SIOX: Simple Interactive Object Extraction in Still Images

Gerald Friedland, Kristian Jantz, Raúl Rojas
Freie Universität Berlin
Faculty of computer Science
Takustr. 9, 14195 Berlin, Germany
{fland, jantz, rojas}@inf.fu-berlin.de

Abstract

The following article presents an approach for interactive foreground extraction in still images that is currently being integrated into the GIMP. The presented approach has been derived from color signatures, a technique originating from image retrieval. The article explains the algorithm and presents some benchmark results to show the improvements in speed and accuracy compared to state-of-the-art solutions. The article also describes how the algorithm can easily be adapted for video segmentation tasks.

1 Introduction

The algorithm presented here has been originally developed for the extraction of an instructor teaching in front of an electronic chalkboard. In the E-Chalk system [15, 8, 9], which is used for recording and transmitting lectures over the Internet, the board content is transmitted as vector graphics while the video of the extracted lecturer is sent as separate stream. The extracted lecturer is then laid over the high resolution board image at video stream rates. The lecturer may be dimmed from opaque to semitransparent, or even transparent. This makes it possible to transmit mimic and gestures of the lecturer in relation to the board without using either too much bandwidth or having blurry artefacts around the board strokes produced by state-of-the-art video compression. Figure 1 shows an example of such a video.

Video segmentation requires speed more than accuracy. In still image segmentation, accuracy becomes the higher priority. This article shows that our algorithm for the segmentation of the instructor is general enough to be used also for foreground extraction in still images.

Section 2 first introduces related work. Section 3 then explains the algorithm and how it is used. We benchmark our results and compare them to the *GrabCut* underlying algorithm presented in [3]. Section 5 shortly summarizes the video segmentation approach before, in Section 6, a conclusion is drawn and possible future work is presented.

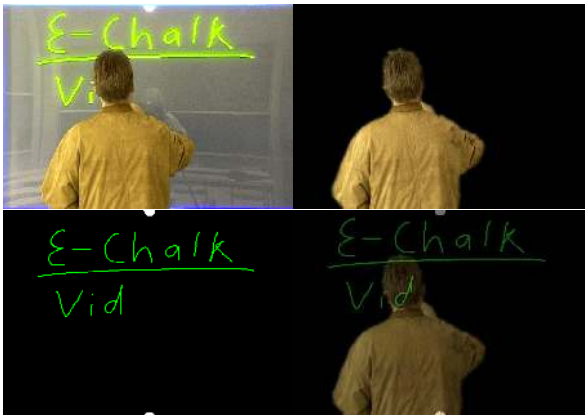

Figure 1. A video of the lecturer is recorded (above left), the instructor is extracted (above right), superimposed semi-transparently on the vector based board data (below left) and replayed together using MPEG4 (below right).

2 Related Work

A nicely written summary and discussion of most still-image foreground extraction methods can be found in [16]. In the following, some approaches for still image segmentation are briefly described. The most popular tool for extracting foreground is *Magic Wand* [1]. *Magic Wand* starts with a small user-specified region. The region grows through connected pixels such that all selected pixels fall within some adjustable tolerance of the color statistics of the specified region. The methods works well for images that contain very few colors, such as comic strips. Intelligent Scissors [13] can be used to select contiguous areas of similar colors in a fashion similar to Magic Wand. The primary dif-

ference is that the scissor tool creates the selection area in one line at a time. Clicking with the mouse creates nodes that are joined using curve shapes that attempt to follow color weights. Once the area is closed, clicking inside the new area renders the selection area. For photographs, finding the correct tolerance threshold is often cumbersome, so a satisfactory segmentation is seldom achieved. *Knockout* is a proprietary plugin for *Photoshop* [6] that is, like the approach presented here, driven from a *trimap* (see Section 3.1). According to [5] the results are sometimes similar to and sometimes of less quality than Bayes matting. Bayes Matting also gets a trimap as input and tries to compute alpha values over the unknown region. A disadvantage is that the user must specify a lot of shape information for the algorithm to work properly. *Grabcut* [16] is a two step approach. The first step is an automatic segmentation step that relies on the work of *Graph Cut* [4] and [3]. The second step is a manual post editing step. The idea of the automatic classification is to build a graph where each pixel is a node with outgoing edges to each of the 8 pixel's neighbors. The edges are weighted such that a max-flow/min-cut problem solves the segmentation. The user only provides the region of interest. *Grabcut*'s manual post processing tools include the so-called background brush, a foreground brush, and a matting brush to smooth borders or re-edit classification errors manually. The disadvantage is that its complicated data structure requires high computational effort. At the time of writing this article, there was no program available to test the performance of the tool, nor is there sufficient information to compare any benchmark results. Since, in this article only automatic classification without manual intervention is considered, Benchmark comparison is done using the results in [3].

3 The Algorithm

The described algorithm solves the task of foreground extraction in a given image. We define foreground to be a single, spatially connected object that is of interest to the user. The rest of the image is considered background. The user has to specify at least a superset of the foreground and the algorithm is to return an image where all background pixels have been set transparent.

3.1 Input

The input for the algorithm consists of three user specified regions of the given image: Known background, unknown region, and known foreground. The user specified regions are called a trimap. The known foreground is optional, but eases segmentation of tricky images. To provide this information, the user makes several selections with the mouse. The outer region of the first selected areas specify the known background while the inner region define a superset of the foreground, i.e. the unknown region. Using additional selections, the user may specify one or more known foreground regions. Figure 2 shows an example of the user interaction and the resulting trimap. Internally, the trimap is mapped to a confidence matrix, where each element of the matrix corresponds to a pixel in the image. The values of the elements lie in the interval $[0, 1]$ where a value of 0 specifies known background, a value of 0.5 specifies unknown, and a value of 1 specifies known foreground. Any other value expresses uncertainty with a certain tendency towards one limit or the other.

Figure 2. The original image from [11], a user provided rectangular selection (red: region of interest, green: known foreground), and the corresponding trimap (black: known background, gray: unknown, white: known foreground).

3.2 Conversion to CIELAB

The first step of the algorithm is to convert the entire image into the CIELAB color space. This color space was explicitly designed as a perceptually uniform color space. It is based on the opponent-colors theory of color vision, which assumes that two colors cannot be both green and red, nor blue and yellow at the same time. As a result, single values can be used to describe the red/green and the yellow/blue attributes. When a color is expressed in CIELAB, L defines lightness, a denotes the red/green value and b the yellow/blue value. In the algorithm described here, the standard observer and the D65 reference white (see [18]) is used as an approximation to all possible color and lighting conditions that might appear in an image. CIELAB may still not be the optimal color space and the aforementioned assumption clearly leads to problems but in practice, the Euclidean distance between two colors in this space better approximates a perceptually uniform measure for color differences than in any other color space, like YUV, HSI, or RGB. Refer to Section 4.3 for a short discussion on the limits and issues of using this color space.

3.3 Color Segmentation

The segmentation method was adapted from [17] who describes the use of color signatures and the Earth Mover's

Distance for image retrieval. The idea behind our approach is to create a kind of color signature of the known background and use it to classify the pixels in the image as those belonging to the signature and those not belonging to it. The known background sample is clustered into equally sized clusters because in LAB space specifying a cluster size means to specify a certain perceptual accuracy. To do this efficiently, we use the modified two-stage k-d tree [2] algorithm described in [17], where the splitting rule is to simply divide the given interval into two equally sized subintervals (instead of splitting the sample set at its median). In the first phase, approximate clusters are found by building up the tree and stopping when an interval at a node has become smaller than the allowed cluster diameter. At this point, clusters may be split into several nodes. In the second stage of the algorithm, nodes that belong to several clusters are recombined. To do this, another k-d tree clustering is performed using just the cluster centroids from the first phase. We use different cluster sizes for the L, a, and b axes. The default is 0.64 for L, 1.28 for a and 2.56 for the b axis. The values can be set by the user according to the perceived color diversity in each of the axes. For efficiency reasons, clusters that contain less than $1\,‰$ of the pixels of the entire background sample are removed.

We explicitly build the k-d tree and store the interval boundaries in the nodes. Given a certain pixel, all that has to be done is to traverse the tree to find out whether it belongs to one of the known background clusters or not. Another tree is built for the known foreground, if the user has specified such. Each pixel is then also checked against the known foreground. If it does not belong to either one of the trees, it is assumed to belong to the cluster with the minimum Euclidean Distance between the pixel and each cluster's centroid.

Using known foreground improves the classification rate dramatically because it lowers the probability that foreground colors that also exist in the background are classified as background.

3.4 Post Processing

The remaining task is to eliminate the background colors that appear in the foreground. By the above definition, the foreground must be a unique, spatially connected region. In addition to several smoothing steps and an erosion/dilation step, a breadth-first-search on the confidence matrix is performed to identify all spatially connected regions that were classified as foreground. We assume that the biggest region is the one of user interest and eliminate all other regions[1]. In addition, the user can provide a smoothness factor to specify, how much smoothing should be applied to the resulting confidence matrix. More smoothing reduces small classification errors. Less smoothing is appropriate for high frequency object boundaries, for example hair or clouds. The values of the confidence matrix are directly used as transparency factors (also known as α-values) for each corresponding pixel. The default value for the smoothing factor is 3. Figure 3 shows the result for an example image directly after color segmentation and after post-processing.

[1] Another approach would be to use all regions containing known foreground.

Figure 3. The result of the color classification (left) and after post processing (right).

4 Benchmarking Results

4.1 Data Set

Rother et. al. [3] presents a database of 50 images plus the corresponding ground truth to be used for benchmarking foreground extraction approaches. The benchmark dataset is available on the Internet [12] and also includes 20 images from the Berkeley Image Segmentation Benchmark Database [11]. In addition to images and ground truth the database also contains user specified trimaps. These trimaps, however, are not optimal inputs for the algorithm presented here because their known foreground is not always a representative color sample of the entire foreground. Furthermore, creating such a trimap would be too cumbersome for the user, as it already contains a lot of shape information. The benchmark then would not represent the results a user could obtain. For this reason, we created an additional set of trimaps better suited for testing the approach. We asked an independent user to draw appropriate rectangles for the region of interest and known foreground in each of the images. These trimaps may still be suboptimal but it is assumed here that they represent the typical input of a user. Using a rough free hand selection instead of a rectangular area, for example, would improve the segmentation result of those images where the smallest possible rectangle already covers almost the entire picture. For the benchmark, the default values for the smoothness factor and the cluster granularity were used. Figure 4 shows an example of an image with both types of trimaps and the ground truth.

Given a perceptual accurate error measurement for foreground extraction approaches would reduce the entire task

Figure 4. From left to right: The original image, the lasso selection, the trimap by a user, and the ground truth

to minimizing this error function. Unfortunately it is difficult, maybe impossible, to create a general error measure. Because we want to create comparable results, we stick to the error measurement defined in [3]. We chose comparison with this article because the solutions presented there are commonly considered to be very successful methods for foreground extraction. The segmentation error rate is defined as:

$$\epsilon = \frac{\text{no. misclassified pixels}}{\text{no. of pixels in unclassified region}} \qquad (1)$$

In low contrast regions, a true boundary cannot be observed. This results in the ground truth database containing unclassified pixels. For comparability, these pixels are excluded from the number of misclassified pixels as in [3]. The overall error when applying the lasso trimaps provided by the database is 9.1 %. As already mentioned, the lasso selections are not optimal for the segmentation algorithm presented here. Figure 5 shows the result for the additional set of trimaps based on rectangular user selections[2]. The overall error is 4.3 % and the segmentations subjectively appear much better. The best case average error rate on the database for the *GrabCut* underlying algorithm is reported as 7.9 %[3].[3] Using different trimaps for classification results in a higher number of pixels to classify. One could object that a higher number of pixels to classify contains more pixels that are easier to classify and thus may beautify the error rate because there is no focus on the critical boundary pixels. This may be true for algorithms that seek an accurate boundary by growing from some center of the picture, or by shrinking a lasso. The algorithm proposed here makes no distinction between critical and non-critical pixels: In the color classification step every pixel has an equal chance of being misclassified no matter where in the image it is located. Having more pixels to classify therefore makes the test even harder.

[2]In this article, the images are always listed in the same order.

[3]At the time of writing of this article, a per image error measurement has not been published.

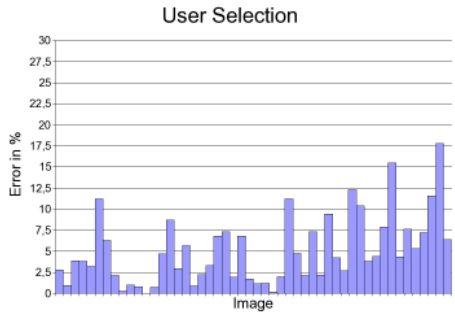

Figure 5. Error rates of segmenting the benchmark images.

Color Space	Worst Case Error	Average Error
LAB	17.8 %	4.3 %
RGB	97.0 %	12.3 %
HSI	54.2 %	6.0 %
YUV	34.7 %	5.4 %

Table 1. Average and worst-case classification results for different color spaces.

4.2 CIELAB vs. YUV vs. HSI vs. RGB

In order to test the impact of using CIELAB as the underlying color space, the algorithm was also applied to the benchmark images using YUV, HSI, and RGB. Otherwise the algorithm remained completely unchanged. CIELAB proves to be better than all other color spaces. Although YUV comes close in average, CIELAB shows a significantly smaller worst-case error. Figure 6 shows the detailed results and Table 1 summarizes the average and worst-case results. Of course, a small worst case error is very important for a generic image manipulation tool.

4.3 Strengths and Weaknesses

The benchmark shows that the presented algorithm performs well on a number of difficult pictures where it is even difficult to construct an accurate ground truth. If the contrast is good, the segmented border is accurate to a pixel (see Figure 8). The classification copes well with noise although the computation needs considerably more time for noisy input. Figure 7 shows the result of classifying a noisy image. However, looking at the resulting pictures also discloses some weaknesses. The segmentation depends heavily on the user provided trimap. The user must select a region of interest containing the whole foreground object. Failing to do so will give unusable results. Difficult im-

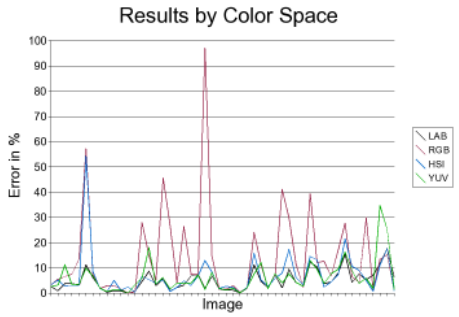

Figure 6. Error rates of segmenting the benchmark images in different color spaces.

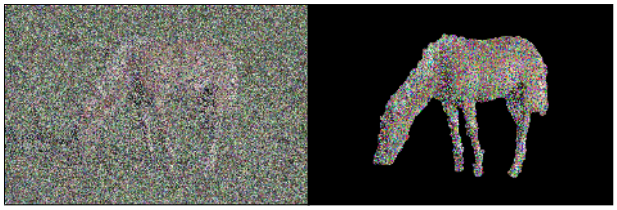

Figure 7. A fairly high signal-to-noise ratio (50 %) has little effect on the segmentation (left: original image from [7] with added noise, right: segmented horse).

ages require a wise selection of representative foreground. Therefore the user must have at least a little knowledge of what could be representative. It is not possible to extract multiple objects at once. For example, extracting multiple clouds from a sky requires several steps[4]. If two very similar objects exist in the picture, and only one of them is to be considered foreground, the segmentation mostly gives bad results. The reason is that most of the colors of the foreground are then considered background because they exist on the second object. The only workaround is to include both objects in the region of interest and to provide good foreground samples. Still, this method may fail when the unwanted similar object is bigger than the desired one. Foreground objects that are connected together with objects of the same color structure (for example, two people embracing each other) are almost impossible to segment using the approach. Most of the misclassified pixels in the benchmark result from objects that are close to the foreground object, both in color structure and in location. The same reason accounts for shadows and reflections. Still another problem is the use of the standard observer and the D65 reference white. Pictures photographed with different il-

[4]A simple trick in this case is to invert the problem. Just consider the sky to be the foreground.

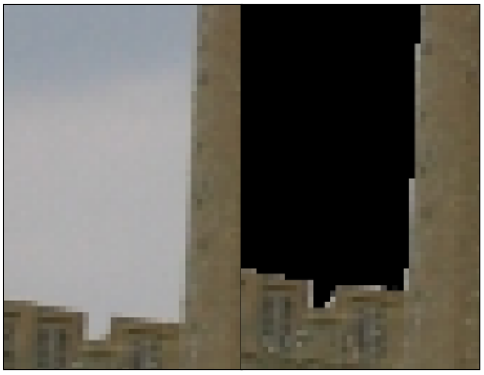

Figure 8. An example of the accuracy of the segmentation in a middle-contrast region (left: original image, right: segmented image).

lumination conditions are segmentated poorly. Especially underwater scenes are awkward to segmentate, because of the natural color quantization underwater [14]. For these pictures, a different model would have to be used. In the case of underwater photography, this model would have to depend on the depth where the picture was taken.

5 Video Segmentation

Due to the fact that the k-d-tree structure enables fast range queries, the classification algorithm can also be used for video segmentation. After converting each video frame to CIELAB space, the first processing step simply uses a Gaussian noise filter and calculates the difference between two consecutive frames pixel wise using Euclidean distance. The confidence matrix is initialized with these distance values normalized between 0 and 1.

The next processing step is to apply exponential smoothing on the last three confidence matrices. This improves the frame rate independence of the algorithm. Now, similar to the first user selection, a representative sample of the background has to be reconstructed.

To distinguish noise from real movements, the following simple but general model is used. Given two measurements m_1 and m_2 of the same object with each measurement having a maximum deviation e of the real world due to noise or other factors, it is clear that the maximum possible deviation between m_1 and m_2 is $2e$. Given several consecutive frames, we estimate e to find out which pixels changed due to noise and which pixels changed due to real movement. To achieve this, the color changes of each pixel over a time period $h_{(x,y)}$ (where x and y specify pixel coordinates) is recorded. It is assumed that during this interval, the minimal change should be one that is caused by

Figure 9. For extracting foreground objects in a video the trimap has to be reconstructed from motion statistics. The images show the original video (left) and known background that was reconstructed over several frames (right). The white regions constitute the unknown region.

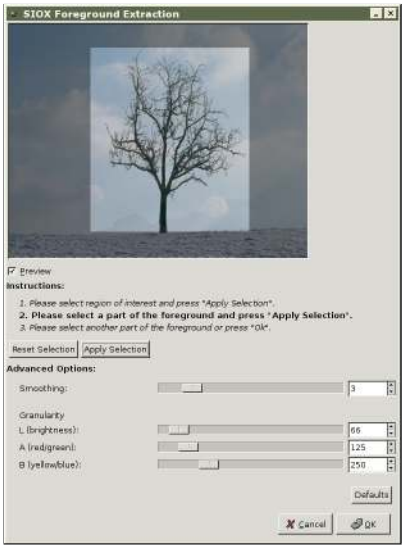

Figure 10. A screenshot of the GIMP plugin.

noise. The frame is then divided into 16 sub-frames and the changes in each sub-frame are accumulated. Under the assumption, that at least one of these sub-frames was not touched by any foreground object $2e$ is estimated to be the maximal variation of the sub-frame with the minimal sum. Then those pixels of the current frame are joined with the background sample that during this history period $h_{(x,y)}$ did not change more than our estimated $2e$. The history period $h_{(x,y)}$ is initialized to one second and is continously increased for pixels that are seldom classified as background, to avoid that a still-standing instructor is added to the background buffer. Figure 9 shows some examples of reconstructed backgrounds. It normally takes several seconds, until enough pixels can be collected to form a representative subset of the background. The background sample buffer is organized as an aging FIFO queue.

The color classification (as described in Section 3.3) is performed using the representative background sample. Once built-up, the tree is only updated when more than a quarter of the underlying background sample has changed. The confidence matrix is then updated by averaging the results of the classification with the old confidence values. This lowers the risk that colors that appear both in the background and foreground are classified in the end as background. Like in still-image segmentation, a connected component analysis is performed for all pixels classified as foreground, i.e. pixels with a confidence greater than 0.5.

The biggest blob is considered to be interesting, and all other blobs (mostly noise and other moving objects) are put back into the background buffer. Again, the elements of the confidence matrix are directly mapped to α-values, specifying the opaqueness of each pixel.

The performance of the algorithm depends on the complexity of the background (entropy) and on how often it has to be updated. The algorithm was applied to extract an instructor standing in front of an electronic chalk board [15, 8]. Using these segmentation videos, the current Java-based prototype implementation processes a 640×480 video at 6 frames per second. This includes a preview window and a motion JPEG compression. A 320×240 video can be processed at 14 frames per second on a standard 3 GHz PC. This rate can be dramatically increased by utilizing the SIMD multimedia instruction sets of modern CPUs.

As the algorithm focuses on the background it provides rotation and scaling invariant tracking of any biggest moving object.

6 Conclusion and Future Work

The article presented a color classification algorithm that can be used for foreground extraction in images as well as in videos. The advantage of the algorithm is that its central data structure is efficient and not spatially bounded to a certain picture, like a graph spanned between pixels. Once built, the structure can be reused for subsequent frames in a video. Benchmark results show that an implicit use of the spatial information provided in the image using region growing suffices to compete with approaches that use this information explicitely by spanning graphs over pixels.

Figure 11 shows some example images from the benchmark. A usable implementation of the algorithm is available as a plugin for the GIMP [10], see Figure 10 for a screen shot of the current prototype. SIOX is open source and currently being integrated into the GIMP. It will be part of the core functionality in GIMP 2.4. This will make the extraction process more effective because of the various manual post editing facilities that are available in GIMP.

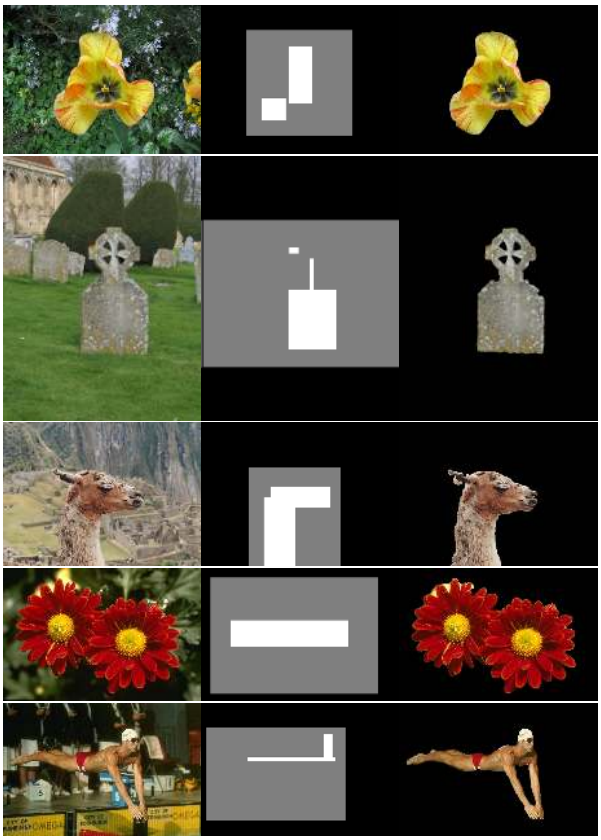

Figure 11. Some of the benchmark images, along with the user provided trimaps, and the segmentation result.

Future enhancements may include automatic finding of the cluster sizes according to the color distribution of the image and a further improvement of the classification speed. Different observers and illumination models may improve segmentation of underwater scenes, space images, or pictures taken at night. We are also experimenting with the integration of color distribution based methods and with using the SCIELAB space [19].

Further information, including detailed benchmark results, demonstration videos, the GIMP plugin, and a preview of the GIMP tool is available at: *http://www.siox.org*

Acknowledgments

We gratefully acknowledge assistance from Sven Neumann (maintainer of the GIMP) who is currently integrating SIOX into the GIMP, Lars Knipping and Tobias Lenz who have provided ideas to improve the project, and Christian Zick for helping to produce the demonstration videos.

References

[1] Adobe Systems, Inc. Adobe Photoshop User Guide, 2002.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[3] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proceedings of European Conference on Computer Vision (ECCV2004)*, May 2004.

[4] Y. Boykov and M.-P. Jolly. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In *Proceedings of the International Conference on Computer Vision*, pages 105–112, Vancouver, Canada, July 2001.

[5] S. D. Chuang Y.-Y., Curless B. and S. R. A bayesian approach to digital matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[6] Corel Corproation. Knockout User Guide, 2002.

[7] Dudo Erny. Free Pictures Download. http://www.bigfoto.com, 2005.

[8] G. Friedland, L. Knipping, J. Schulte, and E. Tapia. E-Chalk: A Lecture Recording System using the Chalkboard Metaphor. *International Journal of Interactive Technology and Smart Education*, 1(1), February 2004.

[9] G. Friedland and K. Pauls. Architecting Multimedia Environments for Teaching. *Computer*, 38(6):57–64, June 2005.

[10] GIMP Team. The GNU Image Manipulation Program. http://www.gimp.org, 2005.

[11] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[12] Microsoft Research. Microsoft Foreground Extraction Benchmark Dataset. http://research.microsoft.com/vision/cambridge/i3l/segmentation/GrabCut.htm, 2004.

[13] E. Mortensen and W. Barret. Intelligent Scissors for Image Composition. In *Proceedings of ACM Siggraph Conference*, August 1995.

[14] D. Richardson. *Adventures in Diving Manual*. International PADI, Inc, Rancho Santa Margarita, CA, 2000.

[15] R. Rojas, G. Friedland, L. Knipping, and E. Tapia. Teaching With an Intelligent Electronic Chalkboard. In *Proceedings of ACM Multimedia 2004, Workshop on Effective Telepresence*, pages 16–23, New York, New York, USA, October 2004.

[16] C. Rother, V. Kolmogorov, and A. Blake. GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts. In *Proceedings of ACM Siggraph Conference*, August 2004.

[17] Y. Rubner, C. Tomasi, and L. J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[18] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley and Sons, New York, NY, 1982.

[19] X. Zhang, J. E.Farrell, and B. A. Wandell. Applications of a Spatial Extension to CIELAB. In *SPIE Electronic Imaging 97*, 1997.