The Art of Image Editing

J. Andrew Bangham,
Andrew Courtenay,
Richard Harvey,
Don McCrae

Contents

Algorithms

5

Creative photos

6

Algorithms for Art

7

Shape

8

Contents

Technology Review of Fo2PiX Development of an image editor and underlying algorithms.

Prof. J. Andrew Bangham
Technical Director, Fo2PiX. Concepts, computer vision research and 'C' coding of the PictureEngine.

Andy Courtenay
Senior developer. Program design and Java coding.

Dr. Richard Harvey
Scientist. Computer vision research into the properties opportunities for the sieve algorithms.

Don McCrae
Chief Executive Officer

Outline of this document

Introduction

The aim of this document is to provide a technical insight into how the Image Comprehension Environment, ICE, works. (Never mind the name the acronym is short and . . .)

The underpinning computer vision algorithms and structures are key for the next generation of image editors and, for image retrieval in two important ways. Their value has been established in other applications and reported in journals ranging from 'IEEE PAMI' (see Chapter 5) to 'Science' (see Chapter 8).

Innovative algorithms: image understanding and retrieval.

A fundamental flaw in existing image editing is: what you see is what you get. But later, can you or your collaborator see how?

Innovative architecture: for slipping seamlessly from editing to creative art this software is peerless.

First and foremost the software described here is designed to address this problem and to make interacting with images easy.

It embodies ideas 'worked up' for *extreme editing*: producing art from photographs. By not competing head on with big name image editors it has been possible to test the ideas on the market.

Innovative structures: Track-changes, ImageDocuments, ImageWizards, blend pipeline and more.

It opens the way to the development of a language of image editing that will raise standards, improve product stability and increase the number of users.

Research: language of editing images.

.

The following table provide guidance on page numbers.

Know how ⇒	**Algorithms** **Sieves**			**Shape**	**GUI (PlainSight)**				**Interface**
Application ⇓	1D	2D	Trees	Model	Blend	Pipe	imDoc	Lang.	
Image editors									
Image editor that slips into art.	7-3	7-3		5-9 8-2		4-16	4-11	3-14	4-3
Re-editing the ImageDocument.						4-16	4-11	3-14	4-3
Object editing edit lists.	5-2	5-2			4-16	4-16	4-11	3-14	4-3
Computer Vision and finding things									
Metadata for indexing images.						4-16	4-11	3-14	
Recognition of feature points.	7-3	7-3	5-9		4-16				
Retrieval for finding images by content.	7-3	7-3	5-9		4-16				

Figure 2.1: Quick guide showing Section-Page numbers relating know-how to applications. New developments are outlined in Figure 4.23

ICE and ImageDocuments

Figure 3.1: *What you see is what you get. But later, can you or your collaboration see how?*

Why ImageDocuments?

Figure 3.2: *Created, how?*

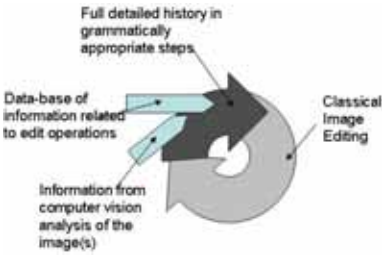


Figure 3.3: *The virtuous circle is completed when people can share the editing of images.*

Popularising the language of image editing is create a fantastic commercial opportunity.

Editing improves images. But once improved, can you remember how and explain it to someone else?

Fundamental flaw This is a fundamental flaw in existing image editing systems. A user has access to the information embodied in the final image, but the information representing the process by which that image was achieved is either lost or unclear.

The evidence from word processing and vector graphics, Page 3-4, is that doing more is better and better is good commercial sense. Somehow, the sequence of image edits should be accessible and editable.

Could the problem of editing images become closer to that of editing narratives? Unlike wordprocessing and technical drawing there was, historically, no culture of widespread sharing of picture editing that developed a language of image processing that would have provided a natural framework for the design of image editing software. Beyond the rubber eraser[1], image editing was rudimentary and software designers had to go forward boldly on their own.

Doing more, therefore, means creating a language for image editing that encourages people to talk about what they have done, share their ideas and generally enrich the creative process. People need to make an edit, have a clear view of what they have done, where they did it, preferably why, exploit useful computer vision algorithms and, critically, easily revisit their work. They need to let someone else go around the editing loop keeping the best and re-editing the rest.

Not having this loop could be inhibiting people from working

[1] 1770, Edward Naime

easily with images. It could be inhibiting the deployment of computer vision and language processing algorithms that would otherwise enable steady improvements in quality, thereby capturing increasing numbers of users and consolidating loyalty.

Against a background of huge numbers of digital camera owners who are starting to look for the next thing to do and computers that are for the first time fast enough (cpu and gpu) to handle images properly. It is now timely to create a new generation of image editor.

So, just how could a colleague change the title on the box, Figure 3.2?

The Approach used in ICE as the Solution

The main aspects the ICE interface dynamics have been tested by thousands of customers. ICE is used as an example because without visibly revealing the underlying mechanisms, ICE based systems are appreciated by independent reviewers and users who notice the difference[2] The GUI will be slicker and more attractive when written in an operating system dependent language: Version 2.

ICE is a tidy source of innovative ideas that has demonstrated that they, the structures and the interfaces are not only feasible, they are practical and effective.

There are four parts to a description of the solution. Firstly, the problem is put into the context of word and vector graphic editing, Page 3-4. Secondly, a solution illustrated with examples of editing and navigating within ICE, Page 3-10. Thirdly, consequences of using the concepts in ICE can be extended to produce a complete system, Page 3-12. Finally, some very attractive new horizons can be seen from the shoulders of ICE, Page 3-14.

What follows could sound a little wild without first reading, or constantly referring to, the remaining chapters of this document.

[2]Tom Arah, Painting by Numbers, in PC PRO, April 2006, pp203:205, Dennis Publishing Ltd, London, reported

- "...combined with layer-based and history based compositing, Photoshop reveals some considerable power However to make the most of it you need to be an expert and, even then, it's very much a process of creative experimentation and trial and error.
- "...What we really want is built on these same principles, but designed from the ground up ...

"...I admit that I've been won over by ArtMaster Pro and the use of ArtWizards in particular. The sheer range of creative choices that Fo2PiX's Source-based approach unleashes, combined with the infinite range of ways in which they can be applied, means that you really need some kind of ever-present guide.

"Moreover, you don't actually feel restricted by the ArtWizards in practice, you can quickly, confidently and consistently give any photograph a particular type of artistic feel, while retaining creative input and ultimate creative control."

Context: Taming Image Editing

Those who do not want to read how ImageDocuments are a natural evolution from word processing, vector graphics and existing image editors, please skip directly to Page 3-6 after a quick look at Page 3.1.

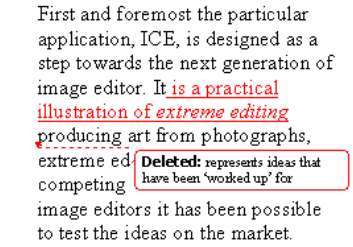


Figure 3.4: *Tracking changes in word processing makes collaboration on documents easier.*

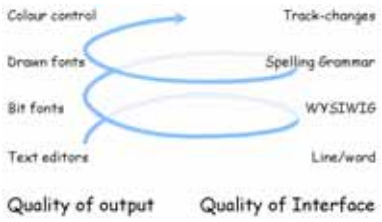


Figure 3.5: *Slowly improving software.*

Setting the scene with word processing The advent of desktop publishing sparked a revolution in the way people worked and played. Inspired by SmallTalk [48] and enabled by laser printers, software designers produced windowed user interfaces, what-you-see-is-what-you-get (WYSIWIG) displays and page oriented layout languages [84]. Graphical software and page layout programs became the standard for word processing and presentation. Programming effort was required in two areas, producing good results and producing a good human computer interface. Tools for organising complex documents into manageable hierarchies were developed. Chapters, headings, figure legends, etc. provide a familiar way to break up documents, layouts and create working templates. Prose also has a natural grammar and that can be exploited by intelligent programs to assist with spelling and grammar. Further improvements were achieved by adding time-stamps to the hierarchy of word, sentence, format, and other objects, so providing a framework for tracking changes and the better sharing of the editing process. Attention shifts around and around between function and useability. It is not a circle, it is an upward spiral, Figure 3.5 that captures increasing numbers of users and consolidates loyalty.

Improved working environments bring other advantages. For example, by making it easy to share documents there has been growth in the availability of worked examples that help people to do things that they would, in the past, have never attempted: templates, boilerplate contracts, pattern essays, CV's, research papers, presentations and articles.

Computing environments for word processing are changing the way people discuss and work with written documents for the better.

With input from vector graphics Computers impacted vector drawing very early [3]. Here, the importance of organising complex work into manageable hierarchies is even more apparent. The

[3] circa 1960, Ivan Sutherland, 'SKETCHPAD', MIT Lincoln Laboratory. Also, 'The Electronic Drafting Machine' by ITEK and 'Design Automated by Computer' by Hanratty at General Motors.

grouping together of lines, curves and shapes into objects is not only physically meaningful, literally the nuts and bolts of engineering, but exactly reflects what is now considered to be good progamming practice. Hierarchical object trees provide a perceptually meaningful way to navigate the work. Objects such as windows, cars and trees are accumulated into libraries to be readily selected, used and adapted. The vocabularies and ways of talking about engineering and architecture reflect the hierarchies and grammars of the data structures. It is not surprising then to find programming practices used in engineering and architectural systems being adopted in vector illustration and presentation programs. Nor is it surprising to find that they not only improve peoples output and working practices but even improve the way they think about the work.

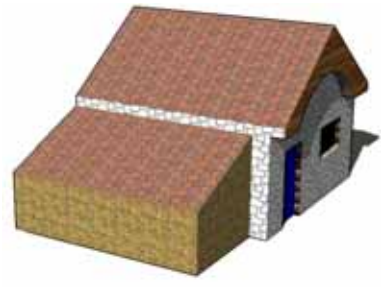


Figure 3.6: *SketchUp demonstration of vector graphics.*

Once again sharing, editing and re-editing are the norm and undoubtably we will soon see the structure exploited in the equivalent of 'track changes' and intelligent tools that check and offer advice on the 'grammar' of the object hierarchies.

Image editors are harder In some ways the development of digital bit image processing has been harder. Paint packages were developed early [4] [75]. Pioneering and useful but you need to be able to paint or at least sketch. Digital images were rare, they had to be scanned. Photoshop was first bundled with a slide scanner. The development of image editing programs then followed the same virtuous circle as word processing, Figure 3.5. It is, however, less obvious how the program should be structured. Images comprise many objects but they are captured as a single, flat, set of pixels. Current systems allow users to adjust, superimpose, blend, mask, adjust tones, balance, crop, blend images and perform an ever increasing number of filter effects. The introduction of 'Layers' and methods for selecting sets of pixels enables a structure to be created manually and 'Actions' to provide some level of automation. The manner in which image editing systems permit users to apply these changes have become very powerful but have remained largely unchanged for several years.

Figure 3.7: *Images are just pixels. What's interesting? The boats, just boat number 7, the topsails, the helmsmen, that they are tacking, racing or that the colour balance is wrong: too blue.*

[4] Richard Shoup, 'Superpaint', Xerox Palo Alto Research Center, 1975

A short introduction to using ICE

For those who have not tried one of the applications derived from ICE, what follows is a short introduction to image editing using ArtMasterPro. To skip on go to Page 3-10.

Starting An image is opened in 'Adjust' and moved directly to the 'Studio' shown in Figure 3.8. There are two ways to load the image into the Canvas, using the manual controls or using an ArtWizard (more generally called ImageWizard). Since the image is a portrait, the 'Portraits:Photo Adjust:Load image to Canvas' ArtWizard has been used (highlighted blue).

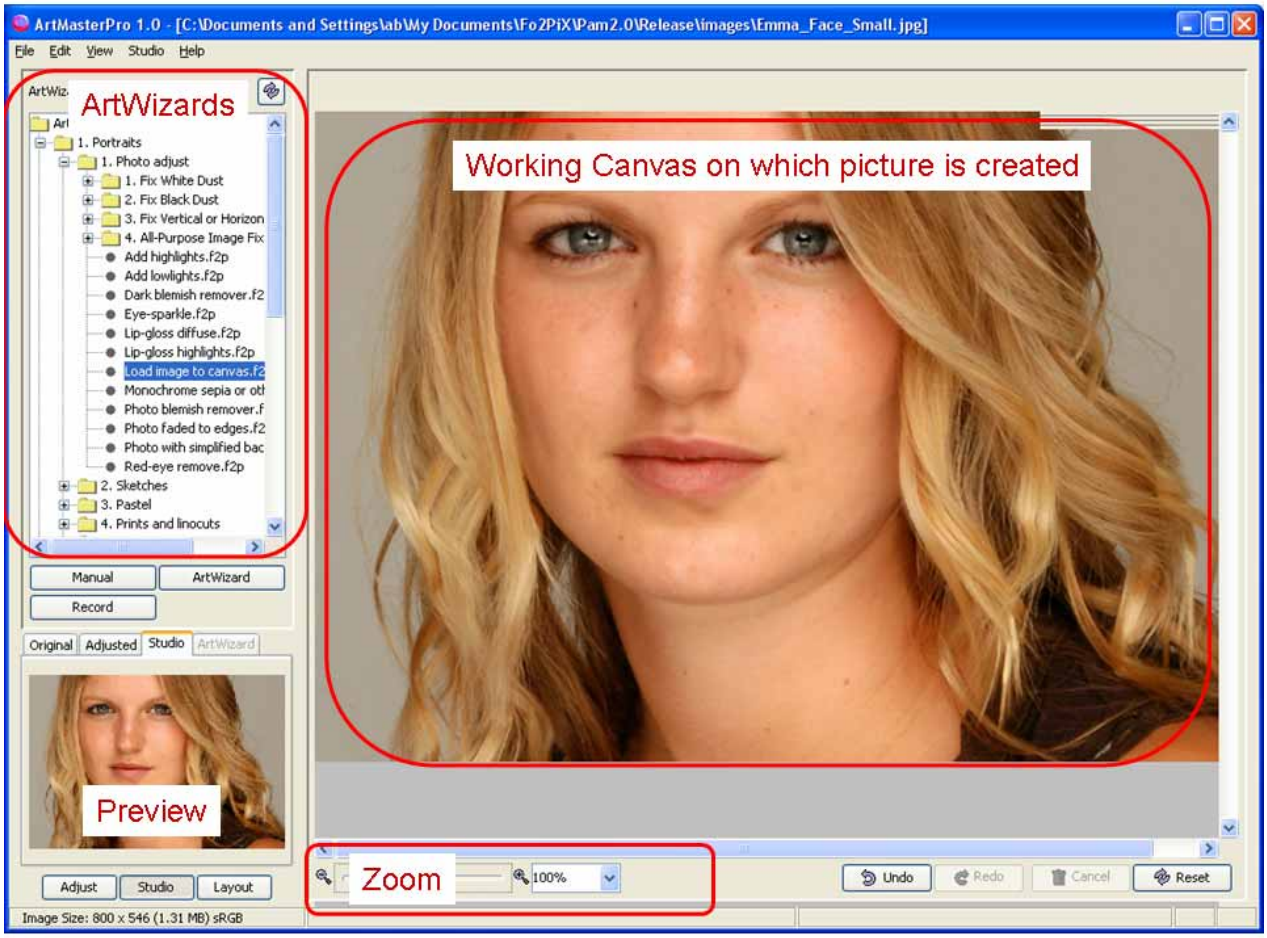


Figure 3.8: *Screenshot of the ArtMasterPro on entering the Studio with an image of Emma. The top left panel shows the ArtWizard library. An ArtWizard has been run that loads the image into the working Canvas. Note (if you can resolve it) that the ArtWizard is in the 'Portraits:Photo Adjust' section of the library, see Figure 3.9. In the course of this example two further ones will be run, 'Eye Sparkle' and 'Add Highlights'.*

Adding sparkle to the eyes The goal is to add sparkle to the eyes and highlights to the hair (both exaggerated for clearity) so the 'Eye Sparkle' ArtWizard is selected and run, Figure 3.10. This shows that the ArtWizard stops when the sparkle is to be brushed over the eyes. Optionally, an information 'Strap' drops down from the top right of the Canvas explaining what to do next and why. The highlights are obtained from a Source. The Sources are special 'layers' derived from the original image, see Page 4-4. Highlights are regionally light areas and the Tab containing Highlights has a set of which the first 6 are of increasing area. Thus the sparkle comes from small, real, highlights that were faintly present in the original image (they are reflections of the studio lights).

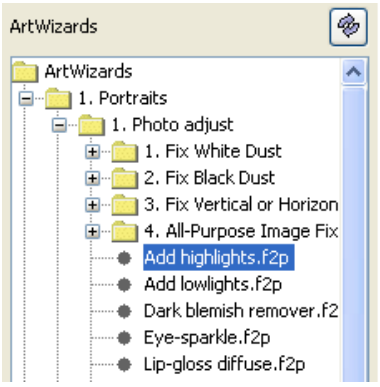


Figure 3.9: *ArtWizards library.*

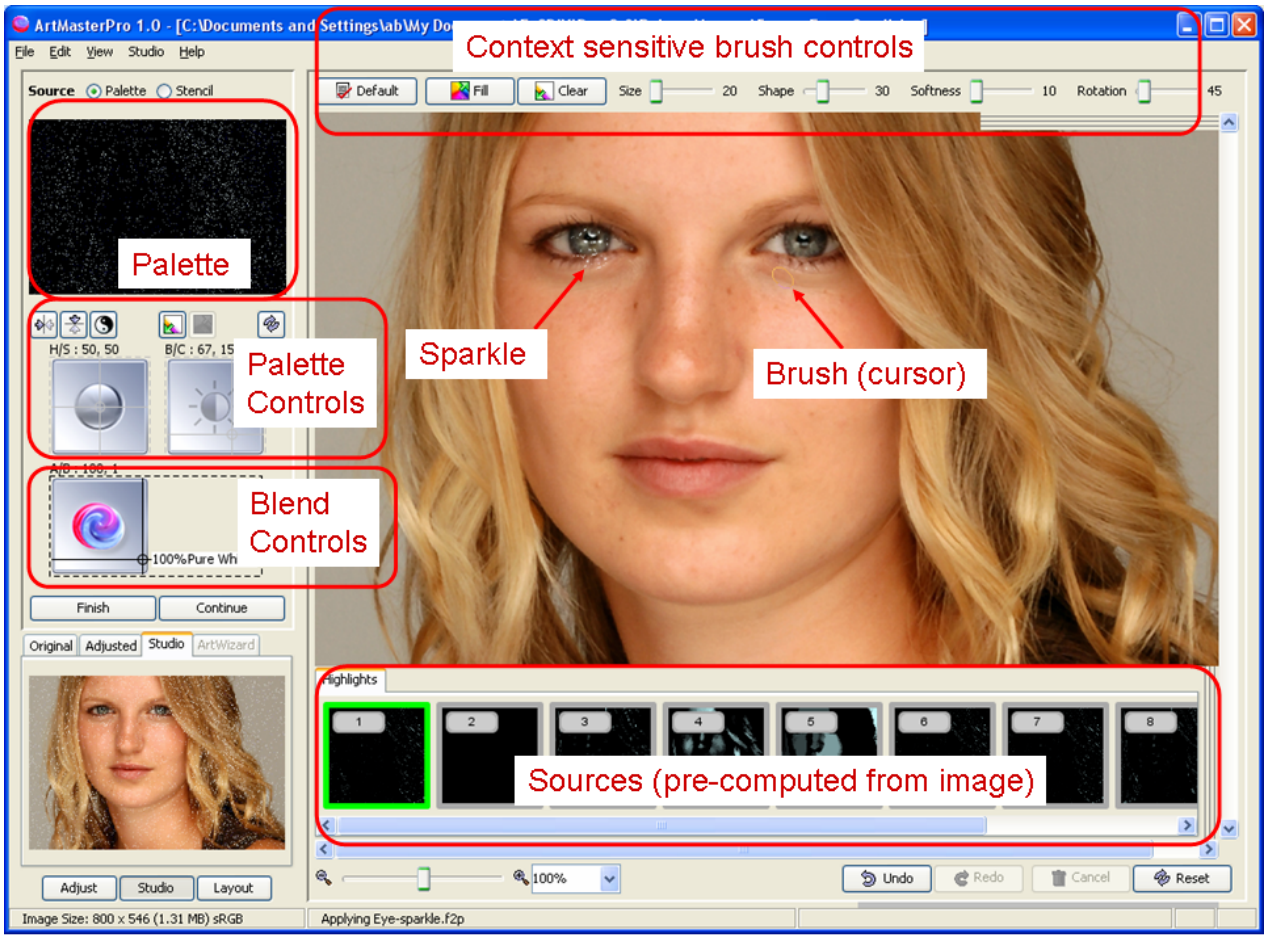


Figure 3.10: *When executing the ArtWizard 'Eye Sparkle' the script stops and invites the sparkle to be brushed into place using the mouse. Here, one eye has been done and the (faint) cursor is next to the other. Context sensitive help is available from a 'strap' that pulls out of the top right of the Canvas (not shown).*

Once selected the thumbnail Sources are recomputed at full size and previewed in the Palette where they are under the control of two dimensional brightness/contrast and hue/saturation pickers (2D sliders).

Adding hair highlights Figure 3.10 shows larger scale highlights being brushed onto the hair. The bottom left preview panel clearly shows all the highlights blended onto the Canvas. Only some are brushed onto the Canvas itself. The ArtWizard selects the appropriate Source and presets all the blend, brush and Palette controls appropriately. However, they they can all be changed. Changes can be undone/redone and aborted.

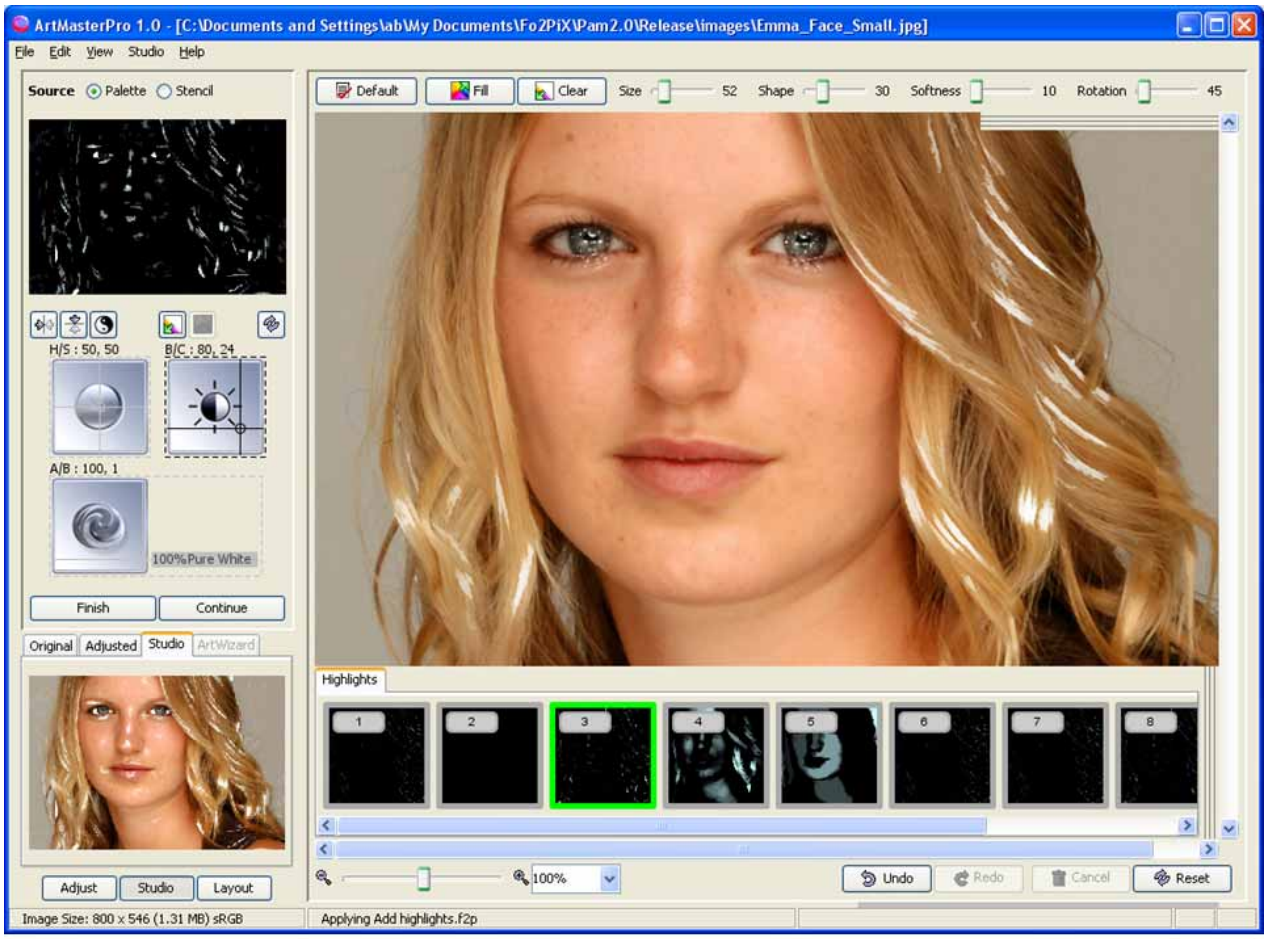


Figure 3.11: *Adding large scale highlights to the hair.*

It is not essential to use ArtWizards although it is much the easiest way to get going. Indeed, they can be treated as tutorials.

Removing skin blemishes without an ArtWizard The alternative is to switch to Manual mode, in which case all controls and 700 Sources are accessible.

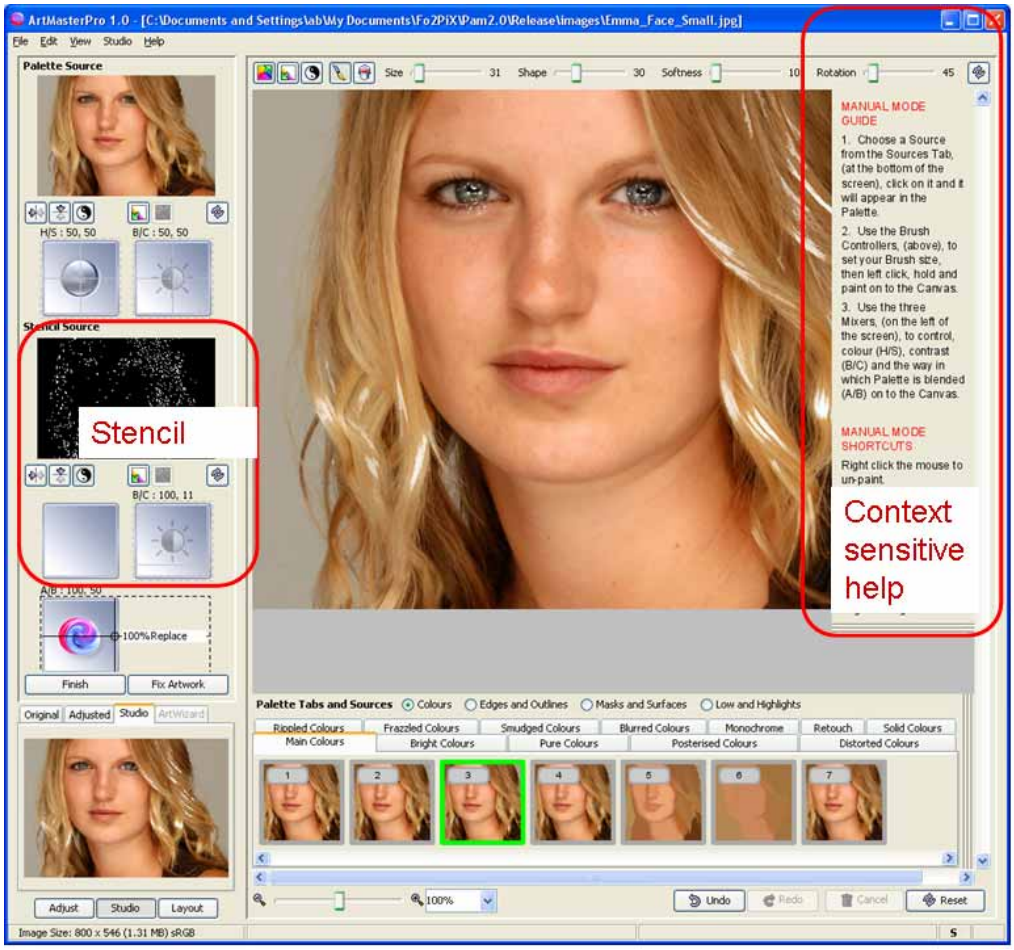


Figure 3.12: *Manual mode. Running on a large screen it is convenient to have the Stencil fully revealed. In manual mode the context sensitive is generic but in an ArtWizard it is specific to each step in each ArtWizard. The Stencil is an α-channel governing the blending of the Palette with the Canvas, see Page 4-16. In this case lowlights of the size of facial moles has been selected and inverted to provide a mask that selects the moles, see Page 5-5. The Palette contains a sieved copy of the image (spots removed). As result, one can brush out the moles. The ArtWizard 'Dark blemish remover' does all this providing help as it goes.*

Simple? 'Creata' is an even simpler, cheap application, derived from ICE that only uses ArtWizards. It is sold with five and new ones are downloadable for a small charge.

Given this short introduction it is now appropriate to examine how ICE addresses the more difficult problem of treating image editing more like word processing.

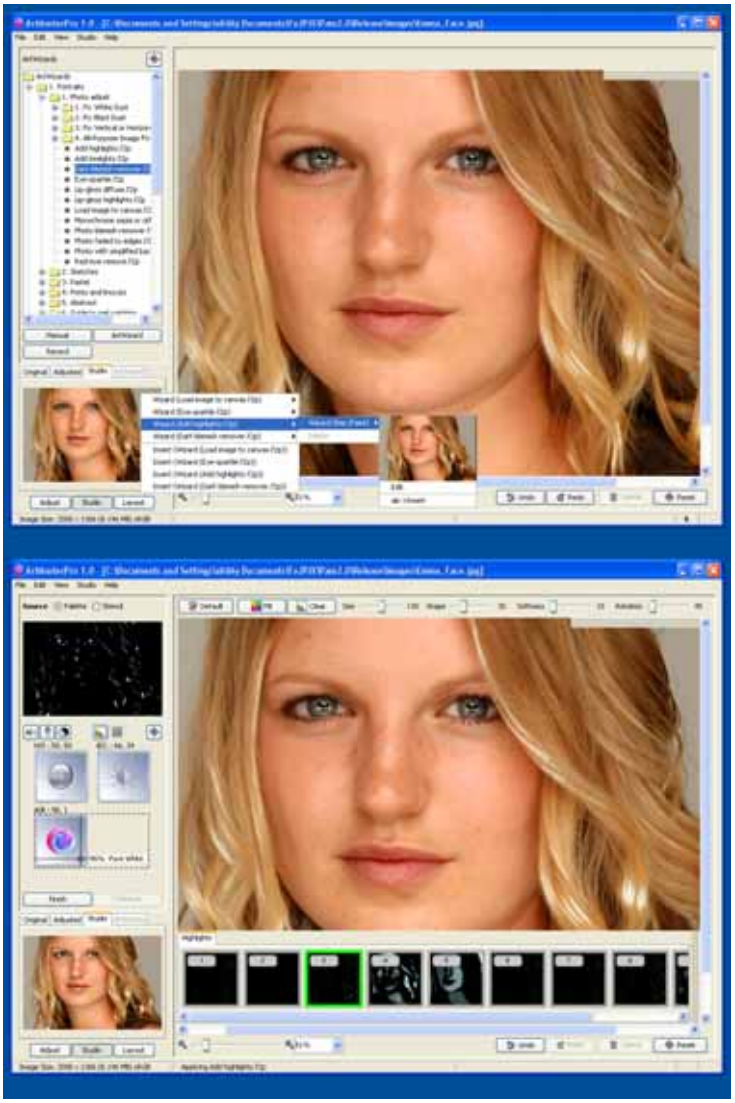


Figure 3.13: *Four ImageWizards were used to produce the top panel. Right clicking on the bottom left panel pops out the history and enables the third ImageWizard to be selected for re-editing. The thumbnail provides a visual reminder of the Canvas at that moment. Bottom panel shows the hair highlights being emphasised more heavily. Clicking Finish causes the final result to be recomputed with the new highlights. Note, the dark skin blemishes have re-appeared. They were removed by the next ImageWizard whose steps will be re-applied when the edit is finished.*

A solution illustrated with ICE

Example of navigating and editing an ImageDocument Typically, ICE is loaded with a set of extracts from the original image, around 700 hierarchically catalogued Sources and with a set of hierarchically catalogued ImageWizards (ArtWizards in ArtMaster Pro). The fact that both are well organised will prove to be important later on.

Consider the image shown in Figure 3.1. The original photograph has been improved by running and interacting with a number of ImageWizards.

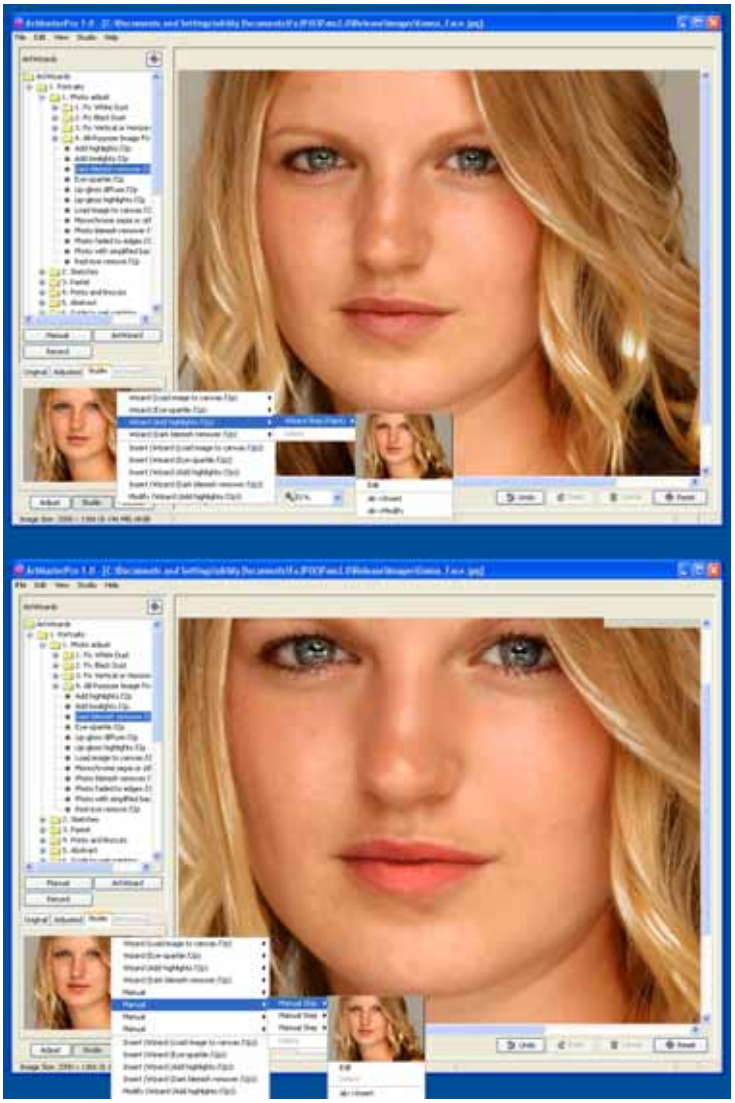


Figure 3.14: *Top panel show the re-edited image. The ImageWizard for removing dark blemishes has been re-applied automatically. Note that it has been recorded that 'ab' did the re-edit. Lower panel, Manual mode has been used by 'ab' to add lipstick, but what else? With ICE it is possible to go back and find out. ImageWizards are labelled, easy to use and people can build and share their own library.*

Figure 3.9 illustrates the library of ImageWizards. The user interacted with the image within a selection of of these ImageWizards and, because the ImageDocument is a complete record of interactions, any of the interactions can be re-edited, see Figure 3.13 and 3.14. The hair highlights on the right of the image were too strong compared to the left. The highlighting is, therefore, being changed by re-editing the appropriate step. ICE allows any interaction, including edits themselves, to be re-edited, augmented or deleted. This is a significant advance. It means that, like a word document, it can be created, changed, saved, re-opened and changed again.

Consequences of ICE: New ways of Navigating Image Editors

The Blend pipeline leads to a sequence of steps that have the grammar of a natural language.

Although pixels are viewed in parallel they are worked on *sequentially*. Books and magazines are full of advice on how to edit images, illustrate, draw and paint. They explain useful sequences of operations and the underlying principles. They demonstrate a rhythm and grammar associated with working with images. When explaining how they work, experts will often start by saying 'Well, I usually start by doing this and then I do that ', in other words there is a pattern in what they do. Can this pattern be turned to advantage?

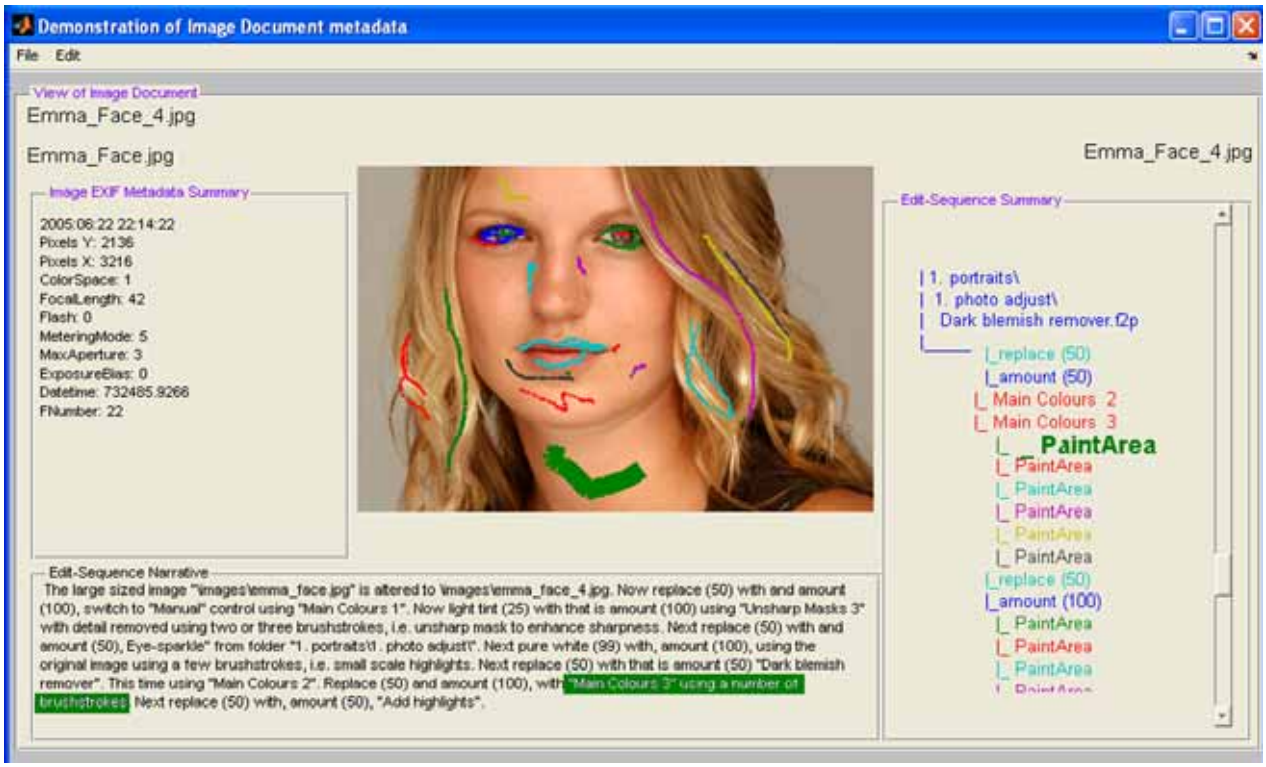


Figure 3.15: *Novel ways to navigate an ImageDocument. The Edit-Sequence Narrative, bottom panel, includes phrases like "The large sized image . . . ", ". . . Now light tint . . . ", ". . . with detail removed using two or three brushstrokes, i.e. unsharp mask . . . ", etc. Clicking on a phrase in the Narrative or a line in the XML ImageDocument summary selects an image editing step. A third alternative is to right-click over a brush mark and select the appropriate step. (Experiment implemented in Matlab).*

Yes, as soon as the sequences were captured it became clear that they resemble other natural languages. It is a rewarding exercise for even just seeing the sequences makes it easier to understand how to work with images. It is this observation that lead to the

Blend pipeline Page 4-16 and sequence of 'steps' Page 4-14.

Note, that it is a characteristic of ICE that the brushes might have been quite wide (because brushing does not 'paint' on the image it modulates an α-channel, Page 4-16).

Feasibility The sequences are explored experimentally with the help of a Matlab demonstration program, Figure 3.15. The centre panel shows the final image. It has been overlaid with lines showing brush marks made during the editing process. The right hand panel shows a summary of the resulting XML ImageDocument created by ICE. (It could be a full fledged XML viewer.) The bottom panel shows a text output. (It could be a full fledged word processor.) It contains a narrative that has been automatically generated from the XML. This is possible because the edit sequence has a natural grammar that can be (loosely) mapped into English. The language generator could/should be improved but this simple implementation makes the point.

There is a 'loose' mapping between image editing sequences and a narrative.

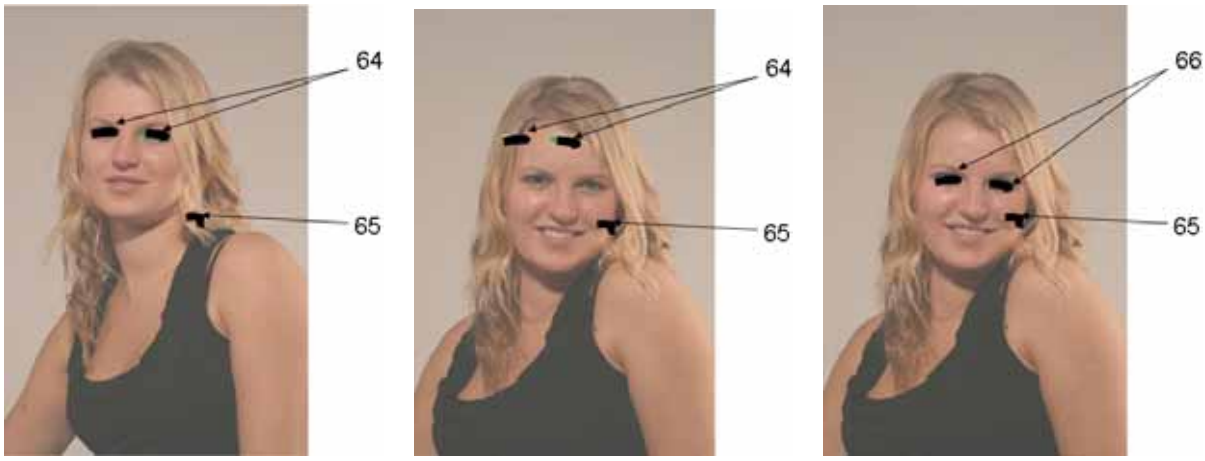


Figure 3.16: *Two frames from a photoshoot. Sparkle was added to eyes in A (64) and a small highlight added to the hair (65). Applying the entire ImageDocument to another image in the photoshoot will work, but only if the eyes are in the same place. In B they are in the wrong place. However, a simple matching algorithm is sufficient to realign the eye effect (region 65 acts as a control).*

Thus Figure 3.15 shows three new ways of navigating the ImageDocument. The first is redolent of selecting Layers in Photoshop. Right-clicking on a brush mark causes the associated elements in the XML sequence and the English narrative to be highlighted (the colour coding helps show the equivalent edit steps). Likewise, the XML document and the English narrative can both be hyperlinked to the brush marks and the associated editor interaction.

Owning the first translator between English narrative and image editing steps will produce a significant commercial advantage.

The first XML ImageDocument to English narrative translator will define the vocabulary of image editing.

Creating the narrative On first examination the narrative looks surprisingly rich given that it was generated from the ImageDocuments. *It is here, however, that the full power of the ICE GUI comes into play.*

Phrases like "The large sized image . . ." is generated from the record of the image filename and the EXIF information. ". . . Now light tint . . ." comes from the BlendPicker setting, ". . . with detail removed using two or three brushstrokes, i.e. unsharp mask . . ." comes from the identity of the Palette Source and details of the interaction.

Clearly ICE has captured lots of metadata. The ImageWizard path and name tells us that the photograph is a Portrait. A combination of ImageWizard name and where the user brushed over eyes, mouth and hair identifies the position of the face. In other words, the combination of the libraries of Sources and ImageWizards and interactions with the image defines 'phrases' of image editing and by increasing the size of both, the language can be made ever richer.

At the moment these image editing related 'words' and 'phrases' are selected from simple hierarchies of Sources and ImageWizards but there are many opportunities to take it further, e.g. probabilistic methods similar to predictive texting.

Dragging Edits around the Image Figure 3.16 shows two photographs from a single shoot. It is not uncommon to improve one only to decide that another would be a better choice. There is an option to apply the ImageDocument to the other image. But the model or photographer might have moved.

Given the rich metadata there are then several ways to realign the changes with the new image. One option is to lasso the edits (as opposed to the regions) and drag them into place. Smarter, would be to have a tool that uses a computer vision algorithm to matches the regions under each change and looks for the closest matched regions in the new image. This is very likely to work well since, within a photoshoot, the images are very similar. Indeed, that was the method used in Figure 3.16.

New horizons: Metadata for Photo organisation and object retrieval

A major problem with computer vision, language and image retrieval algorithms lies in the diversity of possible images. Searching and analysing images on the web is hard, see Page 5-3. The first step then is to divide the problem and conquor. Most people tend

to take photographs of a limited number of subjects in a limited number of ways: they have their own recognisable style. Once, they have edited, interacted with, a selection of their images the resulting collection of ImageDocuments contains a considerable amount of detailed information about their world. From this it is possible to exploit computer vision and language algorithms to set smart 'personalisation parameters' for image editors and image retrieval systems.

Bootstrapping image recognition algorithms Through the mechanisms illustrated in Figure 3.16 the edit sequence can be used to bootstrap software programs or algorithms. For example, if the image editor automatically searched the edit sequence information when an image document was loaded in the viewer, information associated with eyes for example, could bootstrap other programs or algorithms. Thus, the presence of the words 'eye highlights' at a high level of the edit sequence tree and detailed brush stroke information at lower levels of the 'eye highlight' branch when associated by the system with brush and other interactions with objects in the image could be used as training data for a computer vision algorithm to find faces and eyes in an image.

Figure 3.17: *A simple ImageDocument and image browser, click the bottom '< Previous' and 'Next >' buttons to change image. The centre image has 4 derivative ImageDocuments and these are readily browsed by clicking the upper 'Previous' and 'Next' buttons.*

Training content based image crawlers The edit-sequence brings other advantages. For example, a combination of the name 'eye highlights' and the brush movements can 'bootstrap' computer vision algorithms for face finding and recognising similar faces. This can be used to empower 'image crawlers' as they index images by content.

Access to photographs and pictures through a database Each image, and ImageDocument, can be registered into a browsable database, e.g. Figure 3.17, nodes of which are ImageDocuments. An unedited image is the simplest ImageDocument. The browser then allows the user to browse derivative (and sibling) results as easily as the origionals (c.f. Versioning in Aperture).

It will also be possible to search the database for results, . . . "I can't remember the name but know I did . . . and it was cool".

Edit-sequences for image retrieval The Sequence Summary in Figure 5 contains meaningful wizard names. They provide strong clues about image content, here the metadata clearly indicates that the image contains a face. The associated brushing differentiates foreground from background and so forth. Importantly, this metadata was added to the image without the user having to explicitly add labels and it is immediately accessible to image retrieval engines. In other words it automatically contributes to a 'folksonomy'. Of course, it can be augmented by manually added labels such as names. Add a name to the ImageDocument associated with Figure 3.1 and the narrative can mention that Emma is blond with blue/gray eyes.

Non-destructive editing An ICE ImageDocument completely specifies the changes from one image to another. The data structures and design mean that ICE could become become a non-destructive editor. It would need suitably fast, GPU based, image processing algorithms and the Java front end replaced by re-implementing in an operating specific language. (It is relatively easy to move from the Object Oriented language Java and C++ or similar.)

To be done

Full details of the how the ideas developed in this Chapter are integrated into the ICE software and how they will develop are given in Chapters 4. Research that has already influenced the development of the system are given in Chapters 7 and 5.

Software Architecture

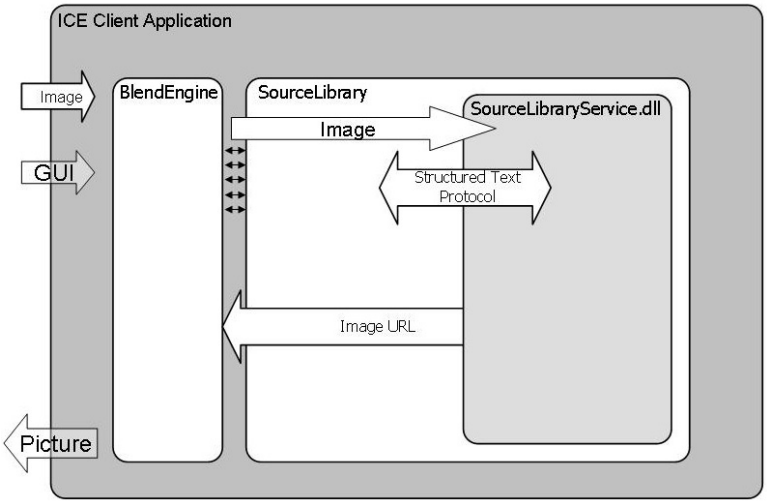


Figure 4.1: *ICE configured as a standalone client application. The image processor (coded in 'C') serves a library of filtered images (Sources) to the ImageEditor (coded in Java) through a structured text interface. The ImageEditor itself comprises a Graphical User Interface (GUI) wrapper for the BlendEngine.*

Components and GUI

The PictureEngine provides over 700 *variants of the original image from which pictures are constructed. They represent an increasingly large, organised,* vocabulary *of picture making elements.*

The design of image editors has been refined and the ideas have been tested in commercial applications derived from the Fo2PiX Image Comprehension Environment, ICE.

What follows is a description of the ideas presented in the context of ICE. By working in Java, the ideas and the associated data structures are couched in a form that provides an clear starting point for operating system specific implementations (C++, Objective-C and .net). It works across multiple platforms but Java on the Mac lags that on the PC.

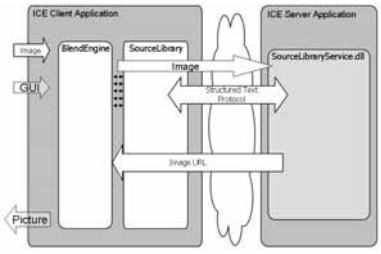


Figure 4.2: *ICE configured as a client/server communicating through structured text where images are accessed through URL's.*

Architecture and image processor

The overall structure of ICE is shown in Figure 4.1. The Graphical User Interface (GUI) and associated BlendEngine together form the ImageEditor that depends on services from the PictureEngine. The interface between the two uses structured text through a single entry point. This has advantages, for example, all changes can be 'soft' allowing multiple generations of the interface to work simultaneously making team programming easier. More importantly the system can also configured as a Client-Server, see Figure 4.2.

Dialogue between client and server During launch the PictureEngine reports a list of services, currently about 700 variants of the input image in multiple sizes. In addition, it provides both user-friendly names and information on how the different Sources

```
ImageEditor request: startprocess=[no] want_picture=[z_0\x0\1_P_74_]
PictureEngine action: perform_action(0,0,,0,,,92,  80)
ImageEditor request: collect_picture
PictureEngine action: perform_action(0,0,,1,,z_0\x0\1_P_74_, 92,80)
no file on disc named z_0/x0/1_P_74_.bmp  so force computation
>>>start computing z_0\x0\1_P_74_
no file on disc named z_0/work2.bmp  so force computation
>>>start computing z_0\work2
perform_action(0,-1,originalwork,1420,,z_0\work2, 92,80)
<<< z_0\work2 .... accumulated secs 0.80 .. i.e. 0.03 cpu secs
perform_action(0,-1,,1,, z_0\x0\1_P_74_, 92,80)
<<< z_0\x0\1_P_74_ accumulated secs 1.03 .. i.e. 0.16 cpu secs
writing z_0\x0\1_P_74_.bmp
ImageEditor request: startprocess=[no] want_picture=[z_0\x0\1_P_75_]
PictureEngine action: perform_action(0,0,,0,,,92,80)
ImageEditor request: collect_picture
PictureEngine reply: perform_action(0,0,,1,, z_0\x0\1_P_75_, 92,80)
no file on disc named z_0/x0/1_P_75_.bmp  so force computation
>>> start computing z_0\x0\1_P_75_
same name so no need to reload z_0\work2.bmp
<<< z_0\x0\1_P_75_ accumulated secs 1.08 .. i.e. 0.05 cpu secs
writing z_0\x0\1_P_75_.bmp
```

Figure 4.4: *Summary of an extract from a dialogue between the ImageEditor and the PictureEngine. Time consuming results are cached on disc. This is particularly effective where many outputs from a single algorithm can be computed in a single pass, e.g. the sieve. In future, hardware acceleration will enable on-the-fly image processing and caching will be bypassed: 'Non-destructive editing'.*

should be categorised. This enables the user interface to be automatically configured.

A sample of the dialogue between the BlendEngine and the PictureEngine, Figure 4.4, illustrates the operation of the PictureEngine. The BlendEngine requests a Source. Then, after it receives an acknowledgement, it waits to collect the result. Meanwhile, debug output shows that the PictureEngine initially looks for the image in disc cache. However, in this example neither it nor a pre-requisite image, 'work2', is available from cache therefore, the request is stacked. Only when 'work2' has been computed is the initial request popped, computed (0.16 cpu seconds) and its' URL returned. The next Source requested by the BlendEngine is also not in cache. But the pre-requisite 'work2' image is in cache and this reduces the overall computing time to 0.05 cpu seconds.

It can be seen that, when asked for a Source, the PictureEngine recurses down through dependencies either loading from cache or computing and caching as it goes. A session, therefore, gets faster as more of the 'vocabulary' of Sources have been computed. This is important because many of the image Sources provided by the

Figure 4.3: *A recurring artistic theme is to simplify the image to remove detail then ease the hard photographic edges by warping using x, y offsets obtained from a smoothed random noise image. The effect is exaggerated in this illustration.*

PictureEngine are the result of what, in classical image editors, would be the result of multiple filtering operations and many have shared dependencies.

The PictureEngine services include unsharp filtering, exaggerated highlights and blurring. The > 700 Sources are provided by the PictureEngine use image processing algorithms that include: Contrast scaling, Histogram normalisation, Object composition, Separable blur using either discrete convolution or an FFT depending on image area, Two dimensional discrete convolution, Edge detection (convolution), Random noise generator, Emboss (convolution), Quantisation, Warp, Nearest neighbour re-sampling, Bicubic re-sampling (up sampling), Morphological filters using standard structuring elements, e.g. erode and dilate, NTSC colour, HSV colour, Tiled texture, Trigonometry, Histogram colour detection, and the non-standard sieve. These algorithms are implemented in 'C' and compile with little change on the PC, Linux and Mac. Speed improvements will be significant when full use is made of pixel shaders.

The difference between successive Sources on a single tab may include, sieve scale, random number seeds and blurring parameters. This is more complex than simply different settings of an effects filter, c.f. Photoshop. Always the goal is to produce a useful, predictable, vocabulary.

Graphical User Interface

In commercial versions of ICE only the 'Studio' is novel and of interest. It has a Graphical User Interface (GUI) that is 'soft'. The functionality and layout are controlled by a combination of an 'ini' file, the launch command line and, most importantly, through encrypted ImageWizards. This has the major advantage that a number of Fo2PiX products, with different values, can be delivered from a single codebase.

The Studio will be illustrated using an extended version of the ArtMasterPro interface, Figure 4.5. The GUI is designed to associate a visual 'pattern' with the current state of the system. This visual approach is successfully used in many areas, for example, software based graphical equalisers for sound systems that emulate their physical conterparts where the pattern of sliders is important. It is how people like to work and critical comment in published reviews suggests that the approach used in ICE is successful.

Image editing is a visual thing. Providing visually obvious and meaningful patterns that reflect what the user is doing at any instant will making it easier to learn, use and relate to the software.

Little should be hidden behind pull-down menus and context sensitive information, showing the full state of the machine, should always be visible. For example (Figure 4.5, 73), the position of the currently selected Source in the hierarchy of 700 Sources is shown through the state of radio-buttons, tabs and through the position of its thumbnail position. Likewise (Figure 4.5, left side and

top panels), the settings for controls over the Palette and Stencil (mask) and blending and the context sensitive Canvas controls create patterns that the user rapidly starts to associate with results and what they want to do.

Sources: pre-defined layers The GUI is organised as follows. Along the bottom are thumbnails (Figure 4.5, 72 and 73) previewing Sources available from the PictureEngine. Typically, the order of Sources on a tab follows the extent of a variation between them and different tabs have different variations on a theme. Here

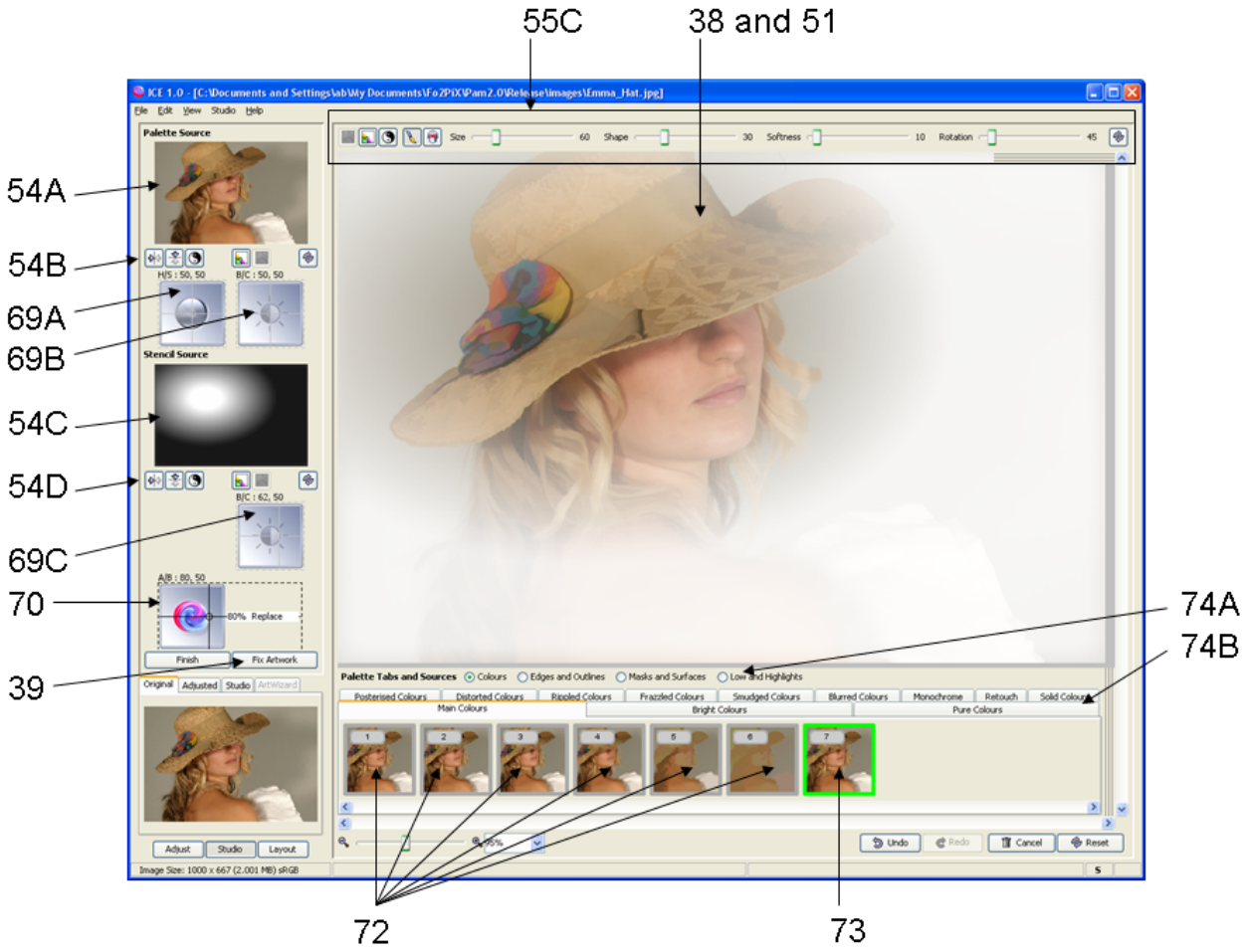


Figure 4.5: *The Graphical User Interface (GUI) is designed to associate a visual 'pattern' with the current state of the system by hiding nothing behind pull-down menus etc.: a* PlainSight *GUI.*

(Figure 4.5, 72), the Sources are increasingly simplified versions of the original generated with the sieve algorithm. (Figure 4.5, 73) shows a Source is a composition of those to the left and on many

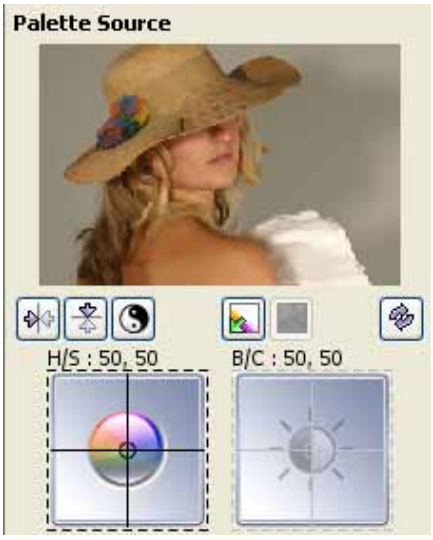


Figure 4.7: *Shows focus on the lefthand two dimensional, 2D, Picker. Horizontally it controls the hue and vertically the saturation. A single touch or drag changes both and the result is previewed in the top panel. Likewise the righthand brightness/contrast picker. Key is that the settings of all four controls together with the toggle flip and inverse buttons form a pattern that becomes visually associated with an effect.*

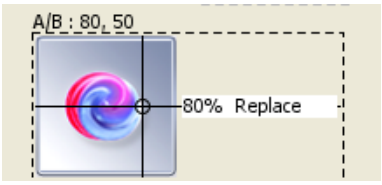


Figure 4.6: *The BlendPicker goes further. Horizontally it controls the amount (opacity) of the Palette blended with the Canvas and vertically it chooses from a number of merged blend functions such as Replace (c.f. Photoshop Normal) and Darken (c.f. Multiply).*

tabs there is an equivalent recurring pattern of 7 Sources. The 700 or so Sources, that represent the vocabulary from which the output image can be produced, are organised into sets of tabs. Thus the region of the GUI organises the vocabulary produced by the PictureEngine and provides a visual pattern related to the Source that is currently selected from the library (vocabulary) of Sources.

Palette: first component in the Blend pipeline Once a Source is selected for the Palette, a full size version is computed and is visible in the preview panel, (Figure 4.5, 54A also see Figure 4.7). The properties of the Palette image are controlled by toggle-buttons and two dimensional sliders, namely the Colour/Saturation Picker and the Brightness/Contrast Picker, Figure 4.7.

Stencil: second component in the Blend pipeline A similar set of controls are also provided for the Stencil. This is a mask or alpha-channel that modulates the amount of the Palette that will be blended with the current Canvas (output from previous step). In this version of the GUI the Stencil and its associated controls are visible below the Palette controls (Figure 4.5, 54C, 54D 69C). The Palette and Stencil are blended with the Canvas, (Figure 4.5, 38).

Blending with the working Canvas The BlendEngine combines these two inputs and the Canvas. In this example, the Stencil, mask or α-channel, fades out part of the image. The blend function itself is controlled by the BlendPicker (see Figure 4.5, 70) and 4.6). Unlike a standard picker, it selects a parameter on the horizontal axis and a function on the vertical axis (US patent application 20040239643). This allows people to smoothly slide between different blend functions such as 'screen' (Pure White), 'multiply' (Darken) and 'normal' (Replace) and simultaneously set the opacity (Amount).

Unique way of painting the α-channel A mask is modified interactively by painting (brushing, stroking) or flood filling (Figure 4.5, 55C). These tools are described in more detail in Figure 4.18. The facility to brush Sources onto the Canvas is intuitive, tactile and entirely consistent with the concept of image editing and users adapt to it immediately. Not only are the tools for interacting with the Canvas visible but the associations between the settings and what the user is doing them are reinforced by acts of brushing the image itself, see Page 4-16.

Different ways of viewing the image

It is not just in caricatures that artists look at their work upside down and in mirrors, it helps check that the composition is not lopsided. ICE therefore provides ways of temporally viewing the Canvas flipped left-right and up-down and with a superimposed grid dividing it into threes, Figure 4.9 and Figure 4.8.

Not everyone sees colour images in the same way. Some people, for example, might be viewing the image after printing in monochrome and many people are red-green colourblind. Holding down function key *F12* instantly provides a new greyscale view.

But which perceptual greyscale colour model? Consider a more subtle purpose inspired by Livingstones interpretation [54] of how Monet handled colour. Perhaps he unconciously (for how could they have described it without the psychophysical science) developed an ability to distinguish between the interpretations of colour and monochrome by the different neural perception systems. Exploiting these difference produces profound effects. Colours by Monet, Matisse, Albers, Warhol and many others since the impressionists are more intense, more vibrant, than is expected from the properties of pigments and dyes alone.

Notice, in example Figure 4.8, that the greyscale associated with the pink border, lips and hat decorations are not perceptably

Figure 4.8: *Why are the lips and pink border somehow both luminous and yet indistinct [54]?*

The best way to interact with an image is to stroke it.

*Figure 4.8 illustrates several artistic features that are easily and seamlessly achieveable in ICE but that are difficult or impossible with standard image editors. The image is highly simplified but not blurred (sieve). The lines follow artistically relevant edges and the thickness is modulated (sieve, warp and brushing the mask). The colour illusion (*F12*) and finally, the semi-automatic border that has complementary colours (ImageWizards).*

different from their surroundings causing the colours to vibrate and the regions to be somehow difficult to fixate. The hat decorations are similar but 'nailed' to the Canvas by black lines that are clearly perceived by both neural systems. The illusion is created with the help of two tools. Firstly, the BlendPicker to selects an appropriate colour space. Secondly, by flicking between colour and greyscale using *F12* and tweaking the Pickers. Incidently, in moderation the illusion adds motion and vibrance, overdone it can be quite disturbing.

For more details on Art see Chapters 6 and 7.

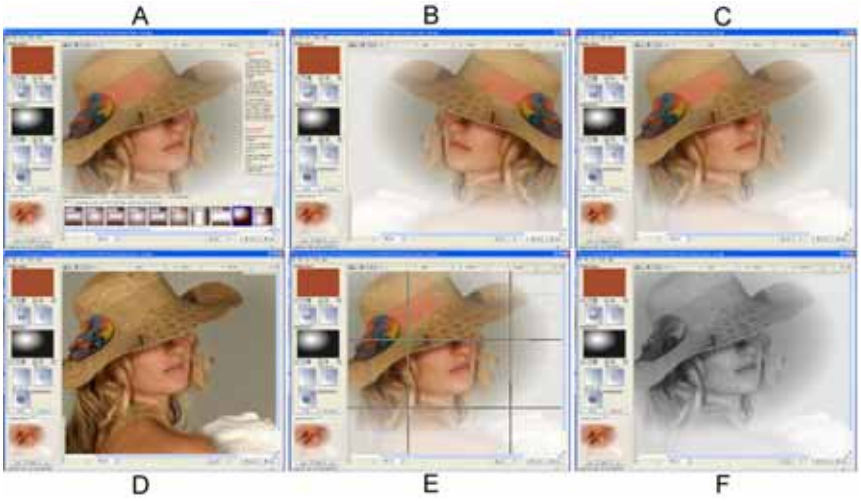


Figure 4.9: *(A) The help and Sources straps can toggled* F6 *and* F6*, c.f.(C). (B) Function key* F11 *temporarily flips the Canvas left to right (*F10*, up down) to help check compositional symmetry. (E)* F8 *superimposes a grid to visually check balance. (D)* F9 *displays the original image. (F)* F12 *provides a temporary perceptual greyscale view, see Figure 4.8.*

Undo/redo Control-z, control-y key behaviour is determined by context: last brush mark, step, manual interaction or wizard, see Page 4-14. Control-c copies the current canvas bit-image to the clipboard and control-v pastes a bit image from the clipboard into the start of the application displacing the current session. The behaviour of of the Help and Sources straps is governed by a persistant preferences and Function keys *F5* and *F6*.

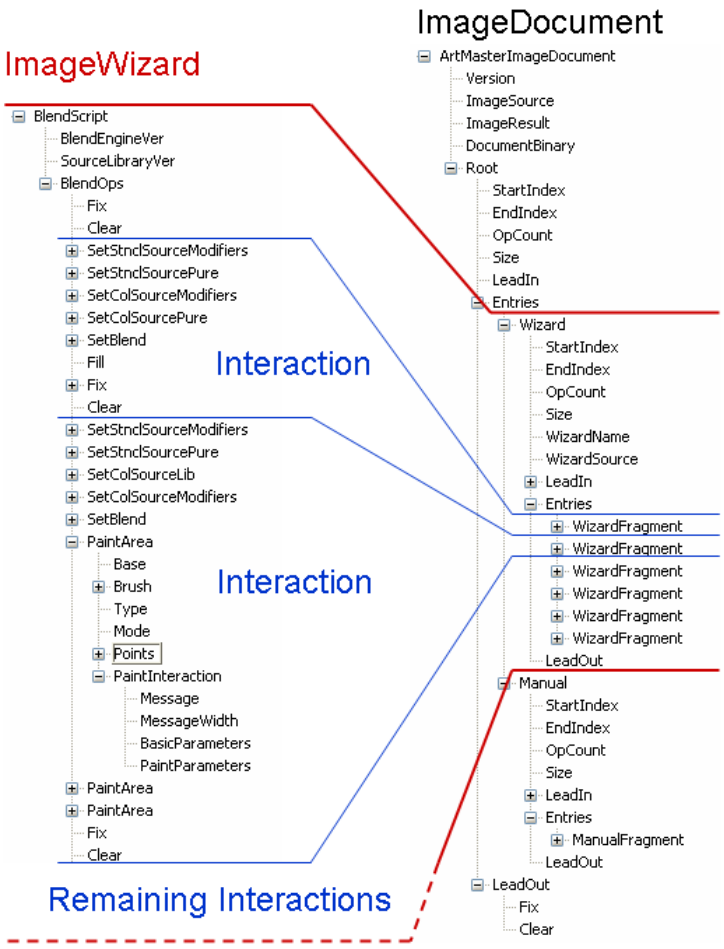


Figure 4.10: *ImageWizard and ImageDocument structures and the relationships between them. The former is a 'flat' set of 'program instructions' the latter is hierarchical and reflects the nature of the 'language of image editing'.*

Data structures

PictureEngine

The key 'C' data structure is persists during a session. Session information for each scale of image being processed is preserved. This includes a set of reusable image buffers and an array of structures containing image information. In the current version all results are cached on disc as .bmp files (for speed there is no compression).

ImageEditor

The Java data structures are best understood by viewing their equivalent XML records, Figure 4.10.

Figure 4.11: *Image used to illustrate how the data in an ImageWizard becomes modified to create an ImageDocument. The ImageWizard will help remove the dust without detracting from the aged graininess of the image. (J-class yacht circa 1926.)*

ImageWizards

An ImageWizard (ArtWizard in ArtMaster and Creata software) can be written in XML using the appropriate DTD. More usually they are generated automatically by recording every interaction in an editing session, see Page 4-19. On saving, only those parts that are sufficient to re-create the output are kept, i.e. the system skips those interactions that were, in the end, replaced.

The ImageWizard has no associated image. It is a *program* that can be *run* in ICE. Following the general header, Figure 4.10, there is a flat sequence of operations that select the Source images (SetStnSourceLib and SetColSourceLib), sets the Palette controls (the 'modifiers') and the Blend controls (SetBlend), etc. The default paint operation simply uses the settings in place when the ImageWizard was created. Run non-interactively an ImageWizard (centre of Figure 4.10) it is equivalent to many 'effects' filters and most Photoshop 'Actions' although the ease of production means that wizards often end up more complex.

What sets ImageWizards apart is the level of interaction that they encourage. The user interacts with the system to customise the effect to the particular image and to the users tastes. ImageWizards are extremely useful, extremely well received by users and represent a great step forward for image editors.

A detailed record of a session is richer than an ImageWizard. It has a hierarchy of events ranging from entire ImageWizards run in their entirety to individual brush marks.

Figure 4.12: *On prompting by the ImageWizard, the user has interacted with the image. The ImageWizard selected appropriate Sources, setup the Palette and Stencil (to mask out all except the dust) and then prompted the user to brush over dusty areas. It was user-friendly and depends on the sieve algorithm.*

ImageDocuments

Figure 4.10 shows part of the XML representation of an ImageDocument after an interaction. The ImageDocument now has a richer hierarchy. The ImageDocument starts with a header. The Wizard Entry starts with a header and is followed by the LeadIn that resets the controls. This is followed by Entries for each interactive step. Each of these contains a LeadIn that contains all of the non-interactive operations and the last WizardFragment usually contains a LeadOut that completes all non-interactive operations. The interactions are in between. There is an option to store the thumbnails associated with each interaction within the document as PNM ASCII format, Figure 4.13. The WizardFragment Entries contains the actual interactive steps. It also contains a list of all the people who have edited that interaction from its original insertion. This forms a start of a full 'Track-changes' facility.

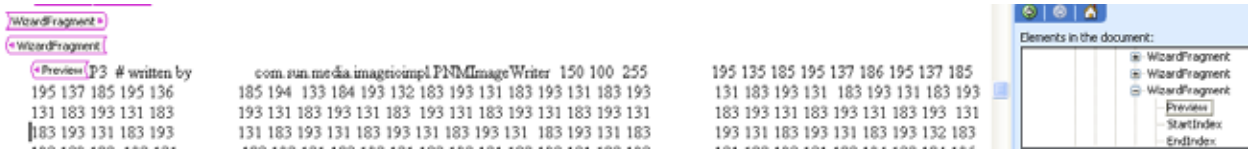


Figure 4.13: *Embedding a thumbnail edit step within the ImageDocument*

The details are typically lengthy because every single brush mark and click has been recorded however they are stored at a lower level and so can be tucked away (actually, in the current implementation they can be stored in a separate XML file and referenced through an 'include').

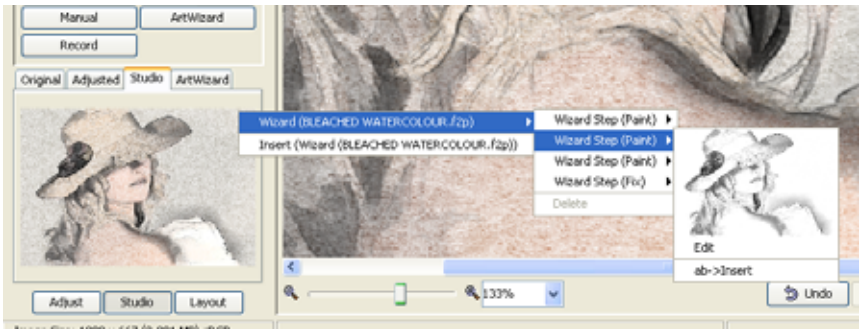


Figure 4.14: *One method for navigating the editable interaction list. In this example a step in the Manual interaction has been inserted by user 'ab', then modified by 'ab'. There is a thumbnail reminder. The edit list history is shown on the left and there is an opportunity to edit interactions within the ImageWizard.*

Thus, after a single session an ImageDocument and the associated Java data structure ...

- Is not flat.
- Represents information about the editing session hierarchically.
- Contains all the information needed to recreate the result starting from the original image and given the relevent PictureEngine services.
- Contains non-interactive elements bundled separately from interactive ones.

Editing itself is an interaction: the ChangeList

Alongside the document structure there is an edit list that contains the undo/redo list. This is currently non-persistant although to pursue 'Track-changes' to its conclusion in the future it will be stored. All *interactions* are undo/redo and editable. And the edit itself is undo/redo'able.

It is reasonable that one cannot backup through the steps required to create a Source since it is not the algorithm that is of intereste but the result. Nor, for the same reason, should one backup through an automatically executed ImageWizard, nor through operations within an ImageWizard that were executed automatically. However, each of these blocks can be undone to return to the previous interaction (and redone). Thus, what is undo/redo'able depends on the session and is properly a property of the ImageDocument.

Given this data structure it is possible to move to any interaction in the document, change it and automatically recompute, from that stage forward, to form a new end result. Call it an 'Edit Anywhere' structure.

Of course, editing itself is an interaction and is also be undo/redo'able and this becomes particularly desirable when working on an ImageDocument in multiple sessions. Thus, Control-z works back through the edit list that modifies the ImageDocument accordingly and redo'ing works forward.

What is needed is a way to navigate these two structures neatly.

Track-changes *becomes a possibility for image editing opening the way to proper sharing of documents and collaborative working.*

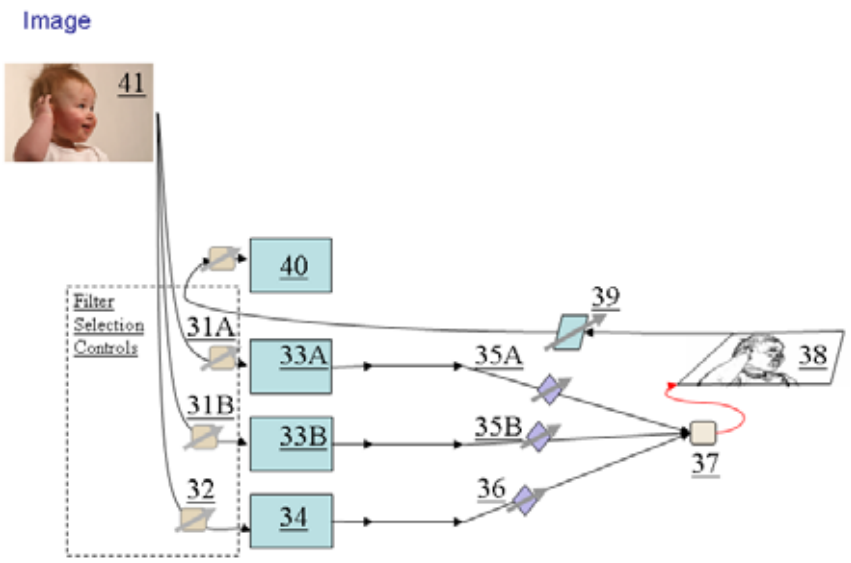


Figure 4.15: *In a classical image editor the original image is transformed into a number of layers that can then be blended to form a new result. The result joins the existing layers and can form a new starting point for further processing. The layers represent both the elements from which new results are derived and a history of things that have been produced earlier in the session.*

Image Data flows

A Classical Image Editor

Layers Figure 4.15 illustrates a classical image editor. An image 41 can be filtered, or objects it contains can be selected, using manual controls (31A, 31B,32) to create layers (spatially aligned objects). For example, a modified version of image 41 could be assigned to layer 33A. Similarly, thresholded version of the image could be assigned to layer 33B. These two can then be blended with layer 34, using the blend controls 35A, 35B and 36 such that layer 33B is an α-channel for layer 33A and the result is stored in Layer 40.

Observations reveal that it is very common to blend just two layers with a third working layer to produce the result. It is worth producing an architecture to make such a **step** *particularly easy.*

Image editors typically allow a large number of layers to be generated and for regions (objects) within the layers to be combined selectively by using intermediate layers as alpha (α-) channels or masks. With an extensive edit, a large number of layers can be formed and it is difficult to keep track of the function and meaning of them all. However, observation reveals that at any time users commonly combine just two layers with a third working canvas and that it is particularly advantageous to be able to adjust the layer properties 35A, 35B, 36 and the blend function 37 and be able to see the result immediately. Such a short sequence of operations is so commonly used that hereafter we will call it a *step*.

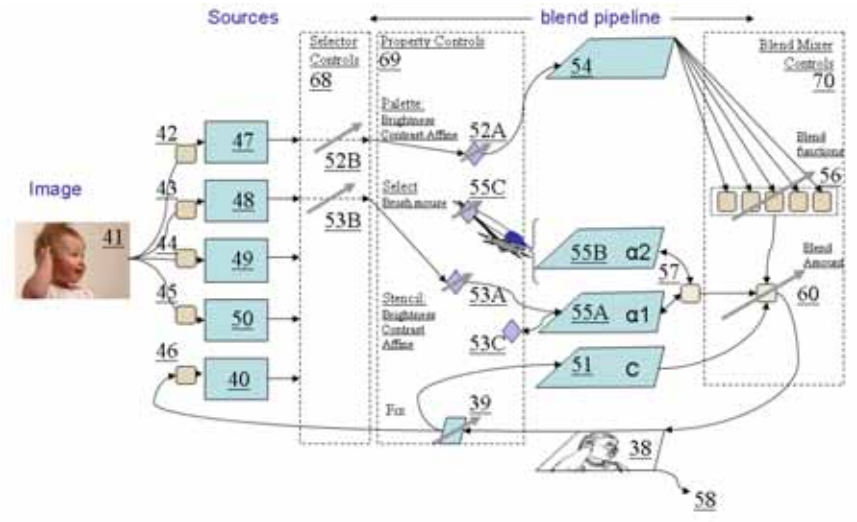


Figure 4.17: *shows a diagram of an example of an image editor which may be used to edit image documents generated in accordance with one example of the system and method of the present invention*

Layers are used in different ways It is also common for the multiplicity of layers to be used to record the results of many such steps and, in combination with a history of operations, they provide some sort of record of the work session. It is not, however, a full history. The history in standard image editors does not make it easy to remember the details of each step in an edit session. It is particularly difficult because of the huge range of possibilities afforded by the manual adjustments such as 31A, 31B, 32, 35A, 35B, 36, 37 and any other associated selection mask or α-channels. What is needed is a good way to organise and navigate all this information.

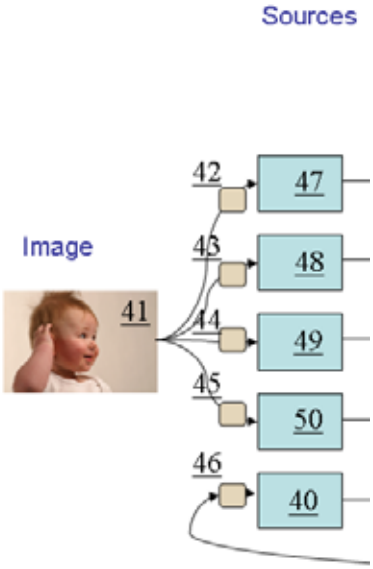


Figure 4.16: *The PictureEngine can generate* > 700 *Sources with carefully chosen parameters replace layers.*

ICE: a Library of Predefined Layers

Layer library An alternative to the manual ad-hoc production of layers is to provide a set generated from the original image 41 by carefully chosen fixed adjustments 31A, 31B, 32 and then organise these into functionally meaningful sets (Figure 4.5). This particular type of layer is called a *Source*. Figure 4.17 shows an example of dataflows through a new image editor, see also Figure 4.16. The two dataflow models can be melded together, Figure 4.20, but before considering the whole structure a new dataflow structure needs to be introduced that builds on the earlier observation that it is very common to blend just two layers with a third working layer.

The goal is a customised dataflow structure, the *Blend pipeline*, that makes it particularly easy to work with a filtered image or

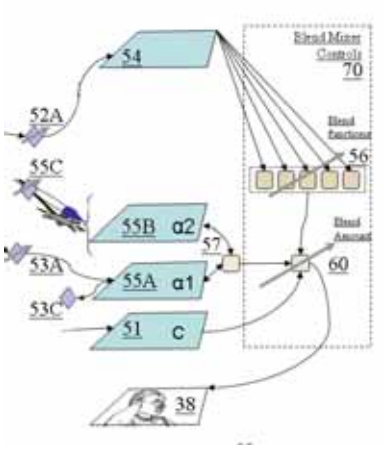


Figure 4.18: *Three special input layers, 54, 55A and 55B are blended with the working layer, 51 to produce the result 38. 55A and 55B are a 'mask' (α-channel. Changing any of the controls on the input layers including brushed changes to the α-channel are immediately visible.*

source, a mask and a working image or canvas. It is convenient to use and allows all controls to be real-time. More importantly, it enforces a regular structure of steps, Page 4-14. This is a subtle point but it is extremely important for perceived user friendliness and for it transforms the editing sequence into a natural language.

Blend pipeline to make 'steps' easier Figure 4.18 shows the flow of data. It is a customised version of the components needed to implement a step, Page 4-14. Layers (Sources) are loaded into a Palette 54 and Stencil 55A (i.e. mask or α-channel) where they are subjec to controls (52A and 53A) that include brightness, contrast, affine transformations, etc. These are then blended with a copy of the working Canvas 51 to produce a new result 38. The choice of blend function and amount (opacity) is controlled by the Blend Picker (US patent application 20040239643, see Figure 4.6).

The entire pipeline is kept open and so subject to real-time changes in any of the controls until a 'Fix' button is clicked. Playing with the result this way is easy because the Picker controls are compact and visible during the blending process, see Page 4-4, and any changes can be (infinitely) undone and redone.

Stroking the image

But there is one further key innovation that makes the system particularly user friendly. It is possible to interact directly with the image, Page 4-4. In particular, the user can brush over the image. For example, imagine that the Stencil (α-channel, 55A) is all zero preventing any of the Palette (54) from showing through. Imagine too that the Palette shows the edges of important objects and finally imagine that the blend is set to 'multiply' (in ICE, 100% Darken). Then brushing (using the mouse or graphics pad) along regions that contain the edges will have the effect of revealing the edge. Notice that the user does not have to be accurate, the illusion is of painting and this makes it natural and rewarding. In reality the computer has done the work of finding the edge line. The system is leveraging the power of computer vision algorithms to compensate for unpracticed manual drafting skills.

People want to interact with the image, stroke it. It is natural and intuitive.

It is not just painting edges that becomes more natural. Unsharp filtering, emphasing highlights and lowlights all benefit. More and better computer vision algorithms will steadily increase the variety of 'component' that can be brushed into place. The idea is getting closer to a vector graphics system, Page 3-4.

It works because the user is actually brushing the α-channel and the result is the $\max$ of (55B and 55A). The quality of the edge is not dependent on careful drawing. It is quick like a professional sketch artist is quick. Provision is also made to flood-fill regions of the α-channel according to thresholds set on the Canvas (51).

Improved blend pipeline

A further improvement to the system is shown in Figure 4.19 (patent applied for but not yet fully implemented in ICE). It is a mechanism for using the brush movements to alter the image in a painterly way. Instead of deriving the Palette image 54 from a Source it is derived from two images 61 and 63 using the function 64. Image 61 is derived from the Source instead. Pixels in the second image 63 are obtained from 61 but are offset or translated according to information in 62. The information in 62 corresponds to two image sized matrices, X and Y that represent pixel positions. Normally, subscripts correspond directly to the pixel positions, i.e. the subscripts at position $[X(303), Y(200)]$ have the values $303, 200$. However, as the brush or pointer is moved over the image being edited so the subscripts in X and Y, over which the brush moves, are changed to remap the pixel values in 61 to a different position in 63. For example, the subscripts to the maximum pixel value in the line from one sampled mouse position to the end of a brush stroke, $[Xm, Ym]$, are copied to all positions from its start position, $[Xm, Ym]$, to the end of the stroke. The effect is a streak, that can be made to fade, along the line of the brush mark.

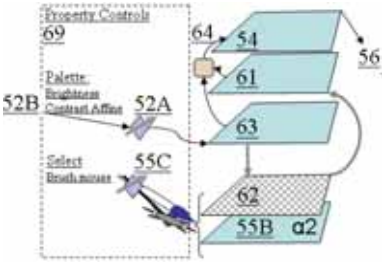


Figure 4.19: *shows a method for enabling further enabling brush stroke information to be used to simultaneously improve the image and improve the record of interactions with the image.*

Thus, this approach uses the brush motion to produce a painting effect whilst maintaining the 'real-time' properties of the blend-pipeline in which the controls 52A and selections 52B, 53B, etc. can be changed and the effect of the brushstrokes is maintained in the final result.

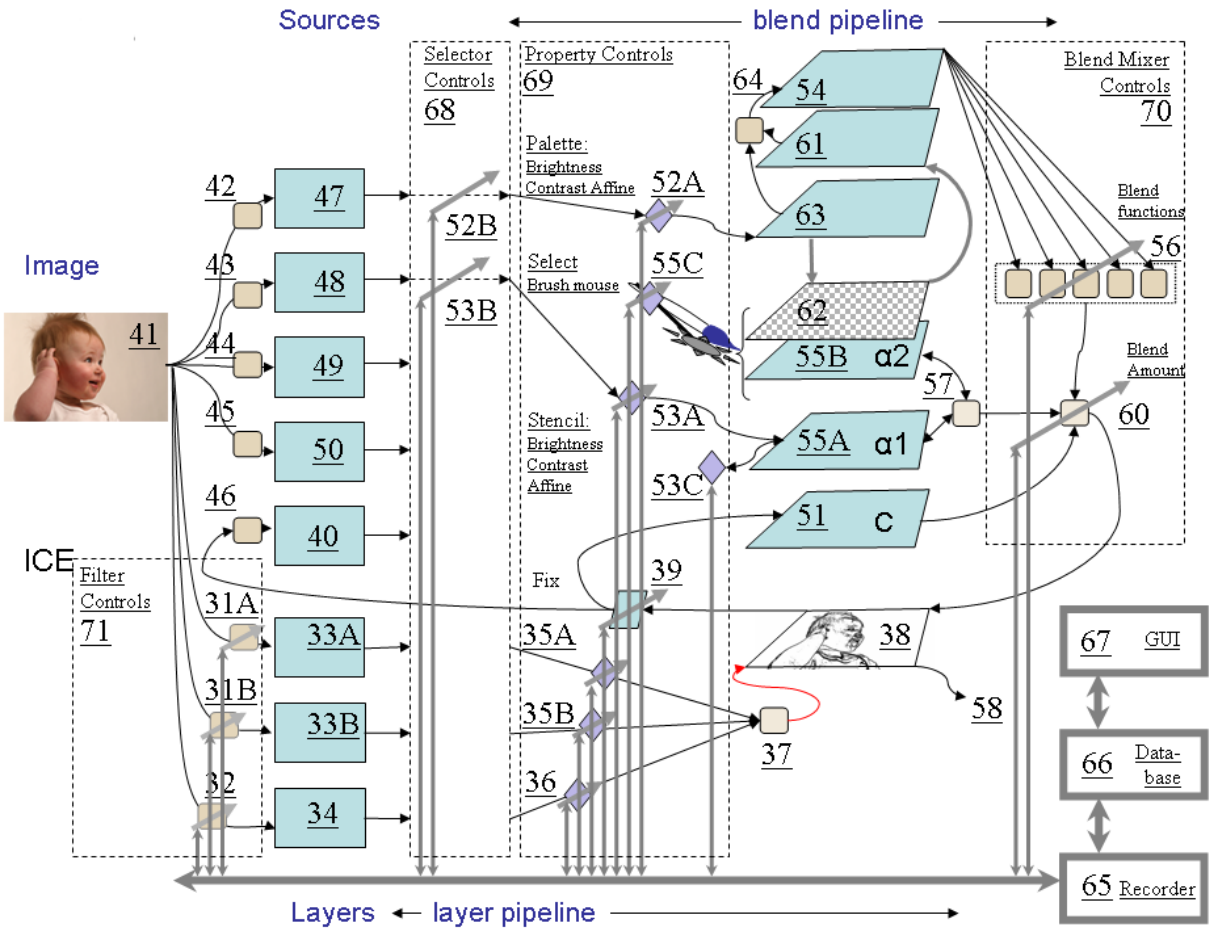


Figure 4.20: *A diagram of the entire system including control data pathways; that are better visualised alone*

The blend pipeline is extremely user friendly particularly when creating artistic results, however, a full image editor needs to be able to do all that current editors do. Figure 4.20 shows a combination of the two described in Chapter 4.2. It is overlaid with the control pathways that enable everything to be recorded and controlled indirectly. These control pathways are shown more clearly in Figure 4.21.

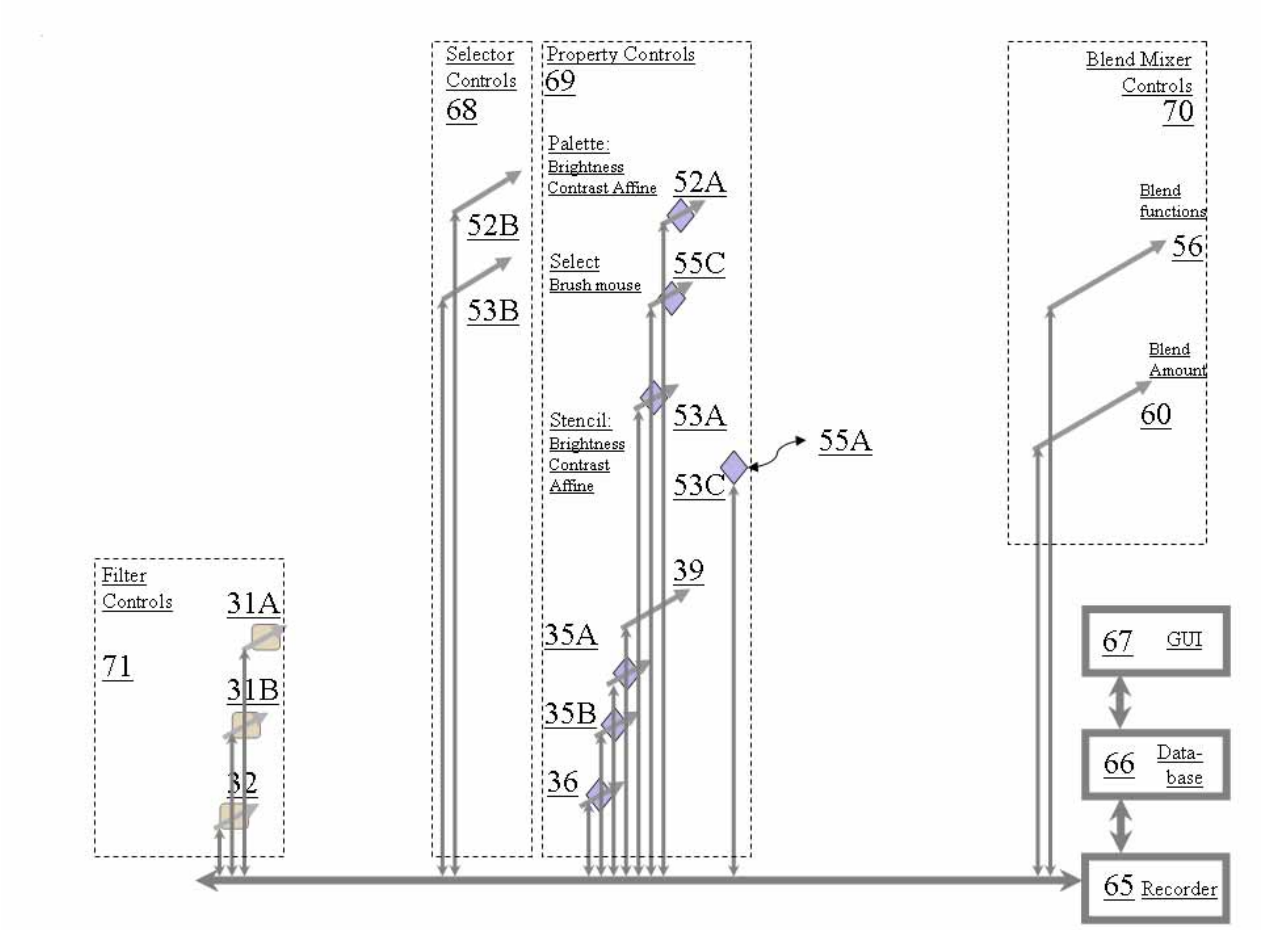


Figure 4.21: *The bidirectional control data pathways*

Control pathways and history

Figure 4.21 shows the editing system overlaid with the bidirectional control data pathways. In this representation of the editing system, all of the manual control systems 31A, 31B, 32, 52B, 53B, 35A, 35B, 36, 39, 52A, 55C, 53A, 53C, 56 and 60 are connected to a recorder 65 and a data base 66 and are controlled through a graphical user interface 67. The values of these controls are recorded during manual use of the editor and can also be set and adjusted either automatically or manually when replaying or using a pre-recorded sequence. These records generate the data structures discussed in Section 4.2.

Among the key records are those from the brush 55C, itself controlled by a pointing device such as a mouse or stylus where not only position but also pressure, tilt angle and rotational angle

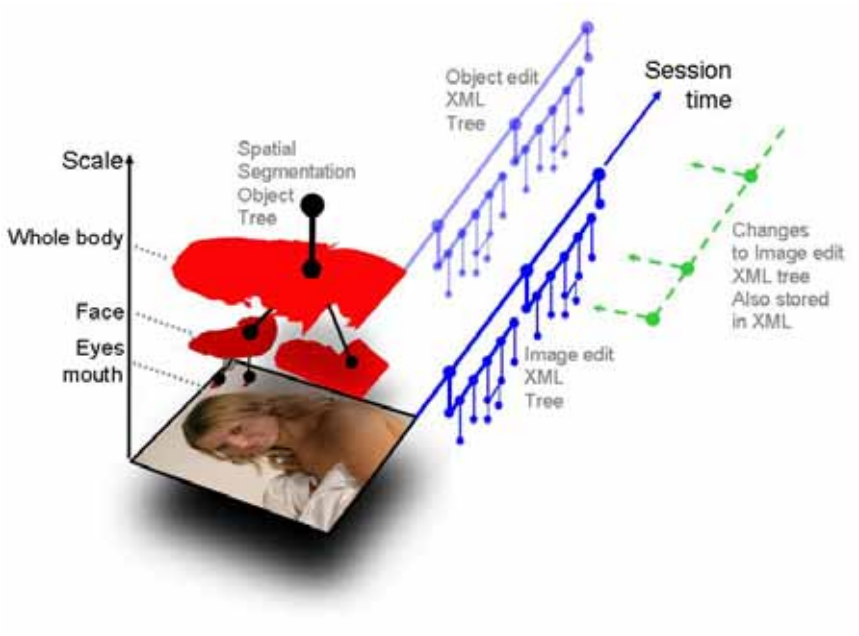


Figure 4.22: *Summary of data structures. On the left, black, is a scale based spatial tree where nodes are layers or objects. Smart computer vision algorithms are leveraged to both improve the the relevance of the nodes and the means for navigating through them. On the right, solid blue, is the ImageDocument and, in dotted green, the associated ChangeEdit data. Smart language processing algorithms are leveraged to both improve the the relevance of the nodes and the means for navigating through them.*

are also important and also the magic wand flood fill tool 55C that is also controlled by the pointing device.

Edit lists associated with objects

In the examples described so far the edit sequence is associated with the entire image. This need not always be the case. For example if 45 is a skin filter then 55A will mask out all except the parts of the face showing skin thereby labelling the face. Thus, those parts of the EditList and the ChangeList then become associated with the face. They are 'face methods'.

The set of data structures in ICE are summarised in Figure 4.22.

Summary

A large library of Sources obtained by analysing an image forms a 'vocabulary' with which photographs can be improved. ImageWizards extend the vocabulary to doing things. Interactions with the photograph are made particularly easy through an innovative Blend pipeline that supports a brushable α-channel. The whole is presented through a 'PlainSight' interface that visually relates the state of the system with the current action and this makes it easy to learn. Finally, ImageDocuments capture everything that is done to create an output and these can be re-edited keeping track of who made which changes and where.

To be done

Having implemented and used ICE new horizons for image editing become clearer. Here are a few.

Development	Enables	Markets
Move away from java. Hardware acceleration.	Non-destructive editor that extends to interesting, creative and profitable works.	Integrated image editor for professional and wider groups of digital camera owners.
First definition of image editing language.	Good ways of sharing editing between professionals and on the the web.	Integrated image editor upgrade 1 and fun, web oriented, visual sweeties.
Good composition recognition.	Smart cropping, colour balance and distribution of detail. Source 'Main Colour' 7 is already making headway in this respect.	Integrated image editor upgrade 2 and fun, web oriented, visual sweeties.
Inference of buildings and rigid bodies from multiple images in a sequence.	Object specific photographic and artistic computer vision based tools. Extension of the vocabulary of image editing.	Integrated image editor upgrade 3.
Integration of global positioning system data with images.	Architectural images to be shared by position. Different interpretations of scenes can be accessed and shared through the web.	Integrated image editor upgrade 4.
Inference of soft shapes for faces and animals from statistical models.	Smart tools for editing faces and creating portraits. Power assisted editing.	Extensive use in the home.
Bootstrapping computer vision recognition algorithms from ImageDocuments.	Extended MPEG-7 compliant metadata tagging permits object finders for faces, people, houses, cars, animals etc .	Image retrieval.

Figure 4.23: Indication of the opportunities that can be grasped starting with the concepts in ICE. It will enable computer vision and language algorithms to be steadily introduced producing a stable product line.

Algorithms for understanding images

Figure 5.1: *Linear blurring, top right, simplifies at the expense of smoothed and indistinct edges. Median filters, bottom left, have clearer edges but still imprint their shape onto the image. Whereas, bottom right, shows that the unique sieve ('buZZ' Photoshop plugin, Fo2PiX, Cambridge, UK, 2001) simplifies without changing the shapes of features.*

Algorithms for image processing

The automatic recognition of objects in two dimensional photographs is critical for image understanding. The ideal system would be able to pass over an image, feature by feature, object by object and label each where objects might range from people and cars to terrorists and malignant cells. It would automatically generate a graph, probably a tree, of relevant objects.

Framing the idea this way works for very specific computer vision problems such as finding defective printed circuits, car number plates and, perhaps, breast tumours and individual suspected criminals. However, in general it is badly framed because a single image means different things to different people. To one person the image might be about a pretty woman, to another her hair style, to a third the car she is leaning against, to the aspirant suitor her ring finger, to a photographer the poor white balance, and to her husband 'who took the photo?'. Clearly the meaning of an image is not universal, it can only be established in the course of a dialogue with the user. It is as much in the eye of the beholder as the image itself.

Thus, the intelligent use of computer vision algorithms to make smart image editors, smart image organisers and perform image retrieval depends on getting people to reveal their interests. In content based image retrieval, for example, the single biggest improvement is to encourage the user to provide feedback on the

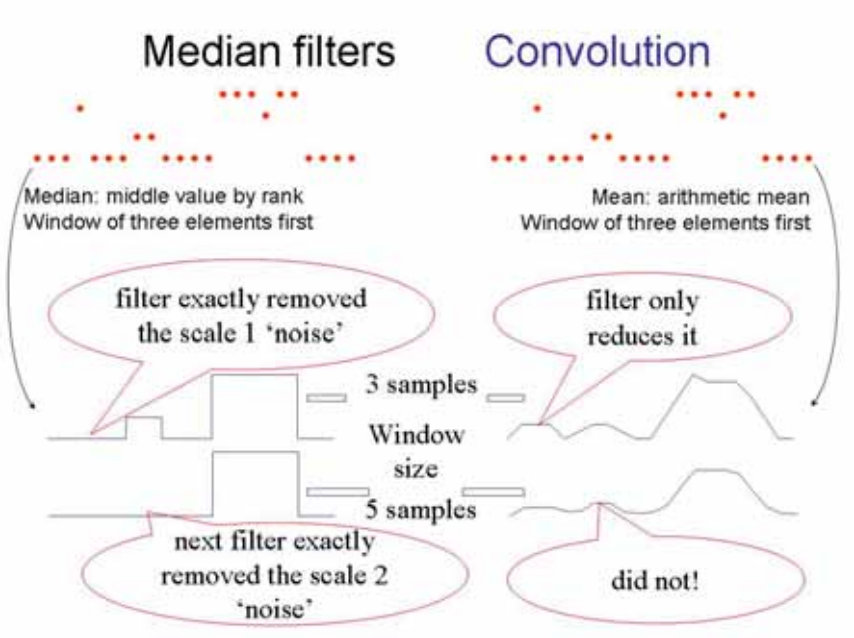


Figure 5.2: *Showing that median filters robustly reject impulsive noise where linear convolution filters do not. The signal, shown as points at the top, is either filtered using a window of 3 samples, upper trace, or 5 samples, lower trace. Whereas the running mean filter smoothes the signal, right column, the median filters do not. Instead, median filters completely eliminate short lived outliers, i.e. medians are robust. However, neither median nor mean filters preserve scale space.*

relevance of an initial search (we find a 30 to 40% improvement that is consistent with work of others, e.g. [90]).

This document presents two important steps that address the original problem. The first is the image editor that is easy and relevant to use and that encourages people to 'stroke', 'touch', 'point to' and address important features of images, just as they do in their lives for their possessions, friends and things of interest.

The second is an innovative image processing algorithm described here.

The problem of finding objects in images

A typical image of 5 Mpixels (15 million 8 bit samples) contains just a few objects of interest. If the goal is to identify say 10 objects, then information is to be concentrated in the ratio 1,500,000 to 1: it is like looking for a needle in the proverbial haystack. Since objects of interest are usually large relative to the image pixels, one approach is to start by using filters to simplify the image, i.e. hopefully concentrating the information by, say, 1000 to 1 leaving 1,500 segments for more detailed analysis. After introducing appropriate filters a twist to this approach is discussed together with ways forward.

Scale Space Filters

Classical signal and image filters are linear and, for these, there are well developed theories. For example, in the 1980's Witkin [88],

The key algorithm leading to the development sieves was not image processing. It arose from a need to remove noise from the tiny currents flowing through single protein molecules, pico-amps through single channels in biological membranes (first described by Nobel prizewinners Neher and Sackmann). However, the demand for a quantitative, statistically optimal, filter was short lived: low noise operational amplifiers obviated the need. However, once discovered it was clear that sieves had much wider application.

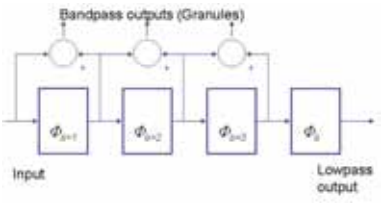


Figure 5.3: *Sieves are a series of increasing scale rank order filters (ϕ_S, where S is the filter scale). Where $max_{S,E}$ is a locally maximum extremum of scale S, similarly $min_{S,E}$, the filters, ϕ, can be c where at each scale $max_{S,E}$ are merged with their nearest neighbours. o Where $min_{S,E}$ are merged. M Where first $max_{S,E}$ then $min_{S,E}$ are processed and N where it is $min_{S,E}$ then $max_{S,E}$, or m where extrema are processed data order. All preserve scale-space, robustly reject outliers and are idempotent.*

Koenderinck [49] and others [77, 52] see also [39] drew attention to the particular importance of scale-space preserving, diffusion, filters. The goal is to simplify the image without introducing any new features. The idea is best illustrated by example.

Imagine projecting a black and white image onto a screen for an hour. Now turn off the projector and turn on a thermal imager. White areas will be warmer than dark areas and the image will be visible. Over time heat diffuses from the warm to cool areas and as locally hot and cold spots are smoothed out so the image becomes blurred and, with fewer thermal maxima and minima, simpler. At no time during the diffusion process are new warm and cold regions created and this property is known as scale-space causality preserving.

Rank filters Note, there is an implicit assumption that important features are associated with local maxima and minima (extrema). Gaussian filters uniquely among linear filters have this property [45]. In the early 1990's the library of causality preserving filters was extended to include non-linear diffusion filters [64, 59] that produce attractive results but confound scale and contrast. Other non-linear filters that were explored included morphological dilation (max) and erosion (min) filters [47]. Like median filters, Figure 5.1, they have structuring elements that imprint their shape on the image. Unlike medians however, they are very sensitive to noise (a characteristic that is exploited elsewhere to generate 'painterly' effects). By contrast non-linear sieves [2, 6](best conference paper and see Figure 5.1 bottom right) also use rank ordering properties but are extremely robust to noise and do not have structuring elements.

Median filters are widely discussed in the literature, and used in industry and image editors. One of the claimed advantages over linear filters in two dimensions is that they preserve the edges of objects, Figure 5.1 and in one dimension they preserve the transients (edges) of pulses (see Figure 5.2).

However, this is only roughly true. Despite the publication of numerous algorithms for optimally weighting median filters and generating optimal structuring elements by the non-linear filtering and mathematical morphology research communities, two problems remain. Firstly, experiments show that the claimed ability of median filters to preserve the shape of pulses (in one dimension) is, on average, no better than linear Gaussian filters [18, 73].

Likewise in two dimensions, some edges 'look' better after me-

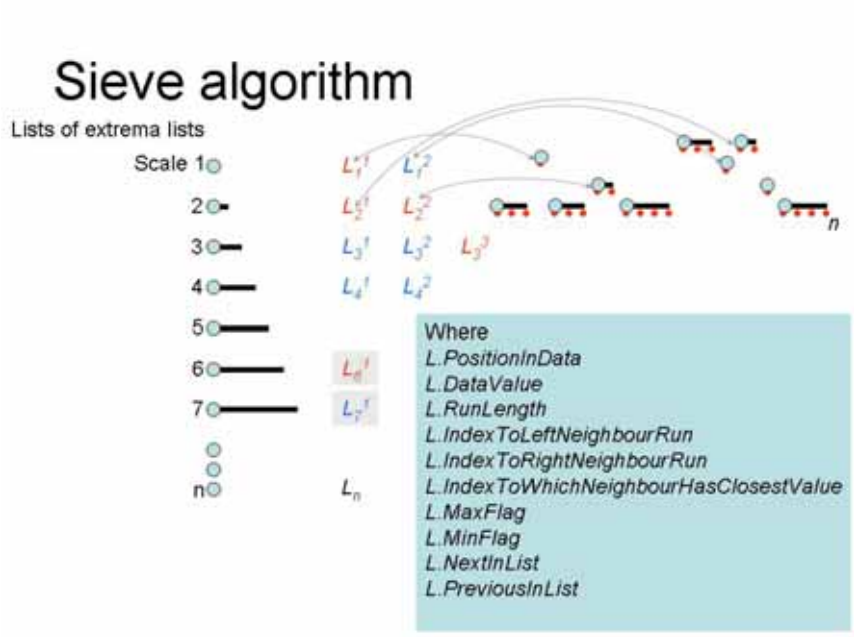


Figure 5.4: *Illustration of the 1D sieve algorithm. In one pass the n data values (red dots) are run-length coded (cyan circles) and local extrema, identified by 'looking' left and right, are mapped into a list of lists. One extrema list for each scale, 1 to n. The filtering process then starts by merging all elements in the scale 1 list with their nearest, in value, neighbouring runs. Each merge typically requires two pointers to be changed as two runs are linked and the new, longer, run is remapped into the list of lists. Filtering stops when the desired scale is reached and the output is rebuilt.*

dian filtering than after Gaussian filtering (blurring) but detailed analysis shows they are located no more accurately than after Gaussian filtering. Worse, in two dimensions the shape of the window (structuring element) is imprinted on the output of the filter and it becomes difficult to disentangle underlying properties of the image from systematic errors that have been added, Figure 5.1. A second consequence is that such independent filters do not preserve scale-space causality and this too means that median filtering obscures underlying, large scale, information in images.

Scale Space sieves solve the problem

Sieves are covered by Segmentis patents US 6,081,617, US 5,912,826, US 5,917,733 together with European equivalents.

Although developed for single channel analysis, the concept of sieves [2] was first exploited for analysing protein structures [3]. It shown to be the best way to distinguish hydrophobic from hydrophyllic regions. The algorithm was criticised because, at the time, it was not understood 'why' the algorithms worked so well. A criticism that precipitated more detailed work, in the 1990's, on the robustness of one-dimensional sieves [4, 6, 7, 5]. In one dimension there is no difference between an algorithm that operates on a window and one that operates on level connected sets, however, in

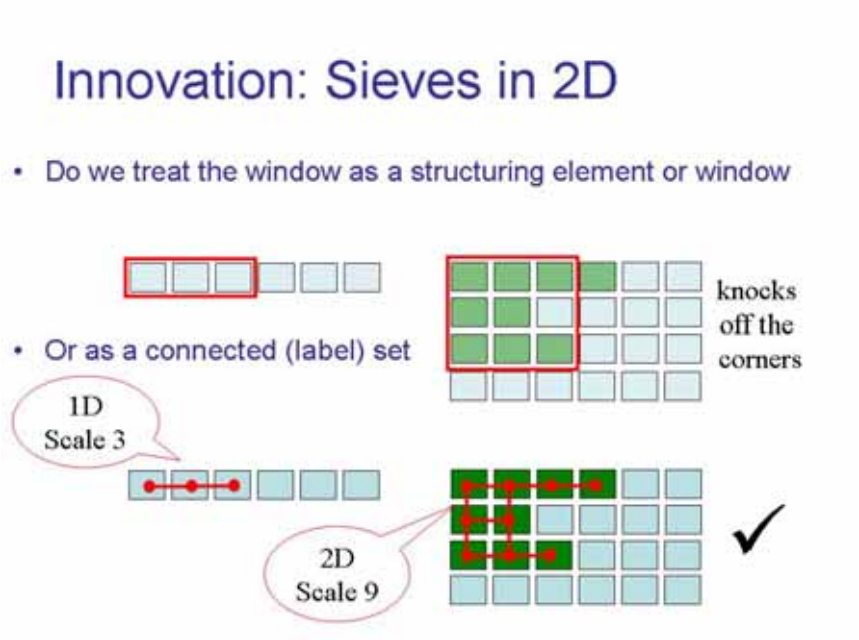


Figure 5.5: *Sieves do not use structuring elements (top). Instead, they operate on level connected sets of pixels, i.e. they follow edges in the data: they are shape free. Bottom right, the sieve is about to remove a set of nine pixels that form a local extremum. Unlike a filter based on a structuring element or window (top) the shape of the selected set follows the edges. In 1D increasing scale sieves remove increasingly long extrema, in 2D they remove extrema of increasingly large area and in 3D, volume.*

higher dimensions the difference between the two is profound [94], Figure 5.5 and Figure 5.1 bottom.

The key properties of sieves arise because they use level connected sets. At this time there was also interest in 'reconstruction filters' that also appear to preserve scale space causality (Vincent, Salembier) that are something of a hybrid between sieves and classical mathematical morphology.

Initial results Simple experiments on segmentation [44, 42, 40, 41, 43] and pattern recognition [69, 6, 44, 68, 70] provided a better insight into their practical value and how sieves might be applied to segmentation [6, 67, 8]. Sieves proved very useful and the work was, therefore, followed up by establishing the mathematical properties of 1D sieves [22, 10, 9, 20], multidimensional sieves [14, 12, 11], their relation to weighted median filters [93, 92, 91] and mathematical morphology [15]. A short formal description of sieves is given in Chapter 7.

Complexity of sieve algorithms It should be emphasised that the obvious ways to implement sieves have a high order complexity, $> O(n^2)$ (where n is the number of samples or pixels in the image) and this did not encourage other research laboratories to work with sieves. Fo2PiX (UEA, Segmentis) did not to draw attention to the patents that explain how it can be implemented.

A fast algorithm is essential. A 1D implementation is summarised in Figure 5.4. Sieving starts by passing though n data points once to map the signal into the list of lists and a second time to pass through each list of the list of lists to both filter, $< n$ elements, and rebuild the processed signal. Note that as the scale becomes larger so the number of lists reduces yielding $O(n)$ [13].

This is quite unlike classical morphological or linear filters where processing time per point usually increases with scale. The 2D case is similar where runs become patches. It is more complex because each patch usually has more than two neighbours requiring a search for the neighbour with the closest value to the current extremum. However, patches are either small and so have few neighbours, or are large in which case they are only analysed when the numerous small regions have been eliminated. The algorithms work out to be approximately $O(N^{1.1})$ in two or more dimensions [11]. Currently, there are two implementations of 2D sieves niether have been optimised to take advantage of modern pipelining

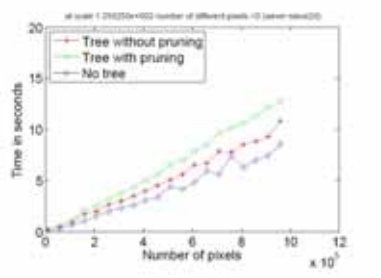


Figure 5.6: *Order complexity of different versions of the sieve, $O(N^{1.1})$*

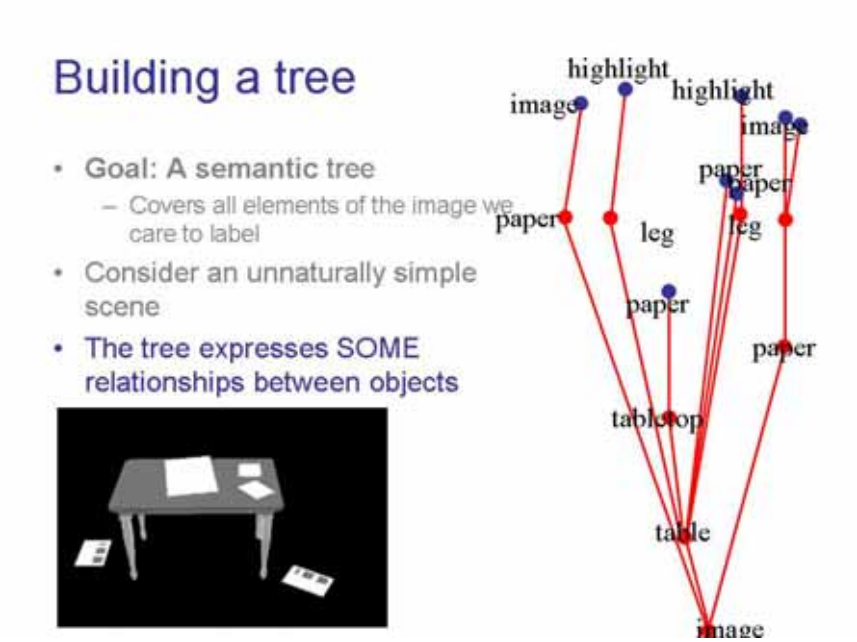


Figure 5.7: *Shows a simple image, left, and the corresponding semantic, object, tree. It is a goal of computer vision to do this automatically. It is not possible using Gaussian filters, wavelets or Fourier transforms.*

Merging at each scale can be done in various ways. If, at each scale, the maxima and minima are merged in data sequence, the result is equivalent to using a recursive median filter at that scale, an m sieve. It is left-right assymetrical, i.e. parsing left to right is likely to yield a different result from right to left. At each scale an alternative is to merge first the maxima then the minima, or vic versa. These two alternatives are also different, i.e. they

are up-down asymmetric. In practice, particularly in two or more dimensions, these four alternatives yield almost identical results and all four are have a robustness comparable to median filters.

A slew of papers on 2D images then followed. Once again their robustness was established [31, 32]. More importantly, a new representation of the sieve transform was developed: a new twist.

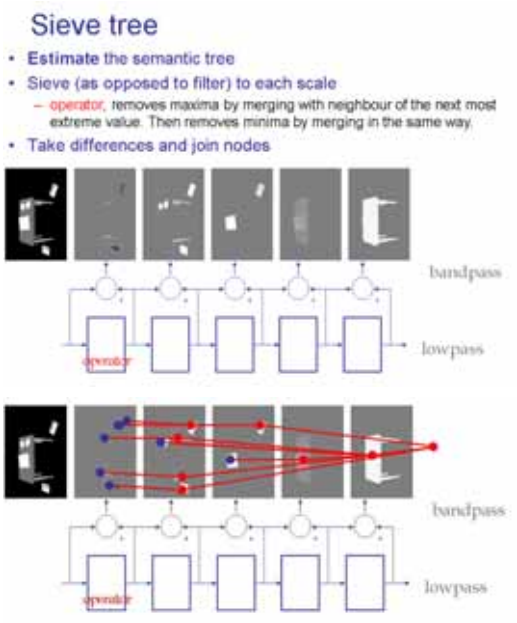


Figure 5.8: *Top, a sieve decomposition produces granules that are closely related to the objects. Bottom, shows that a sieve goes a long way to achieving the decomposition into objects. Thus the sieve looks a promising starting point for analysing real images.*

	1D-sieve	2D-sieve	DTCWT	LBP	Co-occurrance
Mean	0.718	0.943	0.556	0.509	0.692

Table 5.1: *Reliability identifying anisotropic textures,* 1 *is perfect. The 2D-sieve performs the best.*

The Sieve decomposition and generating tree structures

Simply filting an image is useful, about 60% of objects that volunteers identified (about 200 people each marking up 50 images) included large scale extrema and so removing detail by removing small scale extrema is likely to concentrate information. But it is crude. What is really needed is a hierarchy of objects such as that shown in Figure 5.7. Albeit a stylised image it makes a point because, unlike the Fourier transform or wavelet decomposition, the sieve will actually generate such a hierarchy, Figure 5.8 in one pass. This forms a conceptual framework for understanding how sieves might be used.

In reality, it is more difficult. A sieve decomposition of real images produces trees with far too many nodes [16, 17, 36]. It is only recently that methods for simplifying the tree and, as important, efficient code for implementing such simplifications, have become available. Figure 5.9 shows output from the implementation with

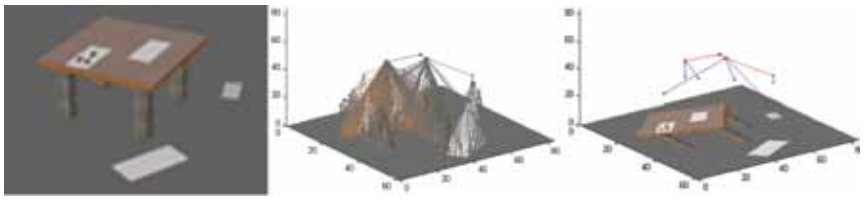


Figure 5.9: *Left panel shows a simple image. Middle panel shows the associated sieve tree. It is a mapping of the image, transformed by no loss of information. Right panel shows the result of a simple clustering algorithm to control the building of the tree during the transform. If a node (granule) is sufficiently similar to its parent the two are simply merged. The concentration of information is greatly increased.*

an order complexity shown in Figure 5.6. It is promising.

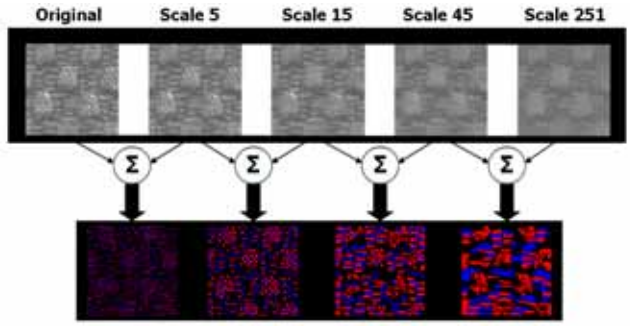


Figure 5.10: *A textured image sieved to five scales using a $\mathcal{M}$-sieve. Resulting Channel images are Bi-polar. Red is used to denote +ve granules and blue -ve granules.*

Applications of sieves

Applied to dynamic shape recognition Recently there have been thorough studies into the relative merits of sieves over the best competitor algorithms. For example, the 1D sieve can generate features for lip reading (hidden Markov models implemented in HTK) and the results remain the best so far, first outlined in [30] and culminating in [57].

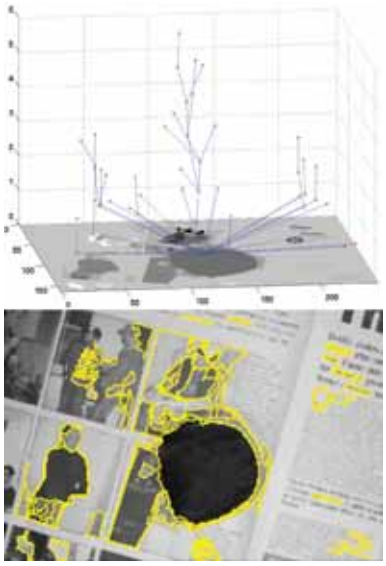


Figure 5.11: *Bottom: an image with its associated stable salient contours. These contours can be assembled into a saliency tree (top) which is an edited version of the full sieve tree.*

Applied to texture recognition The identification of textures is a longstanding computer vision problem. Particularly, anisotropic textures such as cloth where what it looks like depends on the viewing angle. Table 5.1 shows result suggesting that the 2D-sieve is the best. Texture is characterised by sieving each texture image to scales, $[s_1 \dots s_N]$ where $\log_{10} s_n$ are equispaced between 0 and $\log_{10} S_{max}$, S_{max} is the maximum chosen scale and N the number of sieved images.

Tex-Mex features are formed from statistics derived from channel images. Noting that the setting $S_{max} = 30$ removes all the texture from the images and setting $N = 5$ results in five images sieved to scales $[1, 2, 5, 13, 30]$. Five channel images are formed from these sieved images at scales 0 to 1, 1 to 2, 2 to 5, 5 to 13 and 13 to 30. Figure 5.10 shows some example sieved images and resulting channel images. The intensity of the granule, or channel, images as a function of scale is an indicator of the scale-distribution of the texture features.

Applied to feature point detection Early explorations into using sieves to help match stereo pairs of images [33, 61, 60, 62, 16, 21] have been overtaken by a method that uses 'Maximal Stable Extremal Regions', MSER [56]. However, recent work shows that

the MSER appears to be a special case of the sieve (an opening sieve). Actually, alternating sequential and recursive sieves are found to be still more robust [R. Harvey and Y. Lan, in preparation]. Harvey also made a systematic comparison of algorithms [c.f. K. Mikolajczyk, C. Schmid, *A performance evaluation of local descriptors* Technical Report Oxford, to appear PAMI] that show that sieve algorithm outperforms competitor methods for finding salient regions.

Image retrieval Current interest is in finding regions in the image that are likely to remain unaffected by noise, projective transformations, compression and lighting change. In a comprehensive set of trials [58] a type of region known as Maximally Stable Extremal Regions (MSERs) were found to be the best performing. It turns out that MSERs are generated by a variant of the sieve algorithm known as open/close-sieves. It is therefore possible to parse a sieve tree and to generate *Stable Salient Contours* (SSCs) which are carefully selected nodes from the sieve tree that have all the stability and robustness properties associated with MSERs. Thus, as in Figure 5.11 the sieve tree generates stable regions "for free".

Figure 5.12: *Highlights and lowlights (extrema) are important in pictures. So too are edges, but not too many and not too slavishly accurate.*

Related Recently, the sieve algorithm has started to appear in the technical literature under different names apparently reinvented independently. In addition to the MSER (above) a Dutch team reported an algorithm that partially achieves the sieve [86]. A

French team describes the algorithm, they refer to finding image components [63].

Applied to finding components for artistic pictures It appears, therefore, that sieves form a promising starting point for a number of computer vision solutions. For most of these, however, the sieve is just one component of a number of algorithms that are required for the overall functionality.

However, there is one application area where the algorithm makes a particularly large contribution: forming artistic pictures. It is reported in [19] that a sieve based scale-space decomposition replicates a basic aspect of painting. Namely, it identifies light and dark regions of a photograph and does so at multiple scales. It parallels the way artists represent light and dark at multiple scales, Figure 5.12.

The sieve finds extrema of light and dark, preserves the edges of such areas and does so at multiple scales. Experience in the laboratory and commercial market shows that it contributes greatly to producing good digital art. It is this application of sieves that is pursued in Fo2PiX.

See full technical details on Page 7-2.

Initially, the algorithm was released as a simple Photoshop plugin. However, it quickly became clear that the properties of the sieve lead people to use it not as an 'effect' but as a tool.

The wide variety of applications in which the sieve appears to excel suggests that the algorithm will find wide commercial use as the fast sieve algorithm becomes better known.

For the first time it was possible to extract artistic components from the original photograph that could be used to create pictures. Photoshop actions were constructed to make the process easier but it became clear that a better environment for creating pictures, one that used 'steps', would be extremely helpful, see Page 4-16.

Colour sieves Usually the sieve works on the luminance colour channel (grayscale). There is a conceptual problem extending it to colours since it depends on find an *order* (maxima and minima) and one can only order in one dimension, i.e. one colour channel. However, there is a pragmatic work around. The sieve has recently been extended into the color domain [27, 28] via the use convex hulls to define color extrema which are then merged to their nearest neighbours, found using a Euclidean distance measure. Figure 5.13 shows an example color sieve decomposition of a sample image.

Arguably, it is the best 'posterising' (colour segmentation) result yet seen. However, at present the algorithm itself is too slow to be practical. It needs more engineering.

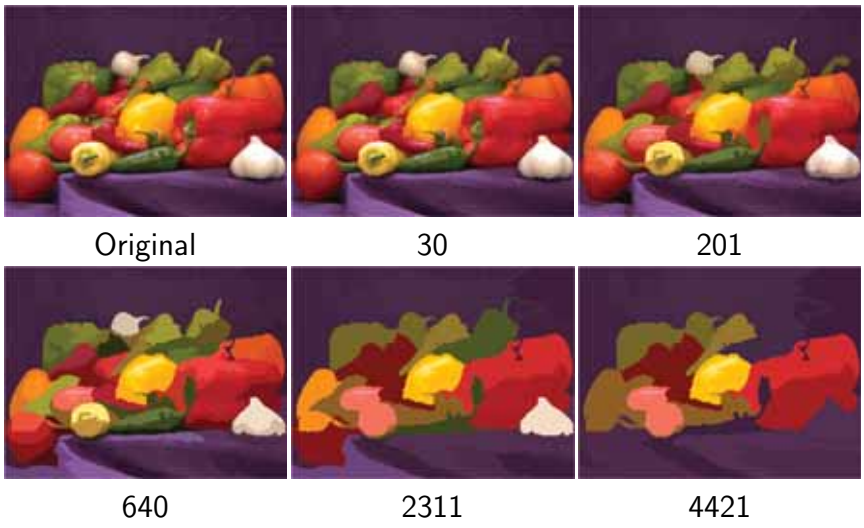


Figure 5.13: *Sample image and RGB space color sieve decomposition to labelled scales.*

Why become creative with digital photos?

Why develop image editors for Art?

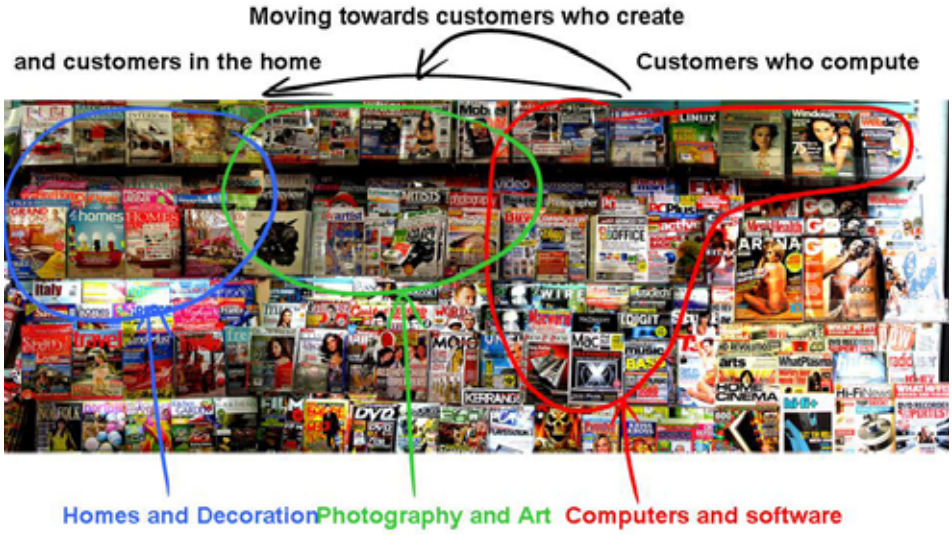


Figure 6.1: *Once only used to* produce*movies, magazine covers, advertisements and illustrations now they are increasingly used to create personalised pictures and decoration. The magazines on the left are increasingly filled with 'how to ... decorative art ... in ten steps'.*

Figure 6.2: *Rubens' daughter: personalised picture but expensive.*

From desktop publishing into the creative home

The price of pictures is falling again. It took an all time world expert, Rubens, months to paint his daughters portrait, Figure 6.2. A person standing would cost 'an arm and a leg' more[1]. Lesser painters took days to produce lesser alternatives. Either way personalised pictures were expensive. Artists responded to a demand for unique pictures. Then of children, wives, mistresses, dogs, horses and estates. Now, just add cars and boats. Unique, because they reflected the lives of individuals.

Technologists respond As soon as they can, technologists also respond to a demand. They produce kit. Then they improve quality and reduce costs. The price of non-personalised pictures, woodcuts, engravings dropped with the advent of the printing press. Even more so for the printed word. Mechanisation had a huge impact.

[1]The artist painted the portrait but subcontracted the arms and legs to a specialist: the origin of the expression

Digital computers, laser and ink-jet printers and desk top publishing software delivered a second blow to the price. They delivered cheap personalised publishing. Experts still do it better, but everyone can now produce reasonably printed material. That leaves pictures.

Figure 6.3: *Cheap photography revolutionised access to individualised images.*

Photography Photographs reduced the price of personalised images fast. Initially the preserve of experts, the Kodak Brownie camera and supporting commercial developing and printing services lead to a century of reducing prices. Photography lead to a new genre of picture. One should distinguish 'snapshots' from the artistic photographic image. Artistic photographs that are sufficiently interesting to hang on the wall are not simply good snapshots.

Figure 6.4: *Photography becoming art (Emmerson, 1890).*

Editing digital Now technologists are precipitating another revolution. Coupled with a new generation of digital printers digital cameras are reducing the cost of personalised images to tenths of cents. Suddenly, one can select the one photograph worth hanging on the wall, from thousands. It is rare to manage perfection straight off. The original is improved with the help of image editing software. Changes are made to red-eye, spots, unwanted objects and bad colour balance. Why stop there? Many direct attention by blurring the backround and masking it out. Why stop there?

Why indeed. Most office pictures and particularly those in the hotels and homes are just that, pictures not photographs.

Figure 6.5: *Often, photographs are the starting point for art (Warhol).*

Slipping into Art Clearly photographs can be art. But artists add more. They might start, and often do, with the projected image, camera obscura, lucida, photograph, and then paint (e.g. Van Eyck to Hockney, etc.[37]). They use skills that most people do not have and the skills, or a lack of them, are a bottleneck for most people. As with desktop publishing, people do not aspire to being better than painters but they do want to be creative, do their own thing and have personalised pictures and decoration.

With Fo2PiX products they can do it. Like the differences between photography, oil painting and watercolours digital pictures are a new genre: desirably interesting, personalised, artistic and accessible. Once professional photographers and others see the software demonstrated they buy. Fo2PiX sales figures uncovered a latent demand.

Figure 6.6: *Now, anybody can produce individualised art but only with the right software.*

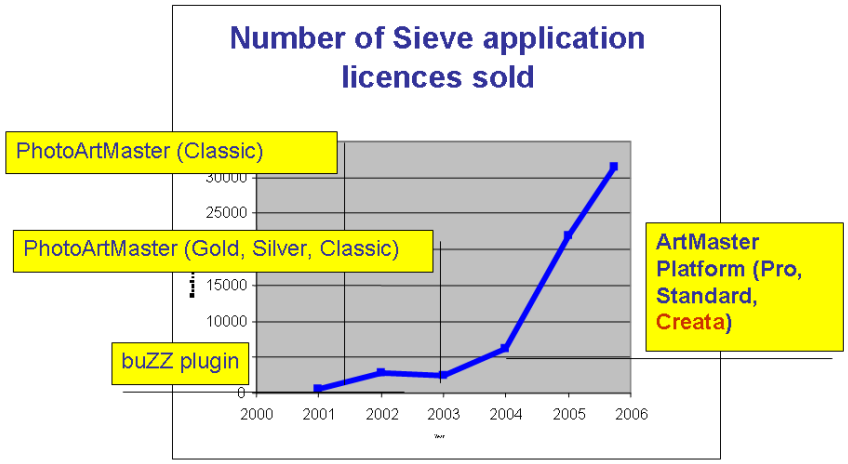


Figure 6.7: *Once people see the Fo2PiX products, they buy. They want to be creative and have personalised art. What is needed now is a bigger marketing machine.*

What can ICE do?

Currently, there are two basic commercial applications derived from ICE. ArtMasterPro, that has been used as an example throughout this document and a cheap, fun version, Creata with functionality limited to a few ArtWizards. New ArtWizards are downloadable for both. Here are a few examples of what ArtMasterPro can do. Of course, the quality depends partly on the skills of the user and partly on the ease with which the user can produce the result. The purpose of the following is to illustrate where some of the algorithms and interface properties are particularly relevant.

Figure 6.8: *This output depends on: the sieve, the blend-pipeline and Gaussian high-pass filtering and the ability to record paint strokes.*

Figure 6.9: *This output depends on: the sieve, Gaussian filtering, Blend-pipeline, image warping.*

Figure 6.10: *Extending the current ICE to include standard editing functions would be useful. Here, the 'torn' edges of the colours that would be output from the current ArtMasterPro are made slightly more substantial by edge enhancement to give a more polished finish.*

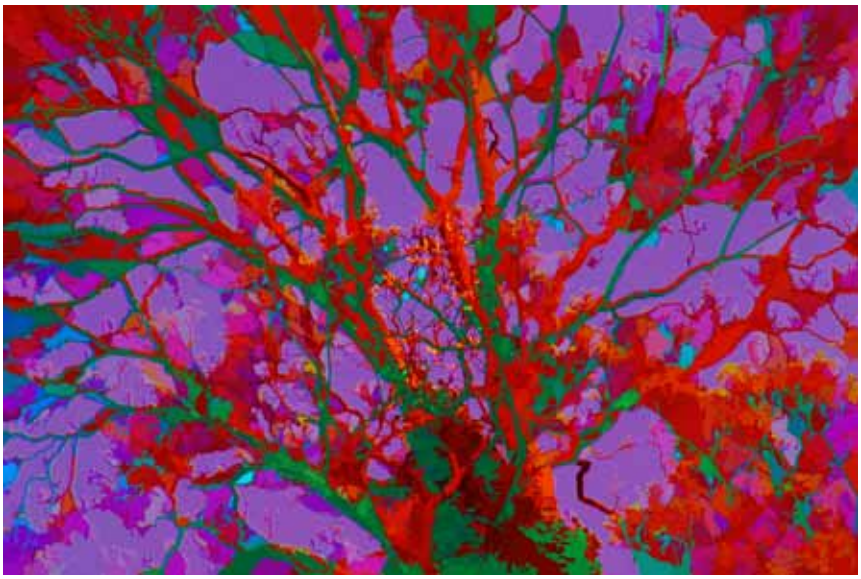


Figure 6.11: *This output depends on: the sieve and the ability to fine tune the balance between colour and tone using* F12*.*

Figure 6.12: *Depends on a photographed pencil texture, sieve, blend-pipeline. The starting photograph was pre-processed in Photoshop.*

Figure 6.13: *Output depends on: the sieve, the blend-pipeline, warped Sources and* F12 *colour checker. The result features complementary colours (face against blue background) and lips in an elusive, tonally invisible, colour that makes their position uncertain. Black produces a strong signal in both the colour and monochrome brain perception systems and 'nails' elusive colours onto the canvas. The eyes assist, they always catch the attention of humans. (When printed, the colour illusion is only visible when the printer is well calibrated.)*

Actually,

- **it is fun**
- **absorbing**
- **rewarding**
- **and printed on A1 canvas, cool**

Algorithms for Art

7 Algorithms for Art

This chapter is extracted from the paper underpinning a prizewinning 'best poster' [19].

Abstract

Artists pictures rarely have photo-realistic detail. Tools to create pictures from digital photographs might, therefore, include methods for removing detail. These tools such as Gaussian and anisotropic diffusion filters and connected-set morphological filters (sieves) remove detail whilst maintaining scale-space causality, in other words new detail is not created using these operators. Non-photorealistic rendering is, therefore, a potential application of these vision techniques. It is shown that certain scale-space filters preserve the appropriate edges of retained segments of interest. The resulting images have fewer extrema and are perceptually simpler than the original. A second artistic goal is to accentuate the centre of attention by reducing detail away from the centre. The process also removes the detail providing perceptual cues about photographic texture. This allows the 'eye' to readily accept alternative, artistic, textures introduced to further create an artistic impression. Moreover, the edges bounding segments accurately represent shapes in the original image and so provide a starting point for sketches.

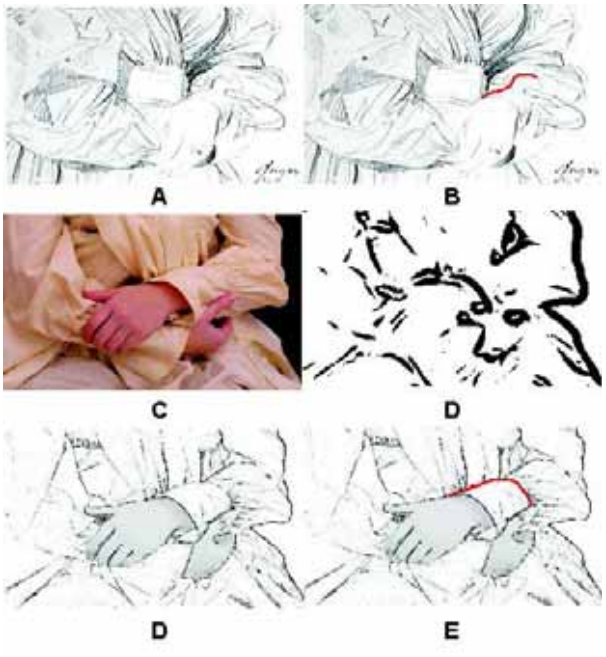


Figure 7.1: *Hockney [37] draws attention to how, in this Ingres drawing (A), "the cuff of the left sleeve is not followed 'round the form' as you would expect, but carries on into the folds". (B) Red overlay indicates the relevant lines. (C) Photograph of a similar subject. A Sobel edge filter (D) does not reveal the artistic line. (E) Shows in red the line line from (D) that follows the cuff rather than the light. There is too much detail. Blurring does reveal the large scale cuff-to-sleeve highlight however it yields edges (D) that are incomplete.*

Introduction

A photographer tends to choose uncluttered backgrounds and make careful use of focus to direct attention. Of course, lens blurring is both easy and effective for it exploits the natural and powerful way in which the brain rejects non-foveated regions of a scene (they are simply out-of-focus). The technique finds its way into rendering, digital art, advertising, and video through the Gaussian blur filter widely used to de-focus background material. But the method is rarely used by painters. Rather, they direct attention by selecting detail and manipulating textures and geometry.

By contrast, a painter starts with a blank canvas, adds paint and the more skilled knows when to stop. It is the progressive addition of detail that characterizes the process of producing representational art in which only some detail directly represents that in the original scene. It difficult to capture representational detail manually from three-dimensional (3D) scenes onto two-dimensional (2D) canvases, but this does not satisfactorily explain why trained artists limit the amount of detail they use. After all two dimensional, photographic quality, images have been traced for over five centuries by those projecting images onto surfaces using concave mirrors and lens [37]). But the evidence from the resulting pictures suggests that artists pick only those details that resonate with their artistic

interpretation. They *choose* to ignore some objects and lots of detail. Painting is not photography.

The importance of controlling detail

Selectively removing detail simplifies a photograph and is implicit in existing methods for producing painterly pictures. Systems for creating pen-and-ink drawings from existing images clearly remove both color, L_c, and spatial detail, L_s, (these two related ideas are lumped together in $L_r = \lambda(L_c, L_s)$) and simultaneously add techniqe and artistic detail in the form of pen strokes [74]. In the case of painting Haeberli *samples* the image and then modulates and spreads the color over a region using a brush, an action that also adds technique detail by simulating the medium and stochastic brush strokes that simulate the artistic interpretation [29]. By modelling the flow of water dragging pigments over paper Curtis [25] removes detail from the original photograph and simultaneously substitutes texture detail (L_t) replicating a watercolor painting. Hertzmann starts by removing detail with large brushes and then uses finer brushes to selectively refine the picture where the sketch differs from blurred photograph [34]. These methods sub-sample the source image either before or after smoothing: the standard way to remove detail and prevent aliasing. In this paper, however, we concentrate on another way to control the level of detail in a digital image. We do not address the separate problems of adding technique and artistic detail.

Of course, form is extremely important and should be exploited in representational art where available, as to an artist working from life or to a digital artist working with a three dimensional graphics system [35]. This, however, is not enough and, anyway, is less easy to come by when starting with a photograph alone. Here, the play between light extrema, light and shade is key. The notion receives some quantitative support from observations on the process of painting, Figure **??**. The artist started with gray paper and subsequent analysis of the fifty images taken at two minute intervals during painting shows that the mean intensity of 80% of paint marks, added by the artist, are more extreme than the mean of their immediate surroundings. In other words, the artist built the portrait by adding ever lighter and darker strokes (indeed it is difficult to see how else it could be done!). Moreover, a quantitative association between light and dark extrema and objects has been reported: when asked to outline objects within photographs, 60% of regions that people demarcated manually were associated

with light or dark extrema [31]. These observations support the contention that control of scale and extrema are important in the creation of pictures from photographs.

The new algorithm decomposes an image by ordering level-connected sets in two ways: first by scale (detail) and secondly by value (extrema of light and shade). It provides the digital artist with access to a choice of differently scaled detail. Unlike blurring, the system simplifies without distorting edges and it can, in principle, be used 'underneath' other painterly algorithms both to simplify the image prior to sampling and to provide, perhaps better, lines for clipping [53].

Selectively removing detail simplifies a photograph and is implicit in existing methods for producing painterly pictures. Systems for creating pen-and-ink drawings from existing images clearly remove both color and spatial detail and simultaneously add artistic detail in the form of pen strokes [74]. In the case of painting Haeberli *samples* the image, effecting a simplification. He then modulates and spreads the color over a larger region using a brush, an action that also adds technique detail by simulating the medium and stochastic brush strokes that are modulated by edge gradients to imply artistic interpretation [29]. By modelling the flow of water dragging pigments over paper Curtis [25] removes detail from the original photograph by a form of blur and simultaneously substitutes texture detail that replicates watercolor. Hertzmann starts by removing detail with large brushes and then uses finer brushes to selectively refine the picture where the sketch differs from blurred photograph, a form of multiscale removal of detail [34]. These methods sub-sample the source image either before or after smoothing: the standard way to remove detail and prevent aliasing. In this paper, however, we concentrate on another way to control the level of detail in a digital image. Scale-space filtering to both remove detail and uncover large scale image maxima (highlight) and minima (lowlights).

Chiarascuro (bright highlights and dark shadows) and its manipulation characterizes the work of many artist's, since the renaissance. Hockney [37] draws attention to the way the line used by Ingres follows the light rather than the form (as evidence of optical assistance of which he gives many other examples). Figure 7.1(A) shows an extract from the original drawing of Madame Godinot 1829. (B) Shows the artist's lines that, Hockney argues, follows the light. We illustrate the problem by analysing the photograph shown in (C).

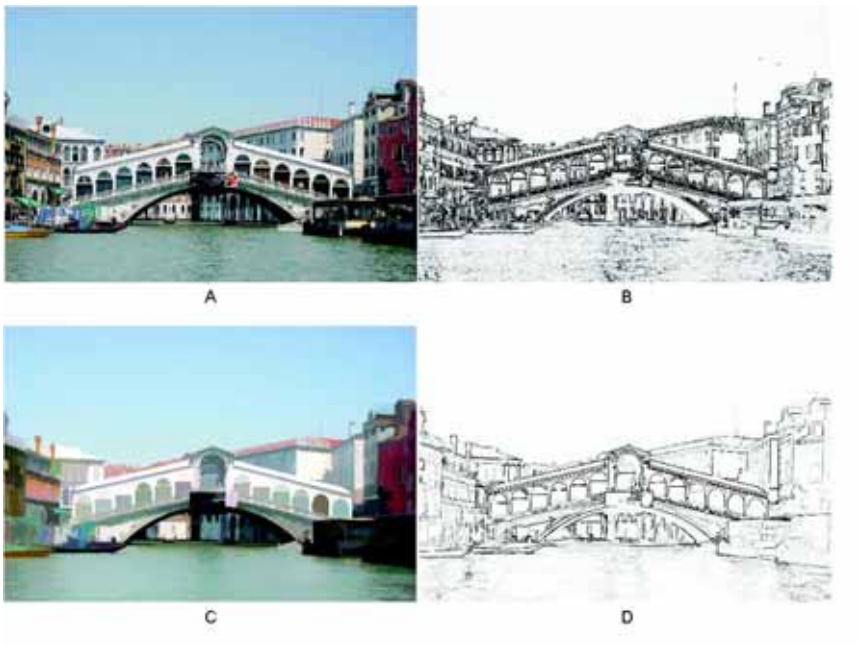


Figure 7.2: *(A) Photograph and (B) associated edges. (C) Sieved to remove detail and (D) fewer edges make a more sketch-like picture.*

Conventional edge detectors (D and E) produce a prominent line along the back edge of the cuff (F): a boundary that was ignored by the artist. The problem lies with the local edge filter. Typically they have a small region-of-support that responds to the strong edges around the form and so cannot 'see' the larger picture (the Canny filter (D) is more complex but has related problems). Simplifying the image by blurring, Figure 7.1(G), increases the region-of-support and does both reveal the expected large scale highlight running from the cuff *into* the sleeve but it removes detail.

Thus Gaussian scale-space filters meet two requirements of a pre-processor for non-photorealistic rendering. As such it is used to segment images and create pictures where the artist's eye-gaze governs the level of detail rendered at different positions in the image [26]. Whilst pleasing, the results are limited in the range of styles they can support because such filters introduce significant geometric distortion reflected in the edges (H) that, whilst graphically interesting, do not form the basis of a sketch: important to the artist. Here we pursue alternative scale-space filters.

Simplification maintaining scale-space causality

In image processing the process of removing detail from a digital image emerged from studies on finding *salient*, edges [55]. The work with Gaussian filters lead to the, theoretically tidy, representation of images known as scale-space [39, 89, 50]. This

important concept is seen as a requirement of image simplification systems since it guarantees that extrema in the simplified image are not artifacts of the simplification process itself. Computation systems that preserve scale-space causality are usually associated with Gaussian filters [1] and diffusion [65] in which the image forms the initial conditions for a discretization of the continuous diffusion equation: $\nabla \cdot (c \nabla f) = f_s$. If the diffusivity is a constant this becomes the linear diffusion equation, $\nabla^2 f = f_s$ which may be implemented by convolving the image with the Green's function of diffusion equation: a Gaussian convolution filter. Of course, care is needed when this equation is discretized [51] but, if it is done correctly, a scale-space with discrete space and continuous scale may be formed[1].

Approximations to this Gaussian blur filter are common in image-editors and graphical rendering systems. The problem with blurring, when finding salient edges at large scales, is that edges wander away from the true edge and objects become rounded: a consequence of convolution, Figure 7.1H. It is better if diffusivity depends upon contrast, as in anisotropic diffusion, but computation then becomes lengthy and unwanted small scale detail with a high enough contrast may nevertheless be preserved. In other words, as with linear diffusion, there is an interaction between the intensity and scale of an object.

More recently the multiscale analysis of images has been explored in the field of mathematical morphology. Two rather different approaches to constructing a morphological scale-space have been suggested. In the first [80, 47] the image is either eroded or dilated using an elliptic paraboloid. As is often the case in morphology (and convolution filters) the shape of the structuring element (window) dominates over structure in the image. That said however, the brush like 'texture' introduced by the structuring element can be useful in digital art and is used in photo-editor plug-ins (Adobe Photoshop Gallery Effects).

The second approach uses those connected-set alternating sequential filters sometimes termed sieves [11]. Sieves [15] appear in a variety of guises but they have their starting point in connected-set graph morphology [72, 82, 83] and watersheds [76]. At small scale they filter out maximally stable extremal points [46] or detail and at larger scale, entire objects. Figure 7.2(C) confirms that fine detail is removed and that edges (D) of remaining features are well preserved. These edges are more sketch-like than those derived

[1] Or with a discrete scale parameter if preferred.

directly from the image (B) or from a Gaussian smoothed image Figure 7.1(H). This, and the more poster-like simplified image provides a reason to explore further how these scale-space filters can be used in non-photorealistic rendering.

Methods

We implement the sieve described in [11]. The algorithm first creates a list of all maxima and minima. These extrema are level 8- (or 4-) connected-sets of pixels that are then sorted by area. A scale decomposition progresses by merging all extrema of area 1 to the next most extreme neighbouring pixel(s), i.e. all extreme values are replaced by the value of the next most extreme adjacent pixel. If the segment remains an extremum it is added to the appropriate scale extremum list. The decomposition continues by merging all extrema of scale 2, 3 and so on. Thus, for example, by scale 100 there are no maxima (white) or minima (black) areas of less than 100 pixels. We use low-pass, band-pass and high-pass filters created by combinations of sieving operations.

The image is represented as a graph [81] $G = (V, E)$. The set of edges E describes the adjacency of the pixels (which are the vertices V). A pixel, x, is connected to its eight neighbours. A region, $C_r(G, x)$, is defined over the graph that encloses the pixel (vertex) x, $C_r(G, x) = \{\xi \in C_r(G) | x \in \xi\}$ where $C_r(G)$ is the set of connected subsets of G with r elements. Thus $C_r(G, x)$ is the set of connected subsets of r elements that contain x. For each integer $r \geq 1$ the operators ψ_r, γ_r, $\mathcal{M}^r$, $\mathcal{N}^r : \mathbf{Z}^V \rightarrow \mathbf{Z}^V$ are defined as $\psi_r f(x) = \max_{\xi \in C_r(G,x)} \min_{u \in \xi} f(u)$,, $\gamma_r f(x) = \min_{\xi \in C_r(G,x)} \max_{u \in \xi} f(u)$, $\mathcal{M}^r = \gamma_r \psi_r$, $\mathcal{N}^r = \psi_r \gamma_r$. $\mathcal{M}^r$ is a connected-set grayscale opening followed by a closing defined over a region of size r.

The types of sieve known as M- or N-sieve are formed by repeated operation of the $\mathcal{M}$ or $\mathcal{N}$ operators that are also known as connected alternating sequential filters. An M-sieve of f is the sequence $(f^{(r)})_{r=1}^{\infty}$ given by

$$f^{(1)} = \mathcal{M}^1 f, \quad f^{(r+1)} = \mathcal{M}^{r+1} f^{(r)}, \quad r \geq 1 \tag{7.1}$$

The N-sieve is defined similarly. It has been shown how connected set openings can be performed in approximately linear time [87] using a modification to Tarjan's disjoint set algorithm and a similar implementation is used here for the alternating sequence of openings and closings that forms the sieve.

$f^{(r)}$ is a low-pass filter removing all extrema up to scale r. $f^{(1)} - f^{(r)}$ is a high-pass filter keeping all extrema from scale 1

to r and $f^{(s)} - f^{(r)}$ is a band-pass filter for extrema (granules) of scales between s and r.

The sieve requires two orderings. Level connected-sets are ordered by value and extrema are removed in order of scale. Where a pixel represents a triple, red, blue and green (RGB), there is no clear way of jointly ordering by value. This is addressed in two ways. We note that all three channels have a high correlation with brightness (unlike hue, saturation, value) and so the three colour planes are sieved independently, the RGB-sieve. The effect of removing detail can be seen by comparing Figure 7.2(A) and (C). It is evident that (B) has less detail yet, in contrast to alternative scale-spaces, the edges of large scale objects are preserved. This is, perhaps, more obvious in the edge images compare Figure 7.2B and D. The resulting image is both grayer than the original (extrema are removed) and the colours change slightly because they arise from the signals obtained from colour channels sieved independently. There is no link between a pixel and its colour.

A new alternative is the convex 'colour sieve' which follows from a geometric interpretation of the colours of a region and its neighbours. A convex hull is fitted to points in the region projected into colour space. All points that lie on the convex hull itself are extreme (ref. to anonymous paper here) and those enclosed are not extreme. This provides an ordering - the distance from the convex hull. This definition is tidy because many typical colour transformations such as gamma correction and linear transformations affect the geometry but not the topology of the convex hull and the extrema inherit the invariance properties.

To simplify the image we merge smaller regions into larger ones without introducing additional extrema by merging to the neighbour with the closest Euclidean distance. Neighbouring regions with identical colour distances are further ordered by computing the difference of their luminance $L = (r + g + b)/3$ and further tiebreaks are achieved by ordering by their G,R and B values. The merging is repeated iteratively until idempotence.

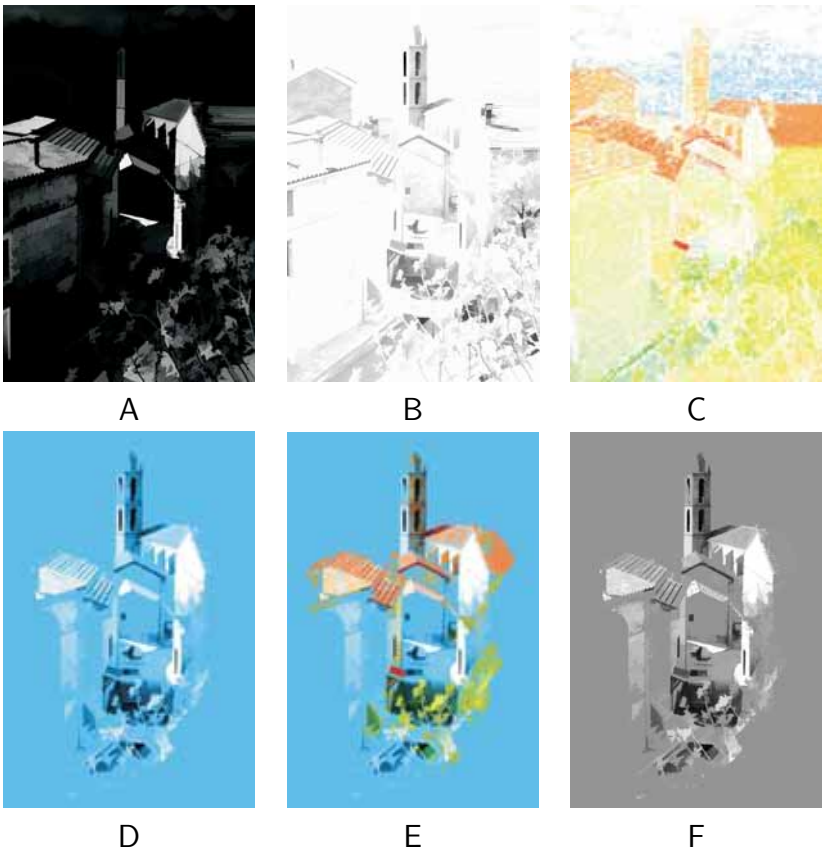


Figure 7.3: *'(A) Band-pass highlights displayed against a black backround. (B) Band-pass lowlights displayed against a white backround. (C) Band-pass HSV saturation channel used to select colours, such as the red roofs, that stand out from backround. (D) Highlights and lowlights incorporated into a picture. (E) Replacing the chroma of (D) with colours from (C) adds colour highlights (visible in colour prints, pdf only) that are not visible in the luminance image (F).*

Results

We consider how high- and low- lights extracted from the image using a high-pass sieve can be incorporated into non-photorealistic image renderings. The grayscale image is band-pass sieved, $q = f^{(s)} - f^{(r)}$, to find the associated scale highlights, where $q > 0$, $y = q$, else $y = 0$. Figure 7.3A shows the result. Likewise, lowlights where $q < 0$, $y = 1 + q$, else $y = 1$, Figure 7.3B. Combined by painting them onto a mid-tone background, Figure 7.3D, the effect is similar to chalk and charcoal. Colour highlights are located at a particular range of scales by sieving the HSV saturation channel and using this to control the chroma, $q_{hue} = f^{(s)}_{hue} - f^{(r)}_{hue}$, where $q_{hue} > t$, $hue = q_{sat}$, $sat = q_{sat}$, $val = 1$, else $hue = 1$, $sat = 1$,

$val = 1$, where t is a threshold that can be adjusted by the user[2]. The colour highlights have been painted by *replacing* the NTSC chroma values on the canvas with those from the colour highlights, Figure 7.3E[3]

Interestingly, as Livingstone [54] points out, by colouring the canvas with the complement (the two colours sum to gray) of, for example, the red roofs an optical illusion is created. The effect is to make the colour appear more interesting that it otherwise might be for two reasons. Firstly, it challenges the viewer's vision system (and monochrome display devices) because the NTSC grayscale (perceptual luminance) does not change even when the chrominance does: all trace of the colour change vanishes in an NTSC grayscale print, Figure 7.3F. Exactly how Figure 7.3E appears on the printed page depends on the printer software. For readers able to see colour this in colour, Figure 7.3E plays to another colour illusion. The sharp boundary between the complementary colours enhances perceived brightness [54].

The brushwork in Figure 7.3A-F places the centre of attention in the centre of the picture by leaving the periphery free of detail. This is typical of many paintings. We, therefore, devise an algorithm that automatically selects a central region to be rendered in more detail than a middleground which, in turn is set against a backround with low detail. In other words, an algorithm that creates foreground, M_f, and middleground, M_m masks.

The process is outlined in Figure 7.4B. The idea is to create masks that exactly follow the boundaries of objects in the image and which place M_f in the centre and M_m around it. Each mask is created separately. The image is sieved to a scale, s, quantised by an amount q and the flat zones labelled. Those zones that intersect the innermost darker-cross and the pale-cross are then marked as shown by the white segments, Figure 7.4B. It white segment has an area A. The part of the pale-cross not covered by the marked zones has an area $\overline{A}$. We then search for a scale, s, and quantisation q that minimises difference between the areas, $A - \overline{A}$. An exhaustive search of only a few s and q suffices. Typical masks for M_f and M_m are shown in Figure 7.4B bottom panel and they have been used to combine images created by RGB-sieving to three scales, Figure 7.4C. The result is more detail towards the centre of the image helps draw the viewers attention. The changes

[2]For digital artists the convention, that user-adjustable thresholds should be avoided, is not relevant.

[3]PDF version of the paper.

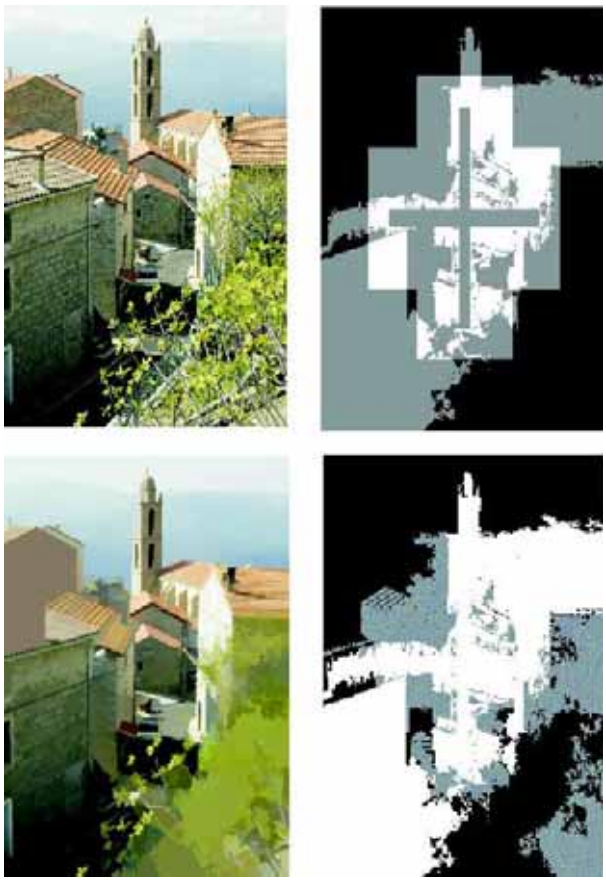


Figure 7.4: *Photograph. (B) Top, the central cross designating the region of primary interest together with its border. Bottom, white segment indicates the region automatically selected to be foreground resolution, gray segment at middleground resolution, black at background resolution. (C) The union of foreground, middleground and background resolution image segments. Controlling the level of detail helps direct attention to the interest points in the centre.*

A B

Figure 7.5: *'(A) Figure 7.4 textured using a photograph of a water-colour wash and pencil cross-hatching (B).*

of scale are subtle since they follow the boundaries of objects in the image rather than some externally imposed mask.

Notice that many of the areas in Figure 7.4C are flat because texture, fine detail, has been removed by the sieve. This creates an opportunity to replace the original texture with another as an artist might do by using paint or pencil. The image was mixed with a photograph of a simple watercolour wash (not shown) (multiplication rather than addition), Figure 7.5A, produces a distinctly watercolour like result. Mixing the same texture with the original is much less effective (easily achieved in Photoshop) because the underlying original detail leaves old texture cues intact. A more extreme example is shown in Figure 7.5B. Here, a photograph of an area of pencil cross-hatching is mixed with Figure 7.4C. Unlike Figure 7.5A however, each labelled level set in Figure 7.4C is filled with a segment of the cross-hatched picked from a random position in the texture image. In other words each of the objects is hatched separately. This is most clearly seen in the large flat areas top left and bottom right. Superimposing the edges completes the effect.

Conclusion

The sieve, particulary the convex-hull colour, algorithm is a useful starting point for non-photorealist rendering of photographs. It provides the digital artist with access to a choice of images with differently scaled detail. Unlike blurring, the system simplifies without distorting edges thus the edges provide a useful starting

point for creating sketches. The large scale level sets it creates provide a mechanism for segmenting the image into regions that, by having different amounts of detail, create a centre of attention. It is data-driven rather than dependent on a pre-defined geometry. Band-pass sieves also allow artistically important high-, low- and bright coloured highlights to be found. Thus in Figure 7.6A the sieve removes small scale detail and the highlights are now treated in a way that is redolent of Figure 7.1A with edges that follow the light Figure 7.6B. We do not attempt to map photographs directly into art: the artist is still essential. Rather the aim is to provide the digital artist with tools. Further automatation might include object recognition to create ways of improving composition and tools to balance colour composition.

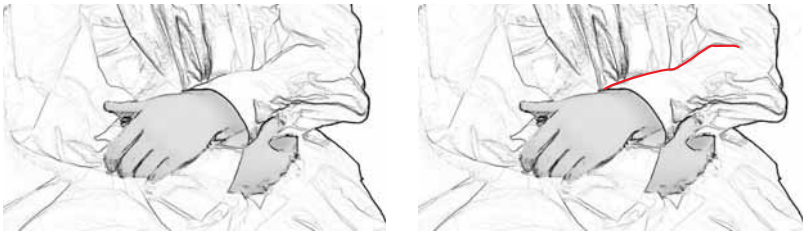
A  B

Figure 7.6: *(A) Sobel edges after sieving RGB Figure 7.1C to scale 2000: the line carries into the folds. (B) Red line indicates the line.*

So what about shape and appearance

Figure 8.1: *Distortions of shape and appearance using active appearance models to make the portrait more like a: teenager, young adult (equivalent to the original), older adult, more feminine, more masculine, Modigliani, Botticelli, Mucha. (Try David Perret's website demonstration* http://www.dcs.st-and.ac.uk/%7Emorph/Transformer/ *)*

Applying Research into Soft Models

Introducing statistical models of images. The ideas applied to the attractiveness of flowers (it is the perception by bees that counts) can be explored using the Matlab toolbox has just been released in association with a paper in Science [85](*http://www.cmp.uea.ac.uk/Research/cbg/Documents/Bangham-Coen-Group/AAMToolbox/AAMToolbox.htm*).

Shape models Two dimensional sieves used for art produce a 'shape free' decompositions of images. This is its great power, the decomposition is affine independent. However, shape is important. The most direct way to analyse and work with shapes is use statistical shape models. These were introduced in botany for

analysing plant parts [38, 71] and recently refined for understanding the evolution of flower colour [85]. Separately, it was developed for analysing medical images [24, 23, 24], speechreading [57], talking heads [79, 78], and tracking people (Matthews). Perret used them to study the psychophysical basis of beauty [66].

Figure 8.1 shows a statistical model developed by Perret distorting a portrait of Emma. Technically, a statistical analysis of shape and shape-free appearance is used to create a model. The images are then generated by the model from different positions in the shape-appearance space.

Figure 8.2 shows the output from a similar statistical model. This time created by analysing a set of photographs. To reshape the original the model was fitted to the original then the result synthesised by shifting closer to the mean shape, keeping the appearance the same.

Figure 8.2: *Using the model to rotate a photograph.*

Applications to photography and digital art It would be desirable to apply the technique to photography and creating artistic pictures. One can imagine 'retouching' photographs by slightly enlarging the eyes, increasing the smile or changing the pose and one can imagine producing pictures with the 'flavour' of old masters (it will not work well, if only because an inkjet print of an oil painting is lamentable). The problem is size. There is a practical constraint on the size of image that can be modelled using the current methods. There are, however, at least two ways in which this limitation can be overcome and these form the subject of current research at UEA.

Figure 8.3: *Shape model definition.*

How the models are created in Matlab Our toolbox for generating statistical shape and appearance models is available together with sample botanical data (http://www.cmp.uea.ac.uk/Research/cbg/Documents/Bangham-Coen-Group/AAMToolbox/AAMToolbox.htm). A set of portraits were centred in a 400x400 pixels image. A point model was created by dotting 199 points around the face, hair, shoulders and hat, (Figure 8.3). Primary points (black) were at easily recognisable features, the corners of the mouth, nose, etc. Secondary points (circles) were equally spaced between the primary points (the pmplace routine helps by automatically sliding the points long a cubic spline fitted to the points).

The positions of n points for each leaf ($[X_j, Y_j], j = 1, , n$) were manually selected using "pmplace" function of the 'Shape

model toolbox' that automates the rest of following process. The points for each portrait are saved in separate files each containing $2n$ data values. The mean shape is calculated from M portraits, $[\bar{X}_j, \bar{Y}_j]$ where $j = 1, \cdots, N$, and the mean $\bar{X}_j = \sum_{i=1}^{i=M} X_{i,j}/N$ and likewise for Y (ignoring the distinction between primary and secondary points). Differences between shapes associated with differing species are reflected in the way portraits shapes differ from the mean. This is captured by subtracting the mean from each point $D_{i,j} = X_{i,j} - \bar{X}_j, D_{i,n+j} = Y_{i,j} - \bar{Y}_j$, notice that the X and Y differences are concatenated into a single data vector forming a column of $2N$ values. D is a $2n$ column by M row data matrix where each row represents a leaf and each column a set of measurements.

The measurements are correlated, i.e. if one compares a wide portrait with a narrow one, adjacent points tend to differ in similar ways. In other words, the measurements do not provide a compact description of shape. To find a compact linear description of shape we can construct the smallest set of linearly independent vectors that span the space of interest. To find independent (orthogonal) measures of shape, the differences for each image, D_i, are represented as a linear combination of orthogonal principal components $D_i = b_{i,0}\mathbf{p}_0 + b_{i,1}\mathbf{p}_1 + b_{i,2}\mathbf{p}_2, + \cdots, b_{i,2n-1}\mathbf{p}_{2n-1}$ where p_l is the first principal component and $b_{i,l}$ is a weight. Thus each portrait shape, i, has a vector of weights b_i. To the extent that D can be represented linearly in this way (there may be underlying non-linearities) the weights, $b_{i,j}$, associated with portraits i are j independent measures of shape that can substitute for $D_{i,j}$.

Principal component analysis Principal component analysis (PCA) is used to find P where $P = [p_0 p_1 \cdots p_{2n-1}]$, such that $\mathbf{b}_i = \mathbf{P}' D_i$ where the superscript tick, $(.)'$, denotes the transpose (a capital T is an alternative). The components are ordered to account for decreasing variance and it is found that a good representation of the shape of the portrait, i, can be made using the weights of just the first three components $[b_{i,0} b_{i,1} b_{i,2}]$. These account for most of the variance of shape about the mean shape. The estimated shape, $\hat{D}_i$, corresponding to just these components, $\hat{b}_i = [b_{i,0}, b_{i,1}, b_{i,2}, \cdots, b_{i,2n-1}]$, can be found from $D_i = \mathbf{P}\mathbf{b}_i$, Figure 3. $\mathbf{P}$ is called the Point Distribution Model (PDM) and it is obtained from $\mathbf{D}$ in Matlab by finding the covariance matrix, $C = cov(D)$, the eigenvectors $(\mathbf{E})$, where $[\mathbf{E}, \mathbf{V}] = eig(\mathbf{C})$, and by sorting $\mathbf{E}$ by decreasing importance according to the eigenvalues,

$\mathbf{V}$ (the covariance of $\mathbf{E}$). Thus, [vals, I]=sort(diag(V), 'descend') and the PDM is the sorted eigenvalues, $\mathbf{P} = \mathbf{E}(:, \mathbf{I})$. To find $\mathbf{b}$ from $\mathbf{D}$ in Matlab use $\mathbf{b}(i, :) = \mathbf{P}' * \mathbf{b}(i, :)$.

Bibliography

[1] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8:pp 26–33, 1986.

[2] J. A. Bangham. *Improved filtering techniques: median filter and datasieve*. Patent, 1987.

[3] J. A. Bangham. Data-sieving hydrophobicity plots. *Anal. Biochem.*, 174:694–700, 1988.

[4] J. A. Bangham. Using the root datasieve to recognise and locate patterns in a data sequence. In *Proc. EUSIPCO'92*, pages 983–987, 1992.

[5] J. A. Bangham and R. V. Aldridge. Multiscale decomposition using median and morphological filters. In *IEEE Workshop on Nonlinear Signal Processing*, pages 6.1–1.1 – 6.1–1.4, Tampere, Finland, 1993.

[6] J. A. Bangham and T. G. Campbell. Sieves and wavelets: multiscale transforms for pattern recognition. In *Proceedings IEEE Workshop on Nonlinear Signal Processing*, pages 1.1–4.1 –1.1–4.6, 1993.

[7] J. A. Bangham and T. G. Campbell. Sieves and wavelets: multiscale transforms for pattern recognition. In *IEEE Workshop on Nonlinear Signal Processing*, pages 1.1–41 – 1.1–4.6, Tampere, Finland, 1993.

[8] J. A. Bangham, T. G. Campbell, and R. V. Aldridge. Multiscale median and morphological filters used for 2d pattern recognition. *Signal Processing*, 38:387–415, 1994.

[9] J. A. Bangham, P. Chardaire, P. Ling, and C. J. Pye. Multiscale nonlinear decomposition: the sieve decomposition theorem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18:518–527, 1996.

[10] J. A. Bangham, P. Chardaire, and P. D. Ling. Multiscale nonlinear decomposition. In *International Conference on Mathematical Morphology*. Kluwer, 1994.

[11] J. A. Bangham, R. Harvey, P. Ling, and R. V. Aldridge. Nonlinear area and volume scale-space preserving filters'. *Journal of Electronic Imaging*, 5(3):283–299, July 1996.

[12] J. A. Bangham, R. Harvey, P. Ling, and R. V. Aldridge. Nonlinear scale-space from *n*-dimensional sieves. In *Proc. European Conf. on Computer Vision*, volume 1, pages 189–198. Kluwer, 1996.

[13] J. A. Bangham, S. J. Impey, and F. W. D. Woodhams. A fast 1d sieve transform for multiscale signal decomposition. In *EUSIPCO*, pages 1621–1624, 1994.

[14] J. A. Bangham, P. Ling, and R. Harvey. Scale-space from nonlinear filters. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 18(5):520–528, 1996.

[15] J. A. Bangham and S. Marshall. Image and signal processing with mathematical morphology. *IEE Electronics and Communication Engineering Journal*, 10:117–128, 1998.

[16] J. A. Bangham, K. Moravec, R. W. Harvey, and M. Fisher. Scale-space trees and applications as filters, for stereo vision and image retrieval. In *British machine vision Conference*, pages 113–122, 1999.

[17] J. A. Bangham, J. Ruiz-Hidalgo, R. Harvey, and G. Cawley. The segmentation of images using scale-space trees. In *BMVC*, pages 1–8, 1998.

[18] J.A. Bangham, T. G. Campbell, and M. Gabbouj. The quality of edge preservation by non-linear filters. In *IEEE Workshop on Visual Signal Processing and Communications*, pages 37–39, 1992.

[19] J.A. Bangham, S.E. Gibson, and R.W. Harvey. The art of scale-space (supporting conference 'best poster' prize). In *British Machine Vision Conference*, volume 1, page 10 pages, 2003.

[20] J.A. Bangham, P.D. Ling, and R. Young. Multiscale decomposition by extrema: the recursive median datasieve and its relation to other sieves. In *IEEE Workshop on nonlinear signal and image processing*, pages 1038–1041, 1995.

[21] J.A. Bangham, K. Moravec, R.W. Harvey, and M.H. Fisher. Scale-space trees and applications as filters, for stereo vision and image retrieval. In *10th British Machine Vision Conference (BMVC99)*, Nottingham, UK, 1999.

[22] P. Chardaire, J.A. Bangham, C.J. Pye, and D.Q. Wu. Properties of multiscale morphological filters namely the morphology decomposition theorem spie nonlinear signal processing. In *SPIE*, volume 2180, pages 148–154, 1994.

[23] T. Cootes and C. Taylor. Active appearance models. *Lecture notes in Computer Science*, 1407:484–498, 1998.

[24] T. F. Cootes, C. Taylor, and A. Lanitis. Active shape models: evaluation - their training and application. *Computer vision and image understanding*, 61:38–59, 1995.

[25] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *Proceedings of SIGGRAPH 97*, pages 421–430, Los Angeles, California, August 1997. ACM SIGGRAPH, ACM Press/Addison-Wesley. ISBN 0-89791-896-7.

[26] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. In *SIGGRAPH'2002 Proceedings*, pages 769–776. ACM, 2002.

[27] S. Gibson, R. Harvey, and G. Finlayson. Convex colour sieves. In *in Proceedings 4th International Conference, Scale-Space 2003*, volume 2695 of Lecture Notes in Computer Science, pages 537–549, 2003.

[28] D. Gimenez and A.N. Evans. Colour morphological scale-spaces for image segmentation. In *Proceedings of the British Machine Vision Conference 2005 (BMVC '05), Oxford, England*, volume 2, pages 909–918, 2005.

[29] P. E. Haeberli. Paint by numbers: Abstract image representations. In *Proceedings of SIGGRAPH 90*, pages 207–214, August 1990.

[30] R. Harvey, I. Matthews, J. A. Bangham, and S. J. Cox. Lip-reading from scale-space parameters. In *IEEE Computer vision and pattern recognition conference*, pages 582–587. IEEE, 1997.

[31] R. W. Harvey, A. Bosson, and J. A. Bangham. The robustness of some scale-spaces. In *Proceedings of British machine vision Conference*, pages 11 – 20, 1997.

[32] R. W. Harvey and J. A. Bosson, A. Bangham. Scale-spaces filters and their robustness. In *Proc. of First Int. Scale-space Conf.: Scale-space theory in Comp. vision*, pages 341 – 344, 1997.

[33] R.W. Harvey, K. Moravec, and J.A. Bangham. Stereo vision via connected-set operators. In *Proc. European Signal Processing Conference*, Rhodes, 1998.

[34] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In M. E. Cohen, editor, *Proceedings of SIGGRAPH 98*, pages 453–460, Orlando, Florida, July 19-24 1998.

ACM SIGGRAPH, ACM Press/Addison-Wesley. ISBN 0-89791-999-8.

[35] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 517–526. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[36] Javier Ruiz Hidalgo. The representation of images using scale trees. Master's thesis, Information Systems, University of East Anglia, 1999.

[37] D. Hockney. *Secret Knowledge*. Routledge and Kegan Paul, 2001.

[38] G. W. Horgan. The statistical analysis of plant part appearance - a review. *Computers and Electronics in Agriculture*, 31:169–190, 2001.

[39] T. Iijima. Basic theory of pattern normalization (for the case of a typical one-dimensional pattern. *Bulletin of the Electrotechnical Laboratory*, 26:368–388, 1962.

[40] J. Impey, Stephen, J.A. Bangham, and J. Pye. Multiscale 1 and 2d signal decomposition using median and morphological filters. In *Workshop on Mathematical Morphology and its Applications*, pages 17–21, 1993.

[41] S. J. Impey and J. A. Bangham. Pattern recognition by area decomposition of hsv components'. In *Signal Processing VIII*, volume 1, pages 169–171, 1996.

[42] S.J. Impey and J.A. Bangham. Improvements to the 'top hat' transform used for analysing pigmented patches on flower petals. In *Signal Processing VII: Theories and applications*, pages 852–855, 1994. Holt, Cowan, Grant and Sandham.

[43] Stephen Impey. *Nonlinear scale-space*. PhD thesis, University of East Anglia, 1999.

[44] Pye J., Bangham J A., Impey S., and Aldridge R V. 2d pattern recognition using multiscale median and morphological filters. In Venatsonopoulos Vernazza and Braccini, editors, *Image Processing: Theory and applications*, pages 309–312. Elsevier, 1993.

[45] M. Babaud J. Babaud, A. P. Witkin and R. O. Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 8:532 – 540, 1986.

[46] M. Urban J. Matas, O. Chum and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, volume 1, pages 384–393, Cardiff, 2002.

[47] Paul T. Jackway and Mohamed Deriche. Scale-space properties of the multiscale morphological dilation-erosion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18:38–51, 1996.

[48] A Kay. The early history of smalltalk. *ACM SIGPLAN Notices*, 28:69–75, 1993.

[49] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363 – 370, 1984.

[50] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[51] Tony Lindeberg. Scale-space from discrete signals. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12:234–254, 1990.

[52] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[53] Peter Litwinowiez. Processing images and video for an impressionist effect. In *SIGGRAPH'97 Proceedings*, pages 407–414. ACM, 1997.

[54] Margaret Livingstone. *Vision and Art: the biology of seeing*. Harry N. Abrams, New York, 2002.

[55] D. Marr. *Vision*. W. H. Freeman and Company, New York, 1982.

[56] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine vision Conference*, pages 384–393, 2002.

[57] I.A. Matthews, T. Cootes, J.A. Bangham, S.J. Cox, and R.W. Harvey. Extraction of visual features for lipreading. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):198–213, 2002.

[58] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[59] M.J.Black and G.Sapiro. Robust anisotropic diffusion: Connections between robust statistics, line processing, and anisotropic diffusion. In *Proc. of First Int. Scale-space Conf.: Scale-space theory in Comp. vision*, volume 1, pages 323–326, 1997.

[60] K. Moravec, R.W. Harvey, and J.A. Bangham. Connected-set filters to improve and obtain stereo disparity maps. In *British Machine Vision Conference*, 1998.

[61] K. Moravec, R.W. Harvey, J.A. Bangham, and M.H. Fisher. Using an image tree to assist stereo matching. In *IEEE ICIP'99*, Kobe, Japan, 1999.

[62] Kimberly Moravec. *Stereo*. PhD thesis, University of East Anglia, 2000.

[63] L. Najman and M. Couprie. Quasi-linear algorithm for the component tree. In *IS&T/SPIE Symposium on Electronic Imaging: Vision Geometry XII*, 2004.

[64] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 12(7):629 – 639, 1990.

[65] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[66] D.I. Perrett, K. May, and S. Yoshikawa. Attractive characteristics of female faces: preference for non-average shape. *Nature*, 368:239–242, 1994.

[67] C. J. Pye and Bangham J A. 2d pattern recognition using 1d multiscale sieve decomposition. In *Signal Processing VII, 'Theories and applications*, pages 856–859. EUSIPCO94, 1994.

[68] C. J. Pye, Bangham J. A., and R. W. Harvey. Using a genetic algorithm to adapt 1d nonlinear matched sieves for pattern classification in images. In *SPIE*, volume 2424, pages 48–55. SPIE, 1995.

[69] J. Pye, J.A. Bangham, S. Impey, and R.V. Aldridge. 2d pattern recognition using multiscale median and morphological filters. In Venatsonopoulos Vernazza and Braccini, editors, *Image Processing: Theory and applications*, pages 309–312. Elsevier, 1993.

[70] Jeremy Pye. *Sieves*. PhD thesis, University of East Anglia, 2000.

[71] T. A. Ray. Landmark eigenshape analysis-homologous contours-leaf shape in syngonium (aracaeae). *Am. J. Botany*, 79:69–76, 2002.

[72] P. Salembier and J. Serra. Morphological multiscale image segmentation. In *SPIE Visual Communication and Image Processing*, pages 620–631, 1992.

[73] P. Salembier, J. Serra, and J. A. Bangham. Edge versus contrast estimation of morphological filters. In *IEEE ICASSP-93*, volume vol. V, pages pp 45–48, 1993.

[74] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In A.S. Glassner, editor, *Proceedings of SIGGRAPH 94*, pages 101–108, Orlando, Florida, July 1994. ACM SIGGRAPH, ACM Press/Addison-Wesley.

[75] A. R. Smith. Paint. Technical Memo 7, N.Y.I.T. Computer Graphics Laboratory, Old Westbury, N.Y., July 1978.

[76] P. Soille and L. Vincent. Determining watersheds in digital pictures via flooding simulations. In M. Kunt, editor, *Visual Communications and Image Processing '90*, volume SPIE-1360, pages 240–250, 1990.

[77] Bart M. ter Haar Romeny, editor. *Geometry-Driven Diffusion in Computer Vision*. Kluwer Academic Publishers, 1994.

[78] B. Theobald, J.A. Bangham, I. Matthews, and G.C. Cawley. Near-videorealistic synthetic talking faces: Implementation and evaluation (submitted on invitation). *Speech Communication Journal*, 2004.

[79] B. Theobald, G.C. Cawley, J.R.W. Glauert, J.A. Bangham, and I. Matthews. 2.5d visual speech synthesis using appearance models. In *British Machine Vision Conference*, volume 1, pages 43–52, Norwich, UK, 2003.

[80] R. van den Boomgaard and A. Smeulders. The morphological structure of images: the differential equations of morphological scale-space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11):1101–1113, November 1994.

[81] L. Vincent. Graphs and mathematical morphology. *Signal Processing*, 16:365:388, 1989.

[82] L. Vincent. Morphological area openings and closings of greyscale images. In *Workshop on Shape in Picture*. NATO, 1992c.

[83] L. Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Trans Image Processing*, vol. 2:pp 176–201, 1993.

[84] John Warnock and Chuck Geschke. Postscript, 1984.

[85] A. C. Whibley, N. Langlade, C. Andalo, A. I. Hanna, A. Bangham, C. Thebaud, and Coen E. Evolutionary paths underlying flower color variation in antirrhinum. *Science*, September, 2006.

[86] M. H. F. Wilkinson and J. B. T. M. Roerdink. Fast morphological attribute operations using tarjan's union-find algorithm. In L. Vincent J. Goutsias and D.S. Bloomberg, editors, *Int Symposium on Mathematical Morphology*, pages 311–32. Kluwer Academic Publishers, 2000.

[87] M. H. F. Wilkinson and J. B. T. M. Roerdink. Fast morphological attribute operations using tarjan's union-find algorithm. In L. Vincent J. Goutsias and D.S. Bloomberg, editors, *Int Symposium on Mathematical Morphology*, pages 311–320. Kluwer, 2000.

[88] A. P. Witkin. Scale-space filtering. In *8th International Joint Conference on Artificial Intelligence*, pages 1019 – 1022, 1983.

[89] A. P. Witkin. Scale-space filtering. In *8th Int. Joint Conf. Artificial Intelligence*, pages 1019–1022. IEEE, 1983.

[90] M. E. J. Wood, B. T. Thomas, and N. W. Campbell. Iterative refinement by relevance feedback in content-based digital image retrieval. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 13–20, New York, NY, USA, 1998. ACM Press.

[91] O. Yli-Harja, J. A. Bangham, and R.W. Harvey. State reduction of recursive median filters. In *IS and T/SPIE 11th Symposium on Electronic Imaging Science and Technology*, San Jose, California, USA, 1999.

[92] O. Yli-Harja, P. Koivisto, J.A. Bangham, G.C. Cawley, R.W. Harvey, and I. Shmulevich. Simplified implementation of the recursive median sieve. *Signal Processing*, 81(7):1565–1570, 2001. 446BW SIGNAL PROCESS.

[93] O. Yli-Harja, I. Schmulevich, J.A. Bangham, R.W. Harvey, S. Dasmahapatra, and S.J. Cox. Run-length distributions of recursive median filters using probabilistic automata. In *Scandinavian Conference on Image Analysis, SCIA'99*, Kangerlussuaq, Greenland, 1999.

[94] J. A. Young, R. L.; Bangham. *Improved data processing method and apparatus: datasieve using a recursive ordinal filter*. Patent, 1994.