previously discussed in Section 3.4.1 and in this Section.

The results of $\%TD$, $\%FDA$, $R$ and $P$ do fluctuate over all scales, however, and this problem can probably be attributed to fixing the extent of the Delaunay neighborhood to $n_d = 3$ with regard to the spatial prior (see Equation 3.7.3). It is this part of the ISI framework that could be affecting the scale invariance. A potential solution to this problem

| Scale | $\%TD$ $0°, 60°, 120°, 180°, 240°, 300°$ | Mean $\%TD$ | $\%FDA$ $0°, 60°, 120°, 180°, 240°, 300°$ | Mean $\%FDA$ |
|---|---|---|---|---|
| 0.6 | 51.15, 55.3, 55.3, 52.07, 54.84, 50.23 | 53.2 | 0.15, 0.07, 0.06, 0.14, 0.06, 0.07 | 0.09 |
| 0.8 | 73.73, 76.96, 74.65, 74.19, 79.26, 72.81 | 75.27 | 0.08, 0.03, 0.03, 0.07, 0.03, 0.03 | 0.05 |
| 1.0 | 87.56, 84.33, 81.11, 82.49, 83.41, 81.57 | 83.41 | 0.03, 0.02, 0.03, 0.05, 0.02, 0.03 | 0.03 |
| 1.2 | 87.56, 86.18, 87.1, 86.18, 87.56, **88.5** | **87.17** | 0.03, 0.02, 0.02, 0.03, 0.02, **0.01** | **0.02** |
| 1.4 | 84.79, 85.71, 86.18, 82.03, 85.71, 86.64 | 85.18 | 0.03, 0.01, 0.01, 0.04, 0.01, 0.01 | 0.02 |
| Scale | $R$ $0°, 60°, 120°, 180°, 240°, 300°$ | Mean $R$ | $1-P$ $0°, 60°, 120°, 180°, 240°, 300°$ | Mean $1-P$ |
| 0.6 | 0.85, 0.9, 0.86, 0.83, 0.88, 0.86 | 0.86 | 0.42, 0.4,0.4, 0.44, 0.39, 0.42 | 0.41 |
| 0.8 | 0.91, 0.93, 0.93, 0.89, 0.92, 0.92 | 0.92 | 0.34, 0.33, 0.35, 0.36, 0.35, 0.35 | 0.35 |
| 1.0 | 0.87, 0.9, 0.9, 0.87, 0.91, 0.9 | 0.89 | 0.33, 0.33, 0.36, 0.34, 0.33, 0.34 | 0.34 |
| 1.2 | 0.87, **0.91**, 0.9, 0.87, 0.91, 0.9 | **0.89** | 0.3, **0.3**, 0.3, 0.29, 0.3, 0.31 | **0.3** |
| 1.4 | 0.82, 0.85, 0.85, 0.82, 0.86, 0.86 | 0.84 | 0.27, 0.28, 0.27, 0.28, 0.28, 0.28 | 0.28 |

**Figure 3.21:** *Experiment 2 for invariance; $\%TD$ versus $\%FDA$, and $R$ versus $1-P$ measured over various scales and in-plane rotations of the Caltech-101 test images. Parameters are $h = -1.5$, $\Lambda_p = 1$, $\Lambda_n = 2$, $t_p = 0.6$, $v_p = 30\omega_p$, $n_d = 3$, and $\alpha_d = 20$.*

might involve an extension to the algorithm that could vary the extent of $n_d$ relative to the scale of the feature being evaluated, $\omega_p$, or combine results over a range of $n_d$ instead. This is an interesting problem for future work. Some of the faces that were detected in Experiment 2 (including one mistake), are shown circled in Figure 3.22. A typical rough segmenation result is also shown at the bottom of Figure 3.22.

The framework was used to experiment with some arbitrary test images that were quickly grabbed from an image search on the Internet. With some tweaking of parameters within the ISI framework, some interesting results were achieved, and these can be seen in Figures 3.23, 3.24, 3.26 and 3.25. Interestingly, SIFT features from hands are misclassified as features from faces in Figures 3.23 and 3.26, probably due to the fact that the SIFT feature descriptors are successfully capturing the appearance of the skin texture that is present on both hands and faces. Figures 3.24 and 3.26 demonstrate how the ISI can improve on results that would be possible using a point-wise machine learning classifier alone. These Figures also show that the algorithm is capable of detecting more than one distinct face in an image, but that bounding the regions segmented as faces with an arbitrary shape (e.g. a circle in this case) is unrepresentative of the underlying state of the feature-wise classification or rough segmentation, as in Figure 3.26 especially. That is why a bounding circle it is useful purely for visualizing the approximate location of face detections here, but it is never compared with a similarly ambiguous ground truth mask in benchmarking test metrics.

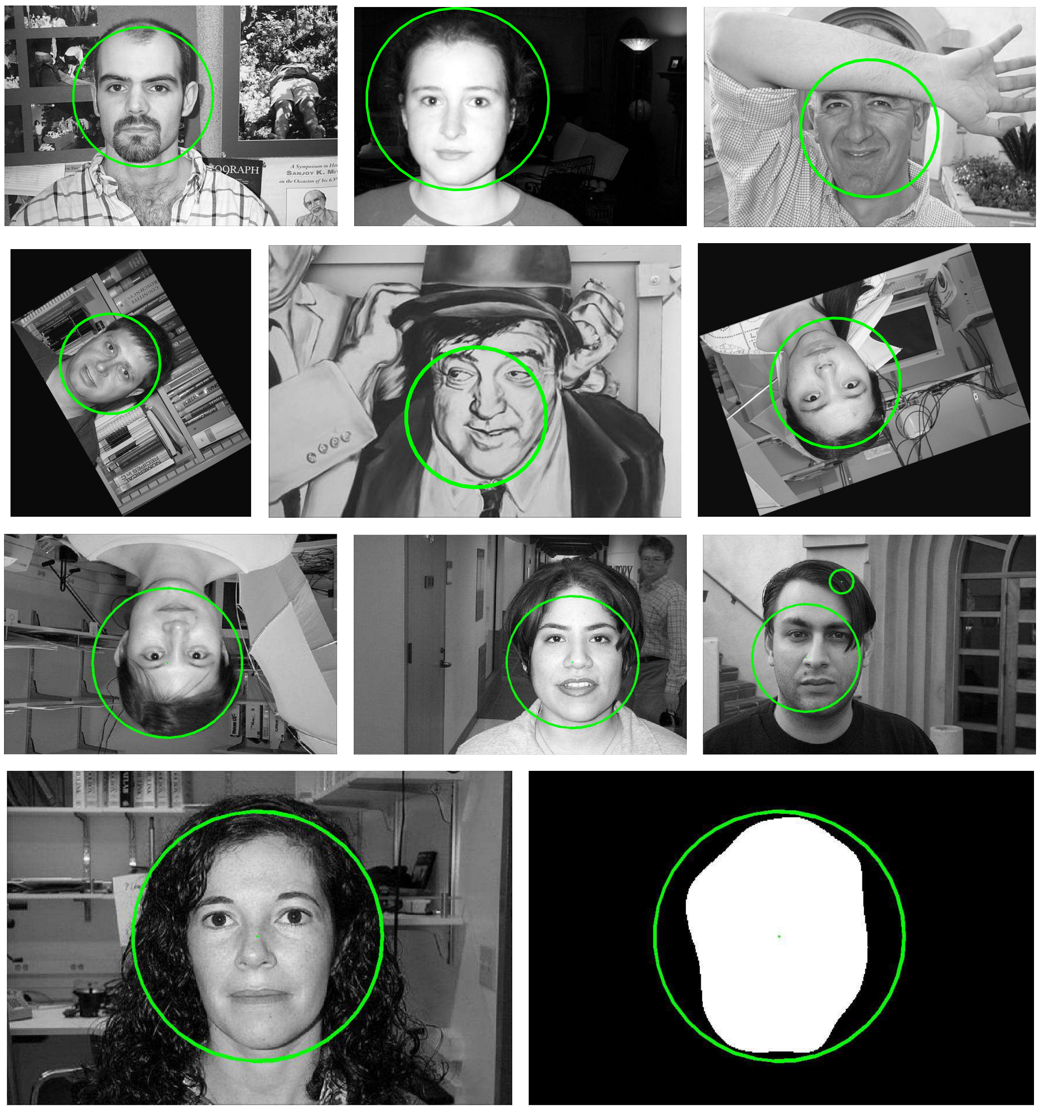


**Figure 3.22:** *Experiment 2 for invariance; various circled face detections over various rotations and scalings, and one rough segmentation result.*

## 3.9 Future Work

Further research could focus on trying to improve on the invariance, training or testing of the proposed face detector. However, the concept of ISI alone is useful in many applications of sparse local feature-based classification, and particularly in the field of Object Detection. Since the use of sparse local features is popular in this field, future work should focus on developing better models and techniques for spatial inference. Delaunay triangulation has been found to be useful for meshing in this instance, but alternative meshing and graphing schemes could be explored. The scheme for spatial inference could take account of the scale attribute of features, such that ISI becomes a multi-scale process, or that the geometric

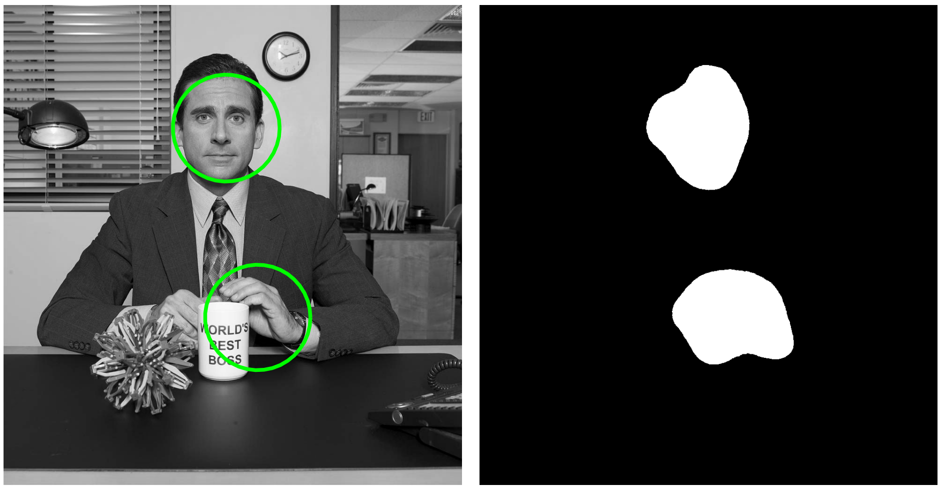


**Figure 3.23:** *Random Internet test image; (left) circled detections, and (right) rough segmentation mask. Parameters are set at* $h = -1.5$, $\Lambda_p = 1$ *and* $\Lambda_n = 2$, $t_p = 0.6$, $\upsilon_p = 30\omega_p$, $\alpha_d = 20$ *and* $n = 3$

grouping of features occurs in some scale-space.

A simple technique for producing a rough object segmentation mask from sparse local features was presented in Section 3.7.5, but this technique could definitely be improved. A natural progression is the exploration of sparse-to-dense or user-assisted inference techniques to obtain a better segmentation matte. The sparse-to-dense geodesic inference work of (Ring & Pitié, 2009), and user-assisted feature correspondence matching technique of (Ring & Kokaram, 2009) are good examples of these techniques at work in the tasks of Motion Estimation and invariant Object Detection respectively. Ideas from these algorithms would also be useful here for geometric inference and segmentation in the task of Face Detection.

## 3.10 Conclusion

A novel algorithm for leveraging the implicit geometry of sparse local features such as SIFT for the purposes of Face and perhaps Object Detection has been presented in this Chapter. The real-valued output of a typical point-wise machine learning classifier is used as a likelihood in the Bayesian framework underlying this algorithm, which has been named Implicit Spatial Inference (ISI) by the author. The main contribution, however, is a novel scheme for the injection of prior contextual geometric information following an initial point-wise sparse local feature-based classification. Working with the implicit geometry of sparse local features is in complete contrast to the rigid geometric shape models underlying many of the state-of-the-art object detectors. ISI is beneficial in that it requires no offline geometry training and is conducive to invariance. A Face Detection task has been used to test

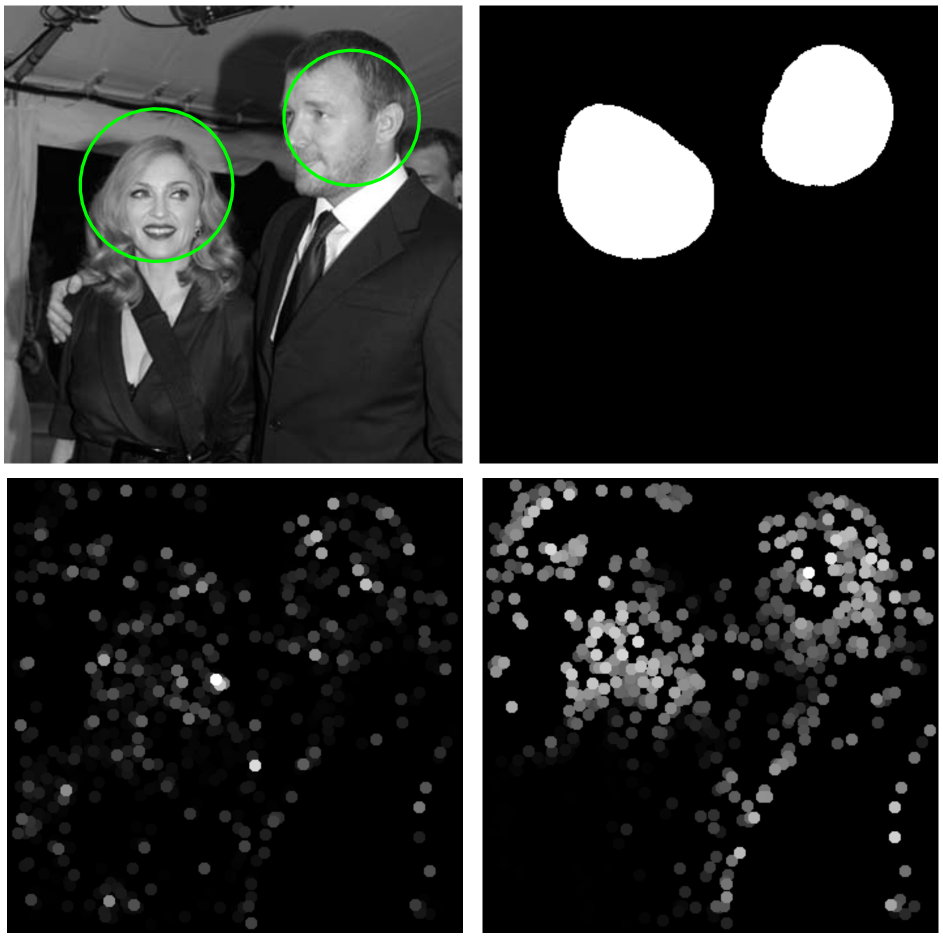


**Figure 3.24:** *Random Internet test image; (top left) circled detections, (top right) rough segmentation mask, (bottom left) the point-wise positive likelihood energy derived from GentleBoost alone (see Equations 3.3 and 3.4), and (bottom right) the ISI-generated positive posterior energy (see Equation 3.6). Parameters are set at* $h = -1.5$, $\Lambda_p = 1$ *and* $\Lambda_n = 1.7$, $t_p = 0.65$, $v_p = 30\omega_p$, $\alpha_d = 100$ *and* $n = 3$.

the algorithm with regard to detection and segmentation accuracy, and invariance. It is apparent from the results of these experiments that ISI can greatly improve on the detection and segmentation results that could be gleaned from the use of a point-wise machine learning classifier alone. Unfortunately, it was found that existing Face and Object Detection benchmarking test databases are lacking, especially with regard to high resolution images that are suitable for sparse local feature-based algorithms, and for testing rotation, scale and pose invariance. Furthermore, ground truth measurements and the metrics used to evaluate results are vague and confusing, and it is therefore extremely difficult to put the results gleaned from the new algorithm in context with the state-of-the-art. However, in terms of the bespoke ground truth masks and benchmarking metrics that have been

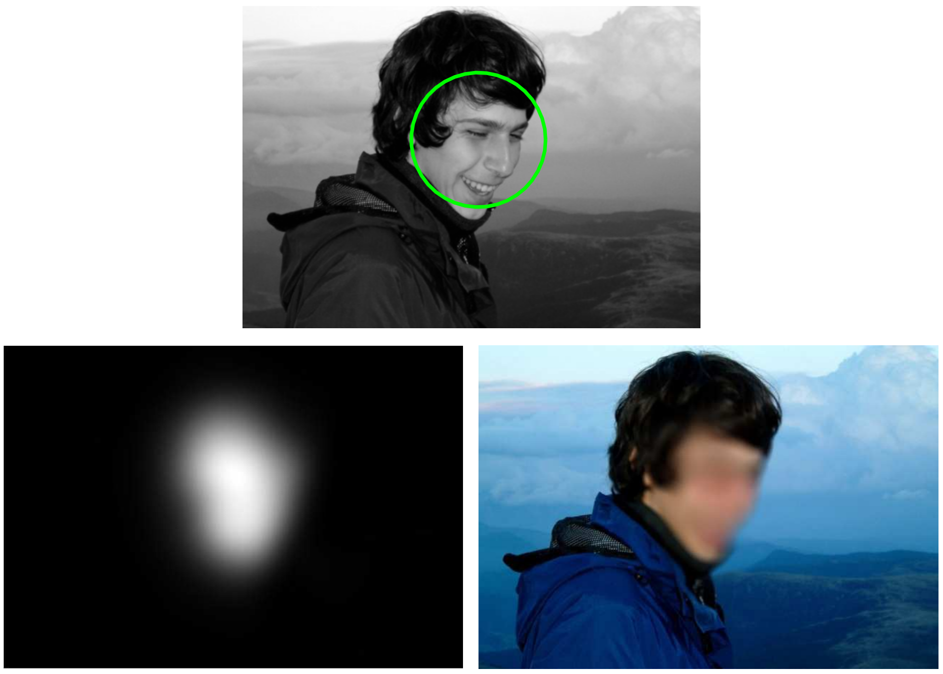


**Figure 3.25:** *Random Internet test image; (top) circled detections, and (bottom left) blur mask created from rough segmentation mask, and (bottom right) the mask used to hide identity. Parameters are set at* $h = -1.5$, $\Lambda_p = 1$ *and* $\Lambda_n = 2$, $t_p = 0.7$, $v_p = 30\omega_p$, $\alpha_d = 20$ *and* $n = 3$

created especially by the author for evaluating this work, the largely invariant detection and segmentation results of this algorithm are very promising. Furthermore, its rough segmentation output could be useful for seeding a finer segmentation, or for the content-based filtering of images.

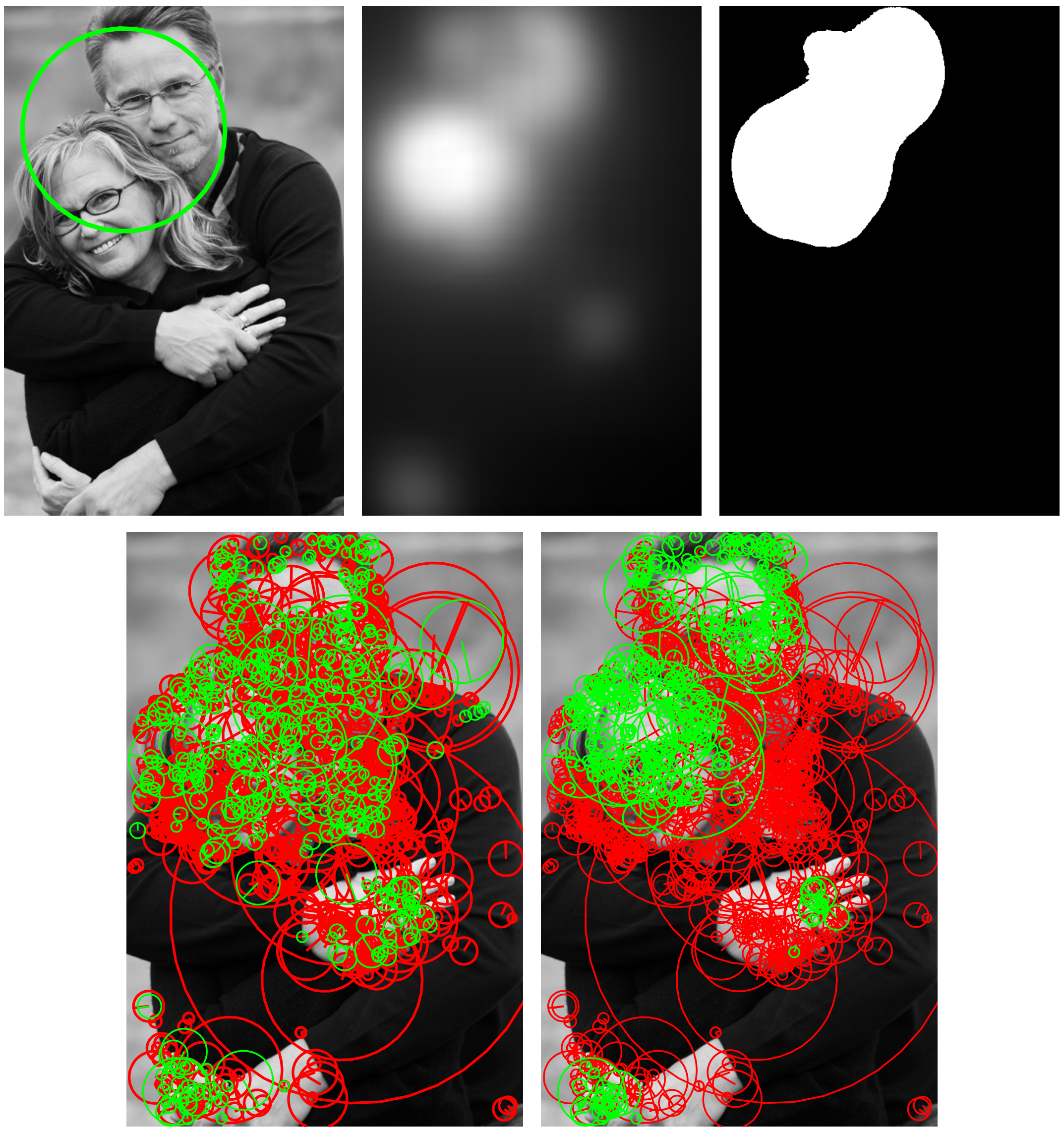


**Figure 3.26:** *Random Internet test image; (top left) circled detection, (top center) positive feature point Gaussian humps (see Section 3.7.4), (top left) rough segmentation mask, (bottom left) initial SIFT feature classification with GentleBoost (see Equation 3.3), and (bottom right) final ISI classification. Positive and negative features are green and red respectively. Parameters are set at* $h = -1.45$, $\Lambda_p = 1$ *and* $\Lambda_n = 1.7$, $t_p = 0.5$, $v_p = 30\omega_p$, $\alpha_d = 50$ *and* $n = 3$

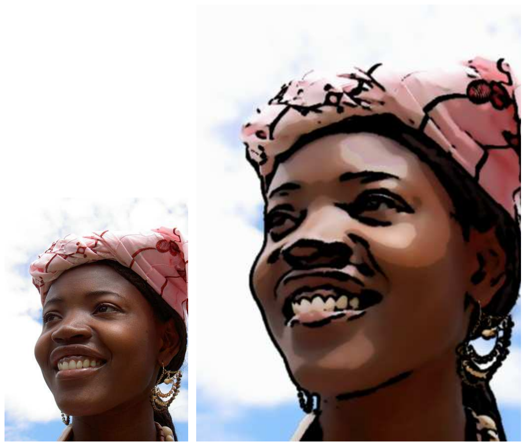
# 4
# On the Stylization of Visual Media

This Chapter presents an introduction to the stylization of visual media. This will consist of an examination of the field of Non-Photorealistic Rendering (NPR), with particular emphasis on the sub-genre of Stroke-Based Rendering (SBR). The methodology and tools constituting the state-of-the-art in this research area will be examined, together with some observations and motivations for novelty in the field. One of the strongest motivations that arises is the idea of a content-based stylization of videos containing head shots of people (e.g. home videos), such that the semantic regions defined by the skin and/or faces of the people in these videos can be stylized more carefully or differently than the less important background regions.

## 4.1 NPR in Context

There has been much research in the area of computer-driven artistic rendering of visual media since the early 1990s. Collected under the title of Non-Photorealistic Rendering (NPR) - a term probably coined in (Winkenbach & Salesin, 1994), the goal is usually the manipulation of digital visual media for aesthetic purposes.

The field seemed to emerge from the desire to simulate real-life painting or drawing effects artificially. NPR has been used to recreate painterly styles such as impressionism (Haeberli, 1990; Meier, 1996; Litwinowitz, 1997; Hertzmann, 1998; Shiraishi & Yamaguchi, 2000; Santella & DeCarlo, 2002; Hays & Essa, 2004; Yang & Yang, 2006; Park & Yoon, 2008), cubism (Haeberli, 1990; Klein et al., 2002) and surrealism (Collomosse & Hall,


(a) Cartoon styling

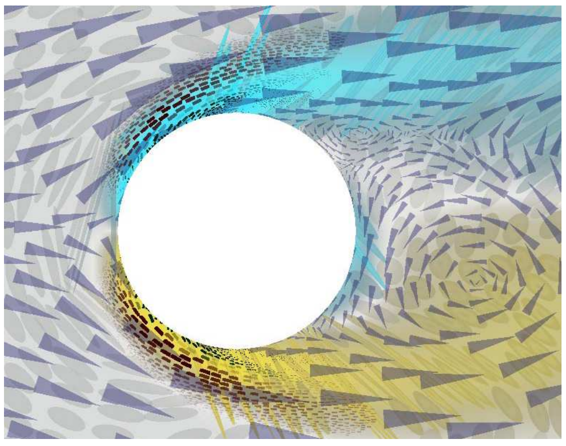

(b) Scientific visualization

**Figure 4.1:** *Examples of NPR; (a) photo stylization with ToonIt! plug-in for Adobe Photoshop CS3, and (b) visualizing a flow stream. Image in (b) is from (Kirby et al., 1999).*

2002; Klein et al., 2002), paint-like effects including oil (Gooch et al., 2002; Hertzmann, 2002) and watercolor (Curtis et al., 1997; Bousseau et al., 2007), drawing styles including sketching (Curtis, 1998; Mignotte, 2003), pen-and-ink illustration (Winkenbach & Salesin, 1994; Salisbury et al., 1994) and half-toning (Haeberli, 1990; Praun et al., 2001; Hiller et al., 2003; Secord, 2002; Hausner, 2005), cartoon-like effects (DeCarlo & Santella, 2002; Wang et al., 2004; Winnemoller et al., 2006; Zhao et al., 2008) and facial caricature (Gooch et al., 2004).

NPR algorithms may be fully automatic (Litwinowitz, 1997; Hertzmann, 2002; Mignotte, 2003; Hays & Essa, 2004; Bousseau et al., 2007), or user-interactive (Haeberli, 1990; Santella & DeCarlo, 2002; DeCarlo & Santella, 2002; Klein et al., 2002). The latter form allows the user some stylistic control, which may be useful for media artists. Animators, for example, must create every frame of an animation from scratch. If some of this painstaking process can be automated, labor is reduced. Semi-automatic NPR tools have been used in the production of animated Hollywood films such as The Lion King (1994, Disney), What Dreams May Come (1998, Polygram Filmed Entertainment), A Scanner Darkly (2006, Warner Independent Pictures) and television series such as Avenue Amy (1999, Curious Pictures). NPR is also useful for scientific visualization (Kirby et al., 1999) - see in Figure 4.1 - and motion depiction (Collomosse et al., 2003; Collomosse et al., 2005) and enhancement (Wang et al., 2006; Liu et al., 2005).

Early NPR algorithms restrict the stylizable input media to graphical geometry objects (Winkenbach & Salesin, 1994; Salisbury et al., 1994; Meier, 1996; Curtis et al., 1997). However, recent years have seen the introduction of image and video processing techniques for the stylization of digitally captured images (Haeberli, 1990; Hertzmann, 1998; Shiraishi

& Yamaguchi, 2000; Santella & DeCarlo, 2002) and videos (Litwinowitz, 1997; Hertzmann & Perlin, 2000; Bousseau et al., 2007). As a result, the consumer demand for NPR effects is growing. Tools for the painterly stylization of images can be found in desktop publishing software as well as professional, creative software (e.g. Adobe Creative Suite). Numerous plug-ins for rendering videos in the style of cartoons exist (e.g. ToonIt! plug-in for Adobe Photoshop, Digital Anarchy[1] - see Figure 4.1). Basic 'toon rendering software has even found its way into applications of some camera mobile phones.

## 4.2 SBR Explained

Stroke-Based Rendering (SBR) is a particular subset of NPR. It is well defined as

> *An automatic approach to creating non-photorealistic imagery by placing discrete elements such as paint strokes or stipples.* (Hertzmann, 2003).

SBR is a simulation of the traditional painting process whereby and artist uses a series of *brush strokes* to capture a scene on a *canvas* by simulated medium of paint, pencil or ink. Computationally, this task is usually emulated in a number of steps, as summarized in Figure 4.2. The pixels of a *source image* are obtained by the computer. Next, there may be some analysis or manipulation of the source image such as gradient detection or smoothing. The outputs obtained from this pre-process are used as *reference data* to guide the painting process. The next stage is the creation of an *ordered list* of simulated paint strokes, and finally, these strokes are composited onto the canvas in the order specified.

An important goal of computer-driven SBR is to limit the number of strokes so that the final rendering looks like a painting or sketchy drawing. Therefore, the output is usually an abstraction of the reference scene, because there is a simplification process inherent to the task of replacing of groups of pixels with larger, stroke-like entities.

## 4.3 Brush Stroke Attributes

One of the earliest SBR algorithms is presented in (Haeberli, 1990). This system - part of which is an online painting applet[2] as can be seen in Figure 4.3 (a) - allows a user to assist in creating an impressionistic representation of a source image, using a series of mouse clicks and gestures. Each click results in the appearance of a paint-like mark the canvas, which the author refers to as a *brush stroke*. The brush strokes and canvas framework introduced in this early work has persisted to current state-of-the-art in SBR, acquiring some enhancements along the way.

[1]ToonIt!: `http://www.digitalanarchy.com/toonPS/main.html`

[2]The Impressionist: `http://www.laminadesign.com/explore/impression/`

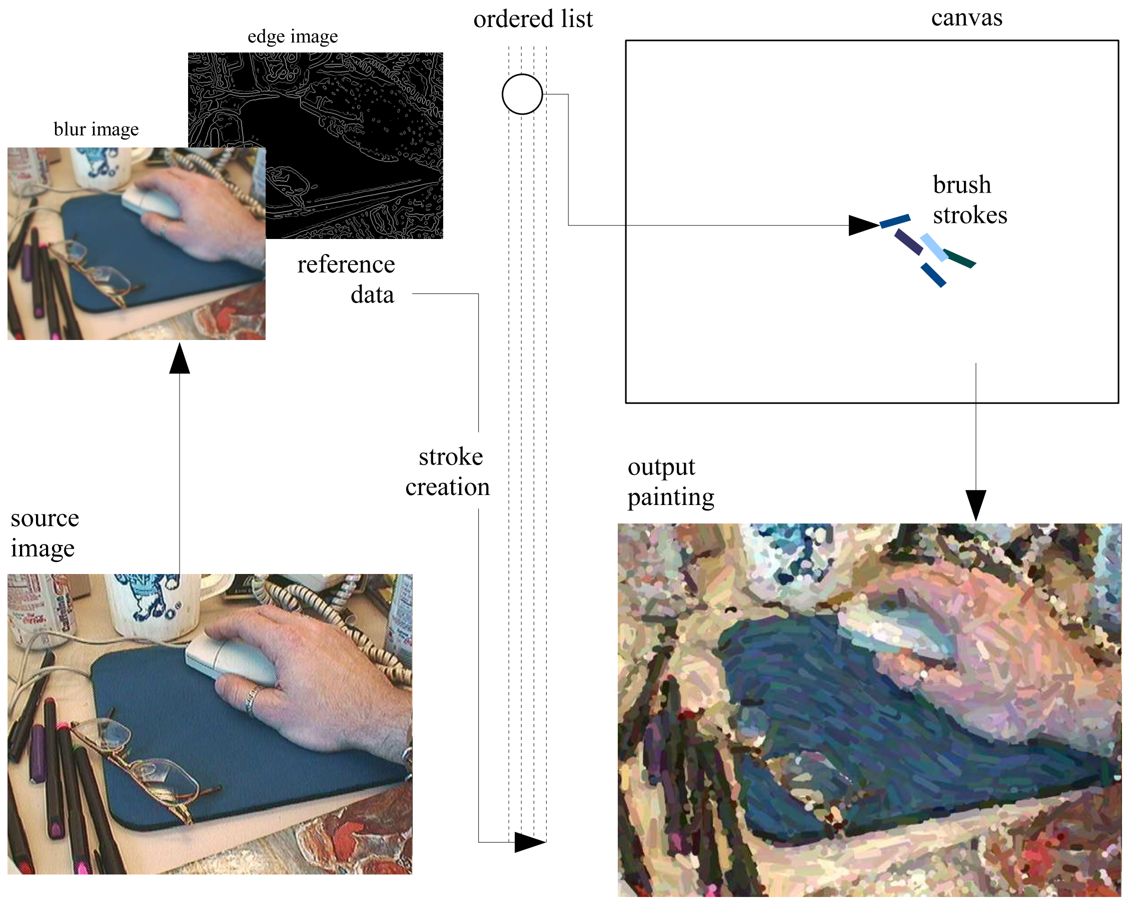
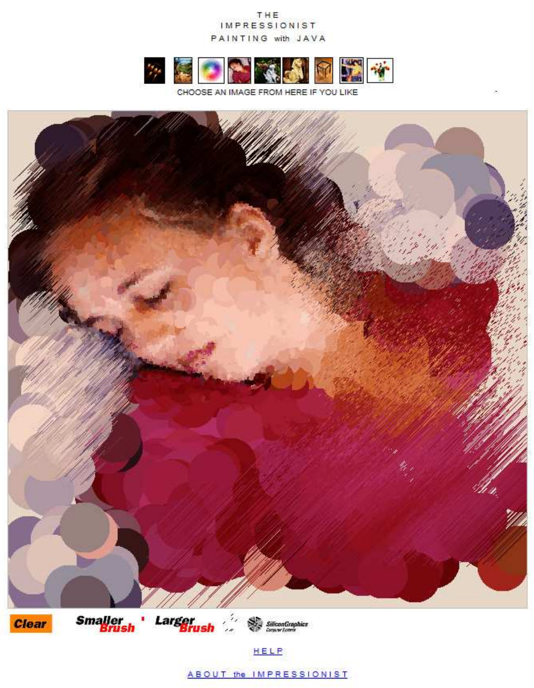


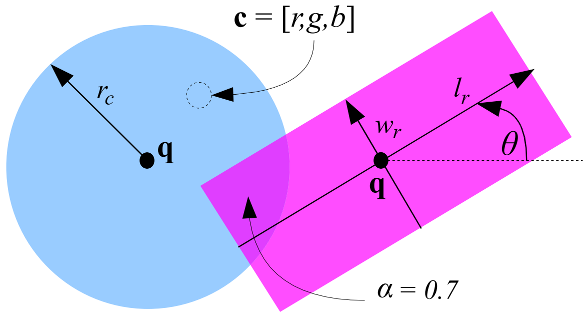
**Figure 4.2:** *The SBR process. Reference data is obtained from analysis of the source image and used to formulate an ordered list of brush strokes. When composited on the canvas, the result looks like a painting. Includes images from (Litwinowitz, 1997).*

Brush strokes are typically implemented as structures having a series of *attributes*, the majority of which can be visualized in a simple diagram such as Figure 4.3 (b). The stroke has an *anchor point* on the canvas image, $\mathbf{q}$, which usually corresponds to the same pixel coordinates in the source image. A stroke may have attributes corresponding to its *size*, such as length, $l_r$, width, $w_r$, or radius, $r_c$. Strokes are painted with a dominant *color*, $\mathbf{c} = [r, g, b]$, and have a dominant *orientation*, $\theta$. The structure may also store a value of *opacity*, $\alpha$, for blending the stroke when compositing. The brush stroke might also have other attributes relating to style (e.g. circle, Voronoi cell, b-spline), texture or shading coefficients. Usually, all of the strokes needed to render a painting are stored in one or more lists ordered for the compositing schedule.

The implementation of SBR, brush strokes and their attributes will now be discussed with regard to SBR literature. The task of painting a single image is explored first, followed by the issues involved in extending NPR/SBR effects to image sequences.


THE IMPRESSIONIST PAINTING with JAVA
CHOOSE AN IMAGE FROM HERE IF YOU LIKE
Clear
Smaller Brush
Larger Brush
HELP
ABOUT the IMPRESSIONIST

(a) The Impressionist



(b) Brush stroke attributes

**Figure 4.3:** *Brush strokes and the canvas; (a) The Impressionist painting applet associated with (Haeberli, 1990) and (b) typical attributes associated with a circular and rectangular brush stroke.*

### 4.3.1 Stroke Distribution Mechanisms

The method of distributing brush stroke anchor points on the canvas is an important stage in SBR. These points should be distributed sparsely to avoid *redundancy* which is the phenomenon of brush strokes piling up, but not to the point that there are undesirable *gaps* in the painting, as can be seen in Figure 4.4. A few different approaches to this task will now be examined in detail.

#### Single-Pass Approach

One of the simplest ways to distribute anchor points on the canvas is to use a single-pass approach. (Haeberli, 1990), for example, generates anchor point locations pseudo-randomly until the painting is complete. This technique can be visualized in Figure 4.5 (a). As discussed in (Hertzmann, 2003), this process could be described as *greedy* in that there is no energy function regulating redundancy in the *canvas coverage*. Redundancy can occur due to strokes overlapping excessively or being placed directly on top of others. This is undesirable because of the extra computing resources needed to store and render brush strokes that may not be visible in the output painting. If brush strokes are semi-opaque

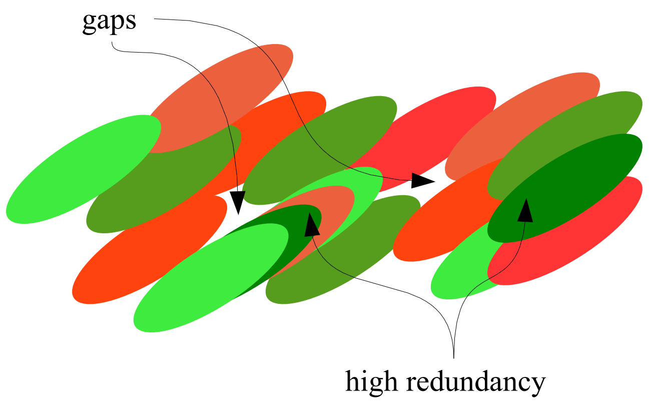
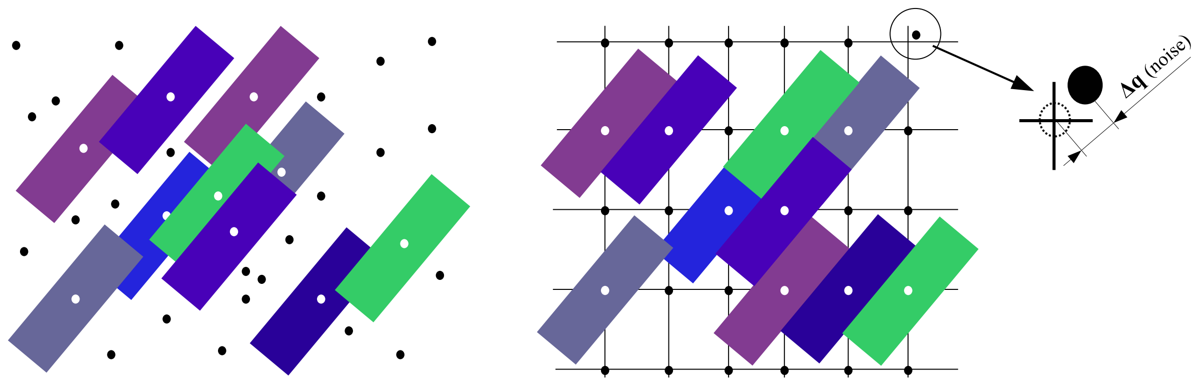


**Figure 4.4:** *Gaps and redundancy (i.e. unnecessary brush stroke overlap) in the output painting.*



(a) Pseudo-random (b) Grid-like

**Figure 4.5:** *Single-pass stroke distribution methods; (a) pseudo-random anchor point distribution, and (b) anchor points centered on nodes of a regular grid. Also in (b), noise can be added for a hand-painted look.*

(i.e. opacity, $\alpha < 1$), however, a controlled depth of canvas coverage may be desired.

An alternative single-pass approach is to tile the canvas with brush strokes in a grid-like fashion, as first implemented in (Litwinowitz, 1997). Brush strokes are anchored to the *nodes* of a *grid* that is spaced to ensure just adequate coverage of the canvas, as can be seen in Figure 4.5 (b). This tiling approach is economical in its distribution of strokes, but the rendered painting can appear too uniform, and *noise* is often added to the anchor point locations to simulate a hand-crafted look. A similar scheme is employed in (Meier, 1996), but since the reference scene is composed of 3D geometry objects, the grid is defined on the volumetric surfaces of these objects. In the rendering stage, the grid is simply projected onto the 2D canvas plane to determine the position of strokes.

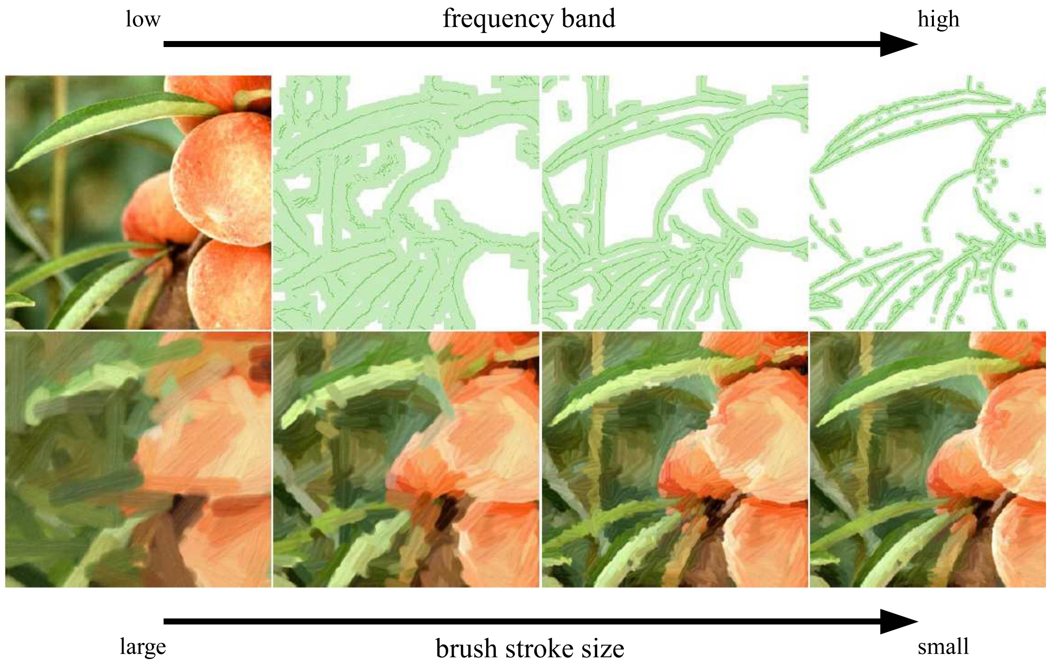


**Figure 4.6:** *A multi-pass anchor point distribution method based on layers. Refinement layers are defined around Canny edges detected at different scales. Includes images from (Hays & Essa, 2004).*

### Multi-Pass Approach

Multi-pass stroke placement is slightly more sophisticated in that there is usually an initial, greedy distribution of strokes, a stage of analysis, and then one or more *refinement passes*. There is an implicit *energy function* inherent in many multi-pass approaches, although this is rarely acknowledged in the literature.

(Hertzmann, 1998), for example, simulates the artistic process of creating a rough *underpainting* of the source image and then "building up" increasingly detailed *layers* of paint using correspondingly finer brush strokes. Therefore, layers are disjoint groups of brush strokes representing successive refinement passes. To create the roughest layer, the source image is blurred with a Gaussian kernel chosen to reflect the size of the largest brush stroke. Brush strokes are then placed using the grid-based technique of (Litwinowitz, 1997). The completeness of the painting is analyzed by computing the Euclidean color distance between the painted canvas and the source image. If the summed global distance is above a threshold, a layer of refinement ensues, but only at sites of significant color distance. This loop continues until the global color distance condition is satisfied. Each refinement pass works with a decreased brush stroke size, and correspondingly less blurred reference image.

A related, layer-based approach by (Hays & Essa, 2004) uses a different threshold to halt the refinement process; the completeness of the painting on each layer. Generally this criterion is difficult to quantify, especially in the use of semi-opaque brush strokes. In (Hays & Essa, 2004), brush strokes are added in pseudo-random locations until no *hole* greater than a pre-defined size exists in the layer. A hole is defined as a connected region of space in the layer completely untouched by strokes. Like (Hertzmann, 1998), this algorithm references blurred versions of the source image and decreasing brush sizes for each refinement pass. However, strokes are only placed at sites around Canny-detected edges in the reference image for each layer, as can be seen in Figure 4.6. The reasoning for this is that artists usually refine paintings in canvas regions near edges in the reference scene. This algorithm, however, covers the canvas more greedily than that of (Hertzmann, 1998).

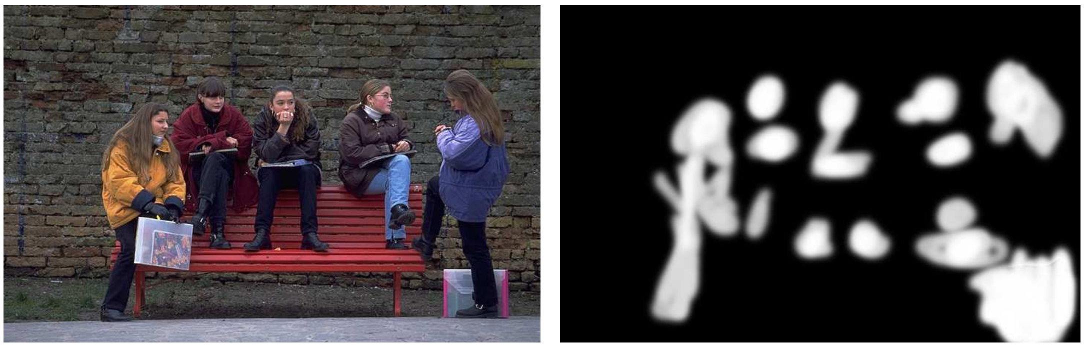
The multi-pass approach of (Santella & DeCarlo, 2002) has an interesting refinement stage that involves a degree of user interaction. Initially, brush strokes are placed in a grid-like fashion on layers representing a chosen set of image *scales* and corresponding brush sizes. For refinement, the user is asked to gaze at the source image shown on a monitor equipped with an eye-tracker, and the locations and durations of the user's *eye fixations* within the image are recorded. This data is presented to a perceptual model that defines a limit on the spatial frequency - in terms of scale and corresponding brush size - visible to the user at the main fixation points. Brush strokes are then pruned from the original distribution according to this analysis.

### Optimization Approach

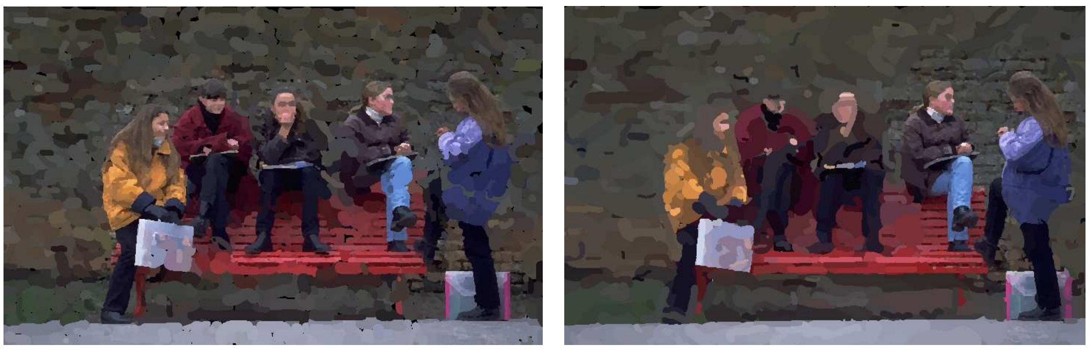
Optimization-based stroke distribution is an attempt to converge upon the best placement of brush strokes according to some optimization criteria. This is usually accomplished by formulating a global energy function to reflect the criteria, and then minimizing the energy over the space of all possible brush stroke attributes and orderings. In practice, the space is usually reduced to a smaller set of candidate strokes and attributes within a sensible range (i.e. prior knowledge).

The process of Stochastic Relaxation is a "trial-and-error" approach (Hertzmann, 2003) that can be used to produce candidate brush strokes. Its first introduction in SBR was an improvement to the greedy pseudo-random stroke-placement algorithm of (Haeberli, 1990), as mentioned at the end of the paper. According to this algorithm, the image is initialized with a pre-defined number of brush strokes, and then the attributes of the strokes are perturbed stochastically while minimizing the root mean squared difference between the source image and the canvas painting.

A more sophisticated relaxation algorithm is implemented in (Hertzmann, 2001). Building on an earlier, layer-based multi-pass approach (Hertzmann, 1998; Hertzmann & Perlin, 2000), the painting is initialized with a rough layer of large brush strokes. Following (Haeberli, 1990), various brush stroke attributes are perturbed to minimize a globally defined



(a) Original photo (left) and weight mask of user-selected semantic regions (right)

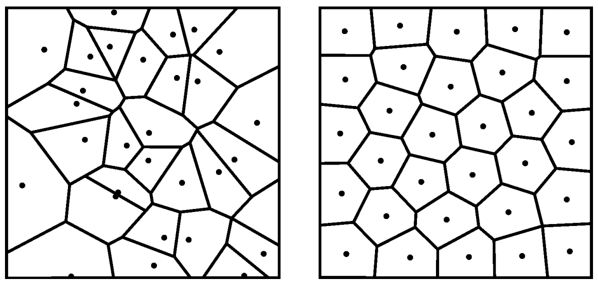


(b) Detail biased to semantic regions (left) and only to the right of the image (right)

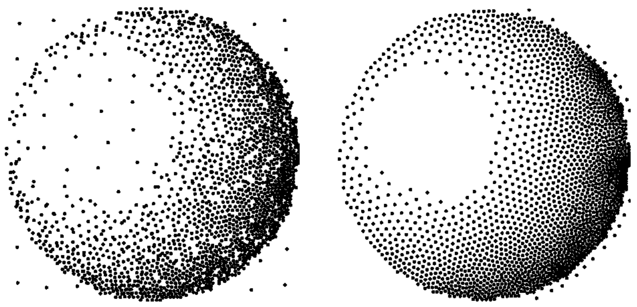
**Figure 4.7:** *Stochastic Relaxation in the painting algorithm of (Hertzmann, 2001). Refinement passes are biased towards semantic regions in the source image. These images are from (Hertzmann, 2003).*

painting energy. This energy function is a weighted sum of four separate criteria. The first is a measure of fidelity to the blurred reference image at each layer, and is defined as the pixel-wise color distance between the reference and painting. This term can be formulated to encourage a denser placement of strokes around edges or semantic regions. For the latter operation, the user must create a mask highlighting the semantic regions explicitly, as can be seen in Figure 4.7. The remaining three energy terms can be used to control redundancy in the canvas coverage, the total number of brush strokes in each refinement layer, and the completeness of canvas coverage respectively. Successive layers of refinement are added and perturbed until the painting energy converges to a satisfactory minimum.

Bayesian techniques for unsupervised style-transfer and sketching are presented in (Mignotte, 2002) and (Mignotte, 2003) respectively. In these works, the concept of relaxation, and the energy function is defined more formally within a Bayesian framework, and optimization occurs via Bayesian inference using well-known local energy minimization techniques such as Iterated Conditional Modes (ICM), as in (Mignotte, 2002). Sketchy pencil-drawn realizations of a source image are created in (Mignotte, 2003). The Bayesian prior is defined as



(a) Voronoi dithering



(b) Stippling algorithms

**Figure 4.8:** *The process of Stippling; (a) the first and last iterations of a Voronoi dithering algorithm, and (b) two Stippling algorithms are compared; fast vs. high-quality. Images in (a) and (b) are from are from (Hiller et al., 2003) and (Secord, 2002) respectively.*

the space of reasonable deformations of a spline-like pencil stroke model, and the likelihood is formed by the statistical distribution of the image gradients. Finding the most likely pencil strokes is a process of minimizing the posterior energy globally, and this is achieved by means of a modified Simulated Annealing algorithm in the case of (Mignotte, 2003). Here, a bias towards stroke placement in textured regions is inherent due to the gradient-based form of the likelihood function. Strokes are also specifically directed to edges by initializing the set of potential anchor points to Canny-detected edge pixels.

The optimization goal of Stippling (Hiller et al., 2003; Secord, 2002; Vanderhaeghe et al., 2007) (see Figure 4.8) and Hatching (Haeberli, 1990; Salisbury et al., 1994; Praun et al., 2001) is to distribute points more densely on the canvas in regions corresponding to high spatial frequencies in the source image. This gives the impression of tone and depth in the rendered output. Stippling algorithms are interesting, in that they attempt to simulate a so-called Poisson Disk Distribution (PDD) of stipples (i.e. anchor points). This means that a disc of given radius could be anchored at any point such that it covers no other point in the distribution. This is usually achieved by Llyod's method; a process of iterative Voronoi dithering on an initial pseudo-random distribution. The dithering can be biased towards high frequencies by restricting the movement of points according to the gradient of the source image, as in (Secord, 2002).

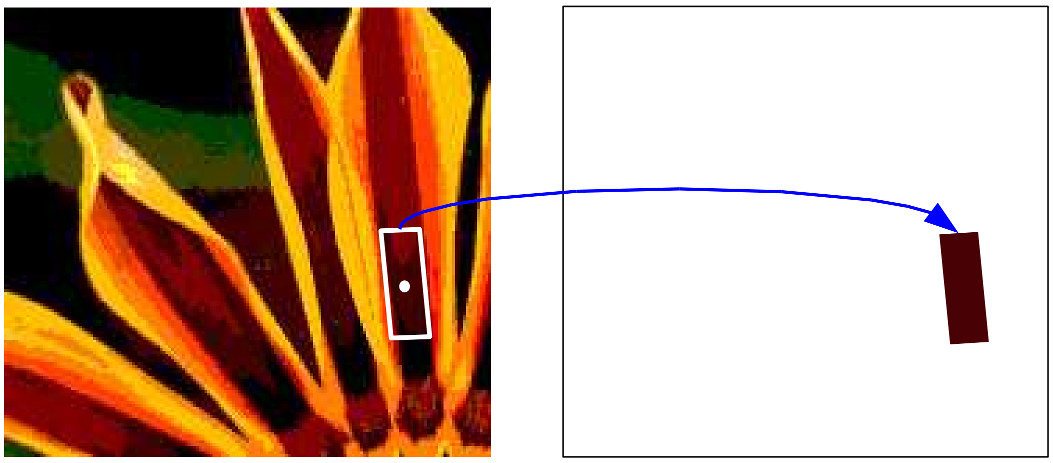


**Figure 4.9:** *Color segmentation-based approach; brush strokes are anchored at the centroids of regions of constant color.*

### Segmentation-Based Approach

Segmentation-based approaches focus on placing brush strokes such that they are anchored in regions of similar intensity (Gooch et al., 2002) or color (Shiraishi & Yamaguchi, 2000; Nehab & Velho, 2002; Santella & DeCarlo, 2002). The latter concept is demonstrated in Figure 4.9. (Gooch et al., 2002) for example, segments the source image using an intensity-based Flood-Fill algorithm. The medial axes of segmented regions are found using a modified thinning algorithm, and these are used to determine the locations of brush strokes.

Much like a Stippling algorithm, a point distribution of brush stroke anchor locations is formulated in (Shiraishi & Yamaguchi, 2000) and (Nehab & Velho, 2002) using a novel dithering algorithm based on color segmentation. First, the image is segmented to local regions of similar color. The extent of the local region is a user-defined parameter that effects the segmentation granularity. A distribution of stroke dimensions over the canvas is calculated by estimating the image moments of these color regions, and this distribution is then inverted to a map of tentative stroke anchors using a Murray space-filling curve algorithm. Finally, these tentative anchor positions are adjusted to the centroids of the segmented color regions. The resulting distribution of strokes is increasingly dense in the high frequency areas of the source image.

## 4.3.2 Stroke Dimensions

Early SBR algorithms model brush strokes with simple style primitives such as circles, points, polygons, straight lines (Haeberli, 1990) or rectangles (Litwinowitz, 1997; Meier, 1996). In early works, the dimensions of the strokes (e.g. radius or width and length) are

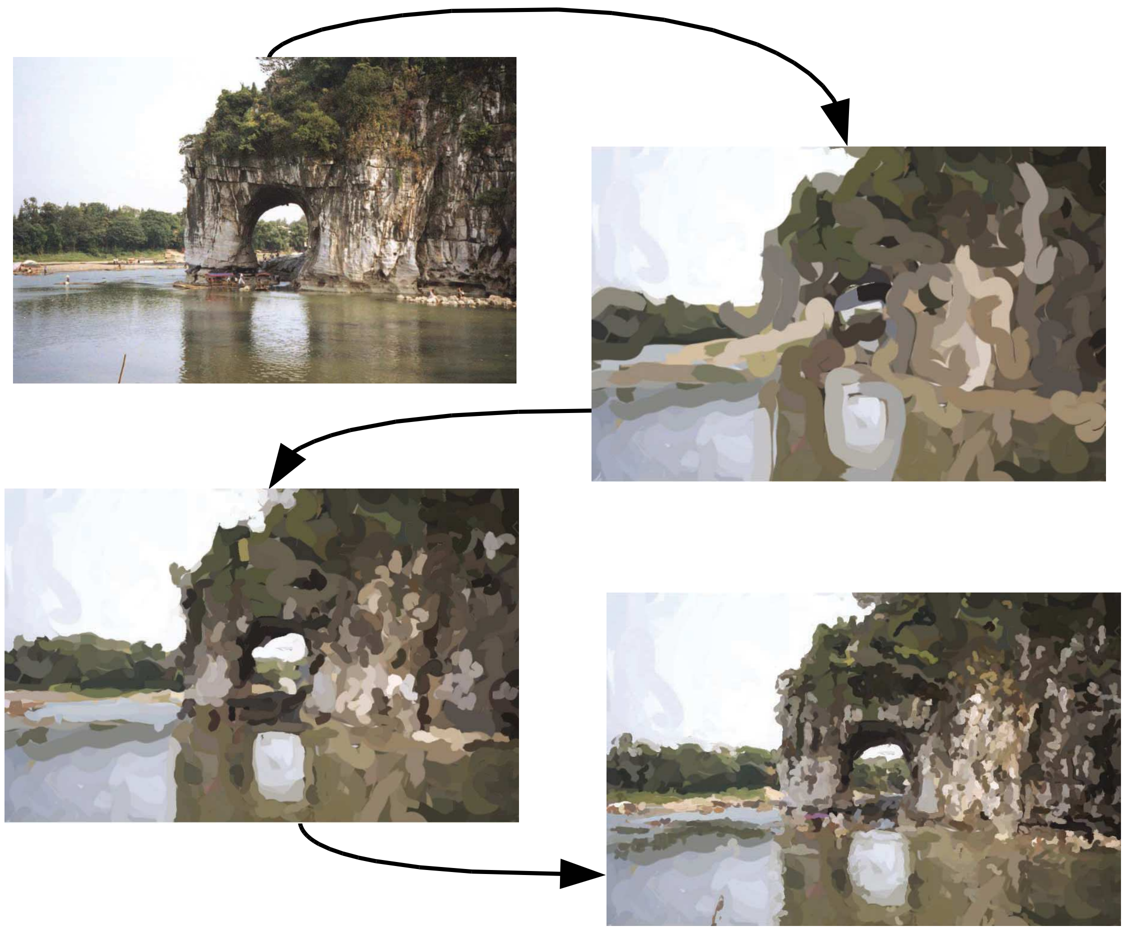


**Figure 4.10:** *In (Hertzmann, 2001) the dimensions of spline-like brush strokes decrease in regions of texture and detail. Includes images from (Hertzmann, 2003).*

uniform across the canvas. Noise is often added to the dimensions to emulate a hand-painted look. (Haeberli, 1990), however, presents an interactive system allowing the user to change both stroke style and size mid-painting. (Meier, 1996) notes that when the reference scene consists of geometry objects, the size of brush strokes can be varied according to depth, tone, or per object. Later SBR algorithms begin to reduce the size of brush strokes near regions of texture or edges in the source image. Brush strokes implemented in (Salisbury et al., 1994; Litwinowitz, 1997) and (Hays & Essa, 2004) are clipped at edge crossings to preserve high frequency information in the painting.

Commonly in layer-based approaches, a particular brush stroke size is used to paint a specific layer, and the size is reduced for each additional refinement pass. In (Hays & Essa, 2004), the area of the rectangular brush strokes decrease on each successive layer of refinement, as can be seen in Figure 4.6. The curved, spline-like brush strokes of (Hertzmann, 1998; Hertzmann & Perlin, 2000) and (Park & Yoon, 2008) have two dimensions; length and radius. These dimensions decrease when painting increasingly detailed areas of the source image, as can be seen in Figure 4.10. In the case of (Hertzmann & Perlin, 2000), this effect is a result of the relaxation process, since it minimizes the painting energy.

Some SBR algorithms encourage a more varied spatial distribution of brush stroke size. In the segmentation-based methods of (Shiraishi & Yamaguchi, 2000; Nehab & Velho, 2002) and (Gooch et al., 2002), the brush size at each point in the painting reflects the size of its anchoring segment. The curved, spline-like strokes in (Hertzmann, 1998; Hertzmann & Perlin, 2000; Hertzmann, 2001) and (Park & Yoon, 2008) are non-uniform within layers because they are grown from their anchor points along the normal of Sobel-detected gradients in the source image. The strokes are terminated when either a maximum stroke length is defined, or the color sampled from beneath a tentative spline control point differs too much from that of its neighboring control point.

### 4.3.3 Stroke Color

Brush strokes are usually colored uniformly over their area. In order to color each stroke, many SBR algorithms sample the color vector at the location in the source image corresponding to the stroke's anchor point on the canvas (Haeberli, 1990; Meier, 1996; Litwinowitz, 1997; Shiraishi & Yamaguchi, 2000; Hays & Essa, 2004). Using this method, the boundaries between neighboring paint strokes can be difficult to distinguish, so noise is often added to the color values (Haeberli, 1990; Litwinowitz, 1997; Hertzmann, 1998). The output color range may also be quantized for the same reason (Haeberli, 1990; Park & Yoon, 2008).

If the paint strokes are semi-opaque, the canvas can be colored to influence the hue of the painting. (Gooch et al., 2002), for example, paints semi-opaque strokes on a hue-adjusted version of the underpainting. The color of the spline-like brush strokes are determined by the weighted average of pixels in a ridge obtained by thinning the segmented region on which the spline is anchored. In (Hertzmann, 2001), the color of a spline-like brush stroke is an average of the colors in the blurred reference image - due to the layer-based algorithm - beneath its painted area.

### 4.3.4 Stroke Orientation

Rather than orientating brush strokes uniformly across the canvas, (Haeberli, 1990) notes that interesting effects are achieved by aligning strokes to the Sobel-detected gradients in the reference image. This technique aligns strokes locally perpendicular to edges, creating a hatch-like effect.

It is argued in (Litwinowitz, 1997) that a more painterly look results from aligning brush strokes normal to the local gradients. Using this basic technique, brush strokes in flat regions remain randomly oriented. A more pleasing effect is achieved when vanishing gradient magnitudes are interpolated from surrounding regions. A thin-plate spline is used as the interpolant function in (Litwinowitz, 1997). Later algorithms build on this idea, and

(Hays & Essa, 2004) uses a radial basis function to align brush strokes normal to Sobel gradients, as can be seen in Figure 4.11 (a).

Only those gradients with magnitudes exceeding a threshold are considered, however. Edge-aligned strokes are almost inherent to segmentation-based algorithms (Shiraishi & Yamaguchi, 2000; Nehab & Velho, 2002; Gooch et al., 2002), and as previously discussed, the curved spline-like brush strokes of (Hertzmann, 1998; Hertzmann & Perlin, 2000; Hertzmann, 2001; Hertzmann, 2002) and (Park & Yoon, 2008) are grown along Sobel gradient normals, and so appear to curve along edge contours. In (Park & Yoon, 2008) the vanishing gradients are globally interpolated in a similar manner to (Hays & Essa, 2004), but from strong Canny-detected edges obtained using a modified thresholding algorithm.

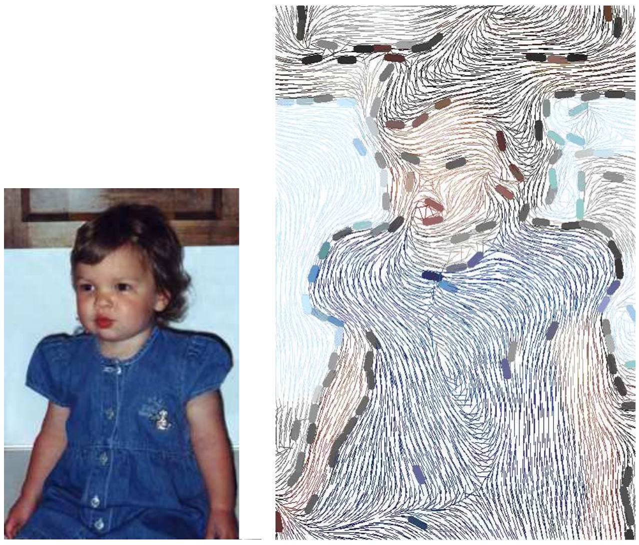
Strokes can be orientated perpendicular to the surface normals of geometry objects, as in (Haeberli, 1990; Meier, 1996) and (Salisbury et al., 1994) so that the painting appears almost to wrap around objects. As usual, noise is often added to stroke orientations to enhance the painterly feel. An interesting effect can be achieved by using the gradient reference of one image to influence stroke orientation in another, as discovered by (Haeberli, 1990). The result can be visualized in Figure 4.11 (b).

### 4.3.5 Other Stroke Attributes

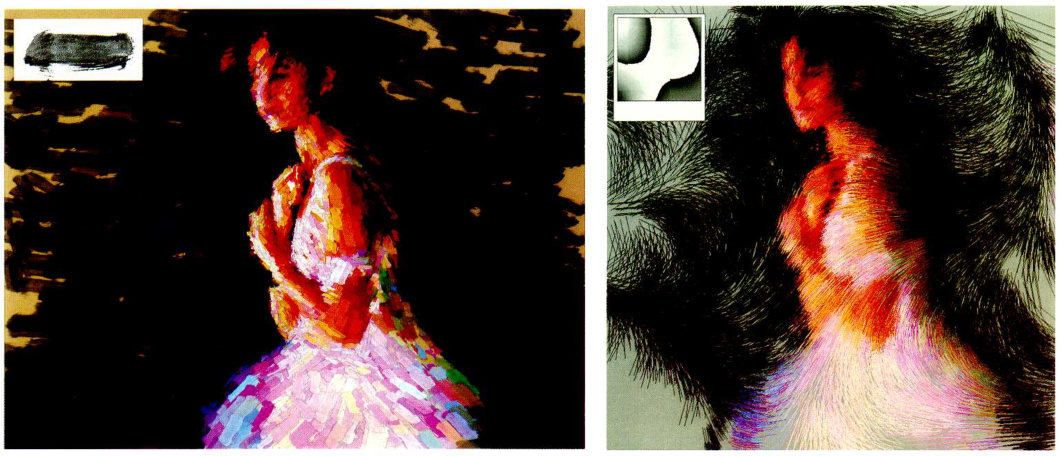
As mentioned previously, brush strokes may have attributes relating to style, compositing effects and paint texture simulation. Style attributes are often stored within the structure of brush strokes, and may be varied spatially or temporally. The canvas may also have attributes relating to color, texture or underpainting.

An alpha ($\alpha$) value for compositing, and texture masks are associated with the brush strokes of (Strassmann, 1986; Haeberli, 1990; Meier, 1996; Gooch et al., 2002; Hertzmann, 2002) and (Hays & Essa, 2004). When used in conjunction, these attributes simulate particular styles of painting, brushes and paint. (Strassmann, 1986), for example, simulates the style and texture of strokes seen in the Chinese painting style of Sumi-e, while oil painting with a Filbert brush is emulated in (Gooch et al., 2002). A more convincing oil-like texture is simulated in (Hertzmann, 2002) by including a height map attribute for simulating lighting on brush strokes.

The appearance of a watercolor painting can be simulated by alpha-compositing semi-opaque brush strokes on a textured canvas. (Gooch et al., 2004) synthesizes paintings in the style of Jackson Pollock using fluid jets, and the isoluminant color-picking algorithm of (Luong et al., 2005) extends nicely to the creation of paintings in the mosaicking style of Chuck Close. The particular look of sand-in-a-bottle art is reproduced in (Neto & Carvalho, 2007).



(a) Radial basis function



(b) Changing the style by varying brush stroke orientations

**Figure 4.11:** *Brush stroke orientation; (a) stroke orientation is determined in (Hays & Essa, 2004) by applying a radial function to a few basis strokes orientated normal to local gradients in the source image, and (b) two styles to paint the same image; strokes aligned with local gradients (left) and using the local gradients from a different picture as reference data (right). Images in (a) and (b) are from (Hays & Essa, 2004) and (Haeberli, 1990) respectively.*

### 4.3.6 Ordering

If the geometry of the source image is known, brush strokes can be composited according to depth. This idea is exploited by (Haeberli, 1990; Meier, 1996) and (Gooch et al., 2002). Depth information is not available in the case of most digital images, however. Yet the compositing order of brush strokes is important. Fan-like artifacts occur when overlapping strokes are composited in scan-line order. This can be avoided by randomizing the compositing order of strokes (Litwinowitz, 1997).

In layer-based, optimization and segmentation-based approaches, brush strokes are usually composited in order of decreasing size, and randomly distributed within layers of refinement. Since refinement is usually concentrated in the region of texture and edges, brush strokes located in these regions are among the last to be composited. Easily visualized in Figure 4.10, this technique is widely regarded as a fair simulation of the processes of human painting and drawing.

## 4.4 Extending Stylization Effects to Video

Extending stylization effects such as painterly rendering to image sequences is a difficult problem. The goal is to produce *stylized sequences* in which motion is coherent with the motion of the source image sequence. Since the video may include local and global motion, the *occlusion* and *uncovering* of objects (both demonstrated in Figure 4.13), transitions, noise or flicker, great care must be taken to ensure that NPR/SBR effects are propagated temporally in a pleasing manner.

### 4.4.1 Animating Anchor Points

In the case of SBR for image sequences, the frame-wise distribution of brush stroke anchor points becomes an interesting problem. If a new, pseudo-random distribution is initialized for each frame, brush strokes in the video will appear to hop around the canvas. On the other hand, if the initial anchor positions are kept constant throughout the sequence, the so-called *shower door effect* is observed; the video will appear to be moving from behind the fixed strokes - as if the viewer is watching the motion through a patterned glass door.

(Meier, 1996) presents a video-based SBR algorithm with motion-animated brush strokes. Since the algorithm is concerned with the stylization of synthetic geometry objects, brush strokes can be anchored to the 3D surfaces of the objects at nodes obtained by a standard graphics triangulation. When these objects are animated, therefore, the brush strokes move coherently as if they are *stuck to the objects*. When projected onto the 2D frames of a video, the painterly effect is visually stunning. As mentioned before, however, most digitally captured image sequences are devoid of geometry.

There are a few ways of animating brush strokes in image sequences where no geometry information is present. This task involves trying to quantify the real-world motion in the image sequence using one of the usual *motion models* for video.

### Rubber Sheet Model

The Rubber Sheet Model regards each frame of an image sequence as a continuously deforming rubber sheet. In other words, every pixel in the previous frame, $n-1$, is also present in the current frame $n$, but simply displaced by a small amount, therefore causing the rubber sheet to deform with the motion. Assuming no error in the estimation, this idea can be described by

$$I^{n+1}(\mathbf{X}) = I^{n}(\mathbf{X} + \mathbf{f}^{n}(\mathbf{X})) \tag{4.1}$$

where $\mathbf{f}^{n}(\mathbf{X})$ is a pixel-wise vector field mapping the image displacement between frames $n$ and $n+1$ with regard to the coordinate system, $\mathbf{X}$. $\mathbf{f}^{n}(\mathbf{X})$ is also known as the forward *motion vector field*, and the backward motion vector field, $\mathbf{b}^{n}(\mathbf{X})$, is defined similarly

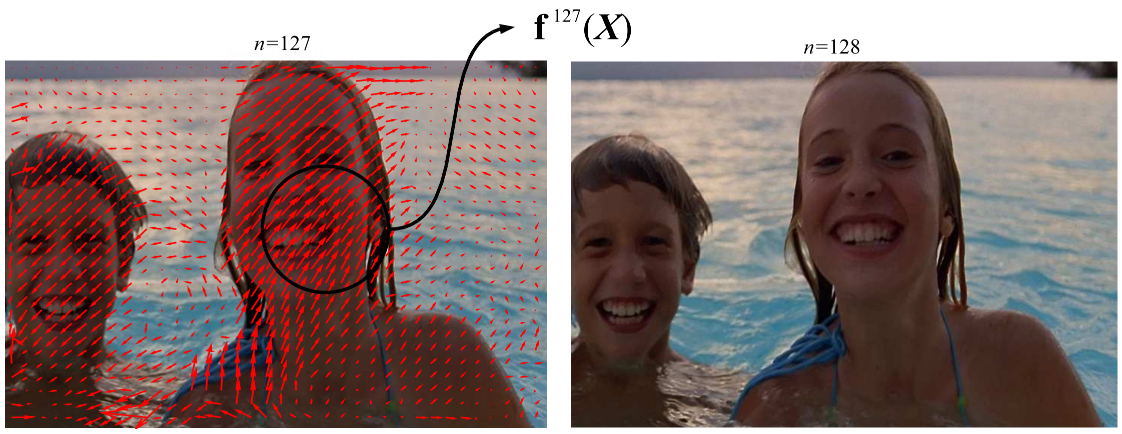
$$I^{n-1}(\mathbf{X}) = I^{n}(\mathbf{X} + \mathbf{b}^{n}(\mathbf{X})) \tag{4.2}$$

These motion fields can be determined through a process of Motion Estimation (Kokaram, 1998), or Optical Flow (Horn & Schunck, 1981). Especially when determined by the latter, the forward motion vector field is often referred to as the *flow field*. This concept can be visualized in Figure 4.12 (a).

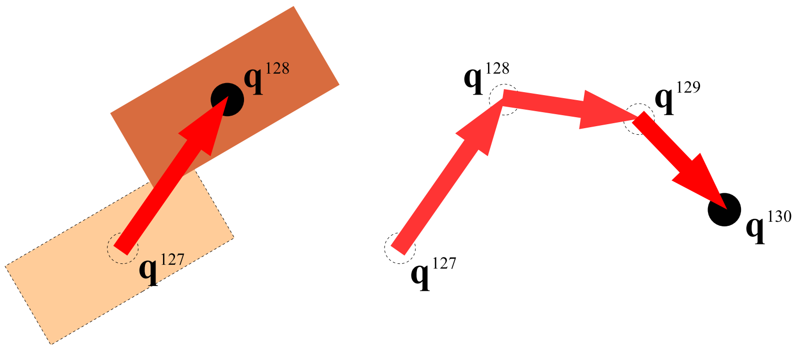
Brush strokes may be animated in video by continually displacing or *motion-compensating* their anchor points according to the flow field defined for each frame. This technique is used in the video-based SBR algorithms of (Litwinowitz, 1997; Hertzmann & Perlin, 2000) and (Hays & Essa, 2004). The motion path of a particular stroke anchor point is known as its *trajectory*, as can be seen in Figure 4.12 (b).

The Rubber Sheet Model does not account for the fact that moving objects in the scene may occlude or uncover each other, conditions that can be visualized in Figure 4.13. Uncovering reveals spatial regions in the source image sequence that were not there in previous frames, and so it causes brush strokes to spread apart leaving increasingly large gaps in the painting over time. Occlusion results in the disappearance of spatial regions of the image sequence, so redundant brush strokes tend to pile up and bunch together in these regions. Therefore, uncovering and occlusion are triggers of painting gaps and redundancy respectively.

Gaps and redundancy are detected and mitigated in the grid-based brush stroke distribution of (Litwinowitz, 1997) by a process involving the Delaunay triangulation of the anchor point nodes. The triangulation imposes a mesh of edges connecting the nodes, and this mesh deforms as the anchor points are displaced by the motion field. To prevent the bunching of strokes from occlusion, the edge lengths in the mesh are monitored. If the length of any edge falls below a minimum threshold, one of the connected points is deleted.



(a) Motion vector field

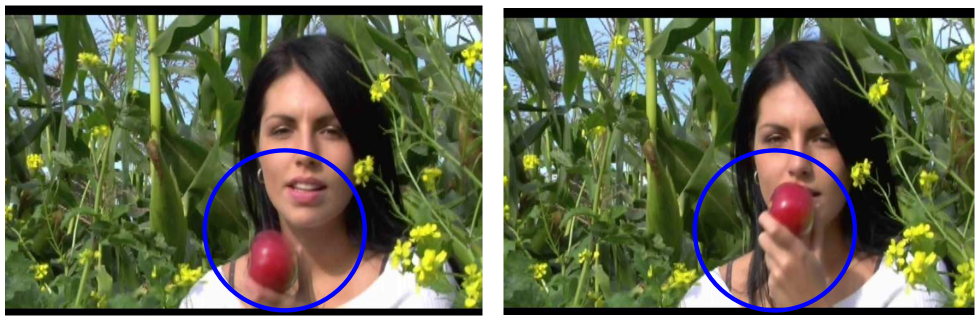


(b) Animating a brush stroke

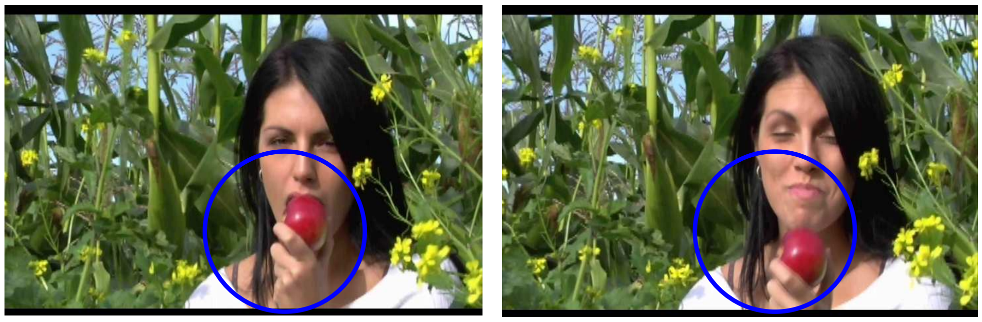
**Figure 4.12:** *The Rubber Sheet Model; (a) a forward motion vector field determined by the process of Motion Estimation (Kokaram, 1998), and (b) animating a brush stroke using the motion field (left) and the anchor point's trajectory over a few frames (right). In (a), the magnitude of motion and spacing of vectors has been enhanced for clarity.*

Furthermore, uncovering is dealt with by sub-dividing the mesh if the area of any triangle exceeds some maximum threshold.

In the layer-based painting of (Hays & Essa, 2004), redundancy is dealt with by monitoring each layer for the bunching of strokes, and gradually reducing the opacity of the strokes causing the redundancy until they have been deleted. Strokes whose anchor points are displaced beyond layer boundaries are also phased out in this way. In the case of uncovering, gaps that emerge in the painting are detected as holes and re-populated using the initial, pseudo-random method of anchor point distribution. These new brush strokes are phased in over time by increasing their opacity values over a few frames.



(a) Occlusion



(b) Uncovering

**Figure 4.13:** *Problems with video stylization; (a) occlusion - the apple moves in front of the woman's mouth and occludes part of her face, and (b) uncovering - the apple is moved away from the woman's mouth uncovering part of her face.*

### Layer Model

The Layer Model assumes that an image sequence consists of a number of moving *layers*. Not to be confused with the layer-based technique of multi-pass anchor point distribution discussed in Section 4.3.1, this motion model defines a layer as encapsulating the motion of a particular object in the scene. Here, the term *object* does not necessary refer to a well-defined semantic entity (e.g. beach ball), but rather groupings of pixels that have some common characteristics (e.g. color, texture) and similar motion in the scene. In the idealized Layer Model, issues with occlusion and uncovering are absent since the motion of each object is encoded in a distinct layer. The motion of pixels within each layer can be determined using the Rubber Sheet Model or using some other technique.

In order to apply the Layer Model, one must first segment the object layers in the video. A popular method in SBR concerns the volumetric segmentation of a video as if it were a 3D cube with two spatial axes (i.e. the frame-wise spatial coordinates), and one temporal axis (i.e. the time line of the image sequence). This concept can be visualized in Figure 4.14.

The video volume is segmented in (Klein et al., 2002) using a number of different

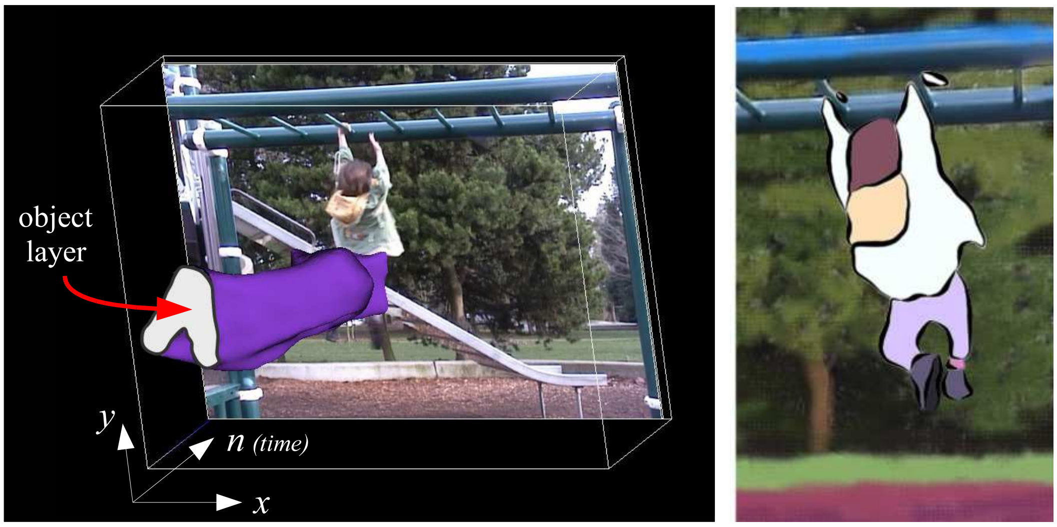


**Figure 4.14:** *The Layer Model; (left) the video is segmented as a 3D cube to determine object layers which can then be stylized independently, and (right) the cartoon-like result of (Wang et al., 2004) with edges highlighted. Includes images from (Wang et al., 2004).*

techniques. First it is pre-processed to find local color gradients and measures of salience (i.e. spatio-temporal variance in intensity). The spatio-temporal object layers are defined in two steps. First, the user may select "cutting lines" on key-frames in the video. These cutting lines are linearly interpolated between key-frames to segment the video volume into a number of sub-cubes. Next, the volume is segmented more finely by constructing a KD-Tree from the color and salience information while respecting the boundaries of the user-defined sub-cubes. Finally, brush stroke anchor point trajectories are defined within the layers by fitting smooth curves to "flow" trajectories extracted by a modified streamline algorithm.

Although not strictly SBR, the algorithm of (Wang et al., 2004) is interesting in that it uses the Layer Model. Color-based segmentation of objects is achieved through application of a modified 3D Mean-Shift algorithm. A locally adaptive anisotropic kernel ensures that the boundaries of segmented objects are well-defined with regard to the underlying motion. Interactive techniques allow the user to group segmented regions to reflect semantic objects in the video. The algorithm of (Collomosse et al., 2005) is comparable, although color segmentation is performed in 2D (i.e. on each distinct frame), and then the spatio-temporal layers are formed by grouping and smoothing the 2D segments temporally using an energy function incorporating measures of object dimensions, shape and color. Brush stroke anchor point trajectories are determined by matching the homography of texture within the object layers from frame to frame. Both (Wang et al., 2004) and (Collomosse et al., 2005) fit smooth surfaces to the boundaries of the segmented object layers to form smooth spatio-temporal object boundaries that have the appearance of stylized cartoon-like edges when rendered.

The drawback of the Layer Model is the computational burden of storing and analyzing video volumes, although this problem is somewhat mitigated in (Collomosse et al., 2005) due to the two-step technique of temporally grouping spatially segmented regions.

**Frame Differencing**

Frame differencing is the detection of motion by comparing consecutive frames in an image sequence for intensity or color differences, usually by means of pixel-wise subtraction. The assumption for SBR is the painting should only be altered in places where the source video changes between frames.

Frame differencing is also known as the "paint-over" or "paint-on-glass" technique in SBR. The general technique is depicted in Figure 4.15. In (Hertzmann & Perlin, 2000) the first frame of canvas is painted with spline-like strokes using the optimization approach of (Hertzmann, 1998). In painting successive frames, a mask is obtained by thresholding the intensity difference between the previous and current frames, and these masked areas are re-painted to create the current canvas. This technique is fast and hence it is useful for real-time applications, as in (Hertzmann & Perlin, 2000). The painted video, however, demonstrates flicker in regions of rapid motion, whereas regions with less motion (e.g. the center of large objects) can appear flat since the anchor points there are not animated. The latter problem is addressed in (Hertzmann & Perlin, 2000) by using Optical Flow to motion warp the previous frame to the current before re-painting the frame difference regions. This effect can be observed in the supplementary videos at the URL associated with (Hertzmann & Perlin, 2000)[3].

A modified frame differencing technique is used in (Park & Yoon, 2008). The algorithm formulates a strong mask or "motion map" capturing differences in the region of edges, and a weak mask capturing differences everywhere else in the frame. These masks are obtained, however, but by thresholding the motion field obtained by Motion Estimation; a process similar to Motion Detection. In the re-painting process, larger brush strokes are painted over regions in the strong motion map, whereas smaller strokes are re-applied in regions in the weak motion map where it is argued that less significant changes occur between frames. The video results of this algorithm are still prone to flicker, as can be seen in the supplementary video associated with (Park & Yoon, 2008).

### 4.4.2 Temporal Coherency and Other Attributes

The method of regulating the painted color of animated brush strokes is an important consideration. (Litwinowitz, 1997) point-samples the color at the anchor point of each

[3](Hertzmann & Perlin, 2000): `http://www.dgp.toronto.edu/~hertzman/videos/npar2000/`

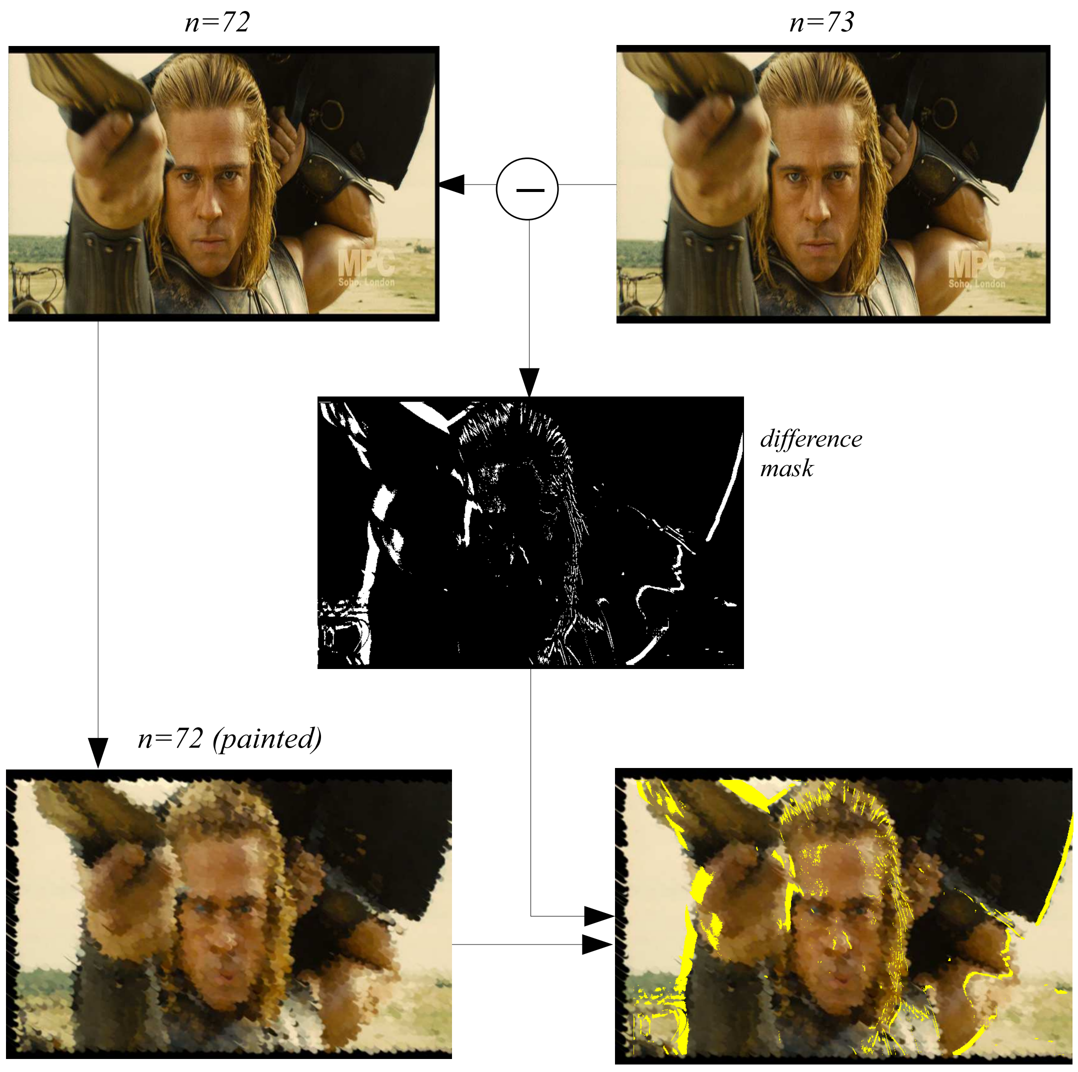


**Figure 4.15:** *Frame Differencing; The current frame, $n = 73$, is stylized by painting brush strokes on top of the previously stylized frame, $n = 72$, in areas of intensity difference between the two frames. Hence, the yellow area must be painted over to create a stylized $n = 73$. Here, stylized frame $n = 72$ has been rendered using the Brush Strokes tool of Adobe AfterEffects CS3 for the purpose of visualization.*

stroke for each frame independently. Frame-wise color-sampling is also used to color brush strokes in (Hertzmann & Perlin, 2000) and (Park & Yoon, 2008). The problem with this technique is that *flicker* is observed as the painted color of the strokes fluctuate throughout the sequence. Attempts at suppressing the effect of flicker are made by spatial smoothing in (Hertzmann & Perlin, 2000) and global color quantization in (Park & Yoon, 2008).

Since the 3D video volume is smoothly segmented in (Wang et al., 2004), the resulting segmented colors can be applied to their associated objects throughout the video. A similar

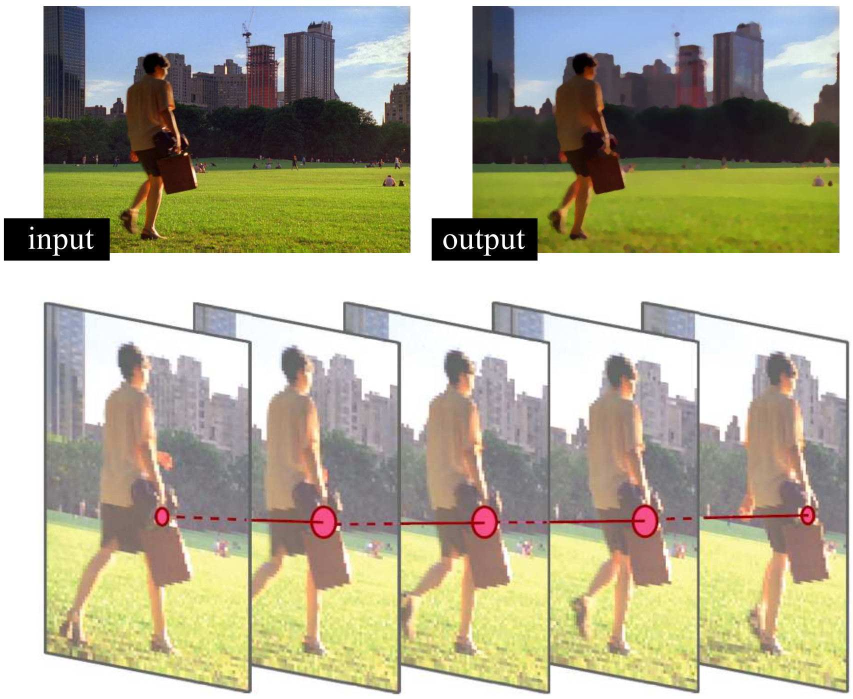


**Figure 4.16:** *Spatio-temporal color sampling. The color of the output painting is sampled from the source image sequence over a non-uniform spatio-temporal window extending forwards and backwards in time. Includes images from (Bousseau et al., 2007).*

technique is used to color the segmented objects in (Collomosse et al., 2005). Coloring semantic objects uniformly is a different problem to SBR, but flicker is also less apparent in the latter when the colors of brush strokes are temporally smoothed, despite the fact that they are being continually displaced with the motion field.

When brush strokes are applied in (Collomosse et al., 2005), they are painted by point-sampling the color from the source image beneath their coordinate trajectories over a number of frames and smoothed temporally by averaging. In (Hays & Essa, 2004), the brush stroke colors are sampled by averaging the color values beneath the painted area of the stroke on the current and a small number of previous frames (see URL associated with (Hays & Essa, 2004) for video demos). (Bousseau et al., 2007) implements an interesting extension to this technique in the form of a non-uniform spatio-temporal filter that averages the color beneath a spatial disk on a number of frames and centered on the current frame. The sampling volume is non-uniform in that the disk's sampling radius tapers outwards from the current frame. The technique can be visualized in Figure 4.16, and its effect can be seen in supplementary videos at the URL associated with (Bousseau et al., 2007)[4].

Other brush stroke attributes may also be smoothed temporally. This idea is key to

[4](Bousseau et al., 2007): `http://artis.imag.fr/Publications/2007/BNTS07/`

the algorithm of (Hays & Essa, 2004) in which the brush stroke orientation and opacity attributes are smoothed temporally by clipping them to a frame-wise maximum derivative of change. The latter is applied to ensure the gradual, smooth introduction and removal of hole-filling and redundant brush strokes respectively. This effect can be seen in the supplementary videos at the URL associated with (Hays & Essa, 2004)[5].

Finally, it has been noted in (Litwinowitz, 1997) and (Hays & Essa, 2004) that brush strokes introduced to fill holes should be distributed randomly in the existing ordered list of strokes to avoid fan-like artifacts appearing in the painting over time.

## 4.5 Semantically-Driven and NPR and SBR

The most recent stylization algorithms are concerned with the *meaningful abstraction* of images and videos for the purpose of efficient information communication and aesthetics.

User interaction is often incorporated to focus the elements of stylization on the semantic content in images (Haeberli, 1990; Hertzmann, 2001; Santella & DeCarlo, 2002; DeCarlo & Santella, 2002; Gooch et al., 2002) and videos (Wang et al., 2004; Collomosse et al., 2005). This idea is demonstrated in Figure 4.17 (a) and (b). An interactive application, the user's gaze fixations influence the SBR in Figure 4.17 (a) such that the people in the image - particularly their faces - are painted with a finer brush and hence more detail. In Figure 4.17 (b), the image is also stylized such that the person is the focal point and cartoon-like edges are used to highlight the important information.

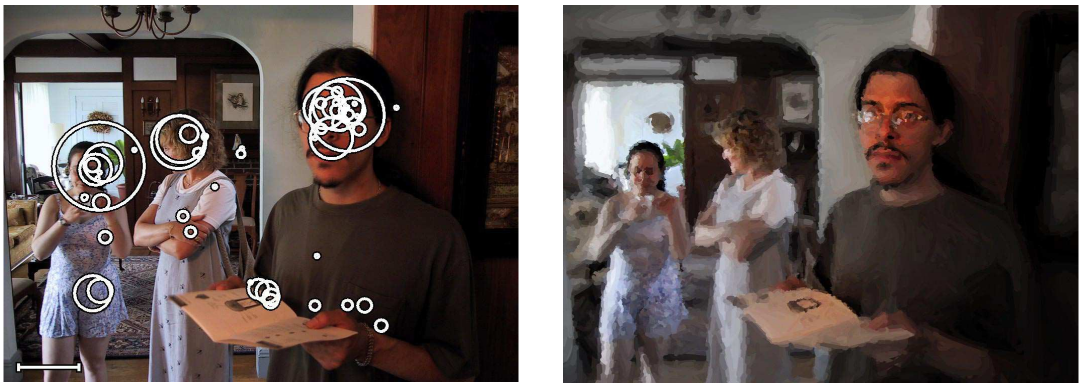
Some content-based algorithms exist, but they are only trained to detect and enhance low-level features such as salience (Collomosse & Hall, 2002; Klein et al., 2002), edges (Haeberli, 1990; Litwinowitz, 1997; Mignotte, 2003; Hays & Essa, 2004; Kim et al., 2008), color heuristics (Hertzmann, 1998; Shiraishi & Yamaguchi, 2000; Hertzmann, 2001; Gooch et al., 2002; Nehab & Velho, 2002; Collomosse et al., 2005; Winnemoller et al., 2006; DiPaola, 2007; Zhao et al., 2008; Park & Yoon, 2008), or interest points (Santella & DeCarlo, 2004), and not high-level semantic content such as people, skin and faces.

Another aspect of content is the behavior of the motion field in a video, and this can also be highlighted, enhanced or meaningfully abstracted. Figure 4.18, for example, shows the results of the algorithm of (Collomosse et al., 2003) which analyzes the motion of the source video. The video action is summarized for the viewer on one output frame as cartoon-like motion trail.

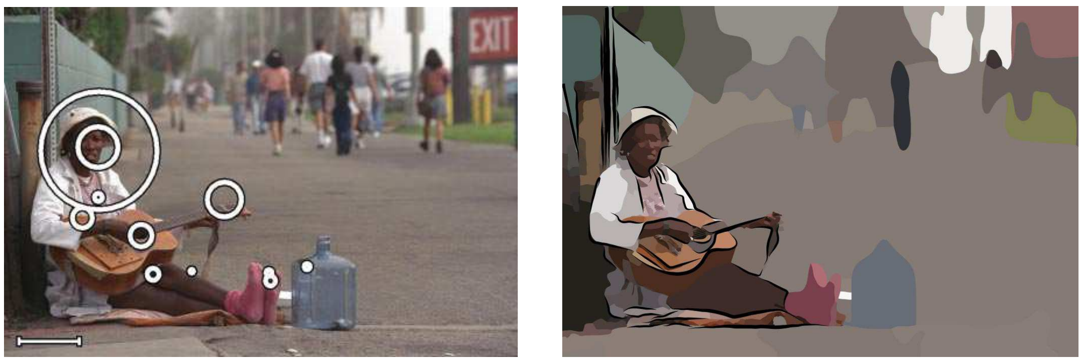
## 4.6 Motivations for a New NRP/SBR Algorithm

Given the subtle relationships between the various techniques in NPR; cartoonization, SBR, semantic stylization and motion summarization - it is interesting to explore the possibility

[5](Hays & Essa, 2004): `http://www.cc.gatech.edu/cpl/projects/artstyling/`



(a) Gaze-driven SBR



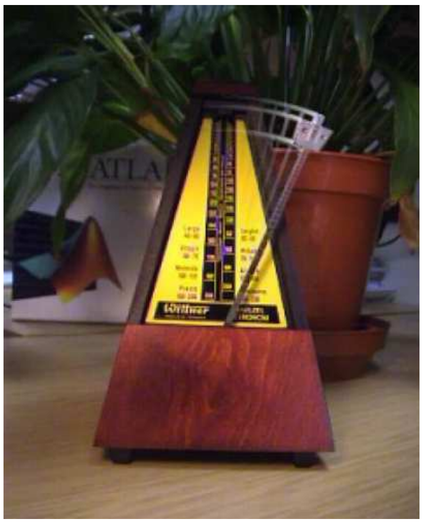
(b) Gaze-driven NPR

**Figure 4.17:** *Meaningful abstraction; (a) the user's eye-tracking data is used to drive finer brush strokes towards semantically meaningful image regions, and (b) toon-like edges are used to enhance the focal point of the image. Images in (a) and (b) are from (Santella & DeCarlo, 2002) and (DeCarlo & Santella, 2002) respectively.*

of merging these ideas into a single framework.

Footage from home movies and camera phone clips could benefit from an enhanced SBR/NPR stylization application. Since much personal visual media includes images and footage of family members, friends, groups, celebrities (i.e. head shots of people), the application could be aware of this content, incorporating it stylistically. It would be interesting to be able to stylize the facial regions of the people in these videos such that they evoke a kind of *animated artist's portrait*, with finer painting detail and perhaps cartoon-like edges highlighting the facial features, these smooth lines encapsulating the outline of the face and neck. This is novel in that it takes the idea of semantic stylization - as discussed in Section 4.5 - to a higher level of content-based manipulation not explored SBR/NPR algorithms previously.

Another aspect of salient content could be highlighted in video-based SBR if an indi-



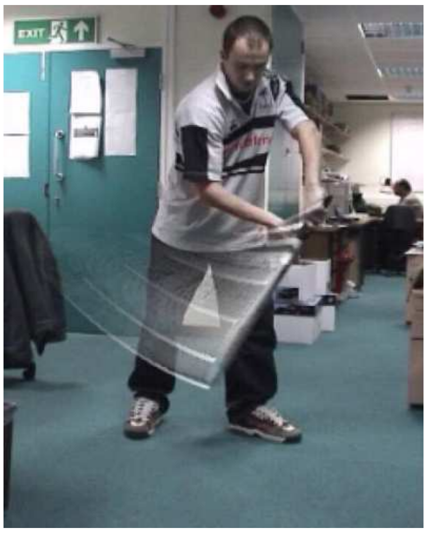


**Figure 4.18:** *Cartoon-like motion trails summarizing the video action in one frame. These images are from (Collomosse et al., 2003).*

cation of the motion trajectory of brush strokes - and hence the underlying video motion - could be visualized on each frame of the painted sequence individually. This effect could be achieved if some of the brush stroke attributes were made to change with regard to their underlying motion trajectories. Brush strokes could stretch according to the magnitude of, and rotate in the direction of the motion vector field from the source sequence. The concept is in a similar vein to the algorithms of (Collomosse et al., 2003; Denman et al., 2003) and (Kokaram et al., 2005) that attempt to summarize the action in a video on one frame for summarization and visualization purposes. Some of the results of these algorithms can be seen in Figure 4.18 and Figure 4.19. To incorporate this concept in an SBR rendering is novel, and the results could be useful for video summarization, cartoon-like styling or sequence story-boarding.

There is scope for general improvement in all aspects of the SBR process; the method of anchor point distribution, animating the brush stroke anchor points, dealing with occlusion and redundancy, uncovering and gaps, with regard to the Motion Estimation, spatio-temporally coherent brush stroke color-sampling, and experimentation with brush stroke attributes by defining style. The inclusion of cartoon-like edge painting - as discussed in Section 4.4.1 and demonstrated in Figure 4.14 - could make for an interesting hybrid effect.

Chapter 3 presents an algorithm for Face Detection in images. It is intuitive to imagine a non-uniform stylization framework incorporating Face Detection, in which a face segmentation mask defines, or helps to define the semantic region(s) for enhanced stylization. Recall that the face detector presented in Chapter 3 is capable of producing only a rough segmentation mask of faces in images (see Section 3.7.5 of Chapter 3 for explanation). To achieve

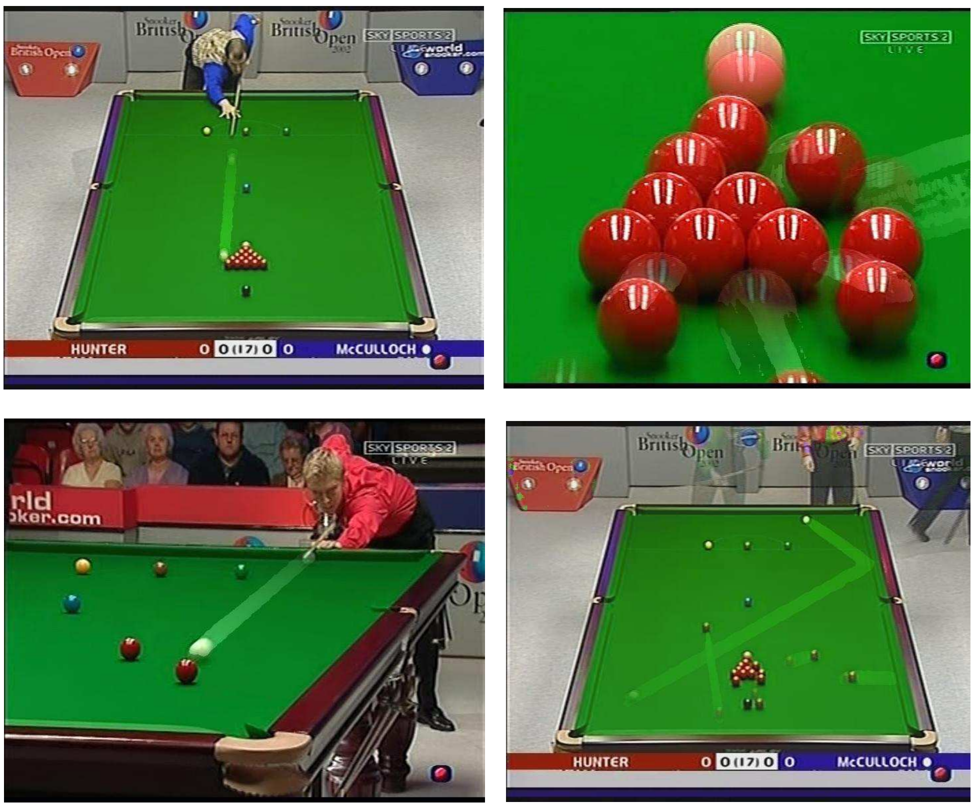


**Figure 4.19:** *Summarization of a game of snooker in a few key frames. All images are from (Denman et al., 2003).*

the desired "artist's portrait" style, however, a more pixel-accurate face segmentation mask would need to be produced. One solution would be to refine the face segmentation mask by applying a color-based pixel-wise skin detector to the image regions encapsulated by the face mask. Since color is definitely desired in the output stylized video sequences, and hence will be present in the input head shot videos, the Face Detection step then begins to seem redundant for this particular application. Furthermore, a skin-color augmented Face Detection step would not segment other interesting skin-colored non-face regions that might appear image sequences of head shots, such as glimpses of a person's neck, shoulders or hands.

Chapter 6 presents a novel framework merging many aspects of NPR; SBR, cartoonization, motion summarization and non-uniform semantic stylization. For reasons discussed above, the semantic content is defined not by Face Detection, but rather by a novel pixel-wise color-based Skin Detection algorithm which will be described in the next Chapter.

# 5 Graph Cut-Based Skin Detection

Chapter 4 presents a review of state-of-the-art techniques in the stylization of visual media and concludes that one of the motivations for improvement in this field is concerned with the semantic stylization of images or videos. Videos containing head shots of people, for example, could be stylized in a non-uniform fashion such that the skin and lines of the face and neck are enhanced. It is useful, therefore, to develop a reliable Skin Detection algorithm for this task. This Chapter presents a brief discussion of techniques in the field of Skin Detection, followed by a novel probabilistic Skin Detection algorithm with Graph Cut-based spatial smoothing.

## 5.1 On Skin Detection

As concluded in Section 4.6 of the previous Chapter, one of the motivations in the field of media stylization is concerned with the semantic stylization of videos, and a proposed application is the non-uniform stylization of videos containing head shots of people. The non-uniform stylization could enhance or emphasize the salient content encapsulated by the skin regions (e.g. cartoonization of the lines on the face and neck), while abstracting the less meaningful background content. What is required for this application, therefore, is a fairly accurate, spatio-temporally coherent skin detector that is capable of smoothly segmenting the skin-colored regions in the frames of a video containing head shots of people, producing a binary *skin mask* for each video frame. This is a problem of estimating the binary label field, $l_j = l(\mathbf{X})$, on image pixels, $\mathbf{X} = [x, y]$, such that

$$l_j = \begin{cases} 1 & \text{if skin pixel} \\ 0 & \text{if non-skin (i.e. background) pixel} \end{cases} \tag{5.1}$$

Skin Detection seems to be a well-researched topic in image processing, and most of the research revolves around probabilistic pixel-wise color segmentation methods. Surveys of some of the techniques and issues in Skin Detection are presented in (Vezhnevets et al., 2003) and (Phung et al., 2005) respectively, for the interested reader. It is notable that many state-of-the-art Skin Detection algorithms are *not* concerned with the creation of skin segmentation masks that are spatio-temporally - or even just spatially - coherent and smooth. Instead it seems that most of the debate in the field is concerned with which *color space* should be used for probabilistic pixel-wise skin segmentation.

There is much argument in favor using of the YCbCr color space for color-based Skin Detection, as opposed to the well-known RGB color space. The YCbCr color space is defined by a linear transformation of the pixel-wise RGB color data $\mathbf{c} = [r, g, b]$ (i.e. red, green, blue) to $\mathbf{c}' = [y, cb, cr]$ such that the luminance (i.e. illumination) information is encoded in the Y field of YCbCr, whereas chrominance (i.e. color) information is encoded in the Cb and Cr fields. In (Chai & Bourzerdoum, 2000), for example, it is claimed that a more illumination and skin-tone invariant detector results from discarding the luminance component, and modeling the conditional Probability Distribution Function (PDF) of skin color in the 2D space of Cb versus Cr. The detector produced by (Chai & Bourzerdoum, 2000), however, is only trained and tested on a handful of images and this hypothesis is refuted in many other studies.

(Jones & Rehg, 1999) estimate the pixel-wise skin and non-skin PDFs as 3D $32^3$-bin histograms of RGB data. These histograms are trained with the *positive* (i.e. skin) and *negative* (i.e. non-skin/background) pixels from tens of thousands of Internet images constituting the now well-known Compaq skin database created by the authors, (Jones & Rehg, 1999). A binary decision is made by thresholding the pixel-wise likelihoods produced by these histograms, and the final skin detector is reported by (Jones & Rehg, 1999) to perform better than one based on modeling the conditional PDFs as Gaussian Mixture Models (GMMs) in RGB space. This performance is measured in terms of *true positive* versus *false positive* skin pixels with regard to the hand-made ground truth masks associated with the Compaq database.

Support for the RGB color space can also be found in the work of (Mason & Brand, 2000), and in the surveys of (Vezhnevets et al., 2003) and (Phung et al., 2005), which both declare the 3D RGB histogram approach of (Jones & Rehg, 1999) to be one of the best performing skin detectors in the field. It is suggested in (Phung et al., 2005), however, that the algorithm is better with $256^3$ histogram bins - rather than the $32^3$-bin histograms suggested by the authors - provided that these histograms are estimated from a sufficiently large training data set like Compaq.

## 5.2 A New Graph Cut-Based Skin Detector

A fairly accurate, but not world-beating skin detector is required for the task of content-based semantic stylization of videos. It is apparent from the literature that a probabilistic technique with the RGB color space should be used, and that a large data set of positive and negative samples is needed for training. Therefore, the RGB values of the skin and non-skin pixels within the $4,000$ positive and $8,000$ negative images of the Compaq skin database (Jones & Rehg, 1999) are collected. Initially following the approach of (Jones & Rehg, 1999) as a first attempt, a *probability map*, $\Psi(\mathbf{X})$, is generated from the likelihood ratio

$$\Psi(\mathbf{X}) = \frac{\Theta p_s(\text{skin}|\mathbf{C}(\mathbf{X}))}{p_n(\text{non-skin}|\mathbf{C}(\mathbf{X}))} \tag{5.2}$$

where $p_s = p(\mathbf{C_s}|\text{skin})$ and $p_n = p(\mathbf{C_n}|\text{non-skin})$ are the $253^3$-bin RGB histogram-estimated likelihoods, $\mathbf{C}_s$ and $\mathbf{C}_p$ the colors extracted from the positive and negative training pixels respectively, and $\mathbf{C}(\mathbf{X})$ is in RGB color space. Since relatively fewer positive than negative training examples are used to form the histograms, $\Theta$ is a weight that can be used to boost the relative influence of the positive likelihood, $p_s(.)$, in the ratio if necessary, and it is therefore often set $> 1$ as in (Jones & Rehg, 1999). The entries in the probability map, $\Psi_j = \Psi(\mathbf{X})$, are useful for visualization, as can be seen in Figure 5.1 (b). Here $\Theta = 1$ for simplicity.

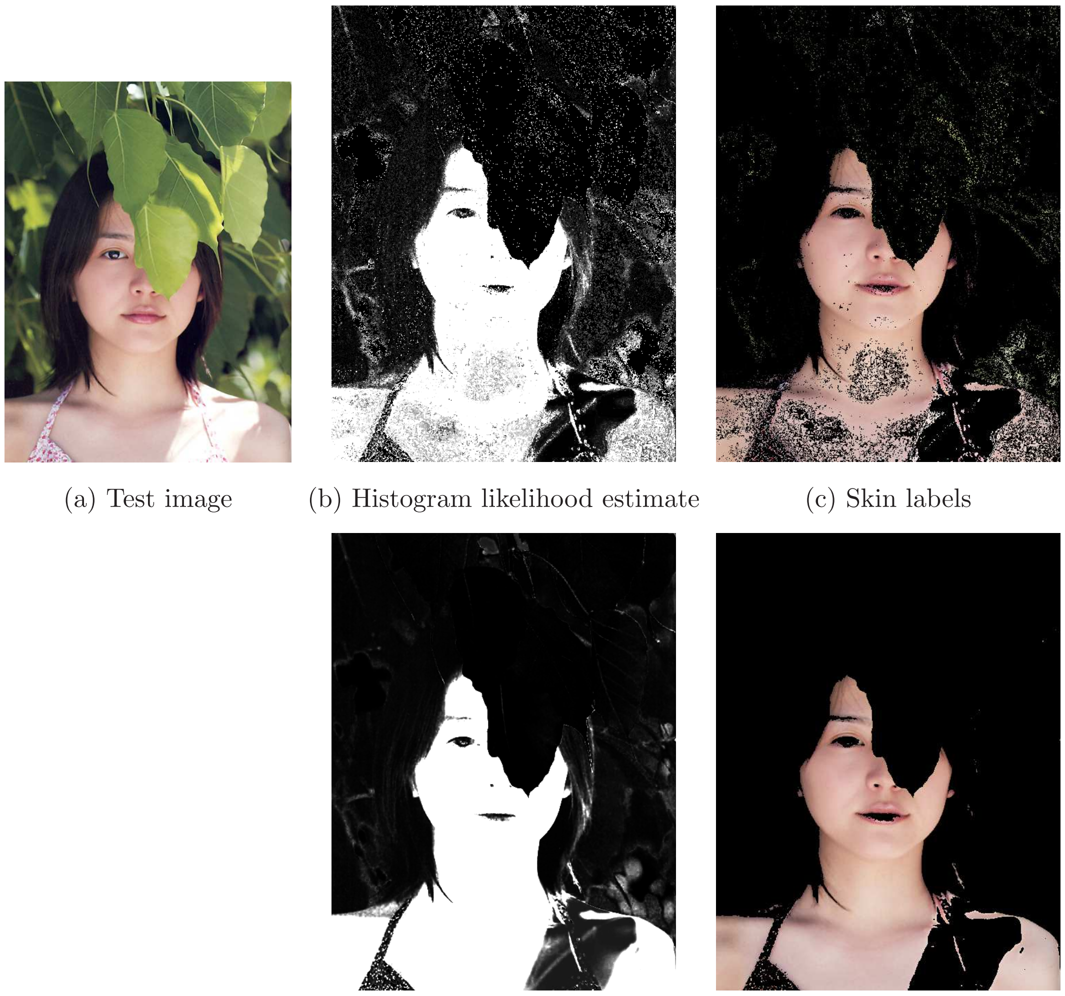
The binary labeling defined in Equation 5.1 is then established by

$$l_j = \begin{cases} 1 & \text{if } \Psi_j > t_\Psi \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

where $t_\Psi$ is a threshold on the probability map, or a trade-off between true and false positive skin pixels, as discussed in (Jones & Rehg, 1999). The skin mask produced by binary thresholding the 3D histogram-based likelihood ratio of Figure 5.1 (b) can be seen in Figure 5.1 (c). It is clear to see that the mask is not very spatially smooth.

### 5.2.1 The Bi-Gaussian Likelihood Model

Experimentation with different forms of the likelihood ratio has led this author to the discovery that the conditional PDFs of skin and non-skin are well modeled as unimodal multi-variate Gaussian distributions in RGB space (i.e. not 3D histograms, or a GMM). This multi-variate Bi-Gaussian (B-G hereafter) likelihood model is defined as


(a) Test image (b) Histogram likelihood estimate (c) Skin labels

(d) B-G likelihood estimate (e) Skin labels

**Figure 5.1:** *Probability maps and skin masks; (a) test image, (b) probability map, $\Psi(\mathbf{X})$, formed by 3D histogram-based likelihood ratio and (c) the positive labeling, $l(\mathbf{X}) = 1$, with threshold $t_\Psi = 0.99$, (d) $\Psi(\mathbf{X})$ formed by Bi-Gaussian (B-G) likelihood ratio, and (d) comparative $l(\mathbf{X}) = 1$ with $t_\Psi = 0.99$. $\Theta = 1$ in both cases.*

$$p_{mg} = \begin{cases} \frac{1}{\sqrt{2\pi}^{\rho/2}|\Sigma_p|^{1/2}} e^{-1/2\left((\mathbf{c}_j-\mathbf{c}_{\mu_\mathbf{p}})^T\Sigma_p^{-1}(\mathbf{c}_j-\mathbf{c}_{\mu_\mathbf{p}})\right)} & \text{for } l_j = 1 \\ \frac{1}{\sqrt{2\pi}^{\rho/2}|\Sigma_n|^{1/2}} e^{-1/2\left(\mathbf{c}_j-\mathbf{c}_{\mu_\mathbf{n}})^T\Sigma_n^{-1}(\mathbf{c}_j-\mathbf{c}_{\mu_\mathbf{n}})\right)} & \text{for } l_j = 0 \end{cases} \tag{5.4}$$

where $\rho = 3$ since there are three variables in the pixel-wise RGB vectors, $\mathbf{c}_j = [r_j, g_j, b_j]$, of $\mathbf{C}(\mathbf{X})$, and $\Sigma_p$, $\Sigma_n$, $\mathbf{c}_{\mu_\mathbf{p}}$ and $\mathbf{c}_{\mu_\mathbf{n}}$, are the covariance matrices and mean color vectors of the positive and negative training pixels of the Compaq skin database respectively. Again, $\Theta$ is a weight that can be used to boost the relative influence of the positive likelihood.

Figure 5.1 (d) is the result of computing the probability map, $\Psi(\mathbf{X})$, from the B-G modeled likelihood ratio, and Figure 5.1 (e) is the resulting binary thresholded skin mask, again with $t_\Psi = 0.99$ and $\Theta = 1$. It is clear to see that the unimodal B-G likelihood ratio results in a more spatially coherent probability map and skin mask than those produced by the 3D histogram-based likelihood ratio, as can be seen in Figures 5.1 (b) and (c). The B-G likelihood model is also less costly to compute than the 3D histogram-based likelihood.

There are still a few blob-like false positives and negatives within the B-G likelihood generated skin mask of Figure 5.1 (e), however, but it is likely that this problem would be largely rectified by injecting contextual spatial information into the binary labeling solution.

### 5.2.2 A Bayesian Approach

Similarly to the sparse local feature point labeling technique in Section 3.7 of Chapter 3, it is a good idea to take a Bayesian approach to this problem of spatial smoothing. The goal, as previously described, is to label image sites as skin (i.e. $l_j = 1$) or background (i.e. $l_j = 0$) class according to the color of the pixel at each site, $\mathbf{c}_j = [r_j, g_j, b_j]$. Proceeding in a Bayesian fashion, this amounts to maximizing the following

$$\underbrace{p(l_j|\mathbf{c}_j)}_{posterior} \propto \underbrace{p(\mathbf{c}_j|l_j)}_{likelihood} \underbrace{p(l_j|\mathbf{L}_j)}_{prior} \tag{5.5}$$

where the two-class likelihood is realized as a unimodal multi-variate B-G as previously described in Equation 5.4, the prior assumes a Markov Random Field (MRF) (similarly to that in Section 3.7.3 of Chapter 3), and $\mathbf{L}_j$ is the spatial neighborhood on which the MRF is assumed.

### 5.2.3 The Prior

The spatial prior is modeled as

$$p_q(l_j|\mathbf{L}_j) \propto \exp -\Big\{ \sum_{s \in S} U_j |l_s \neq l_j| \Big\} \tag{5.6}$$

where $\mathbf{L}_j$ is the 4-pixel spatial neighborhood of labels surrounding $l_j$, and the term $U_j$ is a contrast-dependent smoothness term that ensures maximum smoothness in image regions of low contrast, and vice versa (see (Rother et al., 2004) or (Corrigan et al., 2008) for a more complete explanation of the term $U_j$).

### 5.2.4 Graph Cut-Based Optimization

The Graph Cut algorithm (Boykov & Jolly, 2001) is a recently popular technique for solving 2D binary labeling problems such as the one just defined. Graph Cut is an energy minimization technique, but unlike Iterated Conditional Modes (ICM) (Besag, 1986), it calculates the optimal binary segmentation (i.e true global minimum) with regard to the Bayesian energy function, which in this case is

$$E_j(l_j = 1) = \{E_j(p_{mg}) + \alpha_{gc}E_j(p_q) + E_j(\Theta)\} \tag{5.7}$$

$$E_j(l_j = 0) = \{E_j(p_{mg}) + \alpha_{gc}E_j(p_q)\} \tag{5.8}$$

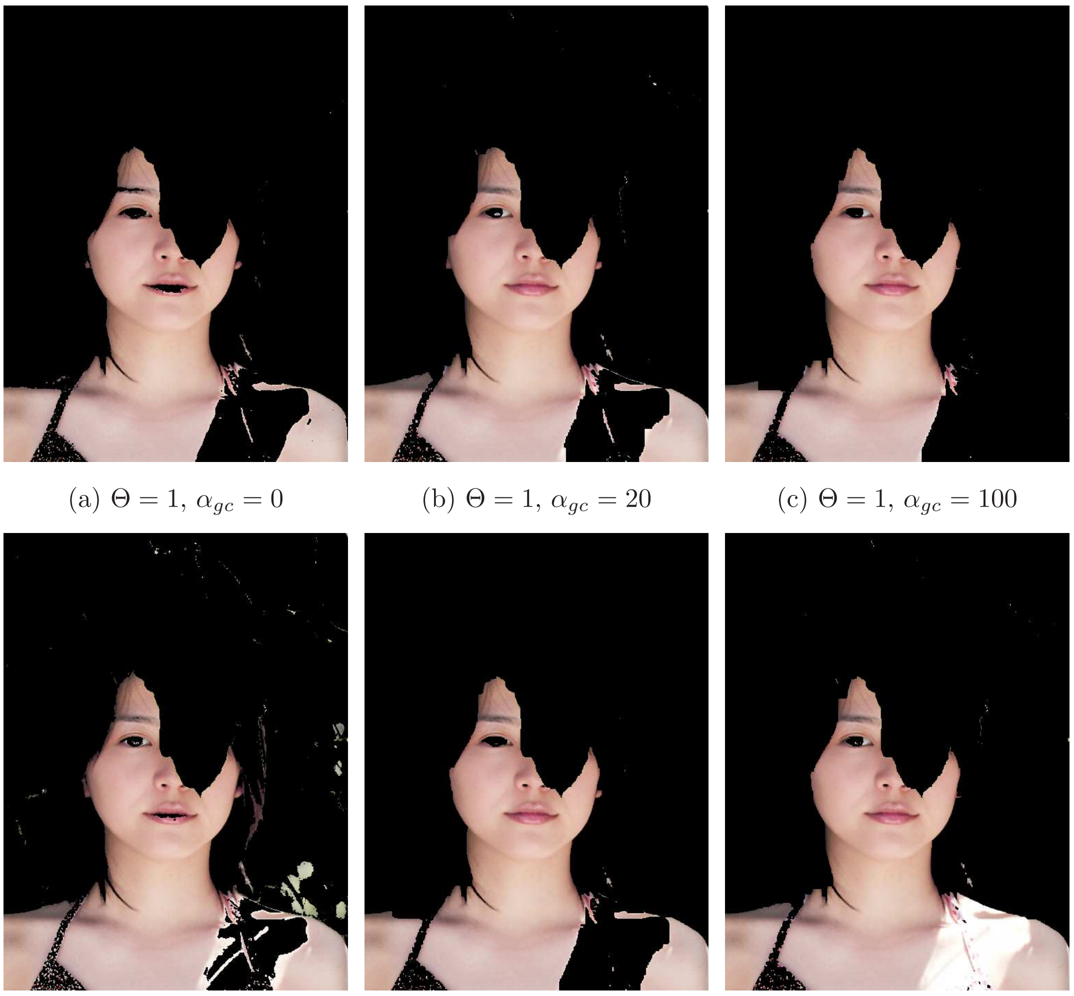
where $E(p) = -\log(p(.))$, $E_j(p_{mg})$ is the B-G likelihood energy, $E_j(p_q)$ is the prior energy, and $\alpha_{gc}$ is a weight controlling the influence of the prior (similarly to $\alpha_d$ in the related Equations 3.6 and 3.7 in Section 3.7 of Chapter 3). Recall that $\Theta$ is a weight that can be adjusted to boost the relative influence of the positive $p_{mg}$ in likelihood ratio of Equation 5.2. Equation 5.7, therefore, ends up with the term $E_j(\Theta)$, which is a result of computing the log energy of the positive likelihood multiplied by the constant $\Theta$. This term has the effect of suppressing the likelihood energy if $(\Theta > 1)$, but it may also be used for the opposite purpose if $(\Theta < 1)$.

The optimum binary skin mask labeling is found by means of the (Corrigan et al., 2008) implementation of the max-cut/min-flow Graph Cut optimization algorithm of (Boykov & Kolmogorov, 2004). As can be seen in Figures 5.2 (a-f), the result depends largely on the values of $\Theta$, and $\alpha_{gc}$. As the smoothing factor, $\alpha_{gc}$, is increased, smaller binary connected regions seem to either merge together to form larger ones, or else disappear from the skin mask entirely.

The former effect is interesting in that it can smooth the skin mask over internal facial features such as the girl's lips, and this may or may not be desirable depending on the application. The latter effect can result in large false negative patches, as can be seen on the girl's shoulder in Figure 5.2 (c) which is critical due to illumination conditions at this region. However, an increase in $\Theta$ can help, because it will boost the relative influence of the skin likelihood such that the likelihood energy becomes less than the smoothing energy at such critical image regions. The best result overall can be seen in Figure 5.2 (f), where the girl's shoulder is correctly included in the skin mask. There are still a few small false positive blobs in the background but these are acceptable for the particular application which will be presented in the next Chapter.

The content-based video stylization application of the following Chapter uses this novel Skin Detection algorithm for obtaining a spatially coherent skin mask for each frame of an input video. This spatially smoothed skin mask sequence is then temporally smoothed by the video stylization framework in a process of spatio-temporal median filtering. The reasons and technique for the temporal smoothing will be discussed later in Section 6.4 of the following Chapter.

Some examples of spatially smooth skin masks resulting from the frame-wise application



(a) $\Theta = 1$, $\alpha_{gc} = 0$ (b) $\Theta = 1$, $\alpha_{gc} = 20$ (c) $\Theta = 1$, $\alpha_{gc} = 100$

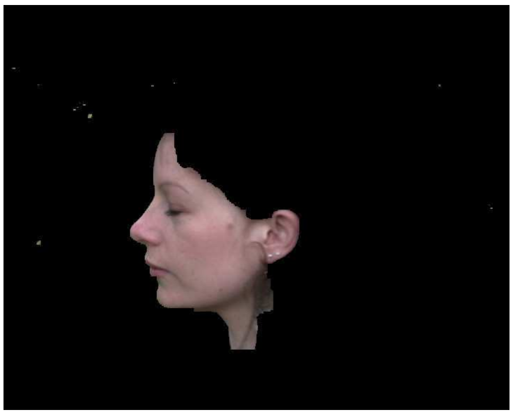
(d) $\Theta = 4$, $\alpha_{gc} = 0$ (e) $\Theta = 4$, $\alpha_{gc} = 20$ (f) $\Theta = 4$, $\alpha_{gc} = 100$

**Figure 5.2:** *Graph Cut-based skin segmentation; (a) binary skin mask with weight* $\Theta = 1$ *and smoothing factor* $\alpha_{gc} = 0$, *(b)* $\Theta = 1$ *and* $\alpha_{gc} = 20$, *(c)* $\Theta = 1$ *and* $\alpha_{gc} = 100$, *(d)* $\Theta = 4$ *and* $\alpha_{gc} = 0$, *(e)* $\Theta = 4$ *and* $\alpha_{gc} = 20$, *and (f)* $\Theta = 4$ *and* $\alpha_{gc} = 100$.

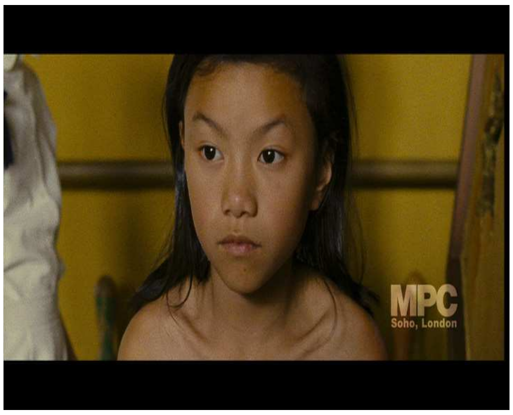
of Graph Cut-based Skin Detection algorithm to two video sequences - *Female1* and *Hollywood* - can be seen in Figure 5.3. The two resulting spatially coherent skin mask sequences - *Female1_GCskin* and *Hollywood_GCskin* - are included on the DVD accompanying this thesis. It is clear to see from Figure 5.3 (a) and (b), that the Skin Detector works well on this particular example in which the background is far from skin-colored. Figure 5.3 (c) and (d), however, demonstrate that this, like many skin detectors, can sometimes fail in difficult illumination conditions and in the presence of skin-colored backgrounds. The *Hollywood_GCskin* mask for the full *Hollywood* sequence also reveals some mis-classification of light-colored hair as skin.

As will become clear in the next Chapter, however, only a rough skin detector is needed

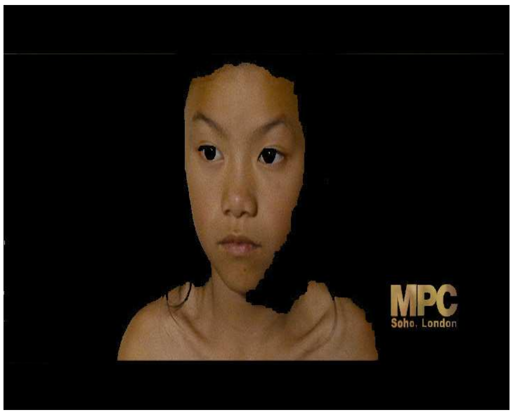
(a) *Female1* video



(b) $\Theta = 10$, $\alpha_{gc} = 500$



(c) *Hollywood* video



(d) $\Theta = 0.8$, $\alpha_{gc} = 10$

**Figure 5.3:** *Graph Cut-based skin segmentation; (a) a single frame of source video Female1, (b) binary skin segmentation with* $\Theta = 10$ *and* $\alpha_{gc} = 500$*, (c) single frame of source video Hollywood, and (d) binary skin segmentation with* $\Theta = 0.8$ *and* $\alpha_{gc} = 10$*.*

for the application of skin-aware non-uniform video stylization, and slight failures of the skin detector are not critical within the video stylization framework.

# 6 Skin-Aware Stylization of Video Portraits

This Chapter presents a new Non-Photorealistic/Stroke-based Rendering (NPR/SBR) framework for the skin-aware stylization of "video portraits" featuring head shots of people, such as home videos, movies, and camera mobile phone clips. Spatio-temporal Skin and Edge Detection are used to locate and emphasize the semantic content in the stylization process. The SBR portion of the algorithm features novel techniques for motion expression with elliptical brush strokes, brush stroke anchor point distribution, spatio-temporal color-sampling, and interesting solutions to state-of-the-art issues in motion-compensated brush stroke animation such as painting redundancy and object occlusion, gaps and object uncovering. A wide user-accessible parameter space and finishing touches such as cartoon-like edge decoration and other quirky effects empowers a variety of artistic outputs. The user can vary the stylization parameters for three uniquely defined spatio-temporal semantic layers in the image sequence, i.e. the background, foreground (i.e. skin regions), and detail regions (i.e edges). The resulting stylized sequences are interesting with regard to compression, summarization, story-boarding, and art. Both the semantic content, and underlying video motion is highlighted and summarized on every frame of the stylized output sequence.

## 6.1 The Stylization Framework

A novel framework for the skin-aware stylization has been created. The framework incorporates elements of Non-Photorealistic Rendering (NPR) and Stroke-Based Rendering (SBR) that were discussed in Chapter 4, and plays with the idea of non-uniform *semantic* styl-

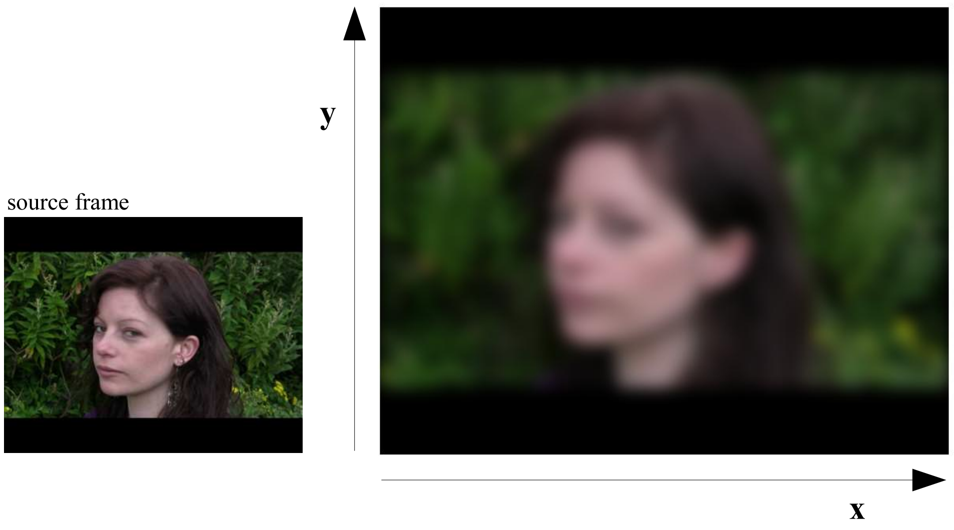


**Figure 6.1:** *Canvas underpainting; the source image blurred with a Gaussian kernel. The source image is from the sequence Female1 included on the DVD accompanying this thesis.*

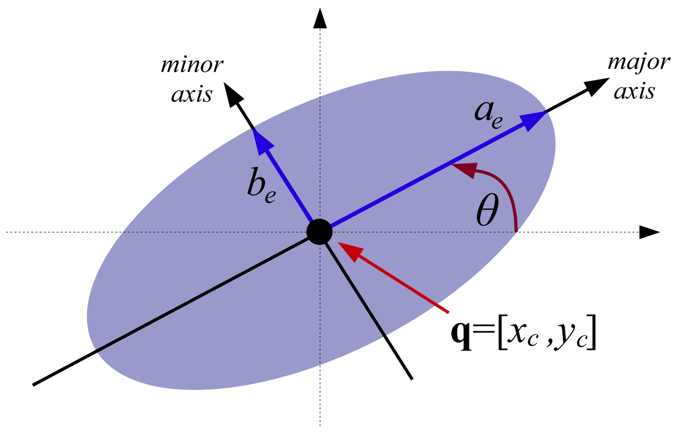
ization of skin regions empowered by the novel Graph Cut-based Skin Detection algorithm discussed in Chapter 5.

## 6.2 Elliptical Brush Strokes

The SBR (i.e. painting) portion of the framework is similar to that of (Haeberli, 1990) in that brush strokes are placed on a canvas, and these brush strokes are implemented as structures with attributes that are established and queried at various stages in the SBR process. The attributes correspond to the brush stroke's anchor point location, shape, dimensions, orientation, color, opacity, and noise with regard to color and orientation. The SBR portion of this framework plays with the dimensions and orientation attributes of brush strokes for motion expression and exaggeration.

Before SBR takes place, the canvas is blank or optionally primed with an underpainting as in (Gooch et al., 2002). This underpainting is simply a blurred version of the source frame with a symmetric Gaussian blur kernel of size $50 \times 50$ pixels and standard deviation, $\sigma_g = 10$, as can be seen in Figure 6.1. Brush strokes are placed on the canvas using a novel anchor point distribution discussed in Section 6.3. As will be discussed in Section 6.4.2, these brush strokes are distributed in a non-uniform, content-based fashion featuring three individual *semantic* (i.e. meaningful) layers defined created by means of spatio-temporal Skin and Edge Detection. Strokes are then organized and stored in three ordered lists; background, foreground (i.e. skin) and detail (i.e. edges), and finally, composited in that order to paint each frame.

Inspired by animated SBR algorithms such as (Litwinowitz, 1997; Hertzmann, 2001) and



(a) Ellipse parameters

(Hays & Essa, 2004), the brush strokes are animated in video using techniques incorporating Motion Estimation (Kokaram, 1998). This part of the algorithm is discussed in Section 6.4.3, along with some novel solutions to the common problems encountered in motion-compensated video-based SBR.

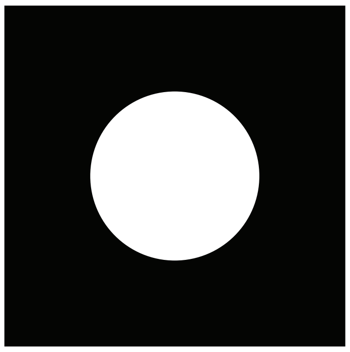
Other elements of NPR such as spatio-temporal cartoonization are also incorporated in the framework to create interesting effects. These ideas are discussed in Section 6.7. Since these effects are considered finishing touches, it is first necessary to discuss the SBR portion of the framework, and all of the steps involved in motion-compensated brush stroke animation.

The chosen brush stroke shape is elliptical. The look achieved by painting with this shape is more like particle simulation than real-life paint modeling. Emulating the look of real paint strokes is not important in this work and therefore brush stroke textures and paint-like lighting models are not implemented. The goal is the distribution and painting of smooth anti-aliased elliptical particles that can move, stretch and rotate to illustrate the magnitude and direction of the underlying video motion. The painted area of an elliptical brush stroke is determined by a binary mask $s(\mathbf{X})$ defined as follows

$$s(\mathbf{X}) = \begin{cases} 1 & \text{where } \frac{[(x-x_c)\mathbf{R}_\theta]^2}{a_e^2} + \frac{[(y-y_c)\mathbf{R}_\theta]^2}{b_e^2} \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

where $\mathbf{X} = [x, y]$ are points on the canvas coordinate system, and $x$ and $y$ are the horizontal and vertical components of these coordinates respectively. The parameters $a_e$, $b_e$, $x_c$, $y_c$ and $\theta$ can be visualized in Figure 6.2 (a), with $\mathbf{R}_\theta$ related to $\theta$.

The centroid of the ellipse corresponds to the anchor point location of the brush stroke, therefore $\mathbf{q} = [x_c, y_c]$. The following rotation matrix determines the orientation of the


(b) $r_{eq} = 100$, $k_e = 1$

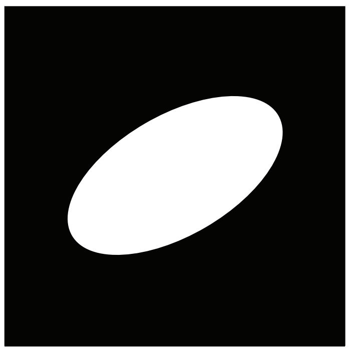

(c) $r_{eq} = 100$, $k_e = 2$, $\theta = 0.52$

**Figure 6.2:** *Elliptical brush strokes; (a) essential parameters, (b) and (c) two brush stroke masks, $s(\mathbf{X})$, with the same circle equivalent radii, $r_{eq}$, but different orientation and eccentricity parameters.*

ellipse, $\theta$,

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \tag{6.2}$$

where $\theta$ is the angle between the horizontal (x) axis and the plane defined by the major axis. The lengths of $a_e$ and $b_e$, the major and minor axes, determines the area, $\phi_e = \pi a_e b_e$ of the ellipse, while the relative lengths of these axes determine the eccentricity. To set $a_e = b_e$ means that the ellipse becomes a circle of radius, $r_{eq}$, as can be seen in Figure 6.2 (b). The ratio $k_e = \frac{a_e}{b_e}$ is related to the eccentricity. One can stretch the ellipse by varying $k_e$ while holding the area, $\phi_e$ constant. The two masks seen in Figure 6.2 (b) and (c) have equivalent $\phi_e$, and so it can be said that each of these masks has an *equivalent circle radius* of $r_{eq} = 100$ in this case.

### 6.2.1 Calculating the Elliptical Stretch and Orientation

The elliptical brush stroke parameters can be manipulated to depict the behavior of the motion field underlying the source image sequence. This idea can be visualized in Figure 6.3. Here, it is useful to have a brief understanding of the process of animating a brush stroke, although this will be described in Section 6.4.3 in more detail later. First, the motion of the source image sequence is obtained by the process of Motion Estimation. There are many Motion Estimation algorithms as this area remains an active research topic. It is the Motion Estimation algorithm of (Kokaram, 1998) that is used in this work. To animate a brush stroke, a stroke's anchor point $\mathbf{q}^n$ is translated according to the underlying video motion such that $\mathbf{q}^{n+1} = \mathbf{q}^n + \mathbf{f}^n(\mathbf{q}^n)$, where $\mathbf{f}^n(\mathbf{X})$ is the forward motion vector field describing the motion of all pixels in frame $n$ of the source image sequence to their displaced positions

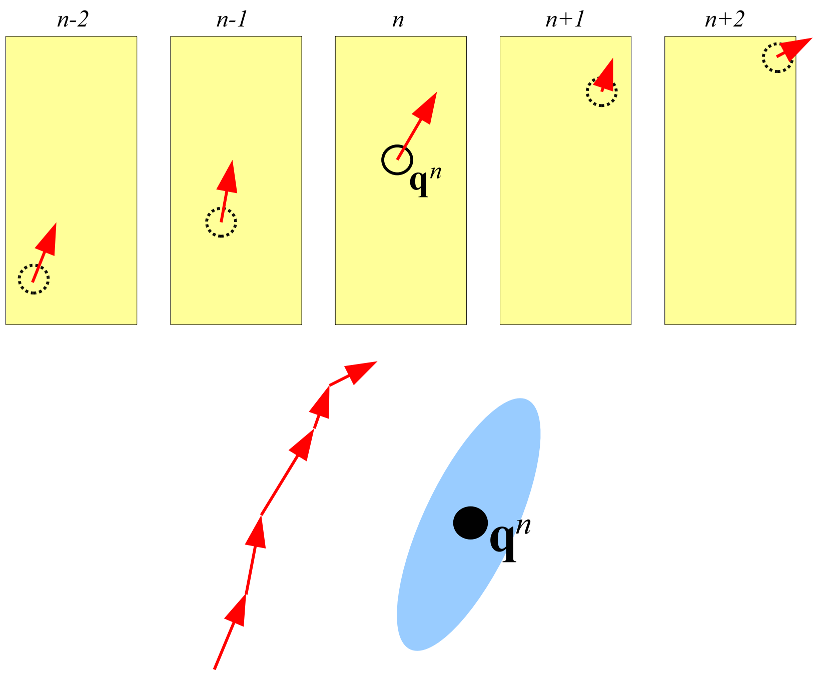


**Figure 6.3:** *Portraying the video motion in the appearance of brush strokes. The motion of the brush stroke's anchor point trajectory is used to determine the stroke's stretch and orientation.*

in frame $n + 1$.

In Figure 6.3, the motion of the brush stroke's anchor point trajectory is used to influence the brush stroke's stretch and orientation attributes - and therefore painted area, $s^n(\mathbf{X})$, for the current frame $n$. The parameters defining these attributes are temporally smoothed by sampling the motion over a temporal window centered on $n$ and extending over a number of frames, $n_u$. Figure 6.3 only depicts the forward motion field, $\mathbf{f}^n(\mathbf{X})$, but a backward motion field, $\mathbf{b}^n(\mathbf{X})$, is also estimated. To implement the idea of Figure 6.3, each brush stroke is associated with a vector encapsulating the following motion information

$$\mathbf{u}^n = \frac{1}{N_u} \sum_{j=-(N_u-1)/2}^{(N_u-1)/2} \left[\mathbf{f}^{n+j}(\mathbf{q}^{n+j}) - \mathbf{b}^{n-j}(\mathbf{q}^{n+j})\right] \tag{6.3}$$

where $\mathbf{f}^n(\mathbf{X})$ and $\mathbf{b}^n(\mathbf{X})$ are the forward and backward motion vector fields over a number of frames, $N_u$, and centered on the current frame, $n$.

Unlike the simplified illustration in Figure 6.3, Equation 6.3 utilizes the bi-directional motion field, i.e. $\mathbf{f}^n(\mathbf{X})$ and $\mathbf{b}^n(\mathbf{X})$, the forward and backward motion vectors respectively. The inclusion of both helps to suppress some errors in the estimation due to the phenomena of occlusion and uncovering inherent to the Rubber Sheet Model (see Section 4.4.1 for explanation). Therefore, $\mathbf{u}^n$ for a particular $\mathbf{q}^n$ can be formed by sampling $\mathbf{f}^n(\mathbf{X})$ and $\mathbf{b}^n(\mathbf{X})$ at coordinates corresponding to the motion translated position of $\mathbf{q}^n$ in each frame of the temporal window, $j$. The magnitude of motion at $\mathbf{q}^n$ is expressed as

$$u^n = \sum_{j=1}^{N_u} \sqrt{(\mathbf{u}_y^j)^2 + (\mathbf{u}_x^j)^2} \tag{6.4}$$

where the terms $\mathbf{u}_y^j$ and $\mathbf{u}_x^j$ refer to the vertical (y) and horizontal (x) components of motion respectively. To make the brush strokes stretch and orientate smoothly in relation to the underlying video motion, a stretch ratio, $k_e^n$, and orientation, $\theta^n$ (in radians), is determined from $u^n$ as follows

$$k_e{}^n = u^n \frac{\hat{k}_e}{\hat{u}} \ , \quad \theta^n = \arctan \frac{\sum_{j=0}^{N_u} \mathbf{u}_y^n}{\sum_{j=0}^{N_u} \mathbf{u}_x^n} \tag{6.5}$$

where $\hat{u}$ is a user-defined parameter reflecting the assumed maximum motion magnitude. The maximum stretch of the elliptical stroke is limited to the user-defined maximum $\hat{k}_e$. The motion field obtained by the Motion Estimation algorithm of (Kokaram, 1998) is spatially very smooth and therefore noise in the range $[-\delta\theta, +\delta\theta]$ may be added to $\theta^n$ to simulate a more hand-painted look.

### 6.2.2 An Alternative Technique

A similar and interesting effect can be achieved by measuring $\mathbf{u}^n$ at the non-motion translated locations of $\mathbf{q}^n$. This changes the form of Equation 6.3 to

$$\mathbf{u}^n = \frac{1}{N_u} \sum_{j=-(N_u-1)/2}^{(N_u-1)/2} \left[\mathbf{f}^{n+j}(\mathbf{q}^n) - \mathbf{b}^{n-j}(\mathbf{q}^n)\right] \tag{6.6}$$

and Equation 6.5 is still used to determine orientation. This idea can be visualized in Figure 6.4. In this case the motion information is no longer motion-compensated, but it is a faster way to achieve motion-expressive brush strokes.

## 6.3 Probabilistic Stroke Distribution

As discussed in Section 4.3.1, there are certain goals when placing brush stroke anchor points on the canvas. To recap, the aim is to cover the canvas with strokes such that they

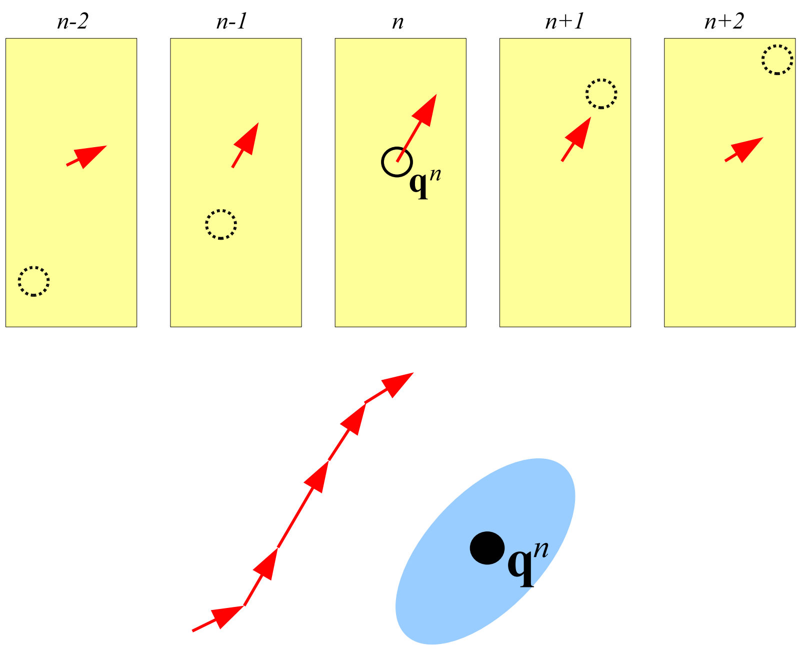


**Figure 6.4:** *Portraying the video motion in the appearance of brush strokes; a faster technique. The motion field at the position of stroke's anchor point in the current frame is used to determine the stroke's stretch and orientation.*

are sparsely distributed with minimal overlap, and to minimize gaps in the painting. Tiling the canvas with a grid-like formation of brush strokes results in a synthetic appearance, and noise must be added to the anchor point locations. A pseudo-random placement of strokes looks more natural, but there is redundancy in the canvas coverage when strokes overlap excessively.

A novel probabilistic point distribution process has been developed to deal with these issues. This process is somewhat related to the idea of Poisson Disk Sampling (PDS), as mentioned in Section 4.3.1. In PDS, point samples are distributed on a plane (i.e. the canvas) in a series of *trials*. In each trial, a candidate sample is generated pseudo-randomly. The candidate is rejected if it falls within a disk of radius $r_{disk}$ surrounding a previously generated sample. This is an interesting idea in that $r_{disk}$ could be associated with the dimensions of the brush strokes. However, brush strokes dimensions might not be uniform over the canvas. Recall that the strokes associated with this video stylization algorithm are elliptical in shape, and that $a_e$, $b_e$ and $\theta$ will be altered to reflect a motion trajectory underlying the anchor point. The following novel point distribution algorithm attempts to soften and adapt the PDS process by posing it in a probabilistic light.

Suppose that we wish to generate anchor points on the canvas to distribute brush stroke anchor points in a series of trials, $i$. The canvas has the coordinate system $\mathbf{X} = [\mathbf{x}, \mathbf{y}]$ as

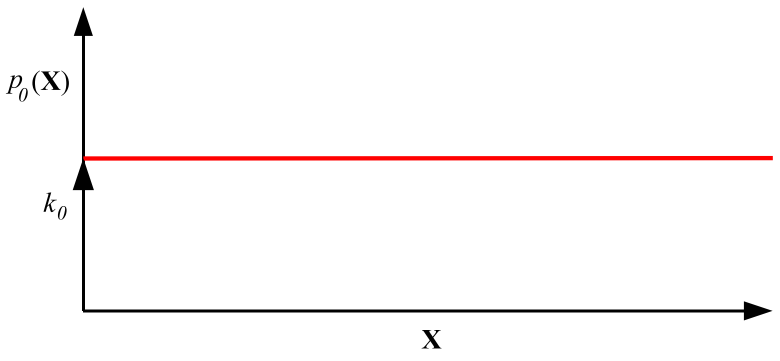


**Figure 6.5:** *Probabilistic anchor point distribution; there is an initial uniform distribution of probability over the canvas coordinates,* $\mathbf{X}$.

before. The anchor point generated by each successful trial will be known as $\mathbf{q}_i = [x_{c_i}, y_{c_i}]$. During these trials, each location on the canvas is associated with a probability. This probability is the likelihood of sampling at that location in trial $i$. At the beginning of the process, therefore, we have an initial uniform distribution of probability

$$p_0(\mathbf{X}) = k_0 \tag{6.7}$$

Figure 6.5 demonstrates this idea. Here, $k_0$ is a constant. To generate the first anchor point, $\mathbf{q}_0$, a sample is drawn from this distribution numerically

$$(\tilde{x}_0, \tilde{y}_0) \sim p_0(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{q}_0 = [x_{c_0}, y_{c_0}] \tag{6.8}$$

The aforementioned PDS algorithm would maintain a uniform distribution throughout the sampling process, but explicitly reject those points whose sampled coordinates fall within radius $r_{disk}$ of an existing point. Here however, the probability distribution is modified after the placement of a brush stroke so as to suppress the likelihood of anchoring another stroke nearby

$$(\tilde{x}_i, \tilde{y}_i) \sim p_i(\mathbf{X}) \mapsto \mathbf{q}_i = [x_{c_i}, y_{c_i}] \ , \quad p_i(\mathbf{X}) = p_{i-1}(\mathbf{X}) p_{b_{i-1}}(\mathbf{X}) \tag{6.9}$$

where the $p_{i-1}(\mathbf{X})$ is the distribution from which the previous anchor point was generated, and $p_{b_{i-1}}(\mathbf{X})$ is the modification that resulted from the previous trial. The modification models a suppression field with peak at the previous brush stroke's anchor point, $\mathbf{q}_{i-1}$, and smoothly decreasing radially outwards from the center. Generally, this Gauss-like suppression function generated in trial $i$ is defined as

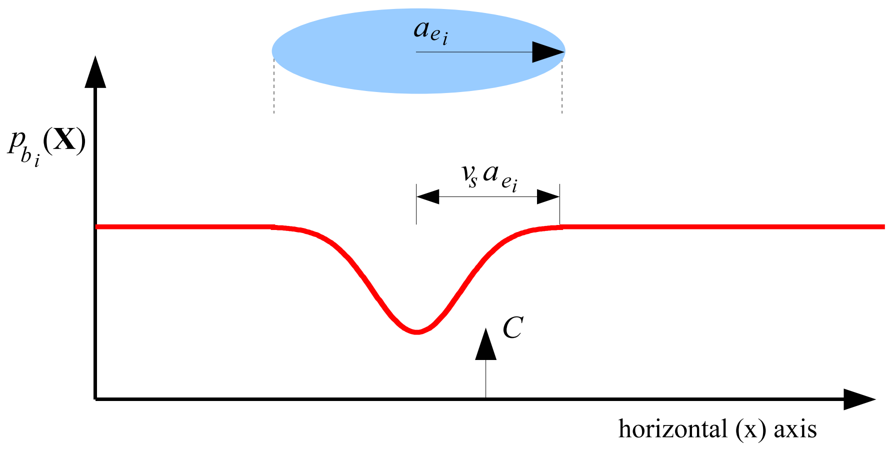


**Figure 6.6:** *Gaussian suppression function for coordinates* $\mathbf{x}$ *when the brush stroke's major axis,* $a_{e_i}$*, is in line with the horizontal axis. In this case* $C > 0$ *and* $v_s = 1$*.*

$$p_{b_i}(\mathbf{X}) = \frac{1}{Z}\left[C - e^{-(v_s \mathbf{X}^T \mathbf{S} \mathbf{X})}\right] \tag{6.10}$$

where $Z$ and $C$ are a normalizing and scaling factor respectively, $\mathbf{S}$ is a 2D covariance matrix whose diagonal entries correspond to $a_{e_i}$ and $b_{e_i}$ - the length of the major and minor axes of - the elliptical stroke anchored at $\mathbf{q}_i$, and $v_s \in \Re$ is a weight. $C = 0$ is the special case preventing two anchor points from ever being placed in the same location on the canvas.

Figure 6.6 shows the effect of this suppression function on the probability distribution projected onto the horizontal (x) axis, for the simplified case of a brush stroke with major axis, $a_{e_i}$, in line with the horizontal axis. Figure 6.7 (a) shows the result of a number of brush stroke placement trials, and Figure 6.7 (b) is an overhead view of the effect of the suppression function on the canvas probability distribution after these trials. Dark areas represent low probability and bright areas represent high probability.

If $C > 0$ there can be an infinite number of trials to place brush stroke anchor points, $\mathbf{q}_{i..\infty}$, so there must be a point when it is decided that the painting is finished. The algorithm should ensure that the canvas is sufficiently covered before halting the trials. In order to do this, the extent of canvas coverage is monitored by maintaining a cumulative count image, $h_i(\mathbf{X})$, with bin dimensions corresponding to the dimensions of the canvas. After each brush stroke is placed, the count image is incremented over the painted area of the stroke

$$h_i(\mathbf{X}) = h_{i-1}(\mathbf{X}) + s_i(\mathbf{X}) \tag{6.11}$$

where $s_i(\mathbf{X})$ is the mask encapsulating the stroke's painted area. Maintaining $h_i(\mathbf{X})$ is useful because is can be used to detect when a painting is still incomplete by searching for gaps in the cumulative count. A gap mask, $g_i(\mathbf{X})$ is defined as

$$g_i(\mathbf{X}) = \begin{cases} 1 & \text{where } h_i(\mathbf{X}) < t_g \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

where $t_g$ is a threshold on the the count. Typically $t_g = 1$, especially when painting with fully opaque strokes (i.e. $\alpha = 1$). The cumulative count image, and gap mask formed by the process of painting the source image seen in Figure 6.1 can be seen in Figure 6.7 (d) and (e) respectively. Gaps are indicated by the white areas in Figure 6.7 (e). The number of trials that have been completed is denoted by $N_t$, and in this case $N_t = 70$.

Suppose that there are trials to place brush strokes until no gaps exist in canvas, i.e. $\sum_{\mathbf{X}} g_i(\mathbf{X}) = 0$. Figure 6.8 (a) has been painted by the probabilistic process with this simple terminating condition. The number of trials to termination, $N_T$ - and hence number of brush strokes placed - was $N_T = 1369$. No gaps exist in the painted output, and Figure 6.8 (b) reveals that the brush stroke anchor points are fairly well distributed with regard to the stretch and orientation of the painted area of the strokes. The final state of $p_i(\mathbf{X})$, and the cumulative count image, $h_i(\mathbf{X})$, can be seen in Figure 6.8 (c) and (d) respectively. The minimum, maximum and mean values of the cumulative count image after the final trial are $\check{h}_i = 1$, $\hat{h}_i = 10$, and $\bar{h}_i = 4.2913$. The extrema of canvas coverage are widely divergent from the mean, and this means that there may be unnecessary redundancy in the coverage.

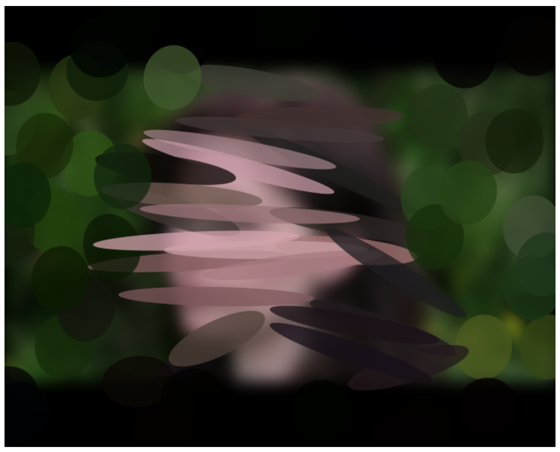
The count image, $h_i(\mathbf{X})$, can be used to influence the probability distribution $p_i(\mathbf{X})$ during the anchor point distribution process. A sensible approach is to suppress the probability of a stroke placement increasingly with count. Hence, the following modification to the suppression function can be performed at each trial

$$p_{b_i}(\mathbf{X}) \propto \frac{p_{b_i}(\mathbf{X})}{k_s h_{i-1}(\mathbf{X})} \tag{6.13}$$

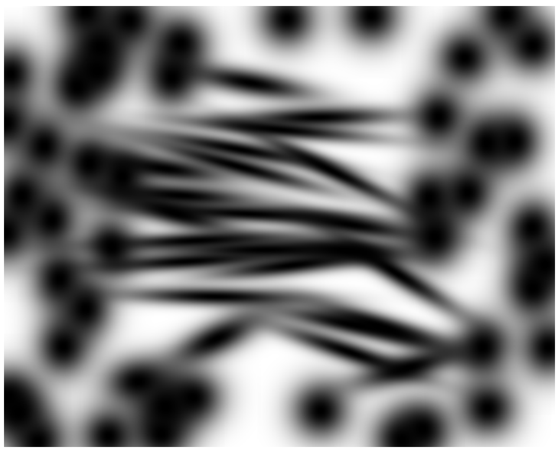
where $k_s \in \Re$ is a weight. Figure 6.9 shows an overhead view of the $p_i(\mathbf{X})$ after a few strokes placed using this modification with $k_s = 1$. It is clear that the probability field is being suppressed in places where the cumulative count is high (seen as a black area in Figure 6.9 (b)), and hence it is less likely that an anchor point will be placed within these regions. However, anchor points are not completely prevented from being placed near to the edges of other strokes, so as to retain a hand-painted look.

What is required from a good anchor point distribution process is economy of brush strokes, minimal redundancy and adequate coverage. To give some idea of how the different

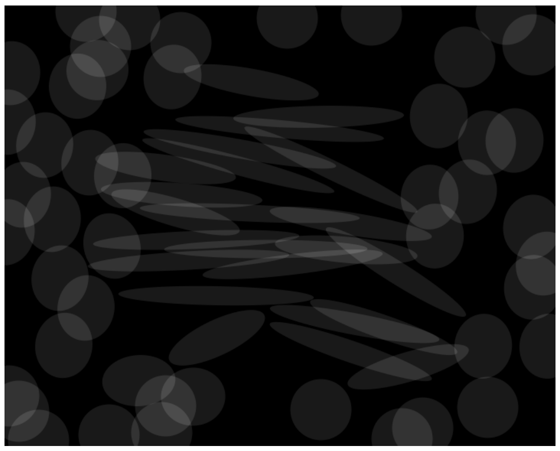
(a) Frames, $N_u = 7$, used to calculate $k_{e_i}$ and $\theta_i$.



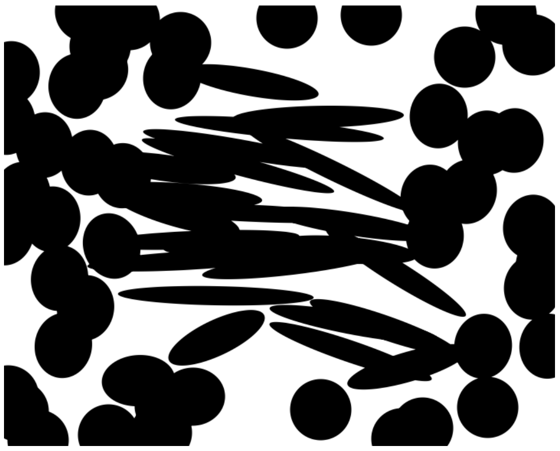
(b) Painting



(c) Probability distribution

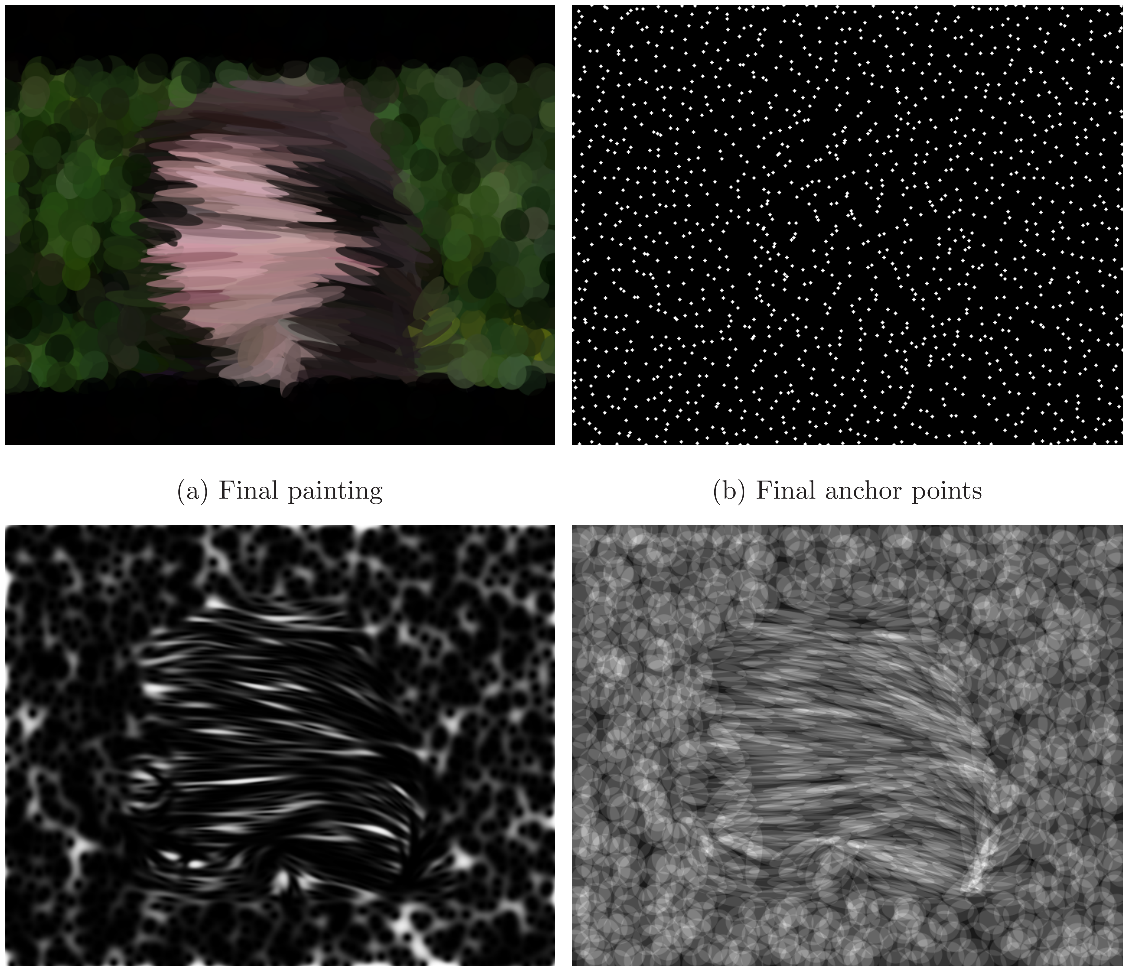


(d) Cumulative count image



(e) Gap mask (gaps=white)

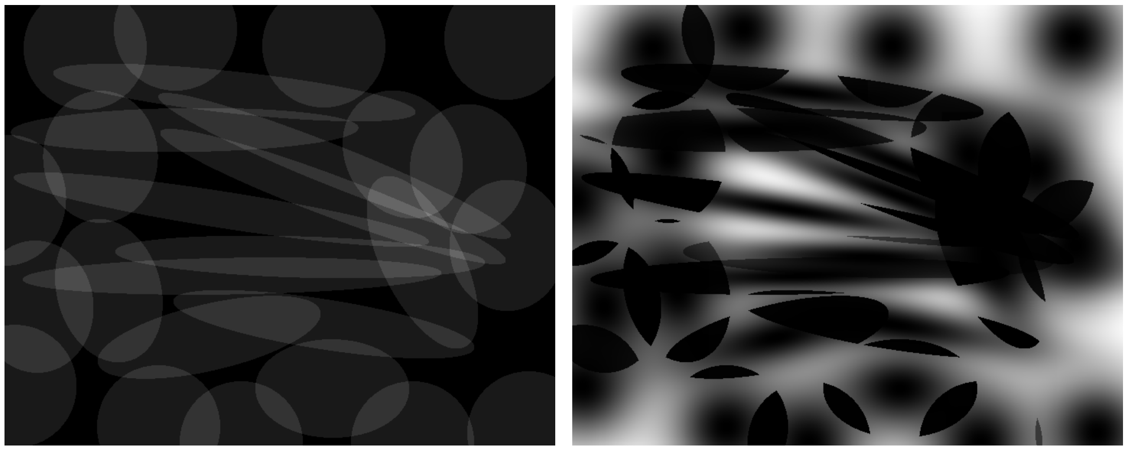
**Figure 6.7:** *The painting process for $N_t = 70$ trials; (a) the motion fields from surrounding frames are used to calculate brush stroke stretch and orientation parameters (see Section 6.2.1), (b) the painted output, (c) probability distribution, $p_i(\mathbf{X})$, (d) cumulative count image, $h_i(\mathbf{X})$, and (e) gap mask, $g_i(\mathbf{X})$ (gaps indicated by white areas). Here $r_{eq} = 40$, $\hat{k}_e = 6$, $\hat{u} = 35$, $v_s = 3$, $C = 0$ and $\alpha = 0.75$. The source images are from the sequence Female1 included on the DVD accompanying this thesis.*


(a) Final painting (b) Final anchor points

(c) Final probability distribution (d) Final cumulative count image

**Figure 6.8:** *Painting while $\sum_X g_i(\mathbf{X}) > 0$; (a) the final painted output, (b) $N_T = 1369$ brush stroke anchor points (dilated for clarity), (c) final probability distribution, $p_i(\mathbf{X})$ (log sigmoid-normalized for clarity), and (d) final cumulative count image, $h_i(\mathbf{X})$. Here $r_{eq} = 20$, $\hat{k}_e = 6$, $\hat{u} = 35$, $v_s = 3$, $C = 0$ and $\alpha = 0.75$. The source image is from the sequence Female1 included on the DVD accompanying this thesis.*

stages of the probabilistic process so far satisfy these requirements, some telling quantities were measured in a trial on the distribution of brush strokes. In this trial, circular brush strokes (i.e. $\hat{k}_e = 1$ and $\hat{u}$ is obsolete) of varying equivalent radii, $r_{eq}$, are placed on a frame of size $576 \times 720$. Three different brush stroke distribution processes are tested: **(1)** a pure pseudo-random anchor point distribution algorithm, **(2)** the proposed probabilistic process, and **(3)** including modification suggested in Equation 6.3. The quantities measured were: the number of brush strokes placed, $N_S$, and the mean, $\bar{h}_{i=N_T}$, maximum $\hat{h}_{i=N_T}$, and minimum, $\check{h}_{i=N_T}$, of the cumulative count image with $i = N_T$ relating to the state of



(a) Cumulative count image (b) Probability Distribution

**Figure 6.9:** *Using (a) $h_{i-1}(\mathbf{X})$, to influence (b) $p_i(\mathbf{X})$, according to Equation 6.3 with $k_s = 1$. Here $r_{eq} = 80$, $\hat{k}_e = 6$, $\hat{u} = 35$, $v_s = 3$, $C = 0$ and source frames are the same as in Figure 6.7 (a).*

$h_{i=N_T}(\mathbf{X})$ after the terminating trial. Economy of brush strokes is obviously related to $N_S$, possible redundancy is revealed by a high value of $\hat{h}$ and/or $\bar{h}$, and $\check{h} >= 1$ is a measure of adequate coverage. Table 6.10 shows the results. The measurements are an average of 100 isolated tests for each process (1), (2) and (3). In (2) and (3) the following parameters are fixed; $k_s = 1$, $v_s = 3$ and $C = 0$.

Table 6.10 clearly reveals the improvements made by the different stages (2) and (3) of the proposed probabilistic process in anchor point distribution for circular brush strokes. Due to the large parameter space of the process, however, the effects of orientation and stretch are more difficult to quantify for all parameter combinations, source image sequences, their associated motion fields, and painting incarnations. It is worth noting that the behavior of these processes is strongly dependent on the behavior of an underlying random number generator[1].

## 6.3.1 Two-Pass Anchor Point Distribution

While the probabilistic process of anchor point distribution discussed in Section 6.3 is effective, the algorithm can be accelerated by taking a two-pass approach. During the first pass, anchor points are placed by running trials until $\bar{h}_i > t_a$, where $t_a$ is a threshold on $\bar{h}_i$. The second pass is a *hole-filling* process. As mentioned previously, it is useful to formulate a map of gaps in the painting, $g_i(\mathbf{X})$. The probability distribution for the second pass can be re-initialized by modifying it with $g_i(\mathbf{X})$ as follows

[1]The Matlab function **rand()** is used in this work

| $r_{eq}$ | Metric | **(1)** | **(2)** | **(3)** | $r_{eq}$ | Metric | **(1)** | **(2)** | **(3)** |
|---|---|---|---|---|---|---|---|---|---|
| 10 | $N_S$ | 16734.2 | 5329 | 3957.5 | 20 | $N_S$ | 5001 | 1356.8 | 1072.5 |
| 10 | $\bar{h}_{i=N_T}$ | 12.2360 | 4.4889 | 3.3175 | 20 | $\bar{h}_{i=N_T}$ | 15.7743 | 4.2602 | 3.3018 |
| 10 | $\hat{h}_{i=N_T}$ | 30 | 10 | 7.6 | 20 | $\hat{h}_{i=N_T}$ | 34.6 | 9.4 | 7.4 |
| 10 | $\check{h}_{i=N_T}$ | 1 | 1 | 1 | 20 | $\check{h}_{i=N_T}$ | 1 | 1 | 1 |
| 40 | $N_S$ | 1034.2 | 351.8 | 272.5 | 80 | $N_S$ | 264.2 | 89 | 74.1 |
| 40 | $\bar{h}_{i=N_T}$ | 12.2451 | 4.0451 | 3.0941 | 80 | $\bar{h}_{i=N_T}$ | 11.6529 | 3.7455 | 3.0054 |
| 40 | $\hat{h}_{i=N_T}$ | 26.7 | 8.8 | 6.6 | 80 | $\hat{h}_{i=N_T}$ | 25 | 7.8 | 6.4 |
| 40 | $\check{h}_{i=N_T}$ | 1 | 1 | 1 | 80 | $\check{h}_{i=N_T}$ | 1 | 1 | 1 |

**Figure 6.10:** *Comparing anchor point distribution processes;* **(1)** *pseudo-random,* **(2)** *probabilistic, and* **(3)** *probabilistic and influenced by* $h_i(\mathbf{X})$ *with* $k_s = 1$ *(see Equation 6.3). The source image size is* $576 \times 720$ *and* $\hat{k}_e = 1$*. Other parameters in* **(2)** *and* **(3)***;* $v_s = 3$ *and* $C = 0$*.*

$$p_i(\mathbf{X}) \propto p_{i-1}(\mathbf{X}) g_i(\mathbf{X}) \tag{6.14}$$

This modification takes place before all trials in the second pass, and has the effect of restricting the sampling space to those canvas locations corresponding to gaps in the painting. In the second pass, sampling trials are carried out on $p_i(\mathbf{X})$ until all gaps are sufficiently filled. Gaps that form larger connected regions are perceived as *holes* in the painting, and the area of a particular connected hole is measured as $A_j$. This provides the halting condition for the second pass. Sampling trials can continue in the second pass while $\hat{A}_j > A_H$, halting when the condition $\hat{A}_j <= A_H$ is met, where $\hat{A}_j$ is the largest hole, and $A_H$ is the maximum tolerated area of a hole.

If desired, the placement of a stroke that would result in $\hat{h}_i > t_m$ can be explicitly rejected, where $t_m$ is a threshold on the maximum of the count image. This stage of the algorithm is inspired by the aforementioned "trial-and-error" style of PDS (see Section 4.3.1). With the addition of this condition, some of the trials to place strokes will not be successful. This implies that more trials are needed before suitable anchor point locations are found, however the number of overlapping strokes can be better controlled. By limiting the total number of strokes, $N_S$, a more memory efficient painting results.

Figure 6.11 shows the results of a two-pass painting with these conditions. The first pass was halted with the condition $t_a = 2$. The halting condition of the second pass was $A_H = 200$, and the resulting gaps can be seen in Figure 6.11 (d). They are not obvious in the painting of Figure 6.11 (a) due to the presence of an underpainting. Due to the condition that $t_m = 4$ in both passes, $N_T = 733$ trials were needed to place $N_S = 652$ strokes with $k_s = 1$ (see Section 6.3). The final cumulative count image has extrema $\bar{h}_{i=N_T} = 2.0323$,

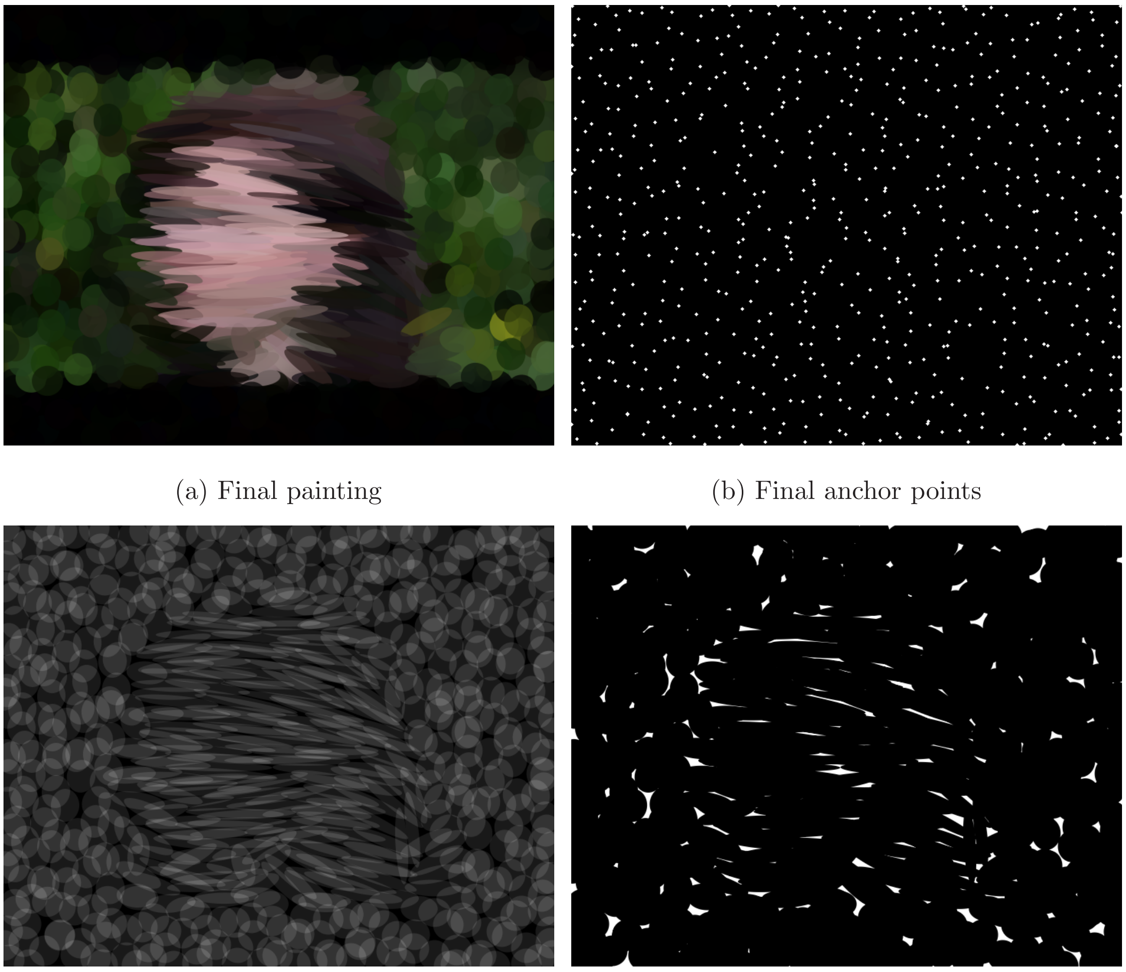


**Figure 6.11:** *Two-pass painting while $\hat{A}_j > A_H$; (a) the final painted output, (b) $N_S = 652$ brush stroke anchor points (dilated for clarity), (c) final cumulative count image, $h_{i=N_T}(\mathbf{X})$ (normalized with regard to $\hat{h}_i$ in Figure 6.8 (d)), and (d) final gap mask, $g_{i=N_T}(\mathbf{X})$ with holes no larger than $A_H = 200$. Here $r_{eq} = 20$, $\hat{k}_e = 6$, $\hat{u} = 35$, $v_s = 2$, $C = 0.1$, $k_s = 1$ and $\alpha = 0.75$. The source image is from the sequence Female1 included on the DVD accompanying this thesis.*

$\hat{h}_{i=N_T} = 4$, and $\check{h}_{i=N_T} = 0$. Interestingly, the value of $\bar{h}_i$ did not change dramatically in the second hole-filling pass (note that $t_a = 2$, and therefore $\check{h}_i \approx 2$ after the first pass). For the purpose of comparison the cumulative count image shown in Figure 6.11 (c) is normalized with regard to the maximum count, $\hat{h}_i$, of Figure 6.8 (d). Compared to Figure 6.8, it is clear that far fewer strokes have been placed, and that there is much less redundancy in Figure 6.11. The painting coverage, however, is still visually acceptable.

## 6.4 Layer-Based Painting

In order to achieve a non-uniform content-based painting, a layer-based approach is proposed. These layers are similar to the layers in the Layer Model for Motion Estimation (see Section 4.4.1), in that they encapsulate the spatio-temporal boundaries of semantic objects. However, since this algorithm is based on the Rubber Sheet Model for Motion Estimation (see Section 4.4.1), provisions will have to be made for the problems of occlusion and uncovering. These issues will be discussed later. Let us first consider the case of layer-based painting on an isolated frame from an image sequence. Three layers are defined for each frame $n$; the background $l_b^n(\mathbf{X})$, the foreground (i.e. a skin mask) $l_f^n(\mathbf{X})$, and details (i.e. edges within the foreground) $l_d^n(\mathbf{X})$. The idea is that each layer can be stylized differently, e.g. the background can be painted coarsely, while the skin and edge layers will be painted more finely or with special highlighting effects. These layers are essential binary label fields that mask the semantic areas within each frame.

### 6.4.1 Spatio-Temporal Skin and Edge Detection

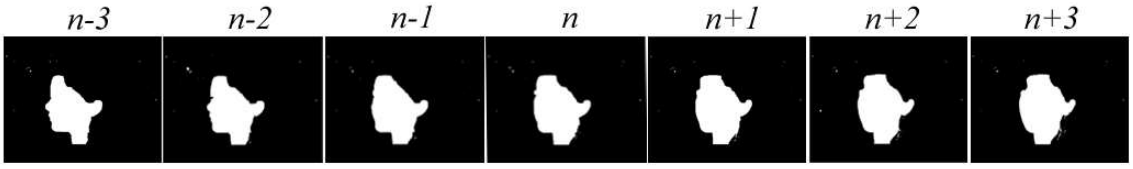
In the case of video in which the main subjects are people, the foreground layer (i.e. skin mask) can be generated using the Graph Cut-based Skin Detection algorithm described in Chapter 5. This results in the binary mask of the form

$$l_f^n(\mathbf{X}) = \begin{cases} 1 & \text{if skin pixel} \\ 0 & \text{otherwise.} \end{cases} \tag{6.15}$$

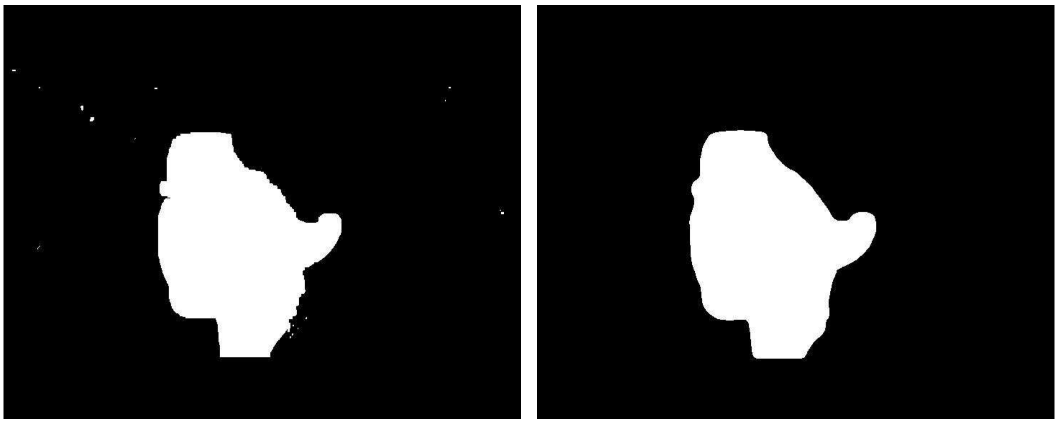
As previously described in Chapter 5, this Graph Cut-based Skin Detection algorithm creates a skin mask for each frame independently. Although spatially smooth, the resulting mask sequence is not temporally coherent.

This problem is addressed through spatio-temporal smoothing of the label fields. Figure 6.12 demonstrates this process. The masks are median filtered over a spatio-temporal window of symmetrical temporal extent $N_{L_f}$, and spatial extent $W_{L_f}$ centered on each pixel in the label field associated with current frame $n$. To further enhance temporal coherency, Motion Compensation is used to warp the pixels of the binary skin masks from the past and future frames back to their locations with regard to that of the current frame according to the accumulated forward and backwards motion vector fields. Figure 6.12 (a) are $N_{L_f} = 7$ skin masks resulting from application of the Graph Cut-based Skin Detection algorithm to the source image sequence seen in Figure 6.7. Figure 6.12 (b) and (c) show the non-smoothed and smoothed versions of frame $n$ respectively.

The background layer is calculated as $l_b^n(\mathbf{X}) = l_f^n(\mathbf{X})^c$. To form the detail layer, $l_d^n(\mathbf{X})$, a Canny Edge Detection is performed on each frame. This results in the binary label field



(a) Frames, $N_L = 7$, used in smoothing $n$



(b) Graph-Cut Skin Detection, frame $n$ (c) Spatio-temporal smoothing, frame $n$

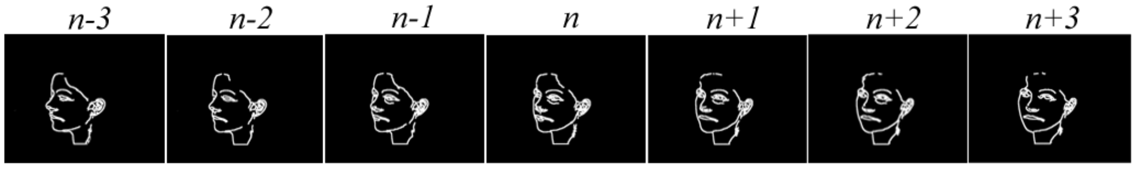
**Figure 6.12:** *f skin-masking; (a) skin masks from surrounding frames are used to smooth $l_f^n(\mathbf{X})$, (b) the non-smoothed version of $l_f^n(\mathbf{X})$, and (c) the spatio-temporal median filtered version of $l_f^n(\mathbf{X})$ with $W_{L_f} = 11$. The parameters used for the initial Graph Cut-based Skin Detection of (b) are $\Theta = 10$ and $\alpha_{gc} = 500$. Both the initial Graph Cut-based Skin Detection and spatio-temporally smoothed skin mask sequences for the source video Female1; Female1_GCskin and Female1_smoothskin - are included on the DVD accompanying this thesis.*

$$l_d^n(\mathbf{X}) = \begin{cases} 1 & \text{if Canny edge pixel with } t_d \\ 0 & \text{otherwise} \end{cases} \tag{6.16}$$

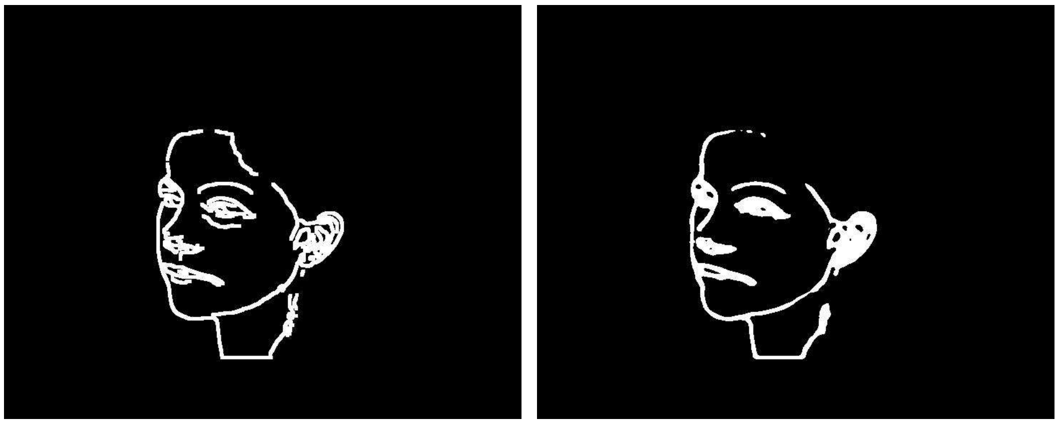
where $t_d$ is a threshold on the Sobel gradient inherent to the Canny Edge Detection process[2]. The Canny detector copes well with noise, and produces thin edge lines that are well aligned to actual edges in the image. From a stylistic point of view, these edge lines are a little too thin for the proposed stylization algorithm. Therefore the edges in $l_d^n(\mathbf{X})$ are dilated morphologically using a disk-shaped structuring element of radius $r_{d_1}$. Next, edges that fall within the background layer are eliminated with the operation $l_d^n(\mathbf{X}) = l_d^n(\mathbf{X}) \cap l_f^n(\mathbf{X})$.

The detail mask sequence is then smoothed using the same median filtering with Motion Compensation process as for the foreground mask sequences. Figure 6.13 demonstrates this

[2]Canny Edge Detection was performed in Matlab with all remaining parameters set to default



(a) Frames, $N_{L_d} = 7$, used in smoothing $n$



(b) Canny Edge Detection, frame $n$ (c) Spatio-temporal smoothing, frame $n$

**Figure 6.13:** *Spatio-temporal edge masking; (a) edge masks from surrounding frames are used to smooth $l_d^n(\mathbf{X})$, (b) the non-smoothed version of $l_d^n(\mathbf{X})$ with $t_d = 0.5$ and dilated with $r_{d_1} = 1$, and (c) the spatio-temporal median filtered version of $l_d^n(\mathbf{X})$ with $W_{L_d} = 7$. The source sequence Female1 is included on the DVD accompanying this thesis.*

process. Figure 6.13 (a) are $N_{L_d} = 7$ skin masks resulting from application of the Canny Edge Detection algorithm to the source image sequence seen in Figure 6.7. Figure 6.13 (b) and (c) show the non-smoothed and smoothed versions of frame $n$ respectively.

### 6.4.2 Separate Stylization of Semantic Layers

The aim of this particular video stylization algorithm is to allow the user to vary the parameters of stylization between semantic layers, where the semantic layers are defined by spatio-temporally coherent skin and edge masks in this case. The spatio-temporally coherent semantic layers defined for a single frame of a typical input video can be seen in Figure 6.14. Note that the foreground (i.e. skin) layer is finalized with the operation $l_f^n(\mathbf{X}) = l_f^n(\mathbf{X}) \cap l_d^n(\mathbf{X})^c$.

Suppose, for example, that the user would like to stylize regions of semantic importance more finely. Recall the properties of elliptical brush strokes and their parameters, as previously discussed in Section 6.2. The user could choose to place strokes with elliptical equivalent radius $r_{eq_b} = 40$ in the background, $r_{eq_f} = 15$ in areas of skin, and $r_{eq_d} = 5$ in detail regions. The probabilistic sampling process described in Section 6.3 can deal with this condition by adjusting the dimensions of the stroke's painted area, $s_i(\mathbf{X})$, and the parameters of the suppression function, $p_{b_i}(\mathbf{X})$, according to the elliptical parameters chosen for the particular semantic layer encapsulating $\mathbf{q}_i$.

The source frame seen in Figure 6.15 (a) is stylized in this way so as to fulfill the conditions $t_a = 2$ and $t_m = 10$ in the first pass, and $A_H = 10$ and $t_m = 10$ in the second pass of anchor point distribution. Here, $N_T = 626$ trials resulted in the anchoring of $N_{S_b} = 153$ strokes in the region marked by $l_b^n(\mathbf{X})$ (i.e. background), $N_{S_f} = 250$ strokes in $l_f^n(\mathbf{X})$ (i.e. foreground/skin), and $N_{S_d} = 223$ (i.e. detail/edge) strokes in $l_d^n(\mathbf{X})$. The final cumulative count image shown in Figure 6.15 (b) has extrema of $\hat{h}_{i=N_T} = 6$, $\check{h}_{i=N_T} = 0$ and $\bar{h}_{i=N_T} = 2.378$.

A problem is clearly observed in the output painting of Figure 6.15 (c) in that too few brush strokes are anchored in regions falling within the foreground, $l_f^n(\mathbf{X})$, and detail, $l_d^n(\mathbf{X})$, semantic regions particularly. Recalling the probablisitic anchor point distribution process described in Section 6.3, this may be due to the large suppression fields, $p_{b_i}(\mathbf{X})$, generated by the large strokes anchored in regions marked by $l_b^n(\mathbf{X})$, and the fact that there are typically fewer pixels marked by $l_f^n(\mathbf{X})$ and $l_d^n(\mathbf{X})$, meaning their sampling likelihood is less. The latter problem would be true even in the case of a purely random sampling process.

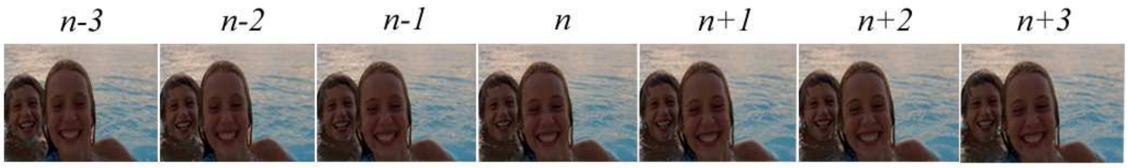
There are two ways to solve this problem. The first is a weighted approach in which the probability distribution $p_{i=0}(\mathbf{X})$ - previously discussed in Section 6.3 - is initialized so as to increase the likelihood of sampling from regions encapsulated by the foreground or detail masks. Specifically,

$$p_{i=0}(\mathbf{X}) = v_b l_b^n(\mathbf{X}) + v_f l_f^n(\mathbf{X}) + v_d l_d^n(\mathbf{X}) \tag{6.17}$$

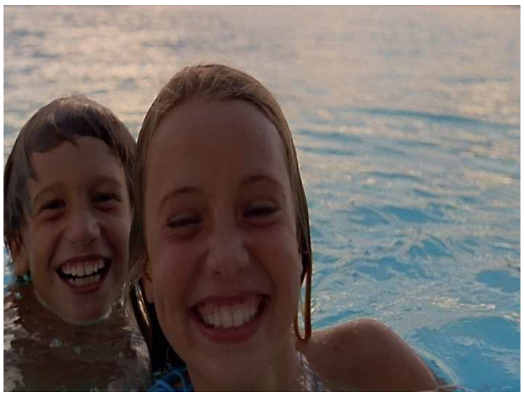
where $v_b$, $v_f$, and $v_d$ are used to weight the initial sampling distribution, $p_{i=0}(\mathbf{X})$, for each frame $n$ with regard to the semantic regions in the source image. These weights should be chosen to reflect the the relative area of their associated semantic mask regions. For example

$$v_d = k_d \frac{\sum\limits_{\mathbf{X}} l_d^n(\mathbf{X})}{\sum\limits_{\mathbf{X}} l_b^n(\mathbf{X}) + \sum\limits_{\mathbf{X}} l_f^n(\mathbf{X}) + \sum\limits_{\mathbf{X}} l_d^n(\mathbf{X})} \tag{6.18}$$

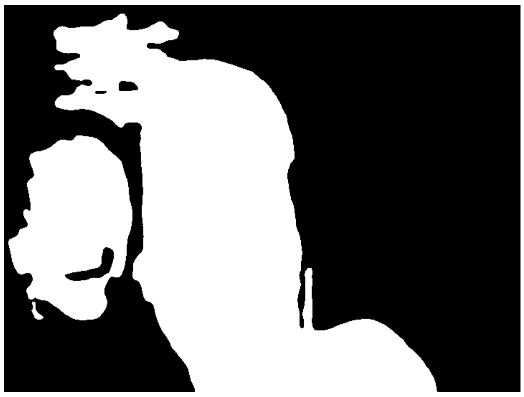
where $k_d \in \Re$ is a weight designed to scale this effect, and similar for $k_f$ and $k_d$ with regard to $v_f$ and $v_d$ respectively. Figure 6.16 demonstrates the use of this technique. Here, $p_{i=0}(\mathbf{X})$ has been weighted with $k_b = 1$, $k_f = 10$ and $k_d = 100$. It is clear to see in Figure 6.16 that more strokes have been anchored $l_f^n(\mathbf{X})$ and $l_d^n(\mathbf{X})$ because the initial sampling likelihood was increased in these regions. Here, $N_T = 1803$ trials were needed to



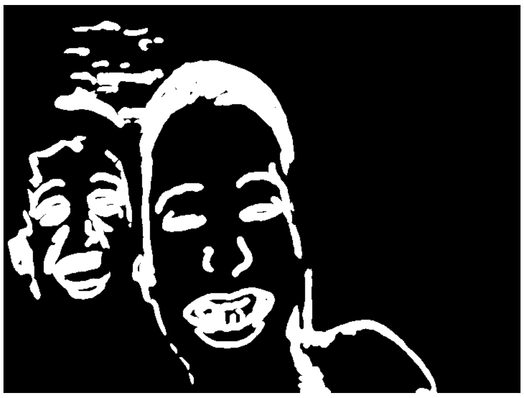
(a) Frames, $N_L = 7$, used in determining semantic layers for $n$



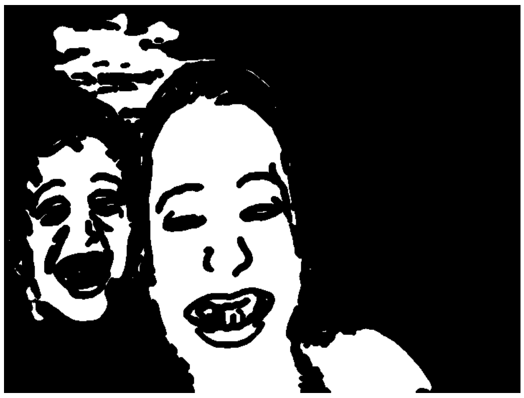
(b) Source frame



(c) Skin mask



(d) Edge mask



(e) Skin mask minus edges

**Figure 6.14:** *Semantic video layers; (a) surrounding frames used to determine masks for $n$, (b) the source frame $n$, (c) spatio-temporally smoothed skin mask, $l_f^n(\mathbf{X})$, (d) spatio-temporally smoothed edge mask, $l_d^n(\mathbf{X})$, and (e) $l_f^n(\mathbf{X}) = l_f^n(\mathbf{X}) \cap l_d^n(\mathbf{X})^c$. Other parameters are $t_d = 0.5$, $r_{d_1} = 3$, and $W_{L_d} = 7$. The source images are from the sequence Swimming included on the DVD accompanying this thesis.*

place $N_{S_b} = 131$ background strokes, $N_{S_f} = 322$ foreground strokes, and $N_{S_d} = 1350$ edge strokes, fulfilling the same conditions imposed on the previous painting seen in Figure 6.15 (a). Now, however, the extrema of the final cumulative count image shown in Figure 6.15 (b) are $\hat{h}_{i=N_T} = 9$, $\check{h}_{i=N_T} = 0$ and $\bar{h}_{i=N_T} = 2.5435$.

The second solution to the problem is to paint each semantic layer separately, with disregard for the painting activity in other layers. Therefore, the three layers of strokes - background, foreground, and detail - are distributed in three rounds of canvas painting, each with a separate set of trials, coverage and terminating conditions and a distinct probability distribution for each canvas region masked by a particular semantic layer mask. This technique is demonstrated in painting Figure 6.17 (a). Here, $N_T = 2010$ trials were carried out to termination. This resulted in the anchoring of $N_{S_b} = 147$ background strokes so as to fulfill the conditions as before, but only in the region masked by $L_b^n(\mathbf{X})$. Similarly, $N_{S_f} = 410$ and $N_{S_d} = 1453$. The extrema of the final cumulative count image shown in Figure 6.17 (b) are $\hat{h}_{i=N_T} = 10$, $\check{h}_{i=N_T} = 0$, and $\bar{h}_{i=N_T} = 2.9482$.

Regardless of the method of layer-based stroke generation, strokes are always stored in three separate lists; background, foreground and detail, organized in the order which they were generated and composited accordingly.

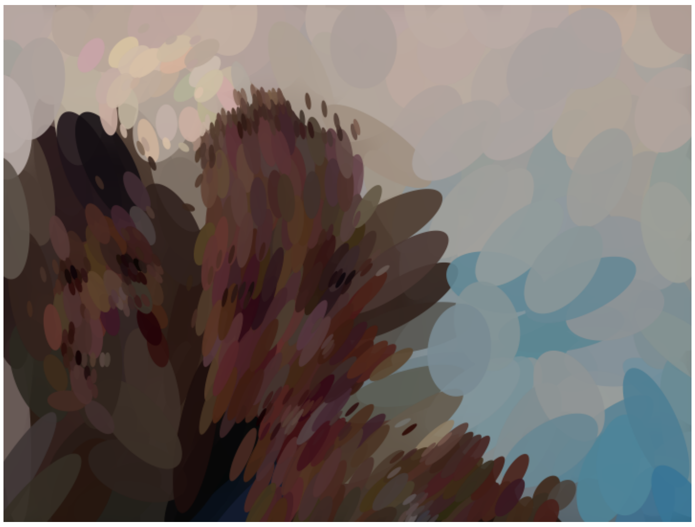
### 6.4.3 Moving the Strokes

The frame-wise motion field obtained by the Motion Estimation is used to move brush strokes from one frame to the next to reflect the underlying motion of the scene. The first video frame (i.e. $n = 0$) is painted exactly as described in Section 6.3. For each consecutive video frame processed the brush stroke anchor points are motion-compensated according to

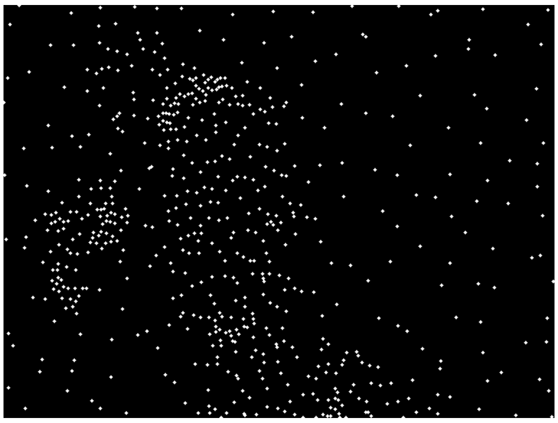
$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n + \mathbf{f}^n(\mathbf{q}_i^n) \tag{6.19}$$

where $\mathbf{f}^n(\mathbf{X})$ is the forward motion vector field estimated between frames $n$ and $n+1$ of the video. This idea can be visualized in Figure 4.12.

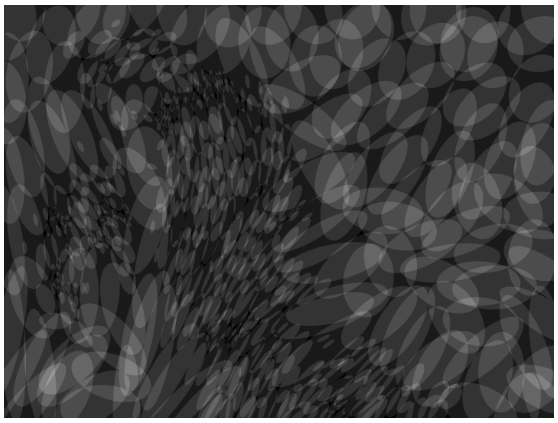
As discussed in Section 4.4.1 of Chapter 4, brush strokes translated using the Rubber Sheet Model will tend to bunch together and overlap excessively in regions of occlusion in the video, and drift apart to reveal gaps in areas of uncovering. The phenomena of occlusion and uncovering are explained in Figure 4.13 (a) and (b) respectively. The problems they create in brush stroke animation can be seen in Figure 6.18, as the camera zooms in on the face of the subject. For visualization, the canvas has not been underpainted and so the uncovering gaps are apparent. The pile-up from occlusion is not easy to visualize because the redundant strokes move on top of one another such that the local maximum, $\hat{h}_i^n$, of the cumulative count, $h_i^n(\mathbf{X})$, in this region is much higher than the overall mean $\bar{h}_i^n$ (see



(a) Output painting



(b) Stroke anchor points

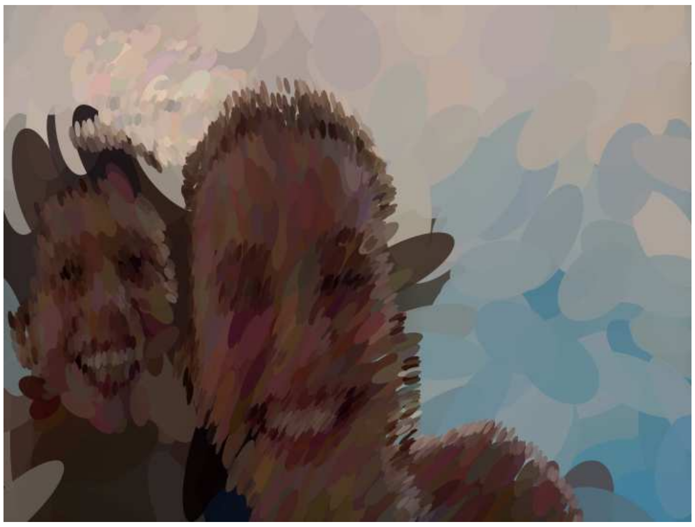


(c) Cumulative count image

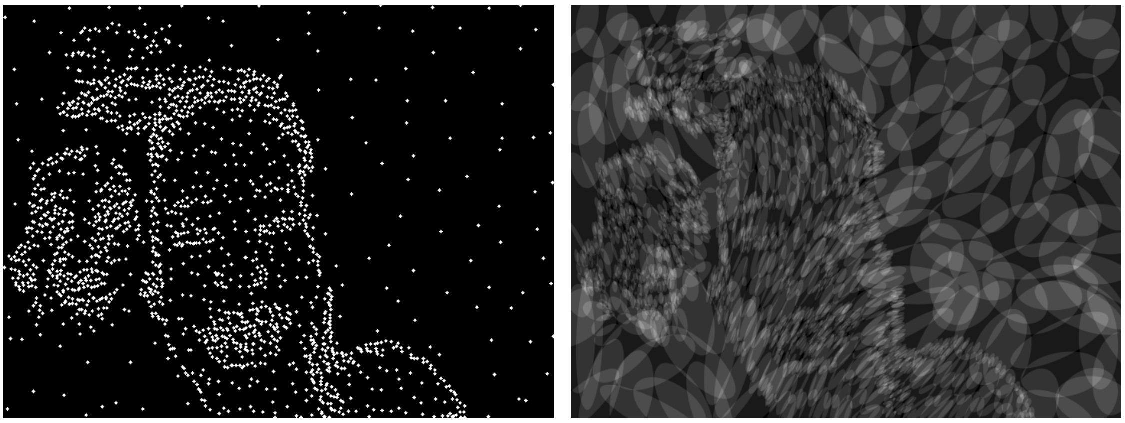
**Figure 6.15:** *Varying brush stroke parameters on semantic layers; (a) the output painting with* $r_{eq_b} = 40$, $r_{eq_f} = 15$, *and* $r_{eq_d} = 5$, *(b) the final cumulative count image* $h_{i=N_T}(\mathbf{X})$, *and (c) the output painting. Very few small brush strokes with* $r_{eq_d} = 5$ *have been placed in the region masked by the detail layer* $l_d^n(\mathbf{X})$, *and similarly with* $l_f^n(\mathbf{X})$. *Here,* $\hat{u} = 35$, $\hat{k}_e = 4$, $C = 0.1$, $v_s = 2$, $k_s = 1$, $\alpha = 0.9$, $t_a = 2$, $t_m = 10$, *and* $A_H = 10$. *The source image is from the sequence Swimming included on the DVD accompanying this thesis.*

Section 6.3 for an explanation of these metrics).

A novel solution to this problem of occlusion pile-up is to detect the frame-wise regions of occlusion, and prevent brush strokes from being translated into these regions. A measure of occlusion is obtained by generating a motion difference map as follows



(a) Output painting

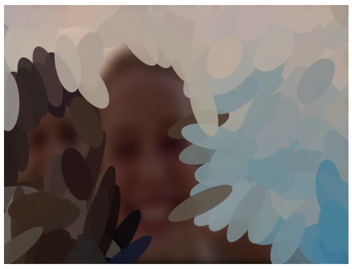


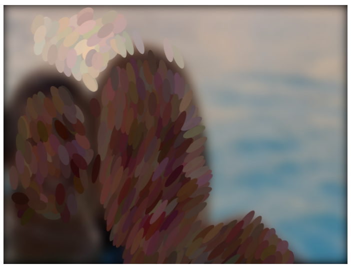
(b) Stroke anchor points (c) Cumulative count image

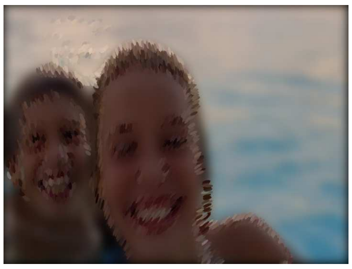
**Figure 6.16:** *Weighting $p_{i=0}(\mathbf{X})$ in styling semantic layers; (a) the output painting with $r_{eq_b} = 40$, $r_{eq_f} = 15$, and $r_{eq_d} = 5$, (b) the final cumulative count image $h_{i=N_T}(\mathbf{X})$, and (c) the output painting. Now more small brush strokes with $r_{eq_d} = 5$ have been placed in the region masked by the detail layer $l_d^n(\mathbf{X})$, and similarly with $r_{eq_d} = 10$ in $l_f^n(\mathbf{X})$. Here, $k_b = 1$, $k_f = 10$, $k_d = 100$, $\hat{u} = 35$, $\hat{k}_e = 4$, $C = 0.1$, $v_s = 2$, $k_s = 1$, $\alpha = 0.9$, $t_a = 2$, $t_m = 10$, and $A_H = 10$. The source image is from the sequence Swimming included on the DVD accompanying this thesis.*

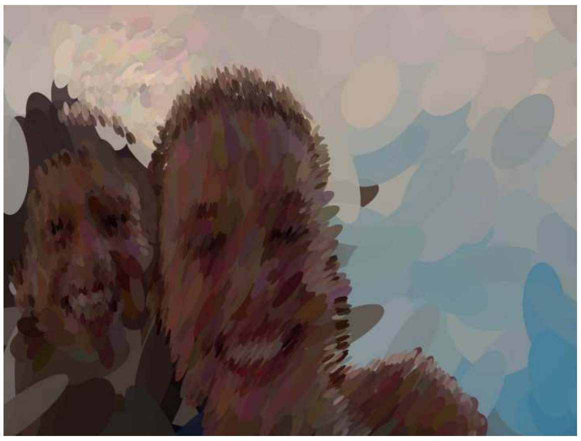
$$o^n(\mathbf{X}) = |\mathbf{X}^{n'} - \mathbf{X}^{n''}| \tag{6.20}$$

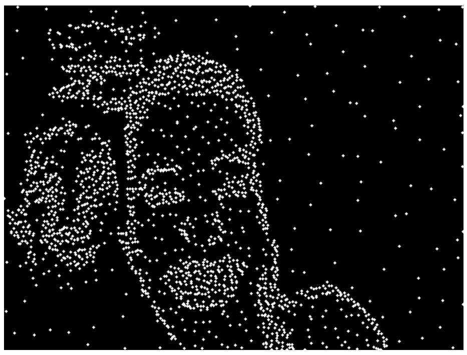
$$\mathbf{X}^{n'} = \mathbf{X}^{n-1} + \mathbf{f}^{n-1}(\mathbf{X}) \ , \ \mathbf{X}^{n''} = \mathbf{X}^{n'} + \mathbf{b}^n(\mathbf{X}) \tag{6.21}$$


(a) Background


(b) Foreground


(c) Detail


(d) Output painting


(e) Stroke anchor points

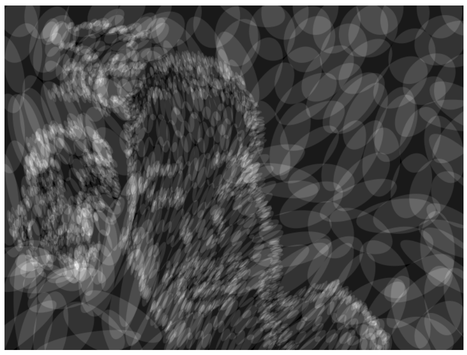

(f) Cumulative count image

**Figure 6.17:** *Distributing strokes in layers; (a) $N_{S_b} = 147$ background strokes in $l_b^n(\mathbf{X})$, (b) $N_{S_f} = 410$ foreground strokes in $l_f^n(\mathbf{X})$, (c) $N_{S_d} = 1453$ foreground strokes in $l_d^n(\mathbf{X})$, (d) the output painting, (e) final cumulative count image $h_{i=N_T}(\mathbf{X})$, and (f) the output painting. Even more small brush strokes with $r_{eq_d} = 5$ have been placed in the region masked by the detail layer $l_d^n(\mathbf{X})$, and similarly with $l_f^n(\mathbf{X})$. Here $\hat{u} = 35$, $\hat{k}_e = 4$, $C = 0.1$, $v_s = 2$, $k_s = 1$, $\alpha = 0.9$, $t_a = 2$, $t_m = 10$, and $A_H = 10$. The source image is from the sequence Swimming included on the DVD accompanying this thesis.*

where $\mathbf{X}^{n'}$ is the translation of the pixel coordinates in frame $n-1$ to $n$ according to the motion estimator and $\mathbf{f}^{n-1}(\mathbf{X})$ the forward motion field estimated for frame $n-1$. $\mathbf{X}^{n''}$ is the translation of the estimated $\mathbf{X}^{n'}$ back to the locations in frame $n-1$ according to $\mathbf{b}^{n}(\mathbf{X})$, the backward motion field estimated for frame $n$. In regions with no occlusion it would be expected that $\mathbf{X}^{n'} = \mathbf{X}^{n''}$, and hence $o^{n}(\mathbf{X})$ would be low. In regions of occlusion, however, $o^{n}(\mathbf{X})$ is higher.

Since the motion estimator will usually fail in areas of occlusion and uncovering, it follows that $o^{n}(\mathbf{X})$ will be non-zero in these regions, and close to zero otherwise. The image is normalized according to a user-defined maximum estimated occlusion, $\hat{o}$. Then, significant regions of occlusion and uncovering can be detected as $o^{n}(\mathbf{X}) > t_o$, where $t_o$ is a threshold. This can be visualized in Figure 6.19. The subject, seen in Figure 6.19 (a) and (b), occludes her eyes by blinking and this is clearly highlighted by the occlusion image, $o^{n}(\mathbf{X})$, seen in Figure 6.19 (c) and thresholded detector seen in Figure 6.19 (d). In animating the brush strokes, anchor points that are motion-compensated into these regions when painting frame $n$ are simply deleted.

Anchor points that are translated beyond the spatial boundaries of the video are also discarded, as are points that are translated from the region of one semantic layer to another. Furthermore, strokes whose motion-compensated translation would excite the condition $\hat{h}_i > t_m$ are deleted. Brush strokes at the rear of the painting (i.e. that were initialized early in the anchor point distribution process) are always deleted before others that were composited on top of. This action prevents the flicker that would occur if brush strokes could be seen popping in and out of the painting suddenly.

The second problem with the Rubber Sheet Model - as previously discussed in Section 4.4.1 - is that brush strokes tend to drift apart in areas of uncovering to reveal increasingly large gaps in the painting which need to be filled. A gap mask, as discussed in Section 6.3, is used to limit sampling activity to gap regions at this point. This amounts to

$$p_i^n(\mathbf{X}) \propto p_{i-1}^n(\mathbf{X}) g_i(\mathbf{X}) \tag{6.22}$$

The entire two-pass distribution process described in Section 6.3 is then carried out on this reduced sampling space until painting completion. This stage will also fill gaps that have been created by the deletion of brush strokes translated into regions marked by the mask $o^{n}(\mathbf{X}) > t_o$. All newly generated brush strokes are added to the head of the list defining compositing order, such that they are painted early in the composition.

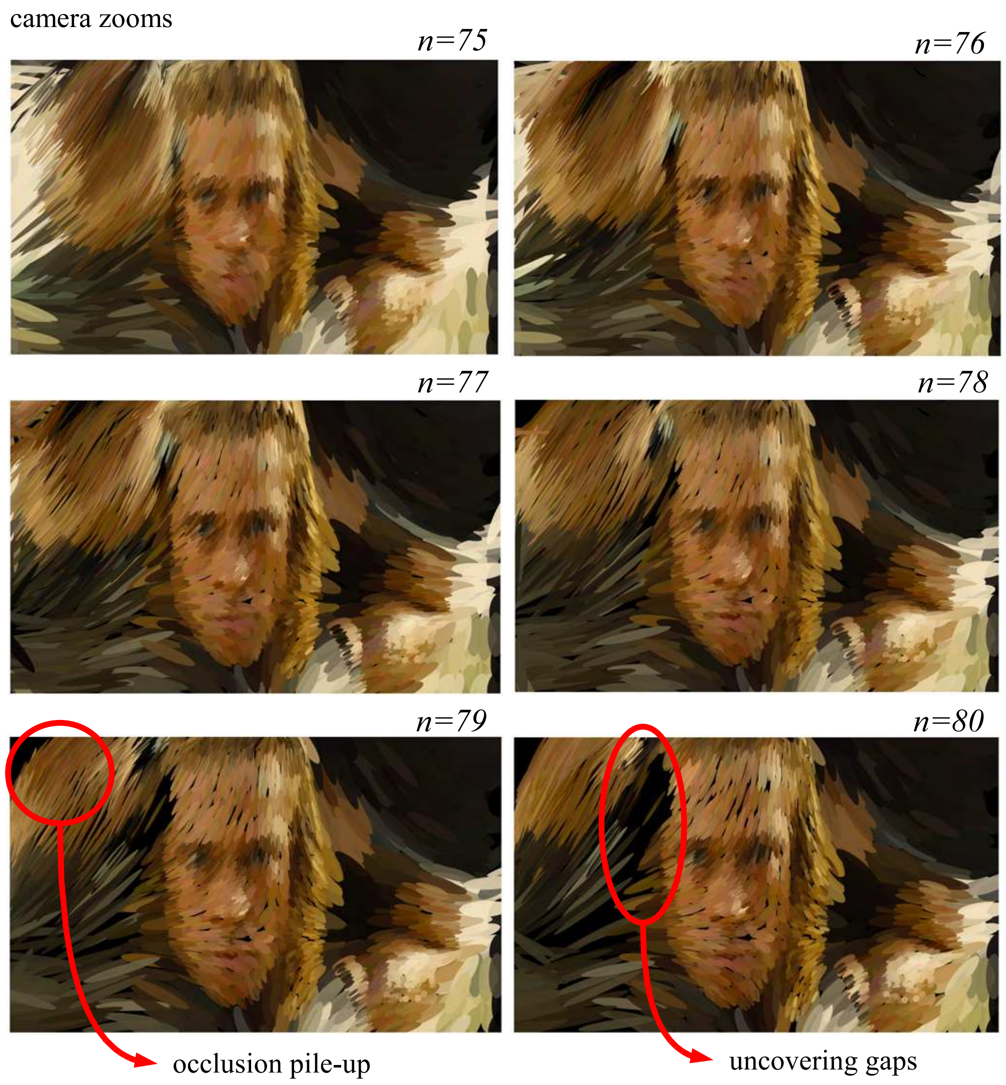


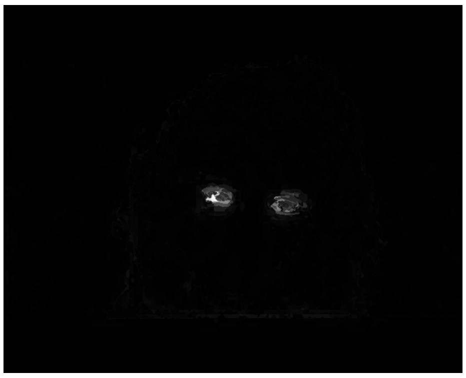
**Figure 6.18:** *Occlusion and uncovering; brush strokes pile up in regions of occlusion and drift apart to reveal gaps in regions of uncovering. Here, a camera zoom demonstrates these problems. Important parameters include* $r_{eq_b} = 20$, $r_{eq_f} = 10$, $r_{eq_d} = 5$, $\hat{k}_e = 4$, $\hat{u} = 32$, $C = 0$, $v_s = 3$ *and* $\alpha = 0.9$. *The source images are from the sequence Hollywood included on the DVD accompanying this thesis.*

## 6.5 Spatio-Temporal Color-Sampling

As discussed in Section 6.5, coloring a brush stroke by point-sampling the color of the image under its anchor point $\mathbf{q}_i^n$ as it moves from frame to frame will result in flicker in the painted sequence. It is also desirable to paint each brush stroke with a color that is representative of the region defined by the stroke's painted area, $s_i^n(\mathbf{X})$, and not just that of its anchor point.

(a) Frame $n-1$

(b) Frame $n$


(c) Occlusion image

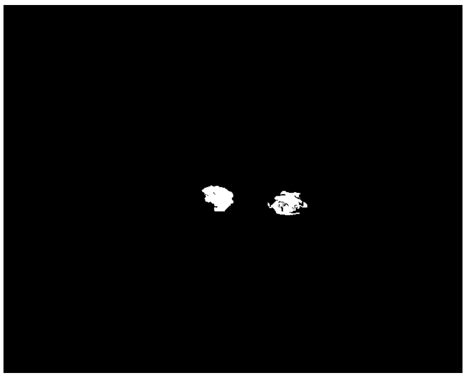

(d) Significant occlusion

**Figure 6.19:** *Occlusion detection; (a) frame $n-1$, and (b) the subject closes her eyes in frame $n$, (c) the occlusion image, $o^n(\mathbf{X})$, captures the occlusion, and (d) regions of significant occlusion are detected with $\hat{o} = 32$ and $t_o = 0.04$. Brush stroke anchor points will not be translated into these regions when painting frame $n$. The source images are from the sequence Female1 included on the DVD accompanying this thesis.*

In order to address these problems, two filters are used in cascade; one spatial and one temporal. A spatial window is used to simultaneously sample and smooth spatial color for each brush stroke, $\mathbf{c}_i^n$, and the temporal window removes variations of this sampled color across the frames. The spatial process can be seen in Figure 6.20, and it yields a color sample

$$\mathbf{c}_i^n = \sum_j \left[ v_j \mathbf{C}_i^n(\mathbf{X} + d_j) \right] \tag{6.23}$$

where $\mathbf{c}_i^n$ is the sampled (rgb) color vector, $\mathbf{C}(\mathbf{X})_i^n$ are the pixel-wise color values for the frame, $d_j$ indexes the samples in the spatial window, and $v_j$ are weights corresponding to a spatial Hamming window, both centered on $\mathbf{q}_i^n$. The Hamming window is in place to

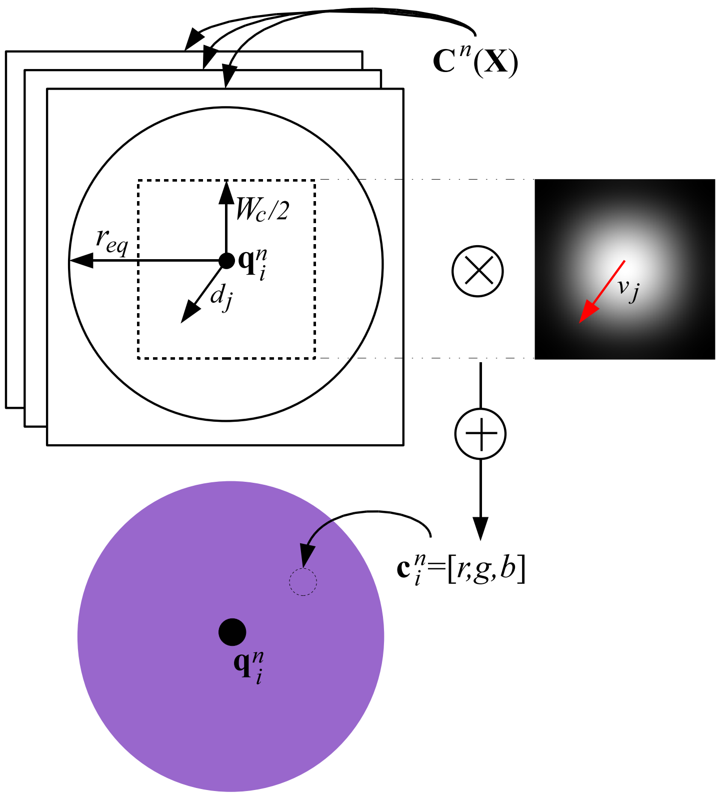


**Figure 6.20:** *The process of spatial color sampling to paint a circular brush stroke.*

ensure that color closer to the center of a brush stroke is weighted with more importance in the spatial averaging process. The width of this spatial window, $W_c$, can be adjusted according to $W_c = 2k_c r_{eq}$, where $k_c \in \{0...1\}$ is a user-defined constant. The smaller the value of $k_c$, the more distinct the color of individual strokes will appear. Noise in the range $[-\delta c, +\delta c]$, can be added $\mathbf{c}_i^n$ to allow the individual strokes to appear even more distinct.

Temporal smoothing is achieved with an Infinite Impulse Response (IIR) Butterworth filter of order $N_b$, and normalized cutoff frequency, $\omega_b$. A third order (i.e. $N_b = 3$) IIR Butterworth filter is a good choice for the task of removing flicker because it has a frequency response that is maximally flat in the pass-band and a reasonably steep rate of attenuation. Furthermore, the flicker-removing performance of a third order Butterworth filter in this task was found to be comparable to that of a temporal mean filter with over twice the number of taps. The taps of a third order Butterworth filter with varying $\omega_b$ can be seen in the Table 6.21 below. Note that $\mathbf{c}_i^n$ refers to the spatially smoothed color sample for frame $n$, whereas $\mathbf{c}_{b_i}^{n-1}$ refers to the temporally smoothed color output of the Butterworth filter for frame $n-1$ and so on.

| $\omega_b$ | $\mathbf{c}_{b_i}^{n-3}$ | $\mathbf{c}_{b_i}^{n-2}$ | $\mathbf{c}_{b_i}^{n-1}$ | $\mathbf{c}_{i}^{n}$ | $\mathbf{c}_{i}^{n-1}$ | $\mathbf{c}_{i}^{n-2}$ | $\mathbf{c}_{i}^{n-3}$ |
|---|---|---|---|---|---|---|---|
| 0.1 | -0.5321 | 1.9294 | -2.3741 | 0.0029 | 0.0087 | 0.0087 | 0.0029 |
| 0.3 | -0.1378 | 0.6959 | -1.1619 | 0.0495 | 0.1486 | 0.1486 | 0.0495 |
| 0.5 | 0 | 0.3333 | 0 | 0.1667 | 0.5 | 0.5 | 0.1667 |

**Figure 6.21:** *The taps of a third order (i.e. $N_b = 3$) IIR Butterworth filter with varying cutoff frequency $\omega_b$.*

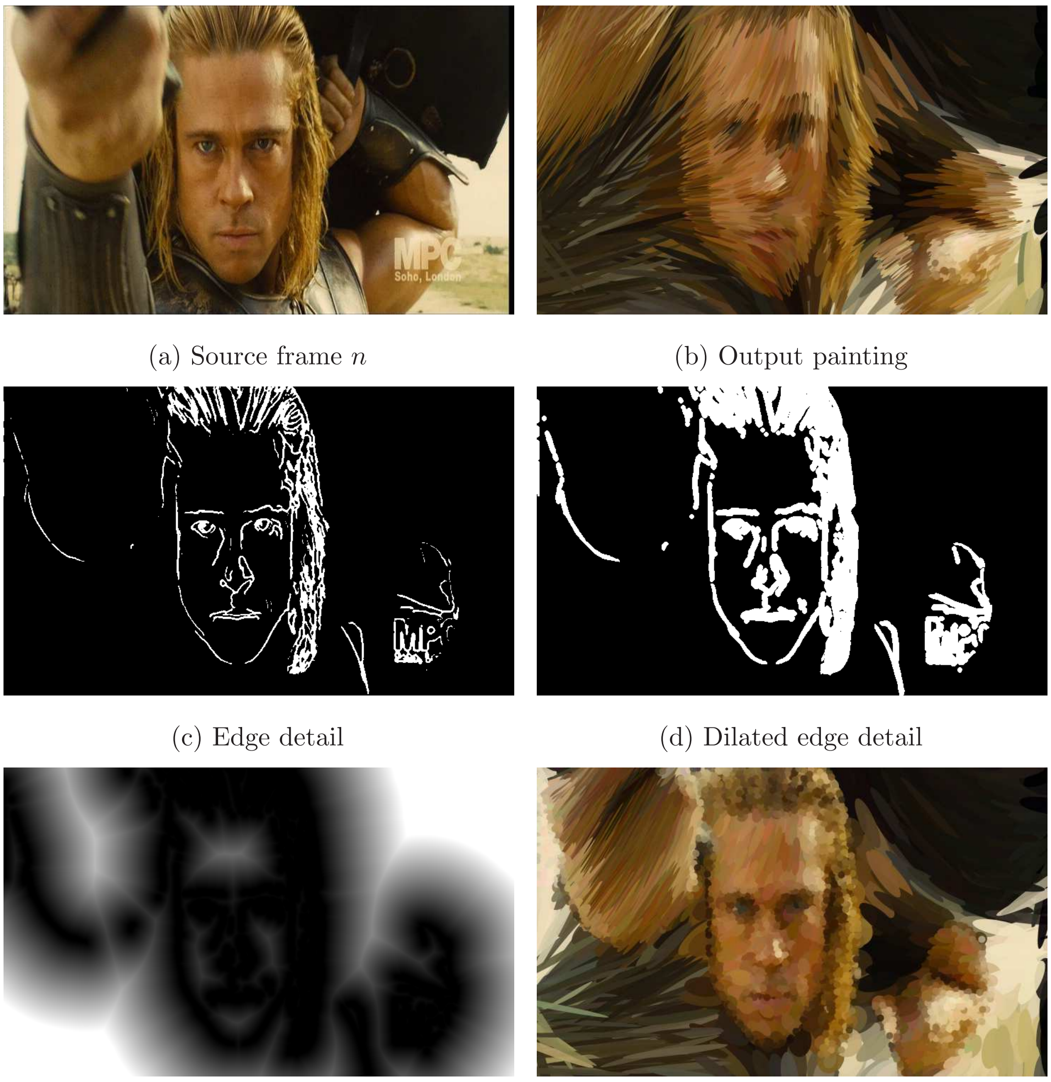
## 6.6 Preserving Some High Frequency Information

Section 6.2.1 of this Chapter presents a novel technique for motion expression in SBR, where the underlying motion of an image sequence is used to smoothly stretch and rotate elliptical brush strokes in the direction of the local motion determined by Motion Estimation. An aesthetic problem results from this technique, however, in that much of the high frequency information is lost when the ellipses stretch across boundaries and edge regions in the painted image sequences. This problem is demonstrated in 6.22 (b).

A (subjectively) better effect be achieved by trying to limit the stretch of the elliptical brush strokes in the region of edges as defined by the spatio-temporal detail label field, $l_d^n(\mathbf{X})$. Figure 6.22 demonstrates this idea. Figure 6.22 (a) shows the original source image. The motion of surrounding frames is clearly expressed by the stretch of the elliptical brush strokes in the initial output painting of Figure 6.22 (b), but much of the high frequency detail is lost where these ellipses are orientated and stretched across edges and boundaries. The original spatio-temporally smoothed detail mask, $l_d^n(\mathbf{X})$, can be seen in Figure 6.22 (c). Recall that the underlying frame-wise Canny edge mask was dilated morphologically with $r_{d_1} = 1$ prior to spatio-temporal smoothing (see Section 6.4 for explanation). This step had the effect of increasing the non-zero regions of the detail mask, $l_d^n(\mathbf{X})$, over which the smallest brush strokes were distributed. Figure 6.22 (d) shows a further dilation of Figure 6.22 (c) with $r_{d_2} = 3$, and from this arises a special detail (i.e. edge) distance filter as can be seen in Figure 6.22 (e). This filter is designed to curb the stretch of the brush strokes at sites near to non-zero values of $l_d^n(\mathbf{X})$ according to

$$k_e^n(\mathbf{X}) = k_e^{n'}(\mathbf{X})d^n(\mathbf{X}) \tag{6.24}$$

where $k_e^{n'}(\mathbf{X})$ describes the stretch profile of the ellipses at potential anchor points across the canvas coordinates $\mathbf{X}$, using the method of calculating the elliptical stretch described in Section 6.2.1. The detail distance filter, $d^n(\mathbf{X})$, is a pixel-wise measurement of the Euclidean distance from each site in $\mathbf{X}$ to the nearest non-zero value marked by $l_d^n(\mathbf{X})$. These values are normalized, scaled by a constant $E_d$, and clipped such that $d^n(\mathbf{X}) > 1 = 1$. The desired effect here is to limit the stretch of the elliptical strokes increasingly close to edges,



(a) Source frame $n$ (b) Output painting

(c) Edge detail (d) Dilated edge detail

(e) Detail distance filter (f) Elliptical stretch control

**Figure 6.22:** *Curbing the elliptical stretch at detail sites; (a) the source frame $n$, (b) output painting with $\hat{k}_e = 9$, and $\hat{u} = 32$, (c) spatio-temporal detail mask $l_d^n(\mathbf{X})$, (d) $L_d^n(\mathbf{X})$ dilated with $r_{d_2} = 4$, (e) detail distance filter $d^n(\mathbf{X})$ with $E_d = 2$, and (f) the output painting also with $\hat{k}_e = 9$ and $\hat{u} = 32$, but now $d^n(\mathbf{X})$ influencing $k_e^n(\mathbf{X})$ according to Equation 6.6. Other important parameters include $r_{eq_b} = 20$, $r_{eq_f} = 10$, $r_{eq_d} = 5$, $C = 0$, $v_s = 3$, $k_s = 1$, $\alpha = 0.9$, $t_a = 2.5$, $A_H = 0$, $t_m = 5$, $t_o = 0.01$, $k_c = 0.3$, $N_b = 3$ and $\omega_b = 0.5$. The source image is from the sequence Hollywood included on the DVD accompanying this thesis.*

reducing them to circles at edge sites. The value of $E_d \geq 1$ determines the stretch-curbing influence of $d^n(\mathbf{X})$ across $\mathbf{X}$. The value of $k_e^n(\mathbf{X})$ at sites where $l_d^n(\mathbf{X}) = 1$ will always be unity, and this is the desired effect. Figure 6.22 (e) shows $d^n(\mathbf{X})$ with $E_d = 2$, and Figure

6.22 (f) shows the painted result after utilizing this filter to influence the parameters of elliptical stretch calculation (see Section 6.2.1) and anchor point distribution (see Section 6.3) according to Equation 6.6. It is clear to see that the elliptical stretch of brush strokes, $k_{e_i}^n$ is increasingly limited near to and at the edges shown in Figure 6.22 (d), and this helps to preserve some high frequency detail in the output painting.

## 6.7 Finishing Touches

There is huge potential for experimentation with the video stylization framework. A cartoon-ish look, for example, can be achieved by alpha-compositing color onto the painted canvas using the spatio-temporal edge layer discussed in Section 6.4 as the alpha mask. This is achieved by the following operation

$$z^n(\mathbf{X}) = E_c z^n(\mathbf{X}) \alpha_d (1 - l_d^{n'}(\mathbf{X})) \tag{6.25}$$

where $z^n(\mathbf{X})$ is the post-SBR painted canvas, $\alpha_d$ is the opacity of the edge decoration, and $E_c$ is a constant defining its color. The mask $l_d^{n'}(\mathbf{X})$ refers to the smoothed version of the detail layer, $l_d^n(\mathbf{X})$ described in Section 6.4. The smoothing is performed with a 2D normalized Hamming window of spatial extent $W_d$. The edge may also be dilated just prior to the final smoothing with a morphological disk of radius $r_{d_3}$. Two examples of edge decoration can be seen in Figure 6.23.

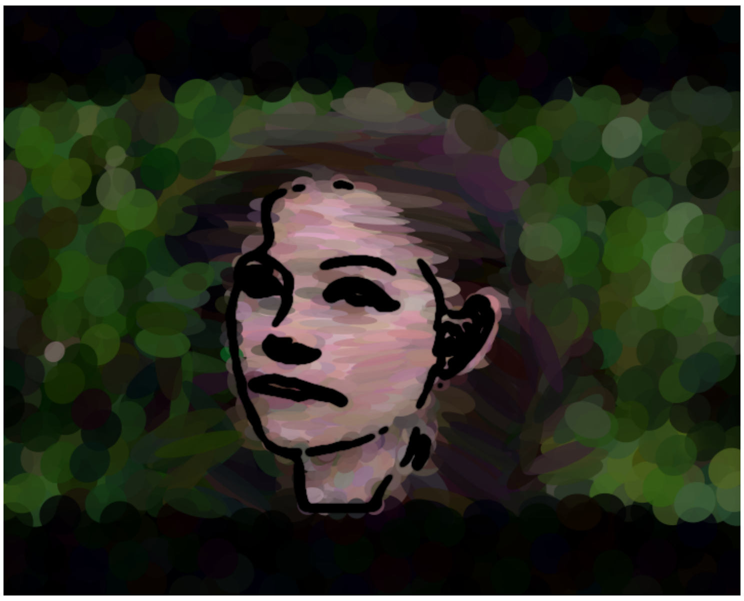
Another interesting effect can be created by sampling the source image beneath the color-sampling window of width $W_c$ defined by $k_c$ (see Figure 6.20), and resizing it to the size of the painted area of its associated brush stroke (i.e. the area defined by $s_i^n(\mathbf{X})$) using bicubic interpolation. The result is a lens-like effect, and the lens-like brush strokes are still animated with the underlying motion of the sequence. Figure 6.24 presents example frames incorporating this effect.

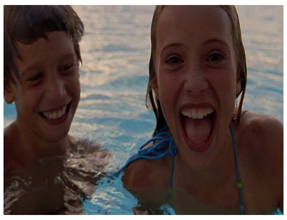
## 6.8 Summary of the Framework

Before describing some video results, a brief summary of the framework for the Skin-Aware Stylization of Video Portraits is needed. Once the user has defined certain initial parameters, the overall stylization process is as follows:

1. First, the source video is analyzed to obtain some *reference data*. Graph Cut-based Skin Detection (see Chapter 5), and Canny Edge Detection are performed on each frame individually, with no temporal smoothing. Motion Estimation (see (Kokaram, 1998)) is carried out to estimate the frame-wise motion field.

(a) Source frame 1


(b) Stylized output


(c) Source frame 2

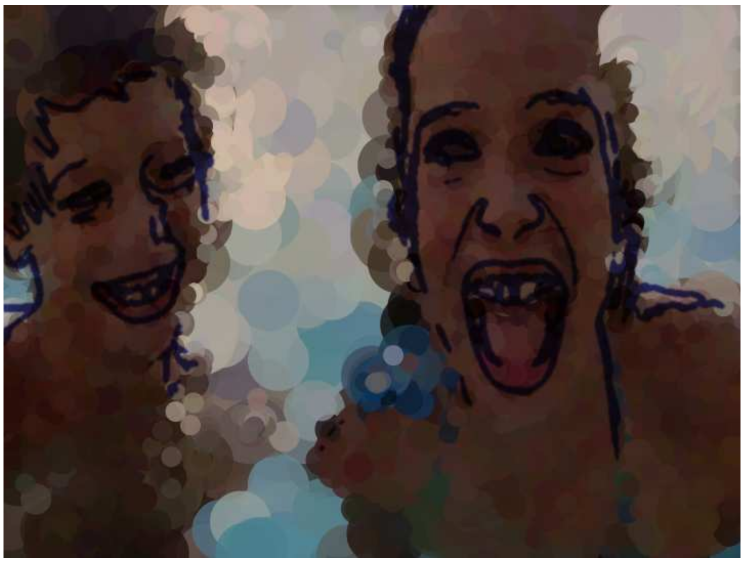
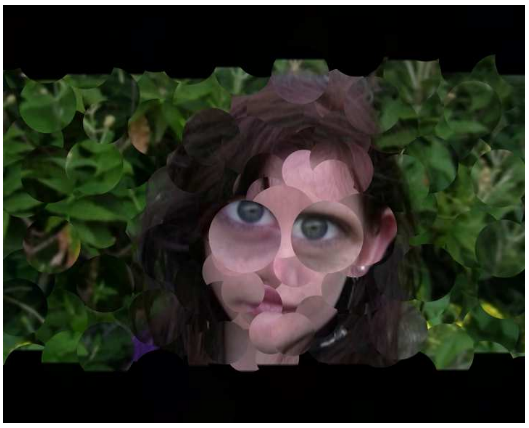
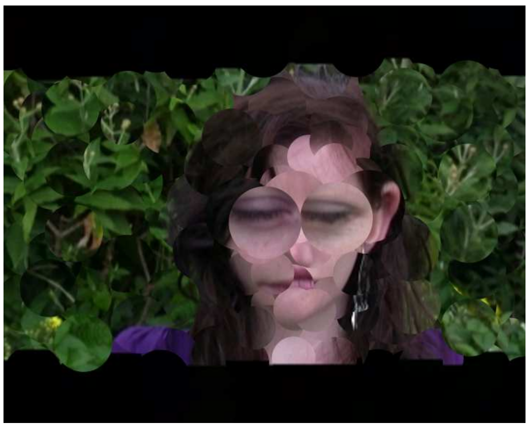

(d) Stylized output 2

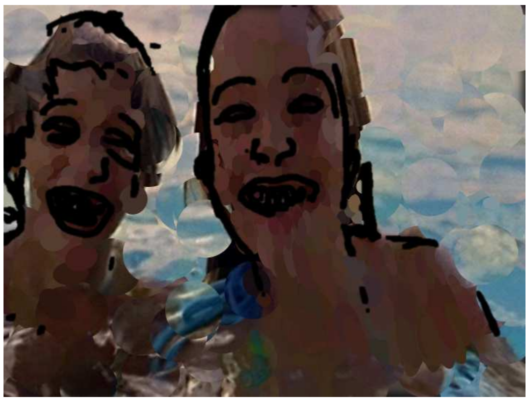
**Figure 6.23:** *Edge decoration; (a) source frame from the sequence Female1, (b) example frame from stylized output sequence Female1_1 with important parameters* $r_{eq_b} = 20$, $r_{eq_f} = 10$, $r_{eq_d} = 0$, $C = 0$, $v_s = 2$, $k_s = 1$, $t_a = 2$, $t_m = 4$, $A_H = 5$, $t_o = 0.04$, $\alpha = 0.75$, $\hat{k}_e = 9$, $\hat{u} = 35$, $E_d = 10$, $\alpha_d = 1$, $r_{d_1} = 3$, $W_d = 5$, $E_d = 10$, $N_b = 3$, $\omega_b = 0.1$, $k_c = 0.3$, $\delta c = 0.01$, $\delta\theta = 0.032$, $N_{L_f} = 5$, $W_{L_f} = 11$, $N_{L_d} = 3$, $W_{L_d} = 7$ *and* $t_d = 0.2$, *(c) source frame from the sequence Swimming, and (c) example frame from stylized output sequence Swimming_1 with with important parameters* $r_{eq_b} = 5$, $r_{eq_f} = 7$, $r_{eq_d} = 15$, $C = 0.1$, $v_s = 2$, $k_s = 1$, $t_a = 1.5$, $t_m = 1.5$, $A_H = 20$, $t_o = 0.01$, $\alpha = 0.5$, $\hat{k}_e = 1$, $\alpha_d = 0.75$, $r_{d_1} = 2$, $r_{d_2} = 4$, $W_d = 5$, $N_b = 3$, $\omega_b = 0.5$, $k_c = 0.5$, $E_c = [0, 0, 0.5]$ *(navy)*, $\delta c = 0.02$, $\delta\theta = 0.064$, $N_{L_f} = 5$, $W_{L_f} = 5$, $N_{L_d} = 3$, $W_{L_d} = 2$ *and* $t_d = 0.1$. *Both source sequences and the stylized outputs are included on the accompanying DVD.*



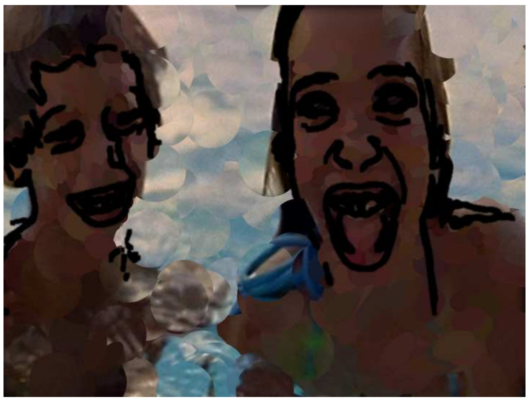
(a) Lens-like effect



(b) Blink success



(c) Lens effect on water



(d) Lens effect on water

**Figure 6.24:** *A lens-like effect; (a) $l_d^n(\mathbf{X})$ has been created by tracking the pupils in the eyes using the motion tracker in Adobe AfterEffects CS3, and (b) the track is not lost when the subject blinks (see Figure 6.19), (c) the lens-like effect applied to $l_b^n(\mathbf{X})$ only, and (d) another frame. A spatial sampling window with $k_c = 0.5$ produces the best results. Examples (a) and (b) are frames from the stylized output sequences sequence Female1_3, and (b) and (b) are from Swimming_2. All parameters are listed in Appendix C.*

2. The frame-wise skin and edge masks are spatio-temporally smoothed by a process of motion-compensated median filtering (see Section 6.4). The edge mask may or may not be dilated slightly prior to this process, depending on the user's preference.

3. The spatio-temporally smoothed skin and edge masks are used to calculate the semantic *background*, *foreground* and *detail* layers (see Section 6.4).

4. A *detail distance filter* is created from the spatio-temporally smoothed detail (i.e edge) layer (see Section 6.6).

5. Using the motion field, regions of *significant occlusion* are detected for each frame

individually (see Section 6.4.3).

6. Next, each frame of the image sequence is stylized sequentially. First, the source image is Gaussian blurred to form a canvas *underpainting*.

7. Brush strokes are distributed over the underpainting (i.e. SBR). Once initialized, the *attributes* of brush strokes are stored as structures in three ordered lists - one for each semantic layer; background, foreground and detail.

8. If frame undergoing SBR is not the first frame of the sequence, the anchor points of brush strokes from the previous frame are animated to the current frame (see Section 6.4.3). The structures of brush strokes whose anchor points fall into occlusion regions after this process are deleted from the lists. Brush strokes which cross the boundaries of semantic layers, or cause redundancy, are also deleted. The attributes of all surviving strokes (i.e. anchor point location, color, orientation, shape) are then re-calculated, and their structures are updated accordingly.

9. The probabilistic anchor point distribution technique is used to distribute further brush strokes until the painting is finished (see Section 6.3 for the completion criteria). The strokes may be distributed in regions masked by each semantic layer separately, or over all semantic regions at once (see Section 6.4.2 for explanation). The user may have chosen to vary certain attribute of brush strokes between semantic layers (e.g elliptical equivalent radius (see Section 6.2). Each trial in the process of anchor point distribution, therefore, is tuned to the equivalent radius (see Section 6.2), and motion-expressive stretch and orientation (see Section 6.2.1), of each candidate brush stroke with regard to the style parameters defined for each semantic layer. It is also aware of the stretch-curbing influence of the detail distance filter (see Section 6.6) at each image site. Once calculated, the attributes of anchor point location, elliptical stretch and orientation are updated in the structure of each brush stroke.

10. The dominant color of each brush stroke is usually calculated using the spatio-temporal color-sampling process (see Section 6.5). The user, however, may have chosen to color the brush strokes of one or more semantic layers using the lens-like effect (see Section 6.7). Once calculated, the color attributes are updated in the structure of each brush stroke.

11. After painting completion, brush strokes are composited. Brush strokes of the background list are composited first, then those of the foreground, and finally detail. The user may have specified an alpha value for each semantic layer, and other parameters such as noise with regard to brush stroke color or orientation. During compositing, the attributes of brush strokes are queried in their structures.

**Figure 6.25:** *Example frames from the source videos; (l-r) Swimming, Hollywood, Male1, Hollywood, Female1, Hollywood.*

12. Final touches are added to the frame. This mainly consists of overlaying the cartoon-like edge decoration (see Section 6.7). Again, the user may have specified a color, and alpha value for compositing the edge decoration.

## 6.9 Results

A number of source image sequences have been stylized using this novel skin-aware video stylization framework. Example frames from these source sequences can be seen Figure 6.25. Small versions (i.e. 50% original aspect ratio) of the source sequences, *Female1*, *Male1*, *Swimming* and *Hollywood* are included on the accompanying DVD, along with a number of stylized output sequences that have been transformed using a variety of parameters and effects within the video stylization framework.

A full list of the parameters used for each stylized video is included in Appendix C. The resulting video portraits are fun and interesting, expressive of semantic content and motion, and temporally smooth with regard to color-sampling. Some interesting frames from the stylized videos can be seen in Figures 6.26 and 6.27, as well as the previously mentioned Figures 6.23 and 6.24.

The *Hollywood* sequence is notable in that it contain *shot cuts*, and the stylized videos associated with it demonstrate the interesting behavior of the motion-expressive ellipses and spatio-temporal color-sampling process over the shot cuts. Only the filters for spatio-temporal smoothing of the semantic layer masks - as described in Section 6.4 - are temporally clipped at the shot cut boundaries during stylization. The original source video was a slow motion demo reel, and so four of the stylized results are presented at the slow frame rate of 10fps (i.e. frames per second); *Hollywood_10fps_(1-4)*. The others are not, however, since they exist for the purpose of comparing the different spatio-temporal color-sampling techniques.

Three of the stylized *Hollywood* videos have been created especially for the purpose of comparing (a) the new spatio-temporal color-sampling process presented in this Chapter (see Section 6.5), with two well-known color-sampling techniques of (b) simple frame-wise point-sampling at anchor point coordinates, and (c) uniform volumetric motion-compensated averaging. *Hollywood_b1*, *Hollywood_point* and *Hollywood_mean*, are stylized using techniques (a), (b), and (c) respectively. For (a), $N_b = 3$, $\omega_b = 0.3$, and a spatial color-sampling

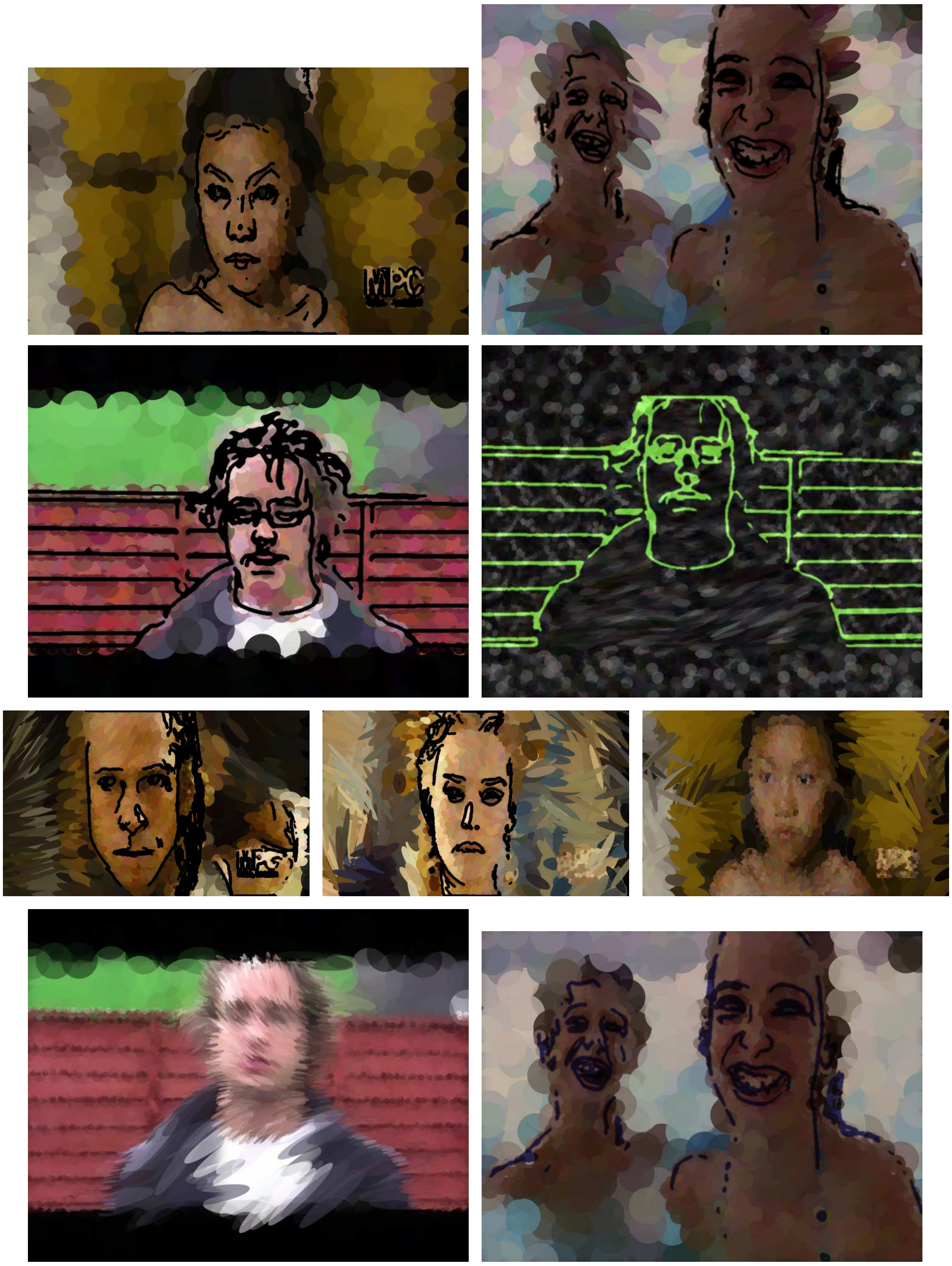


**Figure 6.26:** *Example frames from stylized videos; (l-r,t-b) Hollywood_10fps_1, Swimming_3, Male1_1, Male1_3, Hollywood_10fps_3, Hollywood_10fps_3, Hollywood_10fps_2, Male1_2, Swimming_1. The parameters used to stylize these sequences are listed in Appendix C.*

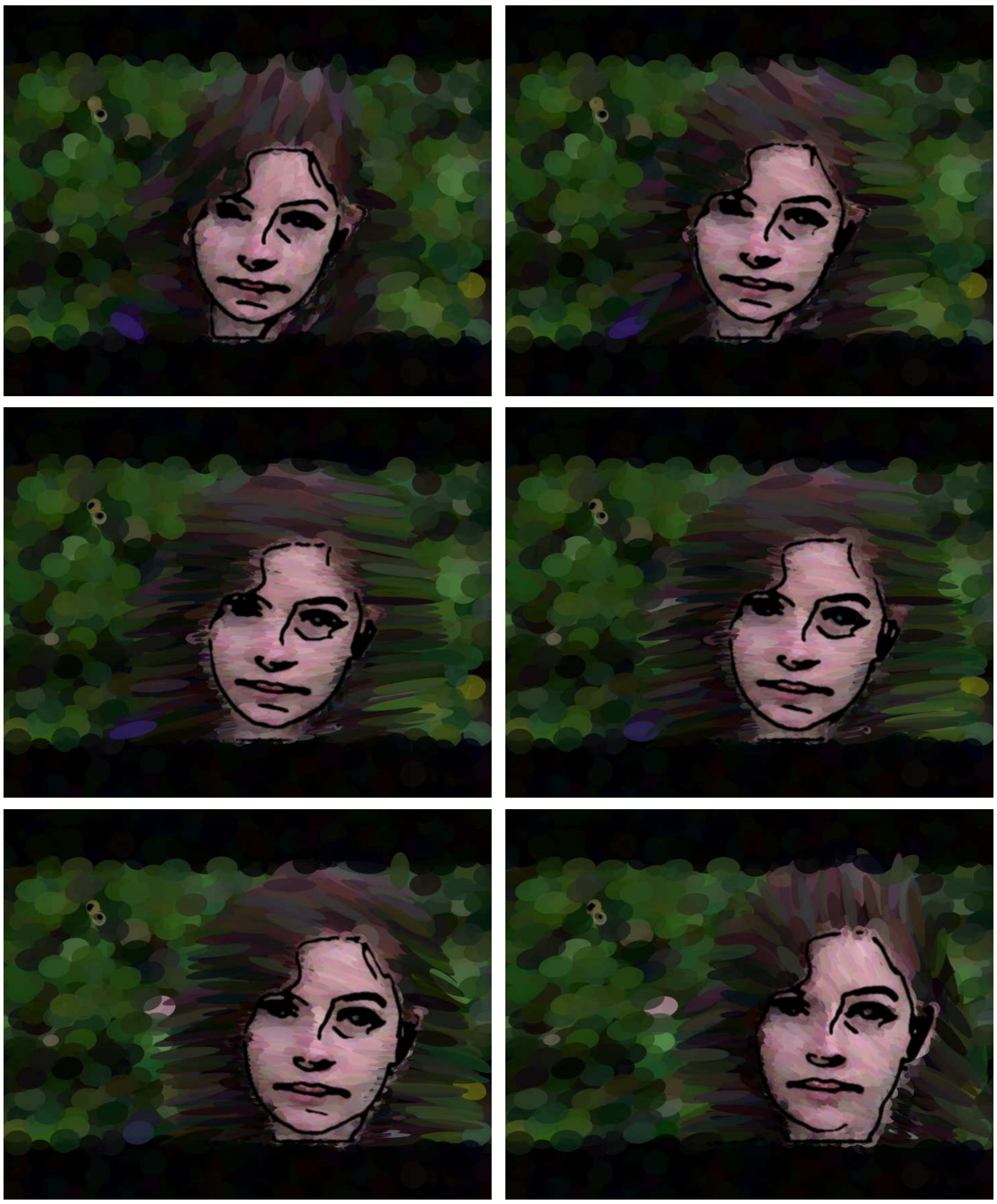


**Figure 6.27:** *A selection of frames from the stylized sequence Female1_3; (l-r,t-b) the overall motion of the character from the center to the right of the shot is revealed in the stretch and orientation of elliptical brush strokes. The specific frames shown here are n=101,104,108,111,114,117 (zero indexed).*

window defined by $k_c = 0.5$ is used with brush strokes of dimensions $r_{eq_b} = 30$, $r_{eq_b} = 15$, $r_{eq_b} = 5$ (see Sections 6.5 and 6.4.2 for explanations of these parameters). In (c), for comparative purposes, uniform symmetrical spatio-temporal volumes of equivalent spatial widths, $W_c = 30, 15, 5$ (i.e. for each semantic layer), and a temporal extent of 7 frames centered on the current frame are utilized for volumetric motion-compensated averaging. These videos demonstrate that (a); the new spatio-temporal color-sampling technique, performs well compared with (c); motion-compensated volumetric averaging, and that the technique of (b); frame-wise point-sampling with no temporal smoothing, simply causes flicker in the stylized output sequence.

The importance of choosing the correct value of $\omega_b$ for the Butterworth filter described in Section 6.5 is demonstrated by comparing the stylized results of *Hollywood_b1*, *Hollywood_b2* and *Hollywood_b3*, in which $N_b = 3$, $k_c = 0.5$, and $\omega_b = 0.3$, $\omega_b = 0.1$, $\omega_b = 0.5$ respectively. The former is the is the most stable value for Butterworth smoothing the colors, whereas the use of $\omega_b = 0.1$ and $\omega_b = 0.5$ produce color distortions and flicker respectively.

The final stylized outputs *Female1_3* and *Swimming_2* demonstrate the lens-like effect described in Section 6.7, and example frames from these stylized videos can be seen in Figures 6.24 (a) and (b), and Figures 6.24 (c) and (d) respectively. For *Female1_3*, the detail layer, $l_d^n(\mathbf{X})$, was created by tracking the pupils in the eyes on the *Female1* source video using the motion tracker in Adobe AfterEffects CS3. In *Swimming_2*, the lens effect is only implemented in the background layer, $l_b^n(\mathbf{X})$. Another strange effect can be seen in the stylized sequence *Male1_3*, in which a single, canvas-sized image of chain mail was repeatedly sampled for coloring the brush strokes in each frame of the painted sequence. The brush strokes are still expressive of motion, however, and the edge decoration still reveals the semantic content of the video, since the male character's portrait is overlaid in neon green.

The sequences *Swimming_3*, *Hollywood_4*, *Hollywood_10fps_4* and *Male1_2* are also notable in that they are stylized without application of the detail distance filter, $k_e^n(\mathbf{X})$, as described in Section 6.7. This somewhat undesirable effect of not curbing the stretch of elliptical brush strokes at edges is intended to demonstrate the usefulness and necessity of $k_e^n(\mathbf{X})$ in creating a more aesthetically pleasing edge-preserving stylization. These videos clearly demonstrate that much of the high-frequency detail is lost in the SBR when the elliptical strokes are allowed to stretch and rotate across edge boundaries. The aesthetic problem can be partly remedied by explicitly superimposing cartoon-like edge decorations, as can be seen in *Swimming_3*.

Also included on the accompanying DVD are two frame-wise spatially coherent skin mask sequences - *Female1_GCskin* and *Hollywood_GCskin* - generated by running the Graph Cut-based Skin Detection algorithm discussed in Chapter 5 on each frame of the *Female1* and *Hollywood* sequences respectively. These can be compared to the corresponding spatio-temporally coherent skin mask sequences - *Female1_smoothskin* and *Hollywood_smoothskin*

- which have been temporally smoothed by the motion-compensated median filtering technique discussed in Section 6.4 of this Chapter.

## 6.10 Future Work

Future work is dependent on the stylistic goals of a user. A variety of brush stroke shapes and styles could be implemented for SBR, with paint-like textures and lighting, as described by Hertzmann (Hertzmann, 2003). The implementation of spline-like brush strokes might be interesting. Perhaps the brush stroke anchor point motion trajectories described in Section 6.2.1 could be used to influence the spline's control points.

The framework could be extended to incorporate awareness of content other than skin color and associated edge features. Aspects of sparse local feature-based Face Detection, as discussed in Chapter 3, could be incorporated into the framework. Sparse local features classified as belonging to a face, for example, might also be considered salient content in the non-uniform stylization. Object Detection or color segmentation could be introduced so that the framework would be capable of automatically recognizing and enhancing even more semantic content, or the the user could make indications of semantic regions by drawing "scribbles" in the first frame of the source video, enabling the sampling of foreground and background features.

The existing skin-aware stylization framework could be refined and extended with improvements to the underlying Skin and Edge Detection algorithms, SBR and cartoonization, and spatio-temporal coherency. Rapid temporal changes in the orientation of elliptical brush strokes, for example, are observable in some of the stylized sequences. The elliptical orientation, $\theta$, could be smoothed in a manner similar to the temporal filtering of color described in Section 6.5. The cartoon-like edges are prone to flicker in periods of rapid motion (see stylized outputs of *Swimming*) or changes in illumination (see stylized versions of *Male1*). Skin-colored hair and background is sometimes falsely masked as skin (see stylized versions of *Hollywood* for misclassified hair and skin-colored text overlay, and *Male1* in which a red-colored background is mis-classified as skin). It is subjective, however, as to whether these failures of the skin detector have a negative effect on the overall stylized appearance of the video results.

## 6.11 Conclusion

A novel Non-Photorealistic/Stroke-based Rendering (NPR/SBR) framework for the skin-aware stylization of videos featuring head shots of people has been presented. This framework merges aspects of SBR, cartoonization, motion expression, and other quirky ideas in NPR. Spatio-temporal Skin and Edge Detection has been used to enable a non-uniform content-based stylization in which the skin and facial features of a human subject can be stylized more carefully and highlighted, while the background region is abstracted. The use of motion-expressive elliptical brush strokes for SBR empowers motion visualization and summarization, such that a snapshot of the underlying source video motion is captured in every frame of the output stylized sequence. Furthermore, novel ideas for probabilistic brush stroke anchor point distribution, spatio-temporal color-sampling, and methods for dealing with gaps and redundancy in brush stroke animation have been presented. The resulting stylized sequences are visually interesting, artistic, expressive of motion and the skin-based semantic content. This framework might be useful for the stylization of home movies, films or camera mobile phone clips, and many directions for future work have been discussed.

# 7 Conclusion

This Thesis has developed a range of tools that rely on the concept of Content-Based Media Processing (CBMP). In each of the three media-based projects that were undertaken in the course of this work, some aspect of audio, image or video content has been detected or exploited. The context and contribution of each of these projects in the field of CBMP will now be discussed.

Sound Texture Synthesis (STS) is extremely useful in audio repair and re-synthesis, compression, and ambient sound production. The latter application is becoming increasingly relevant in the world of computer games and virtual environments where natural background noise or music is required for continuous play. Results in the project of example-based STS suggest that it is a good solution for the production of believable sound textures. This content-based STS technique of estimating the statistics of sound textures by directly measuring from the data in real-world training example clips is distinct from the task of producing sound from model-based synthesis techniques or virtual computer music generation. Example-based STS is conducive to the production of natural, varied, good quality sound textures which can be extended to the long duration required for such applications.

Face and Object Detection are obvious examples of semantic content detection in visual media. The project concerning Implicit Spatial Inference (ISI) with sparse local features for Face Detection highlights many layers of CBMP. It involves the low-level detection of sparse local features, the mid-level task of labeling these features, the higher-level concept of grouping the isolated features according to implicit geometrical context, and the global view of segmenting a face as a semantic image region. The idea that other image processing filters can be applied non-uniformly with regard to the masked semantic region is yet another concept in CBMP, and one of the possible applications of this project. This project, therefore, demonstrates how traditional feature-based techniques in machine learning, graph-based meshing, and Bayesian statistics can be fused in the pursuit of multi-level content detection and media adaptation.

The work concerning the skin-aware stylization of video portraits is an example of a complete framework in CBMP for the artistic semantic stylization of home-movie or cinema-like sequences featuring head shots of people. The project features the analysis of low-level video content such as motion vectors, color and edges, and this information is leveraged for higher level concepts in video processing such as spatio-temporal Skin and Edge Detection, non-uniform skin-aware stylization and cartoonization, motion depiction and expression. The framework incorporates a novel Graph Cut-based Skin Detection algorithm. Video-based content analysis has been used to formulate solutions for various challenges in the field of Stroke-Based Rendering (SBR). These solutions include novel techniques for probabilistic brush stroke anchor point distribution that is non-uniform over spatio-temporally defined semantic image layers, and the use of motion vector-based occlusion detection for the mitigation of gaps and redundancy in motion-compensated brush stroke animation. Also novel is the depiction and enhancement of motion through brush strokes whose attributes are continuously morphed according to the behavior of the underlying local motion field, but high frequency stylized details are still preserved by curbing this behavior at spatio-temporally defined edges.

## 7.1 Future Work

Each of these projects have a huge potential for further research. There is scope in example-Based STS for further complexity reduction, better content-based parameter estimation, and application to real-world problems like the repair of digitized archive audio and ambient sound generation in computer games and installations.

The concept of ISI could be adapted to a multitude of applications of sparse local feature-based classification, invariant Face or Object Detection. It is accepted that sparse local features are highly informative, so it makes sense to use them in the pursuit of invariant Face and Object Detection on a grander scale. Getting these sparse locations to communicate implicitly is an intuitive concept, but difficult to model. The ISI technique proposed here is a good first attempt at this task. A simple example of sparse local feature-based segmentation has also been presented. Natural directions for further work in this project is the incorporation of either sparse-to-dense or user-assisted inference techniques (see (Ring & Pitié, 2009; Ring & Kokaram, 2009) for examples) for better geometric inference, and a better segmentation matte.

The concept of the semantic stylization in visual media processing also has potential for further work. The existing skin-aware stylization framework could be refined and extended with improvements to the underlying Skin Detection, SBR and cartoonization techniques, more exaggerated motion depiction, and improved spatio-temporal coherency in all aspects of the system. Perhaps aspects of sparse local feature-based Face Detection could be incorporated into the framework. Sparse local features classified as belonging to a face, for example, might also be considered salient content in the non-uniform stylization. Face Detection could also be used to re-classify some of the skin-colored background falsely segmented in the Skin Detection, or sparse local feature-based Face Detection could be used to seed a sparse-to-dense skin segmentation involving both sparse local and color features. Object Detection could be introduced so that the framework would be capable of recognizing and enhancing semantic content other than skin. On the aesthetic side, the brush stroke model could be improved with aspects of paint texture and lighting, such that the stylized results appear more painterly.

## 7.2 Issues with CBMP

Some noteworthy issues uncovered by this work in CBMP are as follows:

**The Meaning of Meaningfulness** There are many levels of media content. There are the the low-level heuristics of pixel-wise color, local motion and image gradients, mid-level features such as audio beats, edges and the appearance of local image regions. There are more high-level concepts such as characteristic color classes (e.g. skin color), semantic objects such as faces, and their motion trajectories in videos. It is well accepted that the idea of *semantic content* in media is loosely defined as an attribute, object or "event" that is *meaningful*, important, recognizable or stand-alone to the user and/or application. The issue brought forth by the question "At what level can content can be thought of as semantic?" is a well-debated one in the field of CBMP, which much contention over how it is defined within bodies of audio, image and video data. It is difficult to comprehend bridging the semantic gap when this gap is not easily named. What has been concluded from this work, however, is that there are many different goals and levels of automation in applications of CBMP, and that the term *semantic* is best defined according to the desired application and needs of the end-user. If a user wishes to sort her digital music collection in order of increasing measures of bpm (i.e. beats per minute), then the beats in the music are semantic, and all that is required is a localization of their onsets (i.e. to determine the tempo). There is a wider range of semantic content, however, in the larger task of automated sports summarization for Digital TV - from the faces of sports people, to the trajectory of a soccer ball, to audio/visual events such as the sound of a tennis ball hitting a racket or the sight of a snooker ball disappearing from the table as it is potted.

**Training and Complexity** It seems that many aspects of CBMP are still limited by issues of complexity, bandwidth, and memory. Many algorithms in the fields of Face and Object Detection - such as that of (Viola & Jones, 2004) - for example, involve the extraction of hundreds of thousands of feature vectors and heavy offline training with multiple iterations of computationally expensive machine learning cycles. Because of this complexity, experiments are often scaled down to toy tests on small, well-calibrated, low-quality databases such as the well-known MIT/CMU face database (Rowley et al., 1998b). There is a huge hole in the field, therefore, for more large-scale experiments on realistic data such as high resolution images or hours of video footage captured in home movies or CCTV. As both the volume and quality of data increases, however, so too do the volume of salient features and complexity of the machine learning required. There is also a need, therefore, for more memory efficient representations of feature vectors, as well as increased focus on the optimization or simplification of machine learning algorithms, perhaps by involving some user assistance.

**The Automation Ally** It is widely agreed that any aspect of automation in media processing is invaluable when resulting in a saving of human time and effort. A good foreground/background segmentation matting algorithm, for example, can save hundreds of labor hours in a typical cinema post-production house. Similarly, a reliable STS algorithm does away with the hassle of obtaining hours of ambient field recordings for use as ambient backing tracks in an art installations or computer game-based virtual worlds. Projects in CBMP are dedicated to the goals of automation, labor reduction, time saving and the avoidance of user frustration. However, since the results of these algorithms are seldom perfect, it is quite useful to keep the end-user in the loop to mitigate problems as they arise. The part-automation of user-assisted CBMP, therefore, will be a valuable ally in our increasingly media-saturated personal and professional lives of the future.

## 7.3 Final Remarks

The field of CBMP is a large, and rapidly growing one. It is hoped, however, that the work contained in the three projects of this Thesis has contributed some novel and useful solutions or ideas to each of these topics individually, as well as to the general field of CBMP. It is also hoped that future directions for some further research have been highlighted both here and throughout this Thesis, and that these can, and will be pursued with mindfulness of the issues involved.

# A
# Further Sound Texture Synthesis Results

The Tables on the following pages are a list of parameters associated with some alternative sound textures resulting from the work on example-based Sound Texture Synthesis (STS) discussed in Chapter 2, and included on the accompanying DVD. Unlike those listed in Tables 2.9 and 2.16 of Chapter 2, these were not the best sound textures produced for the listed training examples, $\mathbf{Y}_e$, but they are interesting results nevertheless. Note: The underlying filenames match the parameters in these Tables, taking the form #-$\mathbf{Y}_e$-Training Example_$K$_$W_s$_$\epsilon$.wav.

| # | $\mathbf{Y}_e$ | Training Example | Stereo | Seed (samples$[n^K]$, duration$[s]$) | $K$ | $W_s$ | $\epsilon$ | $t_2[s]$ | Description |
|---|---|---|---|---|---|---|---|---|---|
| a1 | 1 | *drum loop* | N/A | placed in center $(1..87, 1)$ | 8 | 51 | 0.1 | 19 | Repetitive but plausible. Eventually gets locked into a different beat structure. |
| a2 | 1 | *drum loop* | N/A | placed in center $(1..201, 2.3)$ | 8 | 201 | 0.1 | 19 | Heavier quality but plausible. Quicker cymbals. Eventually gets locked again. |
| a3 | 1 | *drum loop* | N/A | placed at start $(1..125, 1.5)$ | 8 | 61 | 0.1 | 63 | Repetitive. Eventually gets locked again. |
| a4 | 1 | *drum loop* | N/A | placed at start $(1..201, 2.3)$ | 8 | 201 | 0.1 | 63 | Perfectly repetitive but with no cymbals. |
| b1 | 2 | *baby crying* | N/A | placed in center $(380..580, 4.6)$ | 8 | 101 | 0.1 | 29 | Eventually gets stuck in a short repetitive loop but plausible. |
| b2 | 2 | *baby crying* | N/A | placed in center $(10..380, 2.1)$ | 6 | 751 | 0.1 | 53 | Training example is smoothly tiled. |
| b3 | 2 | *baby crying* | N/A | placed in center $(100..250, 0.9)$ | 6 | 51 | 0.1 | 53 | Good variation overall. |
| b4 | 2 | *baby crying* | N/A | placed at start $(100..170, 0.4)$ | 6 | 25 | 0.1 | 73 | Good variation overall. |
| c1 | 3 | *traffic jam* | N/A | placed in center $(200..400, 4.6)$ | 8 | 31 | 0.1 | 38 | Training example is smoothly tiled. Original training example was not clipped. |
| c2 | 3 | *traffic jam* | N/A | placed in center $(500..600, 1.7)$ | 8 | 63 | 0.01 | 70 | Some variation, some tiling. Too much honking. |
| c3 | 3 | *traffic jam* | N/A | placed at start $(450..457, 0.04)$ | 7 | 3 | 0.1 | 82 | Training example is tiled. |
| d1 | 4 | *shore splashing* | N/A | placed at start $(400..450, 0.6)$ | 7 | 11 | 0.1 | 78 | Varied, but silences appear abruptly. Original training example was not clipped. |
| d2 | 4 | *shore splashing* | N/A | placed at start $(400..411, 0.3)$ | 8 | 5 | 0.1 | 78 | Repetitive, smooth. Varied silence periods. Original training example was not clipped. |
| d3 | 4 | *shore splashing* | N/A | placed at start $(100..300, 4.6)$ | 8 | 101 | 0.1 | 74 | Good variation. |
| e1 | 5 | *formula 1* | N/A | placed in center $(300..500, 4.6)$ | 8 | 51 | 0.1 | 32 | Training example is tiled. Abrupt silences. Original training example was not clipped. |
| e2 | 5 | *formula 1* | N/A | placed at start $(400..500, 2.3)$ | 8 | 21 | 0.1 | 75 | Good variation overall. One click at end. |

| # | $\mathbf{Y}_e$ | Training Example | Stereo | Seed (samples[$n^K$], duration[$s$]) | $K$ | $W_s$ | $\epsilon$ | $t_2[s]$ | Description |
|---|---|---|---|---|---|---|---|---|---|
| f1 | 6 | *crowd chatter* | N/A | placed in center $(100..200, 0.6)$ | 7 | 51 | 0.01 | 27 | Looping but variation at each side of seed. Acceptable as background noise. |
| f2 | 6 | *crowd chatter* | N/A | placed in center $(100..300, 1.2)$ | 7 | 71 | 0.01 | 41 | Good variation overall. Slight whirring heard after vendor shouting “nuts” event. |
| f3 | 6 | *crowd chatter* | N/A | placed at start $(100..130, 0.3)$ | 8 | 11 | 0.1 | 71 | Smoother. No whirring but looping of vendor then man talking. |
| g1 | 7 | *crowd chatter 2* | C | placed at start $(1..58, 0.7)$ | 9 | 55 | 0.09 | 66 | Good variation of “lady shouting” event over noise. |
| g2 | 7 | *crowd chatter 2* | A | placed at start $(1..58, 0.7)$ | 9 | 55 | 0.3 | 66 | Not full variation. The “lady shouting” event never emerges. |
| g3 | 7 | *crowd chatter 2* | D | placed at start $(1..58, 0.7)$ | 9 | 55 | 0.7 | 66 | Channels fall out of sync after 6$s$ (training example duration). |
| h1 | 8 | *drum loop 2* | A | placed at start $(1..50, 0.3)$ | 9 | 47 | 0.5 | 62 | Not good. Tiled end-to-end. |
| h2 | 8 | *drum loop 2* | C | placed at start $(1..50, 0.3)$ | 9 | 47 | 0.5 | 62 | Good variation. |
| h3 | 8 | *drum loop 2* | D | placed at start $(1..50, 0.3)$ | 9 | 47 | 0.5 | 62 | Channels fall out of sync after 2$s$ (training example duration). |
| i1 | 9 | *baby laughing* | A | placed at start $(1..80, 0.5)$ | 8 | 77 | 0.1 | 70 | Not good. Some tiling is noticeable. |
| i2 | 9 | *baby laughing* | D | placed at start $(1..80, 0.5)$ | 8 | 77 | 0.1 | 70 | Channels fall out of sync after 6$s$ (training example duration). |
| j1 | 10 | *polyphonic music* | A | placed at start $(1..20, 0.5)$ | 10 | 17 | 0.3 | 90 | Acceptable result. Long periods before variation. |
| j2 | 10 | *polyphonic music* | D | placed at start $(1..20, 0.5)$ | 10 | 17 | 0.3 | 90 | Channels fall out of sync after 20$s$ (training example duration). |

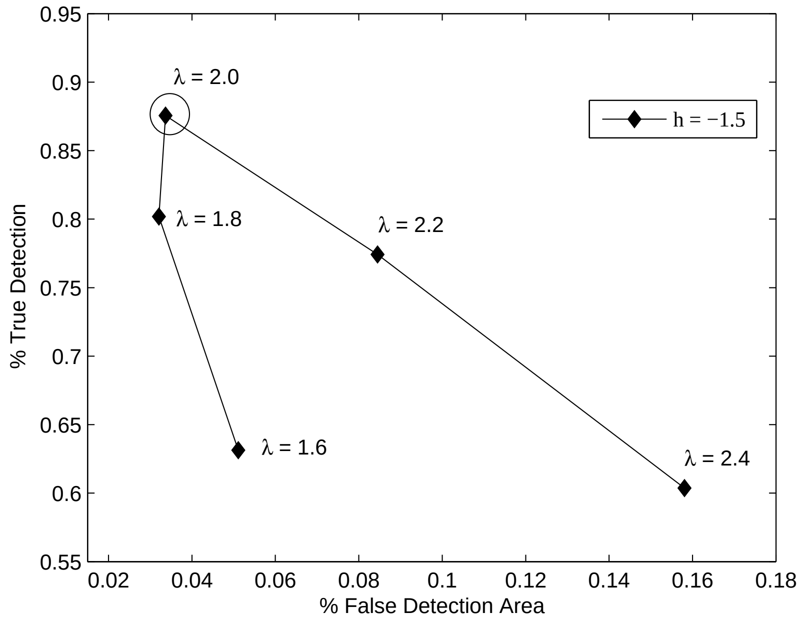
| # | $\mathbf{Y}_e$ | Training Example | Stereo | Seed (samples$[n^K]$, duration$[s]$) | $K$ | $W_s$ | $\epsilon$ | $t_2[s]$ | Description |
|---|---|---|---|---|---|---|---|---|---|
| k1 | 11 | *german speech* | N/A | placed in center $(400..700, 3.5)$ | 8 | 251 | 0.001 | 28 | Looping but variation at each side of seed. |
| k2 | 11 | *german speech* | N/A | placed at start $(1..4, 5.9)$ | 15 | 3 | 0.3 | 60 | Structural decomposition. Good variation. Slight noise. Some annoying looping. |
| k3 | 11 | *german speech* | N/A | placed at start $(1..4, 11.9)$ | 16 | 3 | 0.3 | 60 | Further structural decomposition. No noise. Strange sounds and repetition. |
| l1 | 12 | *english speech bg music* | N/A | placed at start $(1..4, 5.9)$ | 15 | 3 | 0.1 | 70 | Good variation with fast tempo. Periods of major distortion but interesting. |
| l2 | 12 | *english speech bg music* | N/A | placed at start $(1..4, 3)$ | 14 | 3 | 0.1 | 70 | Good variation with fast tempo. Periods of major distortion but interesting. |
| l3 | 12 | *english speech bg music* | N/A | placed at start $(1..4, 1.7)$ | 13 | 3 | 0.1 | 70 | Structural decomposition. Faster tempo. Interesting. Occasional major distortion. |
| l4 | 12 | *english speech bg music* | N/A | placed at start $(1..4, 0.9)$ | 12 | 3 | 0 | 70 | Very fast tempo. Reminiscent of phoneme decomposition. Occasional major distortion. |
| l5 | 12 | *english speech bg music* | N/A | placed at start $(1..4, 5.9)$ | 15 | 3 | 10 | 70 | Even further decomposition. Bizarre. |
| m1 | 13 | *piano phrases* | N/A | placed at start $(100..200, 1.2)$ | 8 | 51 | 0.3 | 146 | Different ordering of the two piano phrases. |
| m2 | 13 | *piano phrases* | N/A | placed at start $(10..14, 5.9)$ | 15 | 3 | 0.001 | 146 | Reminiscent of note-like decomposition. Fairly smooth random variation of notes. Segments of original phrases still emerge. |
| m3 | 13 | *piano phrases* | N/A | placed at start $(10..14, 5.9)$ | 15 | 3 | 0.3 | 86 | Structure more erratic. Slightly more short-term looping. Abruptness. |

# B

# Parameter Trials in ISI with Sparse Local Features for Face Detection

The following two Figures show the results of trials that were necessary for choosing suitable values for the parameters $h$, $\Lambda_p$ and $\Lambda_n$ inherent to the algorithm underlying Implicit Spatial Inference (ISI) with sparse local features for Face Detection, as described in Chapter 3. The definition of these two parameters can be found in Section 3.7 of Chapter 3. Some other similar trials are also discussed in Section 3.8.4 of the Chapter.

Each Figure shows a graph of $\%TD$ versus $\%FDA$ on the top, and one of $R$ versus $1-P$ on the bottom. These metrics of $\%TD$, $\%FDA$, $R$ and $P$ are discussed in Section 3.8.3 of Chapter 3. The trials were carried out by varying the parameters of $\lambda = \Lambda_n/\Lambda_p$ versus $h$ (see Figure B.1), and vice versa (see Figure B.2), in small increments, computing these metrics over all of the subset of $N_T = 217$ test images from the Caltech-101 face database (Fei-Fei et al., 2004). The circled points in the graph reveal the most useful values for $\lambda = \Lambda_n/\Lambda_p$ and $h$ in these trials. Definitions of the other parameters mentioned can also be found in Section 3.7 of Chapter 3.



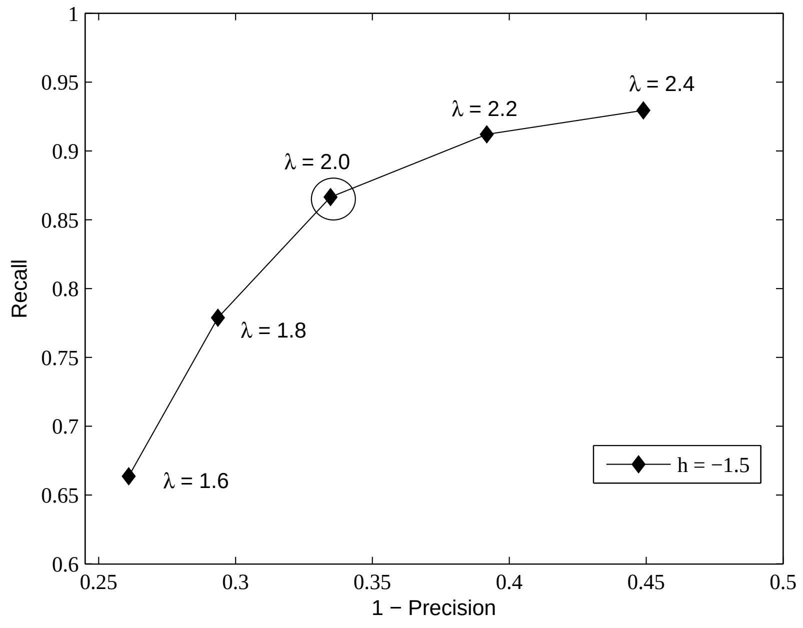


**Figure B.1:** *Choosing a suitable value for* $\lambda = \Lambda_n/\Lambda_p$*; (top)* $\%TD$ *versus* $\%FDA$*, and (bottom),* $R$ *versus* $1 - P$*. Other parameters are set at* $h = -1.5$*,* $t_p = 0.6$*,* $v_p = 30\omega_p$*,* $n_d = 3$*,* $\alpha_d = 20$*.*

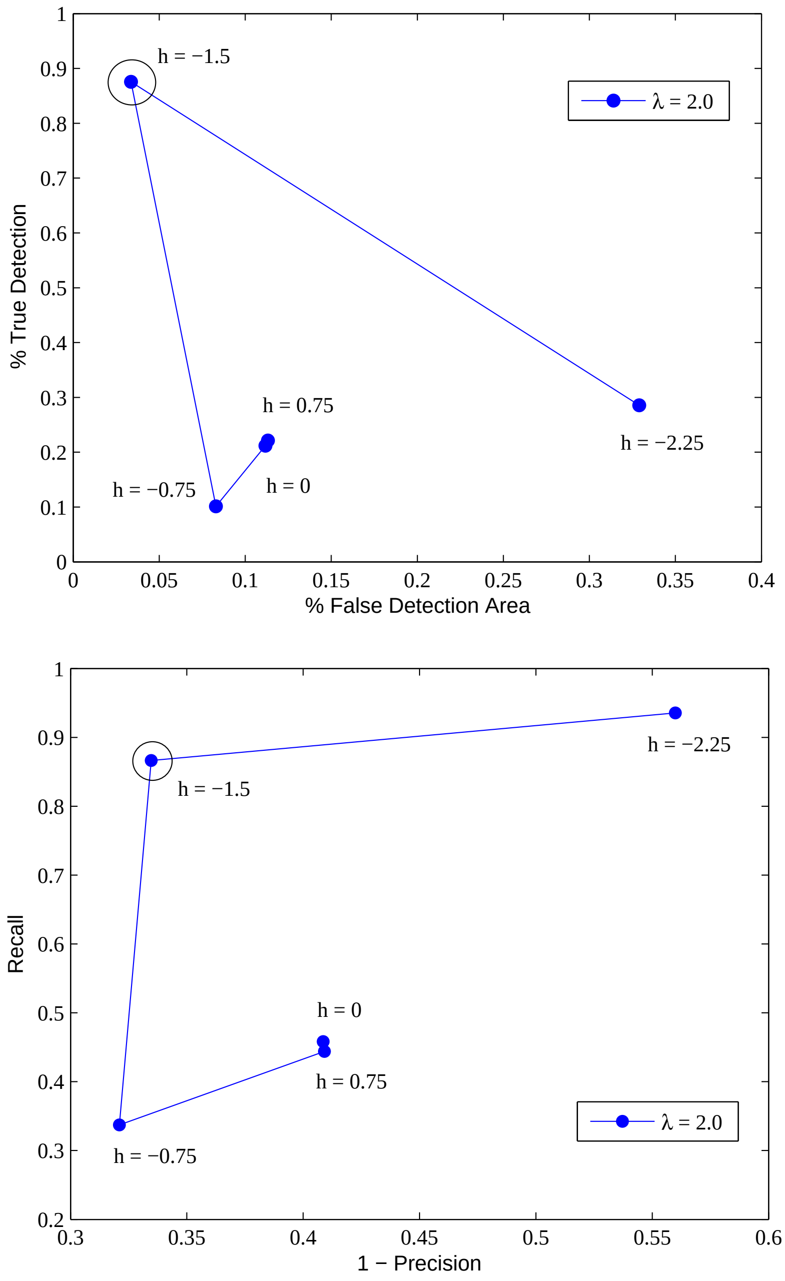


**Figure B.2:** *Choosing a suitable value for h; (top) $\%TD$ versus $\%FDA$, and (bottom), $R$ versus $1-P$. Other parameters are set at $\lambda = 2$ (i.e. $\Lambda_n = 2$, $\Lambda_p = 1$), $t_p = 0.6$, $v_p = 30\omega_p$, $n_d = 3$, $\alpha_d = 20$.*

# C
# Parameters in the Skin-Aware Stylization of Video Portraits

The Table, split over the following three pages, is a list of parameters associated with the stylized sequences resulting from the work on the Skin-Aware Stylization of Video Portraits discussed in Chapter 6, and included on the accompanying DVD. The source sequences are also included on the DVD; *Female1*, *Swimming*, *Hollywood*, and *Male1*.

*M.C./A = Motion-Compensated/Alternative Technique (Equation 6.3/Equation 6.6): The chosen method of determining the elliptical stretch and orientation as described in Section 6.2.1 of Chapter 6.

**W/L = Weighted/Layer-Based: The chosen method of distributing strokes on separate semantic layers. Two separate approaches are described in Section 6.4.2 of Chapter 6. An entry of 1/0 (i.e. the weighted option) has a set of weights beside it in the Table.

N/A = Not Applicable. For example, no detail distance filter, $d^n(\mathbf{X})$, is used in *Hollywood_4* (see Equation 6.6, Section 6.7, Chapter 6), and therefore $E_d$ is N/A.

| Vid/Param | $\alpha$ | $\hat{u}$ | $\hat{k}_e$ | $\delta\theta$ | $N_u$ | $v_s$ | $C$ | $k_s$ | $t_a$ | $A_H$ | *M.C./A |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Female1_1* | 0.75 | 35 | 9 | 0.04 | 7 | 2 | 0 | 1 | 2 | 5 | 0/1 |
| *Female1_2* | 0.75 | 35 | 9 | 0.04 | 7 | 2 | 0 | 1 | 2 | 5 | 0/1 |
| *Female1_3* | N/A | N/A | 1 | N/A | 7 | 2 | 0 | 1 | 2 | 5 | 0/1 |
| *Female1_4* | 0.75 | 35 | 9 | 0.04 | 7 | 2 | 0 | 1 | 2 | 5 | 0/1 |
| *Swimming_1* | 0.5 | N/A | 1 | 0.08 | 7 | 2 | 0.1 | 1 | 1.5 | 20 | N/A |
| *Swimming_2* | 1 | 35 | 3 | 0.08 | 7 | 3 | 0 | 1 | 2 | 500 | 0/1 |
| *Swimming_3* | 0.8 | 35 | 5 | 0.08 | 7 | 2 | 0 | 1 | 1.5 | 2 | 1/0 |
| *Hollywood_b1* | 0.9 | N/A | 1 | 0.04 | 7 | 2 | 0 | 1 | 2 | 10 | N/A |
| *Hollywood_b2* | 0.9 | N/A | 1 | 0.04 | 7 | 2 | 0 | 1 | 2 | 10 | N/A |
| *Hollywood_b3* | 0.9 | N/A | 1 | 0.04 | 7 | 2 | 0 | 1 | 2 | 10 | N/A |
| *Hollywood_point* | 0.9 | N/A | 1 | 0.04 | 7 | 2 | 0 | 1 | 2 | 10 | 0/1 |
| *Hollywood_mean* | 0.9 | N/A | 1 | 0.04 | 7 | 2 | 0 | 1 | 2 | 10 | N/A |
| *Hollywood_10fps_1* | 0.9 | 32 | 6 | 0.04 | 7 | 3 | 0.1 | 1 | 2 | 10 | 0/1 |
| *Hollywood_10fps_2* | 0.9 | 32 | 4 | 0.04 | 7 | 3 | 0.1 | 1 | 2 | 10 | 0/1 |
| *Hollywood_10fps_3* | 0.9 | 32 | 4 | 0.04 | 7 | 3 | 0.1 | 1 | 2 | 10 | 0/1 |
| *Male1_1* | 0.8 | 30 | 10 | 0.08 | 7 | 3 | 0.1 | 1 | 2 | 10 | 0/1 |
| *Male1_2* | 0.5 | 30 | 10 | 0.04 | 7 | 3 | 0.1 | 1 | 2 | 10 | 1/0 |
| *Male1_3* | 0.5 | 30 | 10 | 0.16 | 7 | 3 | 0.1 | 1 | 0 | 10 | 1/0 |

| Vid/Param | $\Theta$ | $\alpha_{gc}$ | **W/L | $r_{d_1}$ | $W_{L(f,d)}$ | $N_{L(f,d)}$ | $r_{eq(b,f,d)}$ |
|---|---|---|---|---|---|---|---|
| *Female1_1* | 10 | 500 | 0/1 | 3 | 11,7 | 5,3 | 20,10,0 |
| *Female1_2* | 10 | 500 | 0/1 | 1 | 11,7 | 5,3 | 20,10,5 |
| *Female1_3* | 10 | 500 | 0/1 | 1 | 11,7 | 5,3 | 50,30,75 |
| *Female1_4* | 10 | 500 | 0/1 | 1 | 11,7 | 5,3 | 20,10,5 |
| *Swimming_1* | 2 | 10 | 0/1 | 2 | 11,7 | 5,2 | 25,15,7 |
| *Swimming_2* | 2 | 10 | 1/0 $k_b = 1, k_f = 10, k_d = 1000$ | 2 | 11,7 | 5,2 | 40,20,10 |
| *Swimming_3* | 2 | 10 | 1/0 $k_b = 1, k_f = 10, k_d = 1000$ | 2 | 11,7 | 5,2 | 25,15,10 |
| *Hollywood_b1* | 0.8 | 30 | 0/1 | 1 | 11,7 | 5,2 | 30,10,5 |
| *Hollywood_b2* | 0.8 | 30 | 0/1 | 1 | 11,7 | 5,2 | 30,10,5 |
| *Hollywood_b3* | 0.8 | 30 | 0/1 | 1 | 11,7 | 5,2 | 30,10,5 |
| *Hollywood_point* | 0.8 | 30 | 0/1 | 1 | 7,7 | 5,2 | 30,10,5 |
| *Hollywood_mean* | 0.8 | 30 | 0/1 | 1 | 7,7 | 5,2 | 30,10,5 |
| *Hollywood_10fps_1* | 0.8 | 30 | 0/1 | 0 | 7,7 | 5,2 | 20,10,5 |
| *Hollywood_10fps_2* | 0.8 | 30 | 0/1 | 0 | 7,7 | 5,2 | 20,10,5 |
| *Hollywood_10fps_3* | 0.8 | 30 | 0/1 | 0 | 7,7 | 5,2 | 20,10,5 |
| *Hollywood_10fps_4* | 0.8 | 30 | 0/1 | 0 | 7,7 | 5,2 | 20,10,5 |
| *Male1_1* | 3 | 10 | 0/1 | 2 | 7,7 | 5,3 | 30,20,10 |
| *Male1_2* | 3 | 10 | 0/1 | 2 | 7,7 | 5,3 | 30,10,5 |
| *Male1_3* | 3 | 10 | 1/0 $k_b = 1, k_f = 100, k_d = 1000$ | 2 | 7,7 | 5,3 | 10,5,5 |

| Vid/Param | $t_o$ | $t_m$ | $k_c$ | $\omega_b$ | $E_{c_{(r,g,b)}}$ | $\delta c_{(b,f,d)}$ | $r_{d_2}$ | $E_d$ | $\alpha_d$ | $r_{d_3}$ | $W_d$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Female1_1* | 0.04 | 4 | 0.3 | 0.3 | 0,0,0 | 0.01,0.01,0.01 | 0 | 10 | 1 | 0 | 5 |
| *Female1_2* | 0.04 | 4 | 0.3 | 0.3 | 0,0,0 | 0.01,0.01,0.01 | 0 | 10 | 1 | 0 | 3 |
| *Female1_3* | 0.04 | 4 | 0.5 | N/A | N/A | 0.01,0.01,0.01 | 0 | N/A | N/A | N/A | N/A |
| *Female1_4* | 0.04 | 4 | 0.3 | 0.3 | 0,0,0 | 0.01,0.01,0.01 | 0 | 10 | 1 | 0 | 3 |
| *Swimming_1* | 0.04 | 3 | 0.5 | 0.5 | 0.5,0,0 | 0.02,0.02,0.02 | 4 | N/A | 0.75 | 1 | 5 |
| *Swimming_2* | 0.04 | 4 | 0.5 | 0.5 | 0,0,0 | 0.02,0.02,0.02 | 4 | 5 | 1 | 3 | 5 |
| *Swimming_3* | 0.04 | 4 | 0.1 | 0.5 | 0,0,0 | 0.06,0.02,0.02 | 4 | 10 | 1 | 0 | 5 |
| *Hollywood_b1* | 1 | 10 | 0.5 | 0.3 | 0,0,0 | 0.01,0.01,0.05 | 4 | N/A | N/A | N/A | N/A |
| *Hollywood_b2* | 1 | 10 | 0.5 | 0.1 | 0,0,0 | 0.01,0.01,0.05 | 4 | N/A | N/A | N/A | N/A |
| *Hollywood_b3* | 1 | 10 | 0.5 | 0.5 | 0,0,0 | 0.01,0.01,0.05 | 4 | N/A | N/A | N/A | N/A |
| *Hollywood_point* | 1 | 10 | 0.5 | N/A | 0,0,0 | 0.01,0.01,0.05 | 4 | N/A | N/A | N/A | N/A |
| *Hollywood_mean* | 1 | 10 | 0.5 | N/A | 0,0,0 | 0.01,0.01,0.05 | 4 | N/A | N/A | N/A | N/A |
| *Hollywood_10fps_1* | 0.04 | 5 | 0.3 | 0.5 | 0,0,0 | 0.01,0.04,0.04 | 4 | 2 | 1 | 0 | 5 |
| *Hollywood_10fps_2* | 0.04 | 5 | 0.3 | 0.5 | N/A | 0.01,0.04,0.04 | 4 | 1 | N/A | N/A | N/A |
| *Hollywood_10fps_3* | 0.04 | 5 | 0.3 | 0.5 | 0,0,0 | 0.01,0.04,0.04 | 4 | N/A | 1 | 0 | 7 |
| *Hollywood_10fps_4* | 0.04 | 5 | 0.3 | 0.5 | N/A | 0.01,0.04,0.04 | 4 | N/A | N/A | N/A | N/A |
| *Male1_1* | 0.04 | 6 | 0.3 | 0.5 | 0,0,0 | 0.02,0.02,0.1 | 0 | 1 | N/A | 1 | 5 |
| *Male1_2* | 0.04 | 10 | 0.1 | 0.5 | N/A | 0.01,0.01,0.01 | 0 | 1 | 1 | N/A | N/A |
| *Male1_3* | 0.04 | 10 | 0.1 | 0.3 | 0.51,0.96,0.26 | 0.04,0.04,0.04 | 0 | 2 | 0.75 | 0 | 5 |

# Bibliography

Avidan, S. & Shamir, A. (2007). Seam Carving for Content-Aware Image Resizing. *ACM Trans. on Graphics, (Proc. of SIGGRAPH '07)*, 26(3).

Bay, H., Tuytelaars, T., & Gool, L. V. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 110(3), 346–359.

Besag, J. (1986). On the Statistical Analysis of Dirty Pictures. *Jnl. of the Royal Statistical Society, Series B*, 3, 259–302.

Bousseau, A., Neyret, F., Thollot, J., & Salesin, D. (2007). Video Watercolorization using Bidirectional Texture Advection. *ACM Trans. on Graphics (Proc. of SIGGRAPH '07)*, 26(3). `http://artis.imag.fr/Publications/2007/BNTS07/`.

Boykov, Y. & Jolly, M.-P. (2001). Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In *Proc. of the Intl. Conf. on Computer Vision (ICCV '01)*, 1 (pp. 105–112).

Boykov, Y. & Kolmogorov, V. (2004). An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9), 1124–1137.

Chai, D. & Bourzerdoum, A. (2000). A Bayesian Approach to Skin Color Classification in YCbCr Color Space. In *Proc. of the IEEE Region Ten Conf. (TENCON '00)*.

Collomosse, J. P. & Hall, P. M. (2002). Painterly Rendering Using Image Salience. In *Proc. of Eurographics '02* (pp. 122–128).

Collomosse, J. P., Rowentree, D., & Hall, P. M. (2005). Stroke Surfaces: Temporally Coherent Artistic Animations from Video. *IEEE Trans. on Visualization and Computer Graphics*, 11(4), 540–549.

Collomosse, J. P., Rowntree, D., & Hall, P. M. (2003). Cartoon Style Rendering of Motion from Video. In *Proc of the Intl. Conf. on Vision, Video, and Graphics '03* (pp. 117–124).

Corrigan, D., Robinson, S., & Kokaram, A. (2008). Video Matting Using Motion Extended GrabCut. In *Proc. of the IEEE Conf. on Visual Media Production (CVMP '08)*.

Curtis, C. J. (1998). Loose and Sketchy Animation. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '98), Electronic Art and Animation Catalogue* (pp. 145).

Curtis, C. J., Anderson, S. E., Seims, J. E., Fleischery, K. W., & Salesin, D. H. (1997). Computer-Generated Watercolor. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '97)* (pp. 421–430).

DeCarlo, D. & Santella, A. (2002). Stylization and Abstraction of Photographs. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH '02)*, 21(3), 769–776.

Delaunay, B. (1934). Sur la Sphre Vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7, 793–800.

Denman, H., Rea, N., & Kokaram, A. (2003). Content Based Analysis for Video from Snooker Broadcast. *Jnl. of Computer Vision and Image Understanding, Special Issue on Video Retrieval and Summarization*, 92(2–3), 176–195.

DiPaola, S. (2007). Keynote Paper: Painterly Rendered Portraits from Photographs using a Knowledge-Based Approach. In *Proc. of SPIE: Human Vision and Imaging, Intl. Society for Optical Engineering.*

Dubnov, S., Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., & Werman, M. (2002). Synthesizing Sound Textures through Wavelet Tree Learning. *IEEE Jnl. of Computer Graphics and Applications*, 22(4), 38–48. `http://www.cs.huji.ac.il/labs/cglab/papers/texsyn/sound/`.

Efros, A. A. & Leung, T. K. (1999). Texture Synthesis by Non-Parametric Sampling. In *Proc. of the Intl. Conf. on Computer Vision (ICCV '99)* (pp. 1033–1038).

Everingham, M., Zisserman, A., Williams, C., Gool, L. V., Allan, M., Bishop, C., & Chapelle, O. (2005). The 2005 PASCAL Visual Object Classes Challenge. In *Proc. of the PASCAL Challenge Workshop*: LNAI, Springer-Verlag.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning Generative Visual Models From Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '04), Workshop on Generative-Model Based Vision.*

Fergus, R., Perona, P., & Zisserman, A. (2003). Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '03)*, 2 (pp. 264–271).

Freund, Y. & Schapire, R. E. (1995). A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. In *European Conf. on Computational Learning Theory* (pp. 23–37).

Gallagher, C. (2006). *Example Based Image Processing*. PhD thesis, University of Dublin, Trinity College.

Gallagher, C. & Kokaram, A. C. (2005). Nonparametric Wavelet Based Texture Synthesis. In *Proc. of the Intl. Conf. on Image Processing (ICIP '05)*, 2 (pp. 462–465).

Gooch, B., Coombe, G., & Shirley, P. (2002). Artistic Vision: Painterly Rendering Using Computer Vision Techniques. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '02)* (pp. 83–92).

Gooch, B., Reinhard, E., & Gooch, A. (2004). Human Facial Illustrations: Creation and Psychophysical Evaluation. *ACM Trans. on Graphics*, 23(1), 27–44.

Haeberli, P. (1990). Paint By Numbers: Abstract Image Representations. *ACM Trans. on Graphics (Proc. of ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '90))*, 24(4), 207–214.

Hausner, A. (2005). Pointillist Halftoning. In *Proc. of Computer Graphics and Imaging '05*.

Hays, J. & Essa, I. (2004). Image and Video Based Painterly Animation. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '04)* (pp. 113–120). `http://www.cc.gatech.edu/cpl/projects/artstyling/`.

Hertzmann, A. (1998). Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '98)* (pp. 453–460).

Hertzmann, A. (2001). Paint By Relaxation. In *Proc. of Computer Graphics Intl. (CGI '00)* (pp. 47–54).

Hertzmann, A. (2002). Fast Paint Texture. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '02)* (pp. 91–97).

Hertzmann, A. (2003). A Survey of Stroke Based Rendering. *IEEE Computer Graphics and Applications*, 23(4), 70–81.

Hertzmann, A. & Perlin, K. (2000). Painterly Rendering for Video and Interaction. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '00)* (pp. 7–12). `http://www.dgp.toronto.edu/~hertzman/videos/npar2000/`.

Hiller, S., Hellwig, H., & Deussen, O. (2003). Beyond Stippling - Methods for Distributing Objects on a Plane. In *Proc. of Eurographics '03* (pp. 515–522).

Horn, B. K. P. & Schunck, B. G. (1981). Determining Optical Flow. *Artificial Intelligence*, 17, 185–203.

Hoskinson, R. & Pai, D. K. (2007). Synthetic Soundscapes with Natural Grains. *Presence*, 16(1), 84–99. `http://www.cs.ubc.ca/labs/lci/naturalgrains/`.

Huang, C., Ai, A., Li, Y., & Lao, S. (2007). High-Performance Rotation Invariant Multi-view Face Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 29(4), 671–686.

Jehan, T. (2004). Event-Synchronous Music Analysis/Synthesis. In *Proc. of the Intl. Conf. on Digital Audio Effects (DAFx '04)*. `http://web.media.mit.edu/~tristan/Blog/Music_Stretching.html`.

Jones, M. J. & Rehg, J. M. (1999). Statistical Color Models with Application to Skin Detection. *Intl. Jnl. of Computer Vision (IJCV)*, 46(1), 81–96.

Kadir, T. & Brady, M. (2001). Scale, Saliency and Image Description. *Intl. Jnl. of Computer Vision (IJCV)*, 45(2), 83–105.

Kim, D., Son, M., Lee, Y., Kang, H., & Lee, S. (2008). Feature-Guided Image Stippling. *Proc. of Eurographics Symposium on Rendering '08*, 27(4).

Kingsbury, N. (2001). Complex Wavelets for Shift Invariant Analysis and Filtering of Signals. *Jnl. of Applied and Computational Harmonic Analysis*, 10, 234–253.

Kirby, R. M., Marmanis, H., & Laidlaw, D. H. (1999). Visualizing Multivalued Data from 2D Incompressible Flows Using Concepts from Painting. In *Proc. of IEEE Visualization '99* (pp. 333–340).

Klein, A. W., Sloan, P.-P. J., Finkelstein, A., & Cohen, M. F. (2002). Stylized Video Cubes. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '02)* (pp. 15–22).

Kokaram, A., Pitie, F., Dahyot, R., Rea, N., & Yeterian, S. (2005). Content Controlled Image Representation for Sports Streaming. In *Proc. of Content-Based Multimedia Indexing (CBMI '05)*.

Kokaram, A. C. (1998). *Motion Picture Restoration: Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video*, chapter 2. Springer Verlag. ISBN:3-540-76040-7.

Kokaram, A. C., Rea, N., Dahyot, R., Tekalp, M., Bouthemy, P., Gros, P., & Sezan, I. (2006). Browsing Sports Video: Trends in Sports-Related Indexing and Retrieval Work. *IEEE Signal Processing Magazine*, 23(2), 47–58.

Kudumakis, P. E. & Sander, M. B. (1993). Synthesis of Audio Signals Using the Wavelet Transform. In *Proc. of the IEE Colloquium on 'Audio DSP - Circuits and Systems'*, 219.

Litwinowitz, P. C. (1997). Processing Images and Video for an Impressionist Effect. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '97)* (pp. 407–414).

Liu, C., Torralba, A., Freeman, W., Durand, F., & Adelson, E. (2005). Motion Magnification. *ACM Trans. on Graphics*, 24(3), 519–526.

Lowe, D. (2004). Distinctive Image Features From Scale-Invariant Keypoints. *Intl. Jnl. of Computer Vision (IJCV)*, 60(2), 91–110.

Lu, L., Wenyin, L., & Zhang, H.-J. (2004). Audio Textures: Theory and Applications. *IEEE Trans. on Speech and Audio Processing*, 12(2), 156–167.

Luong, T.-Q., Seth, A., Klein, A., & Lawrence, J. (2005). Isoluminant Color Picking for Non-Photorealistic Rendering. In *Proc. of Graphics Interface '05*, 112 (pp. 233–240).

Mason, J. S. & Brand, J. (2000). A Comparative Assessment of Three Approaches to Pixel-Level Human Skin-Detection. In *Proc. of the Intl. Conf. on Pattern Recognition (ICPR '00)*, 1 (pp. 1056–1059).

McKinney, M. & Breebaart, J. (2003). Features for Audio and Music Classification. In *Proc. of the Intl. Symposium on Music Information Retrieval* (pp. 151–158).

Meier, B. (1996). Painterly Rendering for Animation. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '96)* (pp. 477–484).

Mignotte, M. (2002). Bayesian Rendering with a Non-Parametric Multiscale Prior Model. In *Proc. of the Intl. Conference on Pattern Recognition (ICPR '02)*, 1 (pp. 247–250).

Mignotte, M. (2003). Unsupervised Statistical Sketching for Non-Photorealistic Rendering Models. In *Proc. of the IEEE Intl. Conf. on Image Processing (ICIP '03)*, 3 (pp. 273–577).

Mikolajcyk, K. & Schmid, C. (2002). An Affine Invariant Interest Point Detector. In *Proc. of the Intl. Conf. on Computer Vision (ICCV '02)* (pp. 128–142).

Mikolajczyk, K., Leibe, B., & Schiele, B. (2006). Multiple Object Class Detection with a Generative Model. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '06)*, 1 (pp. 26–36).

Moorer, J. A. (1995). *New Audio Formats: A Time of Change, and a Time of Opportunity.* Technical report, Sonic Solutions, Novato, CA. Unpublished.

Moreno, P., Marin-Jiminez, M. J., Bernardino, A., Santos-Victor, J., & Blanca, N. P. (2007). A Comparative Study of Local Descriptors for Object Category Recognition: SIFT vs HMAX. In *Proc. of the Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA '07)* (pp. 515–522).

Nehab, D. & Velho, L. (2002). Multiscale Moment-based Painterly Rendering. In *Proc. of the Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 02)* (pp. 244–251).

Neto, L. S. B. & Carvalho, B. M. (2007). Message In A Bottle: Stylized Rendering of Sand Movies. In *Proc. of the Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 07)* (pp. 11–18).

Nowak, E., Jurie, F., & Triggs, B. (2006). Sampling Strategies for Bag-of-features Image Classification. In *Proc. of the European Conf. on Computer Vision (ECCV '06)* (pp. 490–503).

Opelt, A., Pinz, A., & Zisserman, A. (2006a). Fusing Shape and Appearance Information for Object Category Detection. In *Proc. of the British Machine Vision Conference (BMVC '06)*, 1 (pp. 117–126).

Opelt, A., Pinz, A., & Zisserman, A. (2006b). Incremental Learning of Object Detectors Using a Visual Shape Alphabet. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '06)*, 1 (pp. 3–10).

Osuna, E., Freund, R., & Girosi, F. (1997). Training Support Vector Machines: An Application to Face Detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '97)* (pp. 130–136).

Park, Y. & Yoon, K. (2008). Painterly Animation using Motion Maps. *Graphical Models*, 70(1–2), 1–15.

Phillips, P. J., Rauss, P. J., & Der, S. Z. (1996). *FERET (Face Recognition Technology) Recognition Algorithm Development and Test Results, Technical Report ARL-TR-995.* Technical report, Army Research Laboratory.

Phung, S. L., Bouzerdoum, A., & Chai, D. (2005). Skin Segmentation Using Color Pixel Classification: Analysis and Comparison. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 27(1), 148–154.

Ponce, J., Berg, T. L., Everingham, M., & Forsyth, D. A. (2006). Dataset Issues in Object Recognition. *Toward Category-Level Object Recognition*, (pp. 29–48).

Praun, E., Hoppe, H., Webb, M., & Finkelstein, A. (2001). Real-Time Hatching. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '01)* (pp. 579–584).

Ring, D. & Kokaram, A. C. (2009). User-Assisted Feature Correspondence Matching. In *Proc. of the IEEE Conf. on Visual Media Production (CVMP '09).*

Ring, D. & Pitié, F. (2009). Feature-Assisted Sparse to Dense Motion Estimation using Geodesic Distances. In *Proc. of the IEEE Irish Machine Vision and Image Processing Conf. (IMVIP '09).*

Roth, D., Yang, M., & Ahuja, N. (2000). A SNoW-Based Face Detector. *Advances in Neural Information Processing Systems*, 12, 855–861.

Rother, C., Kolmogorov, V., & Blake, A. (2004). "GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH '90)*, 23, 309–314.

Rowley, H., Baluja, S., & Kanade, T. (1998a). Neural Network-Based Face Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 20, 22–38.

Rowley, H., Baluja, S., & Kanade, T. (1998b). Rotation Invariant Neural Network-Based Face Detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '98).*

Salisbury, M. P., Anderson, S. E., Barzel, R., & Salesin, D. H. (1994). Interactive Pen-and-Ink Illustration. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '94)* (pp. 101–108).

Santella, A. & DeCarlo, D. (2002). Abstract Painterly Renderings Using Eye-Tracking Data. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '02)* (pp. 75–82).

Santella, A. & DeCarlo, D. (2004). Visual Interest and NPR: an Evaluation and Manifesto. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '04)* (pp. 71–150).

Scheirer, E. (1998). Tempo and Beat Analysis of Acoustic Musical Signals. *Jnl. of the Acoustical Society of America (ASA)*, 103(1), 588–601.

Schneiderman, H. & Kanade, T. (2000). A Statistical Method for 3D Object Detection Applied to Faces and Cars. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '00)* (pp. 746–751).

Secord, A. (2002). Weighted Voronoi Stippling. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '02)*.

Shiraishi, M. & Yamaguchi, Y. (2000). An Algorithm for Automatic Painterly Rendering Based on Local Source Image Approximation. In *Proc. of the Intl. Symposium on Non-Photorealistic Animation and Rendering (NPAR '00)* (pp. 53–58).

Strassmann, S. (1986). Hairy Brushes. *ACM SIGGRAPH Computer Graphics*, 20(4), 225–232.

Strobl, G., Eckel, G., & Rocchesso, D. (2006). Sound Texture Modelling: A Survey. In *Proc. of Sound and Music Computing (SMC '95)* (pp. 61–65).

Sung, K. & Poggio, T. (1998). Example-based Learning for Viewbased Face Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 20, 39–51.

Terrillon, J.-C., Shirazi, M. N., Sadek, M., Fukamachi, H., & Akamatsu, S. (2000). Invariant Face Detection with Support Vector Machines. In *Proc. of the Intl. Conf. on Pattern Recognition (ICPR '00)*, 4 (pp. 210–217).

Vanderhaeghe, D., Barla, P., Thollot, J., & Sillion, F. X. (2007). Dynamic Point Distribution for Stroke-Based Rendering. In *Proc. of Eurographics Symposium on Rendering '07*.

Vedaldi, A. (2006). SIFT for Matlab. `http://www.vlfeat.org/~vedaldi/code/sift.html`.

Vezhnevets, A. & Vezhnevets, V. (2005). 'Modest AdaBoost' Teaching Adaboost to Generalize Better. In *Proc. of Graphicon* (pp. 320–325). `http://research.graphicon.ru/machine-learning/gml-adaboost-matlab-toolbox.html`.

Vezhnevets, V., Sazonov, V., & Andreeva, A. (2003). A Survey on Pixel-Based Skin Color Detection Techniques. In *Proc. of Graphicon* (pp. 85–92).

Viola, P. & Jones, M. J. (2004). Robust Real-Time Face Detection. *Intl. Jnl. of Computer Vision (IJCV)*, 57(2), 137–154.

Wang, J., Drucker, S., Agrawala, M., & Cohen, M. (2006). The Cartoon Animation Filter. In *Proc. of ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '06)* (pp. 1169–1173).

Wang, J., Xu, Y., Shum, H.-Y., & Cohen, M. F. (2004). Video Tooning. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH '04)*, 23(3), 574–583.

Wei, Y.-L. & Levoy, M. (2000). Fast Texture Synthesis Using Tree-Structured Vector Quantization. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '00)* (pp. 479–488).

Winkenbach, G. & Salesin, D. H. (1994). Computer Generated Pen-and-Ink Illustration. In *Proc. of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH '94)* (pp. 91–100).

Winnemoller, H., Olsen, S. V., & Gooch, B. (2006). Real Time Video Abstraction. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH '06)*, 25(3), 1221–1226.

Xiao, R., Li, M.-J., & Zhang, H.-J. (2004). Robust Multipose Face Detection in Images. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(1).

Yang, H. L. & Yang, C. K. (2006). A Non-Photorealistic Rendering of Seurat's Pointillism. In *Proc. of the Intl. Symposium on Visual Computing (ISVC '06)*, 4292 (pp. 760–769).

Yang, M.-H., Kriegman, D. J., & Ahuja, N. (2002). Detecting Faces in Images: A Survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 24(1), 34–58.

Zhang, W., Yu, B., & Dimitris, Z. (2005). Object Class Recognition Using Multiple Layer Boosting with Heterogeneous Features. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '03)*, 2 (pp. 323–330).

Zhao, H., Jin, X., Shen, J., Mao, X., & Feng, J. (2008). Real-Time Feature-Aware Video Abstraction. *The Visual Computer*, 24(7), 727–734.