A Tool for Extraction of Objects from Digital Images

Dulal C. Kar and Dennis Ma
Texas A&M University-Corpus Christi

Abstract

Many image processing tasks require extracting objects from one image and then combining with another image to create a new one. However, object extraction is a tedious process. A number of disadvantages can be noticed in the traditional process of object extraction. It takes a considerable amount of time and effort to outline the edges of an object before extraction and the process is not very precise, particularly for objects of irregular shapes. There are many commercial image-editing programs that provide the capability for object extraction. One of the most popular programs is Adobe Photoshop. Even though Adobe Photoshop provides a fast edge detection tool, most of the time, the edges found for an object are neither precise nor complete. Furthermore, Adobe Photoshop requires a lot of computer resource and storage to run. In this work, a tool that can be used to extract objects from digital images is presented. By using an extended and modified version of the Sobel edge detection method, the tool detects the edges of the objects in an image and outlines them in the original image. The user has the option to select any object from the outlined objects and save it in a file. The tool reduces the time needed to extract an object from an image. It has a good graphical user interface and extracts objects from an image with good precision. A user can see the original image and the image of the extracted object side by side before saving it in a separate file.

Introduction

In many image processing applications [2], [3], [6], it is required to extract objects from one image and then place and arrange them in another image. However, the manual process of object extraction from an image is tedious and time-consuming, which requires the user to outline the desired object precisely before extraction. The difficulty arises mainly due to irregularity in the shape of the object to be extracted. It is difficult for the user to make sure all the pixels of the object to be extracted are indeed inside the closed outline. The disadvantages in this type of manual process of object extraction can be listed as follows:

- The process is very time-consuming and tedious.
- It requires a great precision and skill for the user to outline the object.
- It is difficult to have a clear perspective on the result until the process is finished.

There are some commercial image-editing programs available that have the capability for object extraction. Adobe Photoshop is one of them. It provides filters and image editing functions to the user and is very popular among many image processing professionals. However, object extraction using Adobe Photoshop remains a tedious work on irregularly shaped objects. Even

Copyright © 2005, American Society for Engineering Education

though Adobe Photoshop provides a fast edge detection tool, most of the time, the edges found are neither precise nor complete. Furthermore, the Adobe Photoshop requires a lot of computer resource and storage for its execution. In this paper, a digital image extraction tool that can be used to automatically extract objects from an image is presented. The tool provides a good graphical user interface that allows successively the user to select an object to be extracted from an image and then save it in a file after extraction. The tool operates on a grayscale version of an image in bitmap format. In the following section, we describe the techniques and algorithms used by the tool to extract objects from an image. The section on results and discussions deals with usefulness and limitations of the tool and suggests ways for future improvement on the techniques and algorithms employed by the tool.

Object Extraction

In order to extract an object from an image, the tool needs to perform several image processing steps on the image. First it needs to detect the edges of the objects embedded in the image. After the edges of a relevant object are found, the tool outlines the edges of the object found in the original image. Then it extracts the object by following some well-defined steps as described later. In the case where there are multiple objects found in a given image, the user has the option to see the images of all objects extracted by the tool successively and save each of them to a specific file. The most important step in this process is the detection of edges of the objects in the image. In the following, we describe the edge detection process.

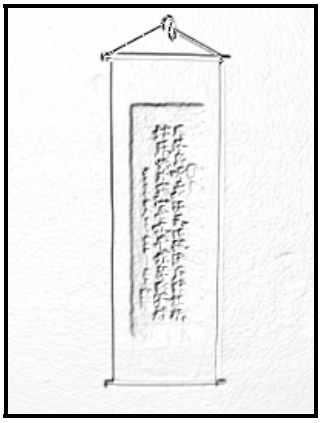
Edge Detection Methods

Edges characterize boundaries and therefore, edge detection is a problem of fundamental importance in image processing [2], [3], [6]. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. There are many ways to perform edge detection. However, the majority of the methods may be grouped into two categories, the gradient-based methods and the Laplacian based methods. A gradient-based method detects the edges by looking for the maximum and minimum in the first derivative of the image. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. Clearly, the derivative shows a maximum located at the center of the edge in the original signal. This method of locating an edge is the characteristic of the “gradient filter” family of edge detection filters, which includes the Sobel method [1]. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, a pixel on an edge pixel has a higher pixel intensity value than those surrounding it. So once a threshold is set, one can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded.

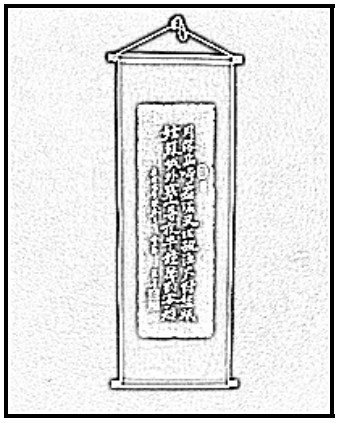
When the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian method [1], [4]. We compare the two well-known edge detection methods i.e., the Sobel method and the Laplace method, for the purpose of object extraction from an image. Figure 1 shows the effect of the methods on a test image and correspondingly Table 1 lists their features in terms of sensitivity on the presence of an edge and noise.

Copyright © 2005, American Society for Engineering Education

(a) Original image



(b) Edge detected by Sobel method



c) Edge detected by Laplacian method

Figure 1. Comparison of edge detection method.

Table 1. Features of Laplace and Sobel edge detection algorithm

	Laplace	Sobel
Noise Level	More sensitive to noise level	Better than Laplace edge detection method. Not only does it give a cleaner look, it also smoothes out the edges.
Completeness	Provides a very complete edge of an object	Weak edges depending on the background of an image.

After comparing the methods, it is found that the Laplace edge detection method is very sensitive to noise, which can cause some problem for the tool to find a valid object edge. Therefore, the Sobel edge detection method is chosen as the primary method of detecting edges for this work since the algorithm produces a clean, smooth image after processing. However, the Sobel method produces an output image with feeble or weak edges. To solve the problem of weak or indistinguishable edge problem, we devise an inverse-and-combine method as described later.

Sobel Method of Edge Detection

The proposed tool makes use of the 2-D Sobel operator that performs a 2-D spatial gradient measurement on an image stored in a 2-D array. By using the 2-D Sobel operator [1], it finds the approximate absolute gradient magnitude at each point in an input grayscale image. As shown in Figure 2, the tool uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). The magnitude of the gradient G is then calculated using the formula:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

where G_x is the gradient calculated in the x-direction using the mask in Figure 2(a) and G_y is the gradient calculated in the y-direction using the mask in Figure 2(b).

Copyright © 2005, American Society for Engineering Education

For fast processing, the evaluation of the square root function is avoided by using the following approximation:

$$|G| = |G_x| + |G_y|$$

The G_x mask highlights the edges in the horizontal direction while the G_y mask highlights the edges in the vertical direction. The mask is slid over the image, manipulating a square of pixels at a time.

-1	0	1
-2	0	2
-1	0	1

a) Mask for G_x

1	2	1
0	0	0
-1	-2	-1

b) Mask for G_y

Figure 2. Convolution masks used by Sobel method.

The mask changes the center pixel's value after the convolution operation on the image pixel array, and then the Sobel mask is shifted one pixel to the right and the process continues to the right until it reaches the end of the row. It then starts at the beginning of the next row. The process continues until it covers the entire image. After the process, the resulting output can be used to detect edges.

However, we observe some weaknesses of the Sobel method when we apply the method to delineate an object's boundary. For the purpose of illustration, we consider the effect of the Sobel method on geometric objects as shown in Figure 3. It seems that the Sobel method is sensitive to the sliding direction of the mask. A particular sliding direction of the mask over a given image seems to have a weak response to this method. As shown in Figure 3(b), the image obtained after the Sobel method tends to have only part of the edges. Also, depending on the background and object color contrast, for images with lighter background compared to the object itself, upper-left edges stand out much more clearly than the other edges. For images with darker background, lower-right edges stand out much more clearly than the other edges.

In order to get all the edges of an object in complete form, we apply a three-step process. First, the original image (see Figure 3(a)) goes through the Sobel edge detection method as stated above, which detects the edges of either upper-left or lower-right corner. For the purpose of discussion, we refer this output image as Sobel image 1. Other edges may appear very faint or may not appear at all in Sobel image 1. To recover the other edges or to enhance the presence of the other edges, we obtain an inverse of the original image (see Figure 3(c)). The inversion of an image is very straight-forward. To inverse an image, each pixel value is subtracted from the maximum pixel value of 255. Next we apply the Sobel method on the inverted image and correspondingly, an example output is shown in Figure 3(d). We refer this output image as Sobel image 2. Finally, the two output images, Sobel image 1 and Sobel image 2, are synthesized to

Copyright © 2005, American Society for Engineering Education

complete the process of edge detection. We refer the combined image as the resultant Sobel image of the original image, which is used as a guide for extraction of objects (See Figure 3(e)).

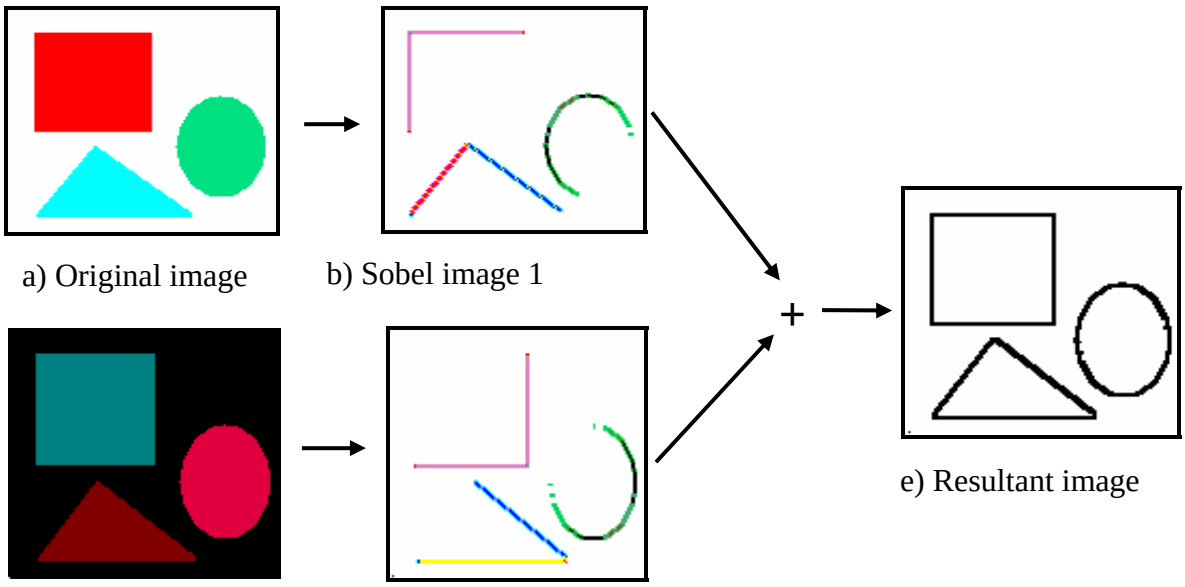
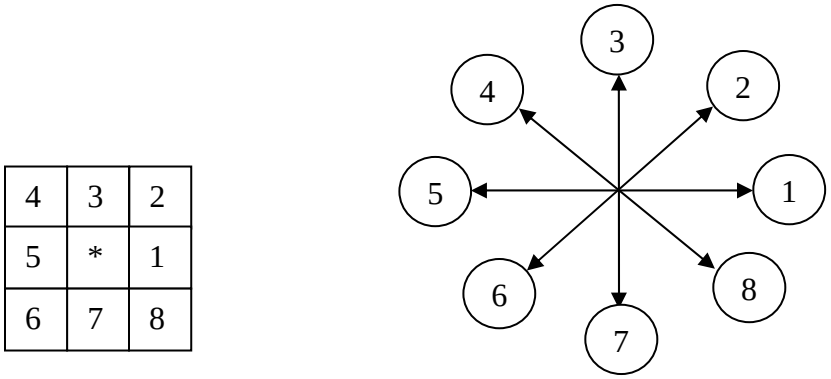


Figure 3. Edge detection by Sobel method.

Outlining Object Boundary

The pixels of the resultant Sobel image are maintained in a two-dimensional array. After edges of the objects in an image have been detected, the next step is to identify the first object, the one closest to the origin (row 0, column 0). A boundary tracing algorithm checks the pixels in the array to identify a closed boundary which is an indication of an object embedded in the image. The following explains how the tracing algorithm works. It is to be noted that a pixel in two-dimensional image has eight neighboring pixels as illustrated in Figure 4. Accordingly, the boundary tracing algorithm checks these eight neighboring pixels of a given pixel for connectivity and moves to the next pixel for further tracing. With the starting point always at the bottom-left corner of an image, the algorithm moves to direction 6 first to find an unmarked pixel on an edge that guarantees the outermost pixel of the edges. If moving onto direction 6 does not provide any pixel on an edge, then the algorithm continues searching for a pixel in direction 7, 8, 1, 2, 3, 4, and 5 successively. However, after a successful search in direction 6, it starts with direction 2 first, then 3, then 4, and so on. If it finds a pixel in direction 7, then direction 5 becomes its next starting direction. Finally, if it finds a pixel in direction 2, then it goes in direction 6 again. Thus, the algorithm ensures that the rightmost pixel is reached when the pixel search is going upward and the leftmost pixel is reached when the pixel search is going downward.

Copyright © 2005, American Society for Engineering Education



a) A pixel (*) and its eight neighbors

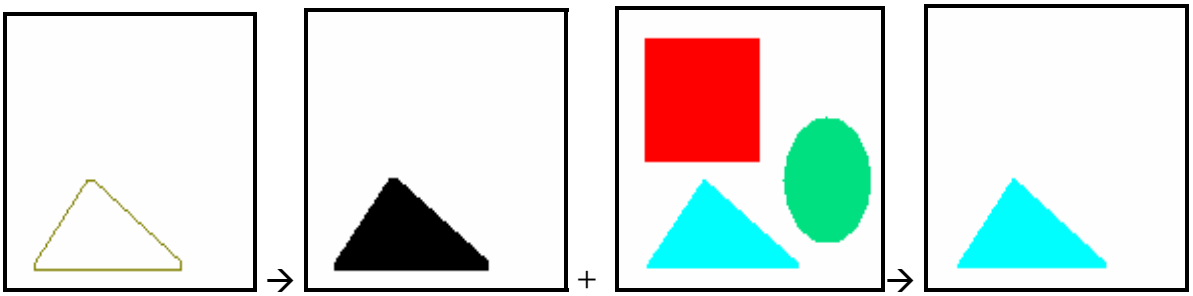
b) Eight tracing directions

Figure 4. Tracing pixels for outlining an object's boundary.

After searching through the pixels of the detected edges, the algorithm forms a loop by marking the pixels for the boundary of the object and stops at the point of the first pixel detected on the boundary. However, the algorithm can fail to mark the boundary of an object properly if the object has a very thin edge extending out in the resultant Sobel image. In this case, the tracing algorithm will not be able to identify the boundary of the object. Because, it will not be able to form a closed loop enclosing the object. The proposed tool stops in this situation and display a message to the user.

Filling

After an outline of the object has been produced, the next step is to fill the object. The approach to fill an object used in this work is very straight-forward. For a given pixel, the algorithm checks pixels in its four directions, up, down, left, and right, then cycles through each direction towards the border of the image. To detect whether a pixel is inside an object or not, the algorithm uses a counter which is incremented when an edge is encountered. If the edge counter is found to be 4 after checking all four directions, which means the pixel is surrounded by the object boundary and the pixel is then assigned a filled value 0. Otherwise, the pixel is set to a white value of 255. The process is illustrated in Figure 5.



a) Outlined Object

b) Filled object

c) Original image

d) Extracted object

Figure 5. Object extraction process.

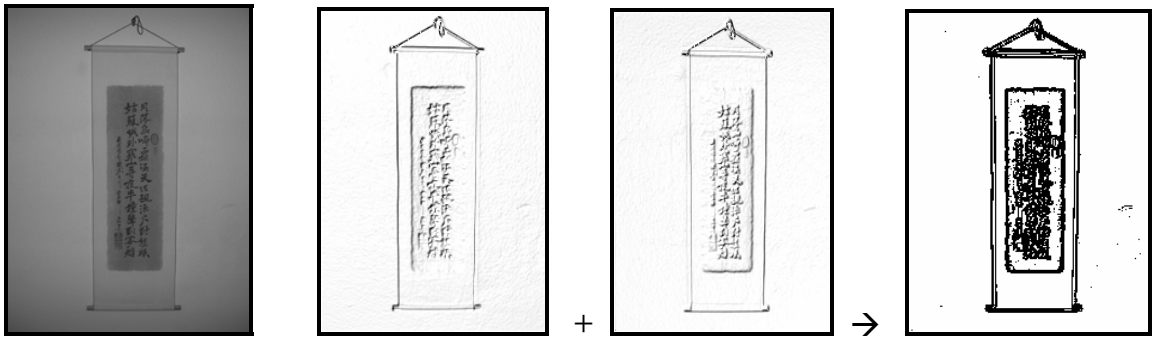
Copyright © 2005, American Society for Engineering Education

Extraction

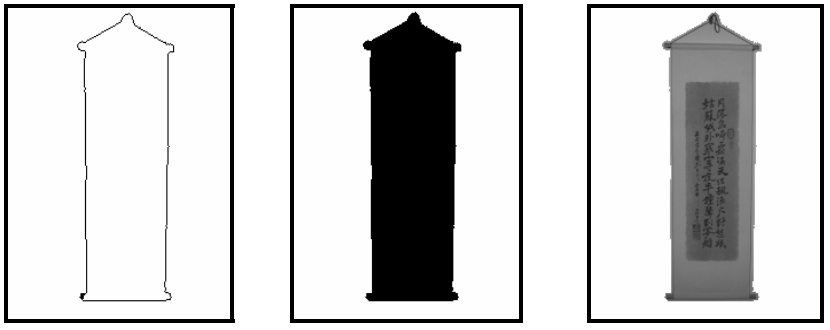
Figure 5 illustrates the final process of object extraction. After having the filled object in place, extracting the original data for the filled area is very straightforward. This is done by checking the original image and the filled image at the same time. Wherever a filled pixel is encountered in the filled image, the algorithm places the exact pixel value of the original image to the output image. Otherwise, it sets the pixel to the value of the white color. After extraction of the first object from the original image, the edges of the extracted object in the corresponding resultant Sobel image are erased by setting them to white color. The resultant Sobel processed image is cycled through the same outlining, filling, and extraction steps to retrieve the next object available in the image. The process goes on until there are no more objects to extract from the image.

Results and Discussions

As stated earlier, the tool is based on the algorithms and methods discussed above. In order to use the tool for an image, the image must be in bitmap format. The tool converts any color image to its corresponding grayscale image for the purpose of applying the Sobel method of edge detection and uses all the steps discussed above for detecting edges, outlining the boundary of an object, filling the object area, and then extracting the object. The steps are shown in Figure 6 on a real image rather than the image with geometric shapes as discussed above.



a) Original image b) Sobel image 1 c) Sobel image 2 d) Resultant image



e) Outlined image f) Filled image g) Extracted object

Figure 6. Object exaction by the tool from a real image.

Copyright © 2005, American Society for Engineering Education

After extraction of each object, the tool allows the user to save the image of the object in a file. A close look at the extracted object in Figure 6 shows that the boundary of the object is little off from the boundary of the original object in some areas. However, this imperfection of the image object can easily be corrected manually with minimum effort. The important feature of the tool is that it does not loose any part of the object itself. The idea of object extraction on a grayscale image can be easily extended to the colored version of the image. In this case, the tool has to select the corresponding colored pixel (in red, green, and blue colors) from the original image for each pixel in the grayscale version of the extracted object.

There are some limitations in this process of object extraction. In the case of an image embedded with overlapped objects, the tool recognizes all overlapped objects as a single object.
An object with an open or broken boundary in the resultant Sobel image cannot be extracted by the tool due to the fact that the tool cannot outline such an object in a complete loop and the object is considered incomplete. However, if it is an incomplete object with thick edges, the tool would simply present the edges as objects. It may be possible to reconstruct the incomplete object by devising a more sophisticated algorithm available in current literature [5], [7]. The background light seems to have some effect on the effectiveness of the tool as well. Even though the tool can detect an object embedded in an image with dark background, but when it does the extraction of the object from the original image, it leaves the area of the extracted object as white. In the subsequent processing of the image with the whitened area in it, the tool considers the whitened area under dark background as an object, therefore, producing an output object with white pixels. This minor problem can be corrected by using some filtering or filling technique on the resultant Sobel image. On the other hand, a light background is ideal for the tool, since it does not cause much of color variance.

Conclusion

In this work, a tool for extraction of objects from an image is presented. It provides an acceptable level of precision in outlining the boundary of an object in an image. It is a small, fast, and easy to use tool that provides an effective way to extract objects from an image file. Furthermore, the tool provides a user-friendly interface, letting the user to preview the images of the extracted objects before and after processing. Objects can be selected and deselected at ease for extraction. The tool allows the user to save a specific extracted object in a file. The tool reduces the time needed to extract an object from an image manually.

References

1. Fisher, B., 2005, “Feature Detector,” URL: http://www.cee.hw.ac.uk/hipr/html/sobel.html.
2. Gonzalez, R. C., Woods, R. E., 2002, *Digital Image Processing*, Addison-Wesley Publishing Company, Reading, Massachusetts.
3. Petrou, M., Bosdogianni, P., 1999, *Image Processing: The Fundamentals*, John Wiley & Sons Publishing Company, New York.
4. Tanimoto, S., 2005, “Digital images,” URL: http://www.cs.washington.edu/research/metip/about/digital.html.

Copyright © 2005, American Society for Engineering Education

5. Chalasani V., Beling, P., 1999, "Road extraction from digital images using linear programming and cost functions," *Proceedings of the Fourth International Conference on GeoComputation*, Mary Washington College, Fredericksburg, Virginia.
6. Baxes G., 1994, *Digital Image Processing: Principles and Applications*, John Wiley & Sons Publishing Company, New York.
7. Qahwaji R., Green R., 2001, "Detecting faces in noisy images," *Proceedings of the ISCA 16th International Conference on Computer Applications*, Seattle, Washington.

DULAL C. KAR
Dr. Kar currently is an Associate Professor of Computer Science at Texas A&M University-Corpus Christi. His research interests include digital signal and image processing, data communications and computer networks, and information assurance. Dr. Kar is a member of Association for Computing Machinery.

DENNIS MA
Dennis Ma obtained his MS degree in computer science from Texas A&M University-Corpus Christi in 2003. Previously he attended Iowa State University for his BS degree in electrical engineering. His research interests in digital image processing includes image enhancement, object recognition, and object extraction.

Copyright © 2005, American Society for Engineering Education