A Novel Image Viewer Providing Fast Object Delineation for Content Based Retrieval and Navigation

S. T. Perry and P. H. Lewis

The Multimedia Research Group
Department of Electronics and Computer Science
University of Southampton, SO17 1BJ, England.

ABSTRACT

In this paper we describe a novel interactive image viewer incorporating a range of image processing techniques that allows inexperienced users to quickly and easily delineate objects or shapes from a wide range of real world images. The viewer is specifically designed to be easily extensible, and this extensibility is demonstrated with the implementation of an iterative user guided segmentation tool. Using this tool objects can be efficiently extracted from images and used as the basis for navigation and retrieval within MAVIS, the Multimedia Architecture for Video, Image, and Sound.

Keywords: Interactive segmentation, content based retrieval, content based navigation, extensible image viewer

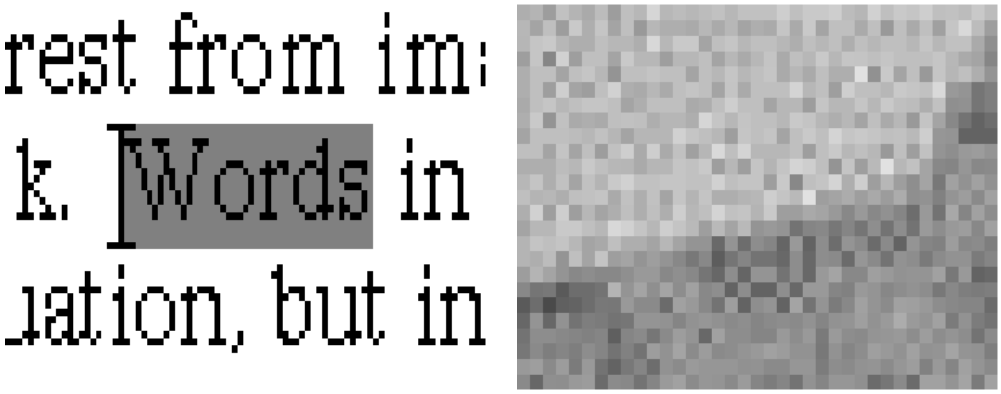
1. INTRODUCTION

Techniques for navigating through and retrieving text documents are widely understood and there are many tools that support their use. By contrast the extension to non-text information such as image, video, and sound is not widely supported at all. This is due to the fact that the fundamental operations of selection and matching are not as well defined as they are for text. Unlike textual media where an object of interest, such as a word or phrase, can be easily selected and separated from the background (the text file), delineating and extracting objects or regions of interest from images, video, or sound files may be a non-trivial task. Words in a text file are clearly delineated by white space or punctuation, but in many cases no such clear distinction exists between features and background in non-textual media. We consider it unrealistic to expect that foolproof automatic segmentation over a wide range of images will become possible in the near future, and so some form of interactive, user guided segmentation will be essential if we are ever to be able to handle objects in images as efficiently and reliably as we can handle text.

When selecting text in an application using some form of pointing device such as a mouse, we typically select an initial character, and then drag a bounding box around the area in which we are interested. When the selection has been made the text within the area is usually highlighted in some way to show that it has been selected, and is available for further processing. If we were to extend this approach to handle the selection of objects in images as well as of words or paragraphs in text, we would expect to able to drag an area around the shape or object, and for that object to then be selected. Unfortunately this is where we encounter one of the fundamental differences between the selection in text, and selection in images. When we are dealing with text the system already has a high level representation of the object that is being selected. It knows its shape, its boundaries and has an easy way in which it may be compared. With images, no such representation exists. All we have is an array of brightness values, from which the system must decide whether a given pixel should be considered part of a shape or not. In addition, text is composed of characters which are relatively uniform in size, and horizontal and vertical extent, whereas objects of interest in images may come in a wide range of shapes, sizes, and orientations. This can be seen from Fig. 1, where the text has easily been highlighted in (a), but (b) shows the fuzzy nature of the boundary between an object and its background.

In their paper on Visual Information Retrieval (VIR),[1] Gupta and Jain identify nine different tools that may serve together as a VIR query language. The first of these is an image processing tool that should allow interactive

Other author information: (Send correspondence to S.T.P.)
S.T.P.: Email: stp94r@ecs.soton.ac.uk; Telephone: +44 1703 594492; Fax: +44 1703 592865
P.H.L.: Email: phl@ecs.soton.ac.uk; Telephone: +44 1703 593715; Fax: +44 1703 592865



(a) (b)

Figure 1. Pixels may easily be classified into object and background when selecting text (a), but the distinction for images is much harder (b).

segmentation and modification of images. Such a tool would seek to overcome the problems of selection in images, and allow the user to accurately extract objects for matching and retrieval. A common approach to interactive object identification is flood filling, which is one of the methods used by QBIC.[2] The main problem with this approach is that of 'leaking', where the area that is filled expands past the boundary of the object, and so interactive pruning and blocking of the area is used to improve results. To overcome this problem of object selection, we have devised a method of allowing a user to extract shapes from images with the minimum of difficulty using a simple point and click interface. The system is implemented using the Generic Image Processor (GIP), a new image viewer which was purposefully designed to be readily extensible so as to facilitate the development of image processing tools, and to ease integration with existing systems. It incorporates a novel approach to viewer design that enables it to remain extremely small and efficient, and yet to be easily extensible in a wide variety of ways with the minimum of effort, and the minimum of disruption to anyone else using, or developing software with, the system.

GIP is described in more detail in the following section, and in section 3 we present our interactive object delineation technique which is based upon assisted splitting and merging of regions, that was developed as an extension to the GIP system. The ultimate goal for this work was to provide a mechanism for the easy extraction of objects from images to be used for Content Based Navigation (CBN) and Content Based Retrieval (CBR) in MAVIS,[3] the Multimedia Architecture for Video, Image, and Sound, the use of which is briefly demonstrated in section 4. Finally, in section 5 we summarise the work that has been undertaken so far and outline some areas for future work and potential improvements to the GIP system, and our interactive object delineation method.

2. GIP — AN EXTENSIBLE IMAGE VIEWER

There are a great number of packages designed for the display and manipulation of images, ranging from relatively simple viewers such as XV, to complex image editing packages such the GIMP and Photoshop. These programs are generally not extensible, and are often often horrendously overlaiden with little used and/or unnecessary features, leading to a situation in which a different program is often used for viewing than is used for processing. Those large packages that are extensible (GIMP, Photoshop, etc.) typically use a plug-in system where code may be dynamically loaded into the program. This means that the code has to be written to a specific API, thus restricting the use of the code to that package and platform, and limiting the functionality of the plug-in to that which is provided by that API. By contrast, our viewer is designed from the outset to be as streamlined and as extensible as possible. It embodies the UNIX programming philosophy that complex systems should be built as a combination of several simpler programs, as this encourages reliability and decreases complexity of design. The system is based on the concept of a radically stripped down core viewer, which by itself provides very little functionality apart from the ability to display an image. It makes no attempt to understand a variety of image formats, it provides no facilities for image editing, or for image processing. It is, in short, an image viewer and nothing else. The key to GIP's extensibility comes through its reliance on external processes for all but the most essential of operations, and the

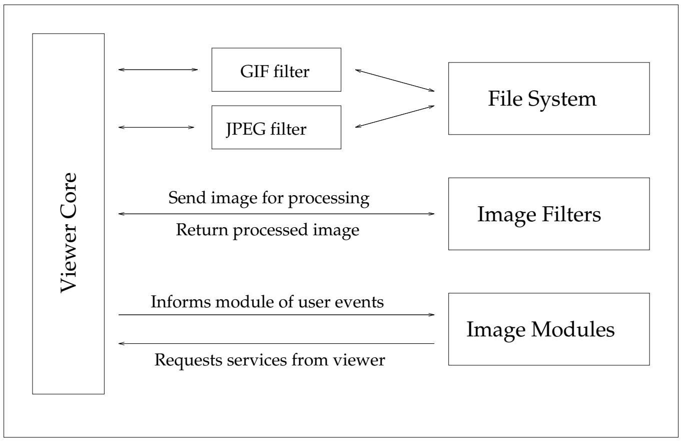


Figure 2. An overview of the GIP system Architecture.

manner in which it provides an easy and flexible way in which these processes may be used to enhance the system. The architecture of the GIP system is described in greater detail in the following section, with section 2.2 showing how basic functionality such as support for multiple file formats, and the processing of images is supplied. More complex processing of images, such as our interactive delineation technique, is enabled with the GIP module system as described in section 2.3

2.1. Architecture of the viewer

The one major facility possessed by the core viewer is the ability to run and to communicate with external processes, and these processes are the way in which extra functionality may be added to the base system. For example, when the viewer is asked to load or to save an image in a format that it does not understand, a process is used to convert the image into a format that it does understand. Similarly, if the user requires to perform some image processing operation on an image, an external process is used, and the result of the operation is displayed in a new window. In total, three types of process are used by GIP to enable it to provide a flexible and comprehensive array of services without becoming prohibitively large or overlaiden with unnecessary features. They are format filters, image filters, and image modules, each of which are described in more detail in the following sections.

When the viewer is started a configuration file is read which contains a specification of the viewer menu hierarchy and lists the various filters and modules that are available for use with the system. By altering this configuration file, the set of operations available to the user from the viewer can be changed to best suit the task at hand. Similarly, adding a new operation to the viewer is simply a case of putting the process in a place where it can be read by the viewer, and then adding it to the configuration file. Typically, image processing operations will require the setting of a number of parameters to be effective. In an application, the separation of the interface from the processing is good software design, and a number of systems exist where interfaces are constructed from text or database information.[4,5] Rather than requiring each process to create its own interface, GIP provides a method whereby it can create a dialogue from a simple description in a text file. Processes that require such information can specify in the viewer configuration file that a dialogue should be displayed, and how the results from that dialogue should be passed as arguments to the process. While this method does not allow full access to the capabilities of the underlying user interface, it greatly simplifies the process of interface construction, and as the interface is stored separately from the process, enables existing command line applications to be easily integrated into the system without any changes. Of course if a process does require an interface beyond the capabilities of the system, it is free to create its own. As can be seen in Fig. 2, the GIP system is effectively separated into two communicating layers — image display and image processing. This separation brings a number of benefits over the traditional integrated approach. The responsiveness and robustness of the viewer are greatly increased, especially when a number of processing operations are taking place. This is particularly useful as some image processing operations may take a long time to complete. In addition, if a developmental process should happen to fail, it has very little chance of adversely affecting the viewer

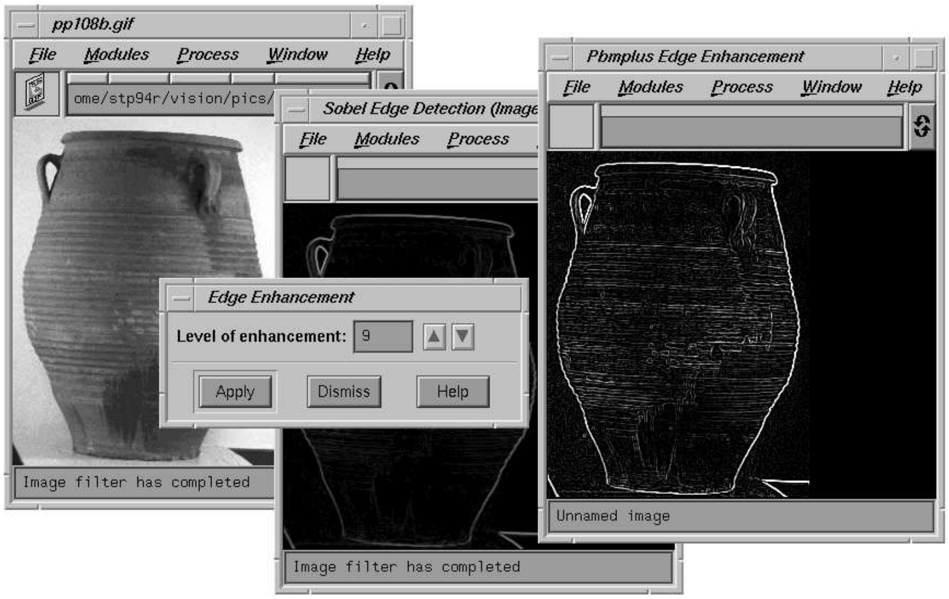


Figure 3. Processing an image using image filters.

as it is an entirely independent process. This is in stark contrast to the use of plug-ins, where the code is loaded directly into the viewer and may easily cause the viewer to crash if something goes wrong. Another major benefit is that the filters and modules available for use in the system may be tailored to suit the particular user or application to a far greater extent than is possible with most viewers. As the configuration of all but the most essential menus is read in at run-time and the set of available options can be easily changed, the viewer never appears 'bloated' or full of features that are not necessary for a particular task, and if more functionality is required another configuration can be chosen and used without ever having to restart the viewer.

2.2. Processing images using filters

GIP allows images to be processed using a filtering concept, with two types of filter being used by the system — format filters, and image filters. Format filters are used to enable GIP to understand a wide range of image formats. When asked to load or save an image in anything other than its native format, the viewer looks for a format filter to handle the conversion. The format filters available for use with the system are specified in the viewer configuration file, and usually consist of two processes, one to convert to GIP's native format, and one to convert from it. When the viewer is asked to load an image in a format that it does not understand, a format filter process is started. This process reads the image and outputs it in the native format, which is read by the viewer and displayed in window. Similarly when an image is to be saved into a non-native format, a format filter process is started which reads the image from the viewer, and saves it to a file. By using external processes to handle these conversions the core of the system is kept as small as possible, and may be easily extended to cope with any new formats, without any recompilation, or even restarting of the viewer.

Image filters are used to provide an image to image processing capability to the system, and enable the viewer to process images in a wide variety of ways. Some typical image processing filters might edge detect an image or perform a histogram equalisation operation, which can be useful when some preprocessing of an image is required before a segmentation or other operation. Given an image in a viewer window, starting an image to image process causes the viewer to start a new filter process and write the image to it. The process manipulates the image in some way, and then outputs a result which is read by the viewer and displayed in a new window. Fig. 3 shows the result of some basic filtering operations on an image — in this case a Sobel edge detection, followed with edge enhancement.

2.3. Advanced operation using image modules

Image modules are similar in concept to image filters, but are far more powerful as they allow a two way dialogue to take place between the viewer and the module. The viewer informs the module of events instigated by the user such as pointer movements and button presses, while the module is able to request services from the viewer such as displaying menus and dialogues, or overlaying some graphics on top of the displayed image. The user guided object delineation system described in section 3 is implemented as a GIP module. A number of modules may be run in a window, although only the module that is currently active will receive user events and have its output displayed.

Modules are the prime way in which the functionality of the system may be extended as they are effectively able to control an image window by responding to user events, and by updating the display. Several modules have been developed for use with the system that are not described in this paper. These include other interactive object delineation tools, such as a shape extraction module suitable for simple polygonal shapes, active contour modules[6] and statistical snakes,[7] and also an MPEG player in which the module generates a stream of images which are displayed through the viewer.

3. USER GUIDED OBJECT DELINEATION

In an ideal world it would be possible to use a mouse to point and click on an object in an image and that object would then be correctly segmented and highlighted, in much the same way as it is possible to click on a word in a text document. Unfortunately this is not yet possible, and is not likely to be for sometime except in circumstances where objects are in some way clearly distinct from their background, or if some prior knowledge of the objects which will be encountered is available. Our object delineation system is written as a module for the extensible image viewer GIP that was described in section 2, and it attempts to correct this problem by providing a number of tools that assist the user in the extraction of objects from images.

The GIP object delineation module consists of a number of tools which may be separated into two categories — those that split an image or regions of an image into smaller regions, and those that take an image consisting of regions and reduce that number by merging. Using only these two fundamental processes it is possible to extract objects from a wide range of images, using a simple intuitive method of the iterative application of the splitting and merging procedures. This approach is somewhat different to the majority of segmentation systems in that the user retains control over the process as it happens and effectively guides the segmentation until the required result is obtained. It is, of course, unrealistic to expect any automated routine to correctly extract complete object boundaries across all images, as the boundary may simply not exist, and for this reason the final option in the delineation process is to manually edit the results of the segmentation and to correct any boundaries that cannot be satisfactorily extracted. In some simple cases the segmentation routines may be able to automatically extract the object correctly without any user interaction, but in the majority of cases at least some editing will be required. In the following sections we describe the methods by which regions may be split and merged, and in section 3.3 we give an example of the system in use.

3.1. Region Splitting

The delineation process starts with a complete image which is broken down into a number of regions using a segmentation algorithm. These individual regions may then split further, or merged together using one of the merging tools. The following section describes the three segmentation algorithms that are currently available for use within the system.

3.1.1. Thresholding

Thresholding can be described as the transformation of an input image f to a binary (segmented) image g such that:

$$g(i,j) = \begin{cases} 1 & \text{for } f(i,j) \geq T \\ 0 & \text{for } f(i,j) < T \end{cases} \tag{1}$$

where T is a brightness contrast or threshold that can be used to discriminate between object and background. A number of techniques exist for the automatic detection of thresholds in images,[8] and the segmentation module uses a technique known as the iterative (optimal) threshold.[9] Although one of the oldest segmentation methods, thresholding is fast and is effective in situations where objects are clearly distinct from their backgrounds, and is often useful for a preliminary segmentation attempt which can be further refined using the other tools.

3.1.2. Region growing

Two types of region growing algorithm may be used in the GIP segmentation module. The first employs a traditional approach in which the user selects a rectangular area of the image that is to be the start of the region that is to be grown. All pixels considered similar to those in the selected region are added to the region, and when no more may be added, the remaining pixels are grown into additional regions that do satisfy the criteria. The homogeneity function used by the region growing algorithm is similar to that given in,[10] and is given below

$$H(p, R) = \begin{cases} \text{True} & \text{if } |p - \bar{x}| \leq k\sigma \\ \text{False} & \text{otherwise} \end{cases} \tag{2}$$

where p is the colour value being tested for homogeneity with region R, and $\bar{x}$ and σ are respectively the mean and standard deviation of the colour values for R. The value k is a homogeneity threshold that may be set by the user to control the amount of merging that occurs.

A more useful interactive region growing algorithm is the Seeded Region Growing (SRG)[11] algorithm. Based on conventional region growing techniques, but in many ways bearing more resemblance to a watershed algorithm, it starts with the selection of a number of seed points to which a region growing technique is applied. Each of the seed points is used as the initial member of a region $R_1, R_2, \ldots, R_n$, and the algorithm then proceeds adding one unassigned pixel to a region in each iteration until there are no more unassigned pixels in the image. This results in a tessellation of the image into the same number of regions as given seed points. The algorithm can be more formally explained as follows. If $N(x)$ is the set of immediate neighbours of the pixel x, then the set T of as yet unallocated pixels which border at least one of the regions can be written as:

$$T = \left\{ x \notin \bigcup_{i=1}^{n} R_i \mid N(x) \cap \bigcup_{i=1}^{n} R_i \neq \emptyset \right\} \tag{3}$$

If, for any pixel $x \epsilon T$, $N(x) \cap R_i \neq \emptyset$ and $|N(x) \cap R_i| \geq 1$, $\delta(x, i)$ can be defined as a measure of how pixel x differs from region R_i. A simple definition for $\delta(x, i)$ is

$$\delta(x, i) = |g(x) - \text{mean}\{g(y \epsilon R_i)\}| \tag{4}$$

where $g(x)$ is the grey level value of the pixel x. In each iteration we take a pixel $z \epsilon T$ such that

$$\delta(z, i) = \min\{\delta(x \epsilon T, i)\} \tag{5}$$

and append z to R_i. This continues until all the pixels in the image have been assigned to regions. The main benefit of this algorithm is that it allows user interaction in the placing of the seed points, and requires no setting of parameters. The user may either select a number of points which will be used as seeds to the process, or they may select two points with the segmentation starting when the second point is selected. If both selected points are within the same region, then the SRG algorithm only processes pixels in that region, effectively splitting it in two. This can dramatically increase the speed of the system, particularly with large images, and is a key part of the interactive splitting and merging procedure in that it provides a way of efficiently dividing a region around the seed points given by the user.

3.2. Region merging

Regions created using the segmentation routines may be merged together at any time using any of the three techniques described in the following sections. Two of these can be described as automatic in that they may be triggered directly after a region splitting operation has completed without any user intervention, while the third is entirely manual and must be instigated by the user.

3.2.1. Area Analysis

Removing regions below a specified size is a simple, but very useful process. The threshold and non-seeded region growing segmentations may well result in a few large regions and a large number of very small regions, caused either by noise or by the fact that they are on the border of inclusion to more than one region. Removing them at an early stage is highly beneficial as they can adversely affect the performance of other operations due to their large numbers. Small regions are removed by merging them with their closest neighbour. The closeness of a neighbour is determined by examining the difference in mean grey levels between the two regions, with the most suitable neighbour being the one with the smallest difference from the region to be removed.

3.2.2. Weak boundary removal

Two heuristics for merging regions based on the edge strength across their boundary have been implemented : the Phagocyte heuristic and the weakness heuristic.[12] For two adjacent regions R_1 and R_2 we consider neighbouring pixels $\vec{x_1}$ and $\vec{x_2}$, on either side of the boundary B. The weakness of the boundary at a given point is then

$$w(\vec{x_1}, \vec{x_2}) = \begin{cases} 1 & \text{if } |f(\vec{x_1}) - f(\vec{x_2})| < T_1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where T_1 is used to threshold the edge strengths. Defining the total weakness $w(B)$ of boundary B to be the sum of the individual weakness values along B, and P_1 and P_2 to be the perimeter lengths of regions R_1 and R_2, the Phagocyte heuristic says that R_1 and R_2 should be merged if

$$\frac{w(B)}{P_1} > T_2 \text{ or } \frac{w(B)}{P_2} > T_2 \tag{7}$$

where T_2 is a weakness threshold. This means that R_1 and R_2 are likely to be merged if one has a perimeter which is short in relation to the size of the weak part of the common boundary. The weakness heuristic says that R_1 and R_2 should be merged if

$$\frac{w(B)}{B} > T_3 \tag{8}$$

where T_3 is another weakness threshold. This means that R_1 and R_2 are merged if a large fraction of the boundary is considered weak. These heuristics may be applied manually, or automatically after every region splitting operation, although the setting of the threshold values may prove problematic.

3.3. Delineation example

In this section we give a demonstration of the system being used to extract an aircraft from an image. Figs. 4 a to c show the six stages of the segmentation process, each of which are described below in greater detail.

(a). As a first attempt, a threshold segmentation has been carried out and some small regions have been removed. While it has not completely extracted the aircraft, it has picked up a fair proportion of it, and so provides a useful starting point for further splitting and merging.

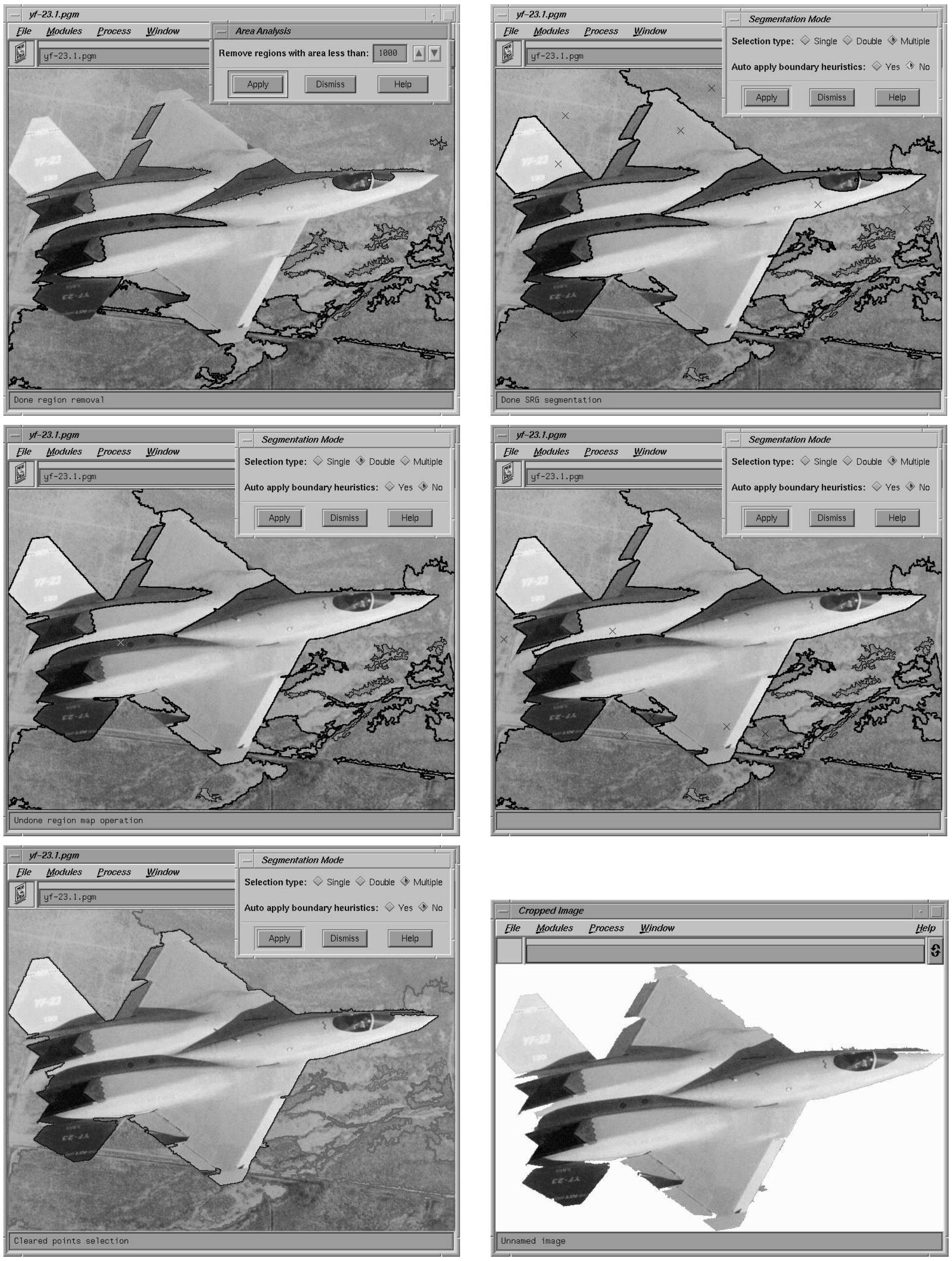
(b). A number of seed points were placed around the boundary of the aircraft, and a seeded region growing operation was instigated. Each seed point grew into one region, and the boundary of the aircraft is now more clearly delineated. It is common to place one seed point within the object, and one outside it along boundaries that need to extracted, as the seeds will typical grow into regions along the boundary.

(c). Some regions internal to the aircraft have been merged to give a better indication of the progress of the segmentation. This is achieved by selecting a region using the mouse and then selecting a number of neighbouring regions using another mouse button, which are quickly merged with the selected region. If a mistake is made a history system is available to allow the user to go back and correct the error.

(d). The seeded region growing algorithm has been used in 'double click' mode, where every two points that are placed in a region are used as seeds to split that region. This method is very useful when there remain a relatively small number of errors to be corrected, as it allows regions to be split very quickly and easily. The entire boundary of the aircraft is now delineated, apart from a small region of the tail which has been incorrectly assigned to be part of the background.

(e). Regions belonging to the aircraft have been merged together, giving the final outline of the boundary. The boundary has been selected, and can now be passed to the MAVIS system for processing, in which case only pixels belonging to the aircraft will be processed. This makes it possible to follow or author links based on characteristics of the aircraft, such as its colour or shape.

(f). Shows the aircraft extracted from the original image. Pixels belonging to the aircraft, and pixels belonging to the background have now been distinguished. There are clearly a number of small imperfections in the boundary of the aircraft, which may be cleared up if required using a manual editing procedure.



(a) (b) (c) (d) (e) (f)

Figure 4. Stages in the extraction of an aircraft from an image, using the object delineation tools in the GIP segmentation module.

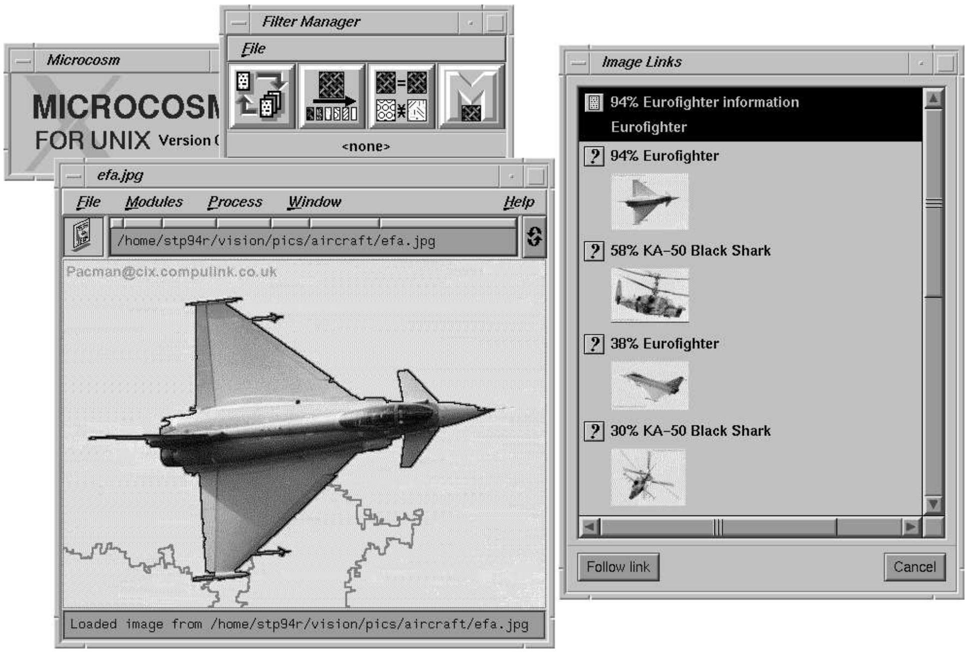


Figure 5. Use of GIP with MAVIS.

Although there may seem to be a large number of stages in the extraction process, each step is in fact very fast and so the overall extraction times compare well with other techniques. The seeded region growing algorithm is highly efficient as it only needs to process pixels within the region that it is splitting, and the region merging process takes effectively no time at all. It is this combination of speed with a relatively large number of interactive stages that makes this method so effective. The fact that the user remains in complete control throughout the entire process and takes an active part in guiding the segmentation to the correct result, in combination with the different tools and the ease with which they may be used together at any time and in any order, means that the user's interactivity with the system is greatly increased, along with the quality of the results that can be achieved with it.

4. CONTENT BASED RETRIEVAL AND NAVIGATION

As mentioned previously, the purpose for this work was to provide an interactive object delineation capability for use with MAVIS.[3] Using the GIP viewer and segmentation module it is now possible to retrieve documents, and author and follow links based upon the characteristics of the extracted object rather than the entire image. At present such characteristics include colour distribution, texture, and outline shape of the object, and may be used together in any combination to be provide a range of query options. In Fig. 5 an aircraft has been extracted from an image using the GIP system, and a follow generic link query has taken place based upon the colour distribution of the extracted aircraft. In the image links window on the right, a number of links to related documents can be seen that were returned by MAVIS as having similar colour distributions. The user may select from the available links, and the destination document of the link will be displayed.

5. CONCLUSIONS

In this paper we have presented our method for the interactive delineation of objects in images with a user guided, point and click, split and merge technique. The system has been implemented as an add on to our extensible image viewer, and has been demonstrated in use, segmenting an image, and acting as a front end to the MAVIS system.

Work continues on the development of both the GIP system and associated modules. While GIP is easily extensible using external processes, improvements can still be made to the module system by opening a wider range of services, primarily including communication between modules. Further work on the delineation module is expected to include the addition of more interactive region editing tools to further assist the extraction process and to improve the quality of the segmentations, as well as some improvements to the user interface.

ACKNOWLEDGEMENTS

Both authors with to thank the EPSRC; the first for the support of a research studentship and the second for support through grant GR/L 03446.

REFERENCES

1. A. Gupta and R. Jain, "Visual information retrieval," *Communications of the ACM* **40**(5), pp. 70–79, 1997.
2. J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic, "Automatic and semi-automatic methods for image annoatation and retrieval in qbic," in *Storage and Retrieval for Image and Video Databases*, *Proc. SPIE* **2420**, pp. 24–35, 1995.
3. P. Lewis, J. Kuan, S. Perry, M. Dobie, H. Davis, and W. Hall, "Navigating from images using generic links based on image content," in *Storage and Retrieval for Image and Video Databases*, *Proc. SPIE* **3022**, pp. 238–248, 1997.
4. P. Kaiser and I.Štetina, "A dialogue generator," *Artificial Intelligence* **12**(8), pp. 693–707, 1982.
5. S. Hudson and S. Mohamed, "A graphical user interface server for UNIX," *Artificial Intelligence* **20**(12), pp. 1227–1239, 1990.
6. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision* , pp. 321–331, 1988.
7. J. Ivins and J. Porrill, "Active region models for segmenting medical images," in *Proceedings ICIP-94*, vol. 3, pp. 227–231, IEEE International Conference on Image Processing, 1994.
8. C. Glaseby, "An analysis of histogram based thresholding algorithms," *Graphical Models and Image Processing* **55**, pp. 532–537, 1993.
9. T. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *IEEE Trans. Systems, Man and Cybernetics* **8**(8), pp. 630–632, 1978.
10. G. Sivewright and P. Elliott, "Interactive regio nand volume growing for segmenting volumes in MR and CT images," **19**(1), pp. 71–80, 1994.
11. R. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. PAMI* **16**(6), pp. 641–647, 1994.
12. C. Brice and C. Fennema, "Scene analysis using regions," *Artificial Intelligence* **1**(3), pp. 205–226, 1970.