A Semi-Automatic Object Extraction Tool for Querying in Multimedia Databases*

Ediz Şaykol†, Uğur Güdükbay and Özgür Ulusoy
Department of Computer Engineering, Bilkent University
06533 Bilkent, Ankara, Turkey

{ediz, gudukbay, oulusoy}@cs.bilkent.edu.tr

Abstract

Considering the complexity and huge volume of image and/or video data, efficient methods need to be developed for querying multimedia databases. A well-known technique used for querying multimedia data is query-by-feature (e.g. color, shape, texture, size) of the objects residing in images and/or video frames. In this paper, we propose a tool for extracting objects from images and/or video frames, called *Object Extractor*, as well as the ways of coping with the object features within extracted objects. The tool is semi-automatic in the sense that the user specifies the colors on the object by clicking the mouse to make the tool capture object pixels. In order to extract objects, an improved version of the *Flood Fill* algorithm for polygon filling is provided. The extraction algorithm uses filtered images to perform better. Moreover, the experimental results obtained for evaluating the performance of the tool in extracting objects are presented. It is shown through these results that a few mouse clicks would suffice to extract objects effectively.

Keywords: Object extraction, flood filling, color median filtering, color space transformations, color quantization.

*This work is supported by Turkish Scientific and Technical Research Council (TÜBİTAK) under the grant number EEEAG-199E025.

†Author for Correspondence

1 Introduction

Advances in multimedia technology accelerate the amount of digitized information so as the data stored as image and video content. Both of these data types require application-dependent processing strategies and easy-to-handle storage methods when stored in a database. As far as the querying process of image and video databases is concerned, spatial (for both image and video data) and spatio-temporal (for video data only) as well as semantic information are taken into account. Semantic querying requires more sophisticated techniques that encode the semantic meaning of the image and/or video content. Spatial and spatio-temporal querying necessitate a query pre-processing phase in which the objects and their corresponding features (e.g. color, shape, texture, size) are extracted. The motivation of the tool presented in this paper is to facilitate the pre-processing phase of the query-by-feature sub-system of the video database querying system being developed at Bilkent University [7].

There exist a considerable number of multimedia data querying systems in the literature [5, 9, 13, 18]. The pre-processing phase in most of these systems includes object extraction in order to figure out the hidden object-based information from the image and video data. Based on the type of user interaction in the pre-processing phase, the object extraction methods can be grouped into three categories:

Fully Automatic Extraction Methods: In these methods, the extraction process for image and/or video data is performed automatically. Since the whole process lacks user interaction, not all types of images and video can be handled with this approach. The QBIC system [9] employs a method based on *Foreground/Background* approach for fully automatic object extraction. This method processes only images and video frames having a separable background [1]. Jain and Vailaya employ edge-detection algorithms for extracting object boundaries from images [11]. Their method for edge-detection is the famous *Canny Edge-Detection* algorithm [3]. Chang et al. describe automatic feature extraction methods for color, texture and shape features of images in [4].

Semi-Automatic Extraction Methods: In these methods, the user assists the object extraction process. One way of assistance is to facilitate the object extraction of an image via clicking on the object pixels to visualize the object area. *Flood Fill* algorithm for polygon filling [10] may be adopted to determine the pixels in the object area for extraction process. Another semi-automatic method is based on the *snakes* concept of computer vision [12]. In this method, the user specifies a bounding polygonal region for an object and the object boundaries are determined from this region automatically. The QBIC system uses these techniques for the object extraction process [1]. The range of images and/or videos handled by these types of methods is larger than that of fully automatic methods but there still exist some types of data that need more user assistance for a proper object extraction.

Manual Extraction Methods: In these methods, the user-computer relation is more than interaction, because the user manages all the extraction process. For example, in order to encapsulate an object region with a (minimum) bounding polygon, the drawings of the users will be used as is. Obviously, any type of image and video data can be handled manually since any kind of data is perceptually comprehensible to the user. However, the manual extraction is a very tedious process and cannot be applied to very large datasets.

The more the user participates in the extraction process, the less the image restrictions and requirements for proper object extraction are needed. The basic aim in object extraction is to minimize the user interaction throughout the extraction process without discarding any type of image or video data. Thus, a powerful extraction system should include tools for each of the above extraction methods not only to extract objects but also to extract the corresponding object features.

The *Object Extractor* tool proposed in this paper extracts objects in images and/or videos semi-automatically. In order to increase the quality of the processed images and/or video frames, color space transformations, quantization and color filtering are employed before processing the input. The filtered input leads to an increase in the performance of the object extraction algorithm. *Flood Fill for Extraction (FFE)* algorithm is employed for extracting objects and as a result of the filtering steps, the tool provides an environment for processing images and/or videos effectively.

The organization of the paper is as follows: Section 2 summarizes the techniques related to the principles of our *Object Extractor* tool. Sections 3 explains the design of *Object Extractor*, Section 4 describes the object extraction algorithm. Section 5 presents the experimental results obtained for evaluating the performance of the tool. Finally, Section 6 concludes the paper.

2 Background Information

One of the most important features of objects in image and video data is *color*. Each pixel in an image has a three-dimensional color vector and different color spaces encode color information based on different approaches. The most famous color space model is the *Red-Green-Blue Model (RGB)* where the color vector of a pixel p is the compound of red, green and blue channels $v_p = (r, g, b)$. Another color space model is the *Hue-Saturation-Value Model (HSV)* that is based on color descriptions rather than individual color components and makes the model unique among other color space models $v_p = (h, s, v)$. The RGB model has a major disadvantage: it is not perceptually uniform. Therefore, most of the systems use color space models other than RGB, such as YIQ [10].

The color regions are perceptually distinguishable to some extent. The human eye cannot detect some little color differences and may perceive these very similar colors as

the same color. This leads to the *quantization* of color, which means that some pre-specified colors will be present on the image and each color is mapped to some of these pre-specified colors. One obvious consequence of this is that each color space may require different levels of quantized colors, which is nothing but a different quantization scheme. In Figure 1, the effect of color quantization is illustrated. Figure 1 (a) is the original image with RGB color space and (b) is the image produced after transformation into HSV color space and quantization. A detailed explanation of color space transformations (from RGB into HSV) and quantization can be found in [17].

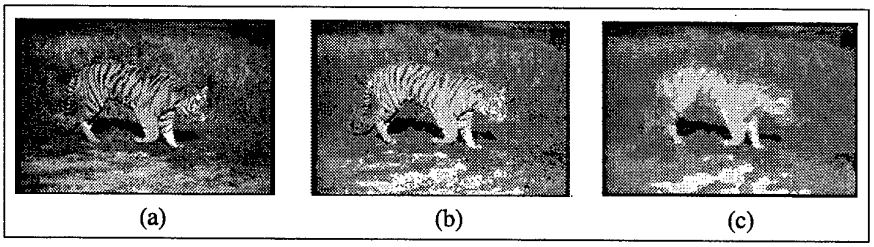

Figure 1: Transformation, quantization and color median filtering of an image. (a) Original image. (b) Image produced by applying RGB to HSV color transformation and quantization. (c) Image produced after applying color median filtering.

Moreover, not all the colors in the image are dominant. Dominance is in the sense that some of the colors may reside in a region relatively small than the others. The *color median filtering* technique [14], a famous method for neighborhood ranking, eliminates these non-dominant colors and produces a filtered image (Figure 1(c)). This technique facilitates the object extraction process because it also eliminates the noise of the color on the object boundaries to some extent.

3 Design of Object Extractor

3.1 Overall Architecture

The *Object Extractor* tool extracts objects from images and videos semi-automatically with the help of the improved version of the flood fill algorithm. The overall architecture of the tool is shown in Figure 2. Videos are segmented with *Fact Extractor* in the system and keyframes of the videos are processed as images. An image is passed through quantization and color median filtering steps and then the final image, where the object is to be extracted, is produced. This filtered final image is processed with *Flood Fill for Extraction Algorithm* to extract the objects along with their features. Since we deal with realistic images and videos in the system, automatic extraction of objects and features would be inadequate, so that the user intervention becomes inevitable. The last step in the process is storing the features of the extracted objects in the object feature database.

Thus, *Object Extractor* is one of the basic tools that cooperate with the query-by-feature sub-system of our video database and querying system [7].

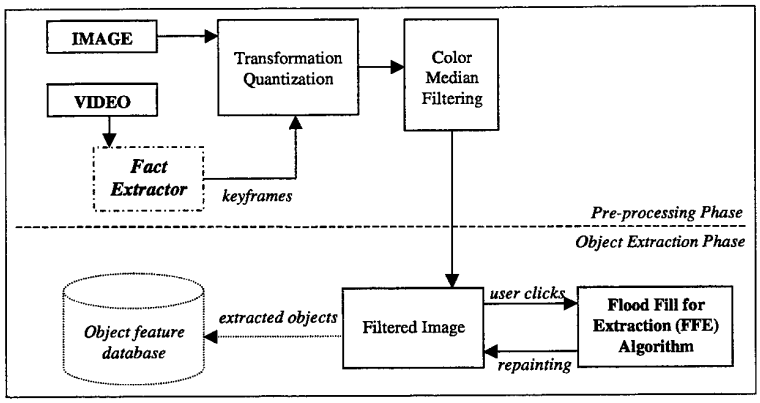

Figure 2: Overall architecture of *Object Extractor*

The image whose objects to be extracted passes through a color space conversion step and the color vectors of all of the pixels are transformed from RGB color space into HSV color space. Then, this data is used in the quantization step yielding 18 hue, 3 saturation, 3 value levels and 4 gray levels. The color quantization step employed here is very close to the one proposed in $VisualSEEk$ [18] and $VideoQ$ [5]. After completing this step, the median filtering algorithm is applied to eliminate the non-dominant color regions.

The *Flood Fill for Extraction (FFE)* algorithm, an improved version of the flood fill algorithm, is designed to specify object regions. The color of the user-clicked pixel initiates the process and it forks into four branches corresponding to the four neighboring pixels (north, east, south and west). As soon as the difference between the processed pixel's color and the initiative pixel's color exceeds a pre-specified threshold, the execution stops. The user may continue to specify new initiative pixels as necessary to extract the object.

In the pre-processing phase of the *Object Extractor*, color space conversion and color quantization are applied to the input, thus the FFE algorithm performs better. This yields an increase in the effectiveness of the technique. Moreover, since the objects are extracted separately, images containing more than one image are handled successfully. The *Object Extractor* provides an environment where a wide range of images and/or videos are handled effectively by the help of the adapted and improved techniques.

3.2 The User's Assistance

Due to the semi-automatic nature of *Object Extractor*, the user assists the extraction process. The tasks of the user are the identification of the colors in the extracted object and then labelling the object. An appropriate indexing scheme can be adopted for these labelled objects and the indexed data can be queried for the extracted object features such

as color and shape. Faloutsos et al. propose an effective querying methodology that is based on quadratic color histogram distance considering the perceptual similarity of colors via cross-correlation [8].

Definition 1 *(t-neighborhood) The t-neighborhood of a pixel p with respect to color is a contiguous set of pixels t_p, where the Euclidean distance between color vectors of p and the pixels on the line segment pp_i, such that $p_i \in t_p$, is not greater than a color difference threshold value t. It is obvious that $p \in t_p$.* □

In the current implementation of our tool, the user clicks on a pixel p_c on the image and the FFE algorithm is initiated with the pixel p_c and the current color difference threshold t. During this execution, the pixels in t_p are repainted for p_c. However, if the object bounds unprocessed pixels, the user may click onto another pixel for extracting other parts of the object. Having satisfied with the extracted object region, the user labels the object.

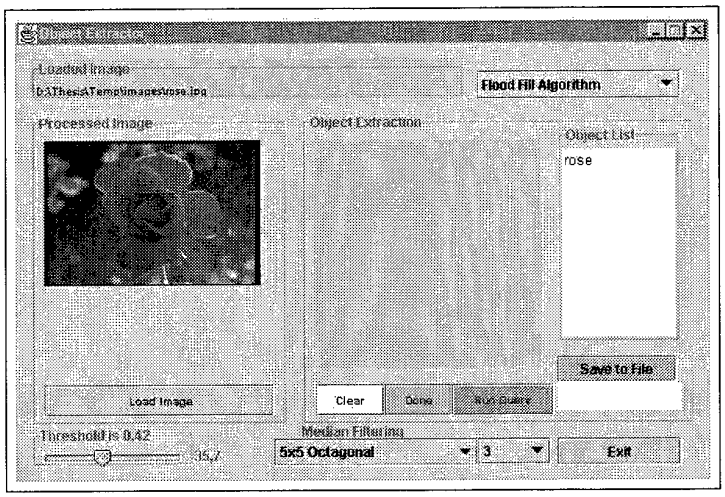

Figure 3: The user interface of *Object Extractor*

3.3 The User Interface of Object Extractor

The user interface of *Object Extractor* has been developed in *Java* programming language and it provides the following functionalities. First of all, since median filtering alleviates color quantization on the image and filters out the non-dominant color regions, the user can see the effects of the median filtering algorithm separately in the tool. This gives the opportunity to the user to decide whether to use median filtering or not. Based on this decision, the color transformation and quantization steps produce a better image. The default color difference threshold is 0.4, which is determined by a reasonable number of experiments. The user interface of the Object Extractor tool is shown in Figure 3.

Moreover, the main usage of this tool is to extract objects for the query-by-feature subsystem of the rule-based video querying system that we develop [7]. As seen in Figure 3,

'run query' button activates this querying operation with the previously extracted objects. The image to be queried is also segmented with this tool and the objects are extracted. Since all of the extracted objects are handled with a proper indexing mechanism, the extracted objects of the query image can be queried with the existing objects. The details of the querying methods can be found in [16].

4 The Object Extraction Algorithm

The *Object Extractor* tool processes both images and/or video frames. The method it employs for both types of data is very similar since each video frame can be treated as a single image. The *Fact Extractor* tool inside the video database system handles video data and produces video keyframes. Thus, videos can be processed in the *Object Extractor* tool through their keyframes.

```
procedure FloodFillforExtraction(Pixel p)
// INPUT: a single pixel p
// the INITIATIVE_PIXEL is global to the method and
// it holds the user-clicked pixel

1.   if (pixelProcessed(p))
2.       return;
3.   endif
4.   setProcessed(p);
5.   if (thresholdPassed(p, INITIATIVE_PIXEL))
6.       paint(p);
7.       FloodFillforExtraction(left(p));
8.       FloodFillforExtraction(right(p));
9.       FloodFillforExtraction(up(p));
10.      FloodFillforExtraction(down(p));
11.  endif
endprocedure.
```

Figure 4: Flood fill for extraction (FFE) algorithm

As discussed above, *Flood Fill for Extraction (FFE)* algorithm works with a transformed, quantized and median filtered image. When the algorithm halts, it repaints some of the pixels on the image and the user may continue the extraction as many times as he/she wants. The pseudo-code of the FFE algorithm is given in Figure 4. When the user clicks on a pixel in order to initiate the algorithm, this pixel is stored in the $INITIATIVE\_PIXEL$ and used globally in the procedure. Line 1 checks the stopping condition and the lines $4-11$ correspond to the recursive part. Each pixel is processed only once due to the **if**-statement at the beginning. Since a pixel may be visited more than once, this fastens the algorithm significantly. On the other hand, the test for the threshold in line 5 is performed by evaluating the Euclidean distance between color vectors of the two pixels, namely p and $INITIATIVE\_PIXEL$. If the test succeeds, the pixel p is repainted and the algorithm calls itself recursively for the neighboring four branches. The whole process stops when there is no executing branch in the recursion tree.

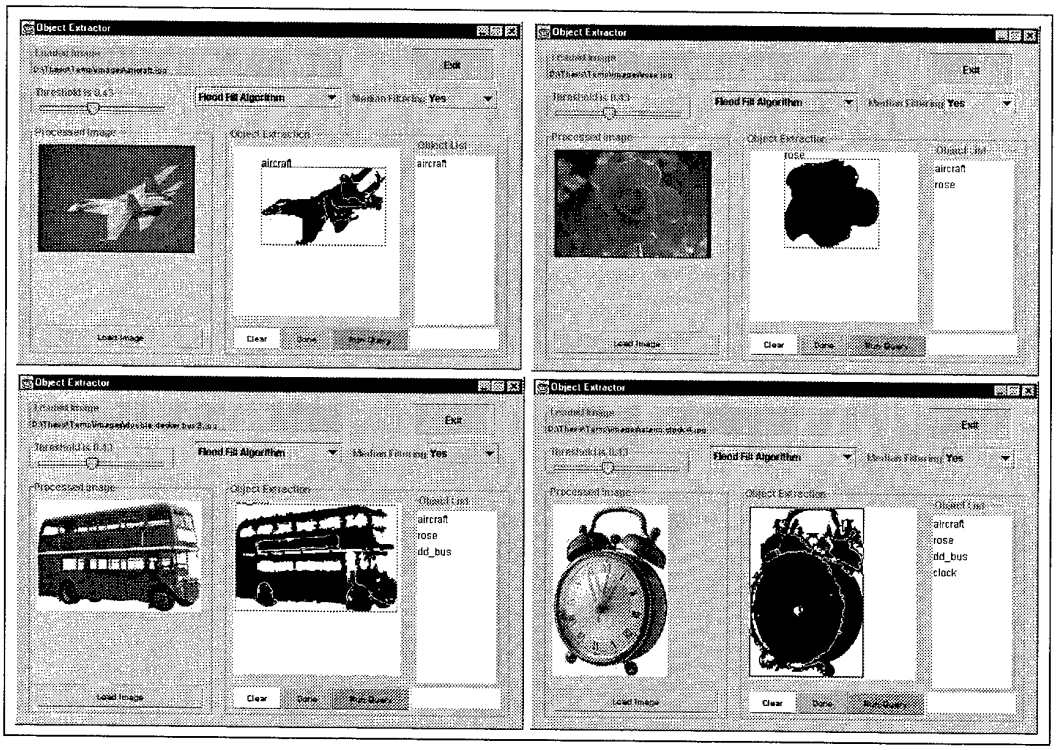

Figure 5: Experimental snapshots for four images sampled in *Object Extractor*

5 Experimental Results

The experiments to evaluate the performance of the *Object Extractor* tool are conducted with the images from *Berkeley University Blobworld Project* [2] and *CoffeeCup Software Photo Gallery* [6]. In the experiments, color median filtering is enabled with 5x5 box filters and applied three times to improve smoothing. The results are presented in Table 1 where each row shows the number of mouse clicks during extraction of an object with different threshold values. A detailed analysis and evaluation of *Object Extractor* with various median filters as well as comparisons with some of the existing object extraction methods and photo editing tools using similar techniques in terms of effectiveness can be found in [15].

The tool gives promising results when the objects are on a separable background in the image. This restriction is inevitable but softened with quantization and color median filtering. This lies in the observation that median filtering facilitates the separation of background from the foreground of the objects. However, the tool performs better in extracting objects containing noise or non-uniformity on the boundaries as well as objects containing holes since it is usually agreed that automatic object extraction tools have difficulty in extracting such objects. Moreover, most of the objects are extracted with a few clicks in different color regions of an object and the semi-automatic nature of the tool

Table 1: Objects versus color difference threshold t. X means that the repainted region is larger than the object region when extracted with the threshold.

	Number of Clicks		
Object	t=0.21	t=0.42	t=0.56
Rose	3	1	X
Aircraft	5	2	X
DDeckerBus	7	3	1
AlarmClock	8	2	2
BlackGirl	8	2	X
WorldMap	9	2	X
Tiger	3	1	X

does not have a considerable effect on the speed of the extraction process.

6 Conclusion and Future Work

The *Object Extractor* tool is a semi-automatic tool used for extracting objects from image and/or video data. In our video querying system, the queries that specify object features is processed with the help of this tool. The extracted object features are basically the color content and the shape information (in fact the boundary) of the objects. For the former, the color content of the whole image can be stored in order to respond to image-based color queries. For query-by-shape, the boundary information of the extracted objects can be stored as well as some other shape features such as turning angles, area, center coordinates, etc. The operations necessary for query-by-feature sub-system and the improvements on object extraction tool are going to be performed in parallel. One possible improvement in *Object Extractor* will be enabling the users to specify boundary polygon rather than clicking onto the object pixels.

Along with the described work, we have also studied extracting objects automatically not only from images but also from video frames. Since disabling user interaction throughout the object extraction process requires the use of more automatic image processing methods on the images and/or video frames, this is a relatively hard task to achieve. Adopting appropriate indexing structures onto the extracted objects based on the features that are stored in the object feature database is another ongoing project for our rule-based video database and querying system.

References

[1] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, D. Petrovic. Automatic and semi-automatic methods for image annotation and retrieval in QBIC. *Proceedings of SPIE-Storage and Retrieval for Image and Video Databases III*, Vol. 2420, 24–35, 1995.

[2] Berkeley University Blobworld Project Start Images, Berkeley University. `http://elib.cs.berkeley.edu/photos/blobworld/start.html`.

[3] J. Canny. A Computational Approach to Edge-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, 679–698, November 1986.

[4] S.F. Chang, J.R. Smith, H. Wang. Automatic Feature Extraction and Indexing for Content-Based Visual Query. *Technical Report CU/CTR 414-95-20*, Columbia University, January 1995.

[5] S.F. Chang, W. Chen, H.J. Meng, H. Sundaram, D. Zhong. VideoQ: An Automated Content Based Video Search System Using Visual Cues. *ACM Multimedia'97 Conference Proceedings*, 313–324, Seattle, WA USA, 1997.

[6] CoffeeCup Software Photo Gallery. `http://www.coffeecup.com`.

[7] M.E. Dönderler, Ö. Ulusoy, U. Güdükbay. A Rule-Based Approach to Represent Spatio-Temporal Relation In Video Data. *ADVIS'2000 Proceedings*, LNCS Vol. 1909, T. Yakhno (Ed.), 409–418, 2000, Springer-Verlag.

[8] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petrovic, R. Barber. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[9] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petrovic, D. Steele, P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer Magazine*, 28(9), 23–32, September 1995.

[10] D. Hearn, M.P. Baker. Computer Graphics. Prentice Hall, Inc., New Jersey 1994.

[11] A.K. Jain, A. Vailaya. Image Retrieval using Color and Shape. *Pattern Recognition*, 29(8), 1233–1244, August 1996.

[12] M. Kass, A. Witkin, D. Terzopulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 321–331, 1998.

[13] A. Pentland, R.W. Picard, S. Scarloff. Photobook: Tools for Content-Based Manipulation of Image Databases. *Proc. Storage and Retrieval for Image and Video Databases II*, Vol. 2,185, 34–47, SPIE, Bellingham, Washington 1994.

[14] J. Russ. The Image Processing Handbook. CRC Press in cooperation with IEEE Press, 1999.

[15] E. Şaykol, U. Güdükbay, Ö. Ulusoy. A Semi-Automatic Object Extraction Tool for Storage and Retrieval in Multimedia Databases. *journal paper in preparation*, 2001.

[16] E. Şaykol. Web-Based User Interface For Query Specification in a Video Database System, M. Sc. Thesis, Bilkent University, September 2001.

[17] J.R. Smith, S.F. Chang. Tools and Techniques for Color Image Retrieval. *IS&T/SPIE Proceedings*, Vol 2670, In: Sethi, I.K., Jain, R.C., eds., *Storage&Retrieval for Image and Video Databases IV*, 426–437, February 1996.

[18] J.R. Smith, S.F. Chang. VisualSEEk: A Fully Automated Content-Based Image Query System. *ACM Multimedia'96 Conference Proceedings*, 87–98, NY 1996.