# Extending the SIOX Algorithm: Alternative Clustering Methods, Sub-pixel Accurate Object Extraction from Still Images, and Generic Video Segmentation

Gerald Friedland, Kristian Jantz, Tobias Lenz, and Raúl Rojas

Freie Universität Berlin, Institut für Informatik
Takustr. 9, 14195 Berlin, Germany
{fland, jantz, tlenz, rojas}@inf.fu-berlin.de
January 2006

**Abstract.** This paper presents a practical approach for object extraction from still images and video sequences that is both: simple to use and easy to implement. Many image segmentation projects focus on special cases or try to use complicated heuristics and classificators to cope with every special case. The presented approach focuses on typical pictures and videos taken from everyday life working under the assumption that the foreground objects are sufficiently perceptual different from the background. The approach incorporates experiences and user feedback from several projects that have integrated the algorithm already. The segmentation works in realtime for video and is noise robust and provides subpixel accuracy for still images.

## 1 Introduction

This paper is concerned with extracting objects from photographies and videos. Many projects focus either on very special cases, like traffic video sequences where a constant background can be assumed and no camera movement takes place, or try to use complicated algorithms with many case distinctions. The presented approach focuses on typical home-made pictures and videos taken from everyday life. The algorithm tells apart foreground from background by certain color characteristics, therefore it requires the assumption that the foreground objects are sufficiently perceptual different from the background. Fortunately, digital cameras typically try to optimize color variance resulting in perceptual dissimilarity of different objects [1]. The goal is to provide a generic segmentation engine that works for both, video and still-images. The user interaction should be kept as simple as possible and the algorithm as general as possible.

Because the implementation of the presented algorithm is open source, there has already been a lot of user feedback. The algorithm has already been integrated into several projects, most prominently into the GIMP. Solutions for the early reported problems like segmentation of highly detailed textures and removal of spill colors are also presented in this paper.

After a brief discussion of the related work, section 3 introduces the original algorithm which has been derived from a technique used in image retrieval. Section 4 discusses application specific optimizations. Sections 5 and 6 discuss the different creation of the initial input for still images and video. Section 7 presents an extension of the algorithm for subpixel accuracy, before section 8 draws conclusions and presents future work.

## 2 Related Work

The standard technologies for extracting foreground objects onto a given background are chroma keying and background subtraction [2]. These techniques are often not applicable, because the background of a given scene is not always monochromatic and/or fixed.

Much work has been done on tracking objects for computer vision (like robotic soccer [3], surveillance tasks [4], or traffic applications [5]). Most of these approaches concentrate on special features of the foreground and in these domains, realtime performance is more relevant than segmentation accuracy as long as the important features can be extracted from each video frame. Numerous computationally intensive segmentation algorithms have been developed in the MPEG4 community, for example in [6].

The use of stereo cameras for the reconstruction of depth information has been thoroughly investigated. Not only is disparity estimation a calculation intensive task, it also involves texture matching. Thus this method is affected by the same problems as 2D segmentation algorithms: very similar or homogeneous areas are very difficult to distinguish [7].

A nicely written summary and discussion of several foreground extraction methods for still images can be found in [8].

The most popular tool to extract foreground is *Magic Wand* [9]. *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 good for images that contain very few colors, such as drawings. Intelligent Scissors [10] can be used to select contiguous areas of similar color weight in a fashion similar to Magic Wand. The primary difference 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. For natural images, finding the correct tolerance threshold is often cumbersome. A satisfactory segmentation is only achieved for very primitive pictures.

Bayes Matting [11] gets a known foreground, known background, and an unsure region 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.

*Knockout* is a proprietary plugin for *Photoshop* [12]. According to [11] the results are sometimes similar, sometimes of less quality than Bayes matting.

*Adobe Photoshop* contains a tool called *extract*, the user interaction required is similar to Knockout and the tool gives similar results.

*Grabcut* [8] is a two step approach. The first step is an automatic segmentation step that relies on the work of *Graph Cut* [13, 14]. 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 computes the segmentation. The user only provides the region of interest. *Grabcut*'s manual post processing tools include a so-called background brush, a foreground brush and a matting brush to smooth borders or re-edit classification errors manually. *Grabcut* surpasses all the algorithms mentioned earlier, but can only select one object at a time. The algorithm minimizes a global cost function which cannot distinguish between fine local details and noise. It completely fails for highly detailed regions, see figure 1, and noisy pictures.

**Fig. 1.** Grabcut's bad result on highly detailed textures.

## 3 Our Approach

Our goal is to separate a user intended set of pixels (=*foreground*) from the rest (=*background*). This classification is basically done in a binary fashion—every

pixel is either foreground or background—but to improve the quality of the results we also allow fractional *confidence values* for each pixel. This is especially helpful at the transition regions between foreground and background where anti-aliasing in the original image usually blurs the change.

Our algorithm outputs a selection of the original image with modified alpha channel and works as follows. All the color operations are done in LAB space [15] due to its natural distance measure.

1. User classifies sure foreground and sure background regions in the image.
2. Create a set of representative colors for foreground and background by clustering the color space driven by the user input. → *signatures* $S_F, S_B$
3. Assign all image points to foreground or background by a weighted nearest neighbor search in $S_F$ and $S_B$. → *confidence matrix*
4. Apply some standard image processing operations like erode/dilate and blur to the confidence matrix to remove artifacts.
5. Find the connected components with high confidence which are either large enough or marked by the user. → *selected foreground*

(a) The original has very smooth transitions making it even hard for a human to find the exact boundaries.

(b) The result has clearly visible dents and holes.

**Fig. 2.** If color signatures $S_F$ and $S_B$ overlap, the result would be a bad segmentation.

If the results are not satisfying, the user will be able to refine its initial classification inductively.

The most important part is the clustering in step 2. It bases on the two stage kd-tree construction by Rubner et.al. [16]. The adaption for segmentation purposes is described in our preceding paper [17]. The general kd-tree was introduced in 1975 by Bentley [18]. Alternative methods for this clustering are discussed in section 4. The signatures resulting from the clustering typically have a few hundred points or less which makes the subsequent steps very fast.

The most critical drawback of this approach is the color dependence. Our basic method to distinguish foreground and background relies on the color signatures derived from the user input. Although many photos are well-separable

by color, the algorithm cannot deal with camouflage. If the foreground and background share many identical shades of similar colors, the algorithm might give a result with parts missing or incorrectly classified foreground as seen in figure 2.

On the other hand the approach offers some unique advantages compared to other algorithms. A very powerful and maybe underestimated strength is the simplicity of the building block concept. If we consider the initial user input as confidence values 0, 0.5 and 1 respectively, each of the well-separated steps would have a confidence matrix as input and a refined version of this matrix as output. Every step might become replaced on its own. For video applications the user input step is replaced by an automated learning algorithm and the kd-tree clustering is replaced by the faster clustering from section 4 for example. The complete algorithm is easy to implement efficiently with standard data-structures.

(a) The original image with 50% random noise.

(b) The result is almost as good as from a source without any noise.

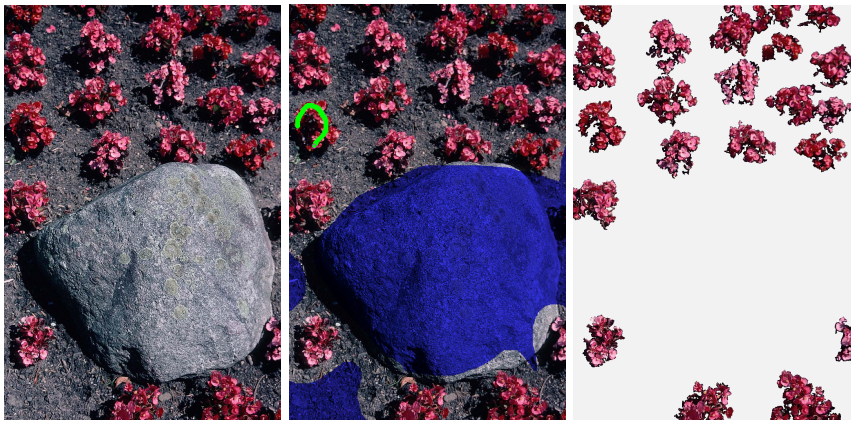
**Fig. 3.** Even a large amount of noise has negligible influence on the result.

Other unique features are high noise robustness (see figure 3) and the ability to select multiple components at once as seen in figure 4.

Running the algorithm against a benchmark set up for Grabcut [14] produces 3.90% of misclassified pixels. The details of this result can be found in our Technical Report [19].

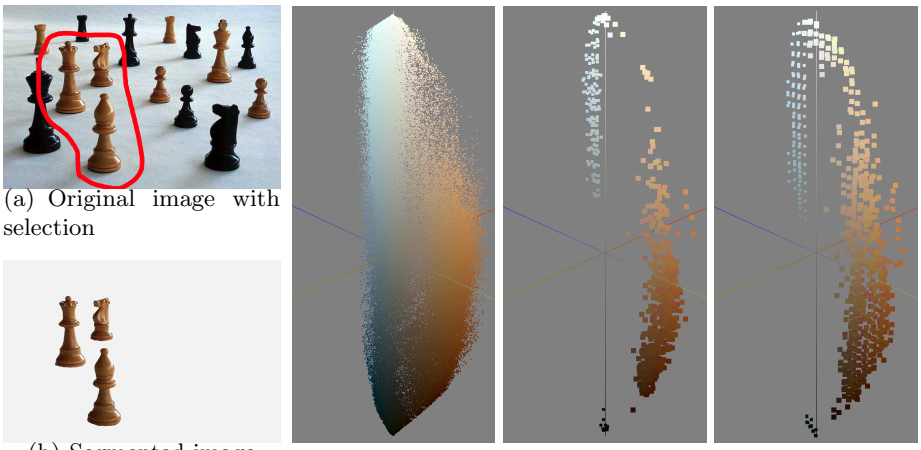
## 4 Alternative Clustering Strategies

The algorithm originally used for color clustering utilizes a kd-tree as its basic structure [17]. This tree is build up in two stages to achieve a good distribution and to obtain a representative signature with few points. This version performed best in our tests with respect to quality but the two stage process is the bottleneck when it comes to runtime.



(a) Original image (b) User specified regions (c) Result

**Fig. 4.** Simultaneously selecting multiple components is sometimes a very useful feature and saves a lot of time.



(a) Original image with selection

(b) Segmented image

(c) All colors (d) Signature from the kd-tree (e) Signature from the array

**Fig. 5.** In (c) all colors from (a) are visualized as points in LAB space, (d) shows the color signature from the kd-tree clustering algorithm and (e) the signature from the faster array based algorithm. The fat and small dots are the foreground and background signature respectively.

For applications where speed is a major issue, like in video segmentation with high frame rates or when a large batch of images is processed at once, we can offer a much simpler and 8-10 times faster clustering algorithm.

This speedup is achieved by exchanging the dynamic splitting rule from the kd-trees by a fixed discretization of the spherical LAB space. This can be realized as a simple three dimensional array which does not only allow very fast access to every cell but also allows incremental updates when new foreground or background is selected without the need to rebuild the whole structure.

Optimizing even further we replaced the usually used LAB colors by standard RGB colors to save the costly conversion for each pixel. This approach gave still very good results for example in the Microsoft benchmark [14] (4.4% error rate) but has some counterintuitive effects in the interactive version.

To compare different clustering techniques we look at the clusters they create as seen in figure 5. The discretized LAB space yields a very regular arranged signature compared to the kd-tree approach due to its array implementation which allows further geometric optimizations.

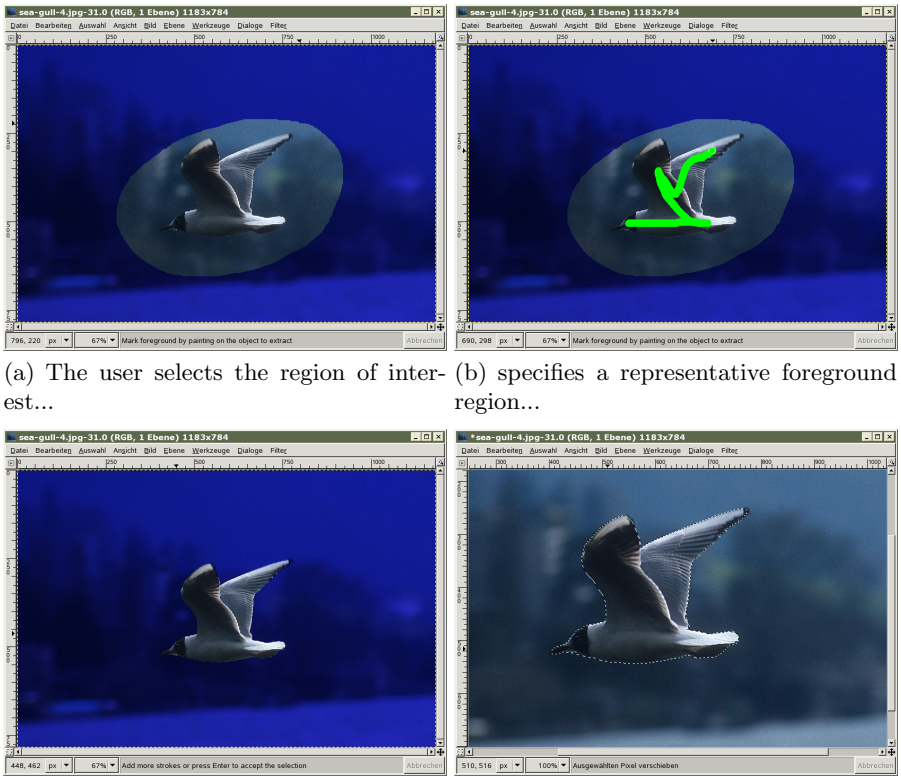
## 5 User Interaction

Figure 6 shows the user interaction necessary to create the initial confidence matrix as it is implemented in the GIMP. The user uses a freehand selection tool to specify the region of interest (figure 6a). It contains all foreground objects to extract and as few background as possible. The pixels outside the region of interest form the sure background while the inner region define a superset of the foreground, i.e. the unknown region. The sure background is visualized as dark area.

The user now uses a foreground brush to mark representative foreground regions (figure 6b). Internally, this input is mapped into 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. Once the mouse button has been released, the selection is shown to the user (figure 6c). The selection can be refined by either adding further foreground markings or by adding background markings using the background brush. Pressing enter results in the creation of the final selection mask (figure 6d).

## 6 Video Object Extraction

For object extraction in videos the confidence matrix is learned from motion statistics. The input is a sequence of digitized YUV or RGB video frames either from a recorded video or directly from a camera. The first processing step simply uses a Gaussian noise filter and calculates the difference of two consecutive frames pixelwise using Euclidean distance. The confidence matrix is initialized with these distance values normalized to $[0; 1]$. The next processing step is to apply




(a) The user selects the region of interest...

(b) specifies a representative foreground region...

(c) verifies and optionally refines the result...

(d) and is provided with a tight selection.

**Fig. 6.** User interaction to provide initial confidence matrix.

exponential smoothing on the last three confidence matrices. In our experiments this improves the frame rate independence of the algorithm.

It is often non-trivial to build a model of the background since it might be changing continuously. To distinguish noise from real movements, we use the following simple but general model. 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, we record the color changes of each pixel $(x, y)$ over a certain number of frames $t(x, y)$. We assume that in this interval, the minimal change should be one that is caused by noise. The recorded data is continuously evaluated. The frame is divided into 16 equally sized regions and changes are accumulate in each region. Under the assumption, that at least one of these regions was not touched by any foreground object, $2e$ is estimated to be the maximum variation of the region with the minimal sum. We then join all pixels of the current frame with the background sample that during the recording period $t(x, y)$ did not change more than our estimated $2e$.

The recording period $t(x, y)$ is initialized with one second and is continuously increased for pixels that are seldom classified as background, to avoid that a still-standing foreground object is added to the background buffer. In our experiments, it took few seconds until enough pixels could be collected to form a representative subset of the background. We call this time period the initialization phase. The background sample buffer is organized as an aging FIFO queue.

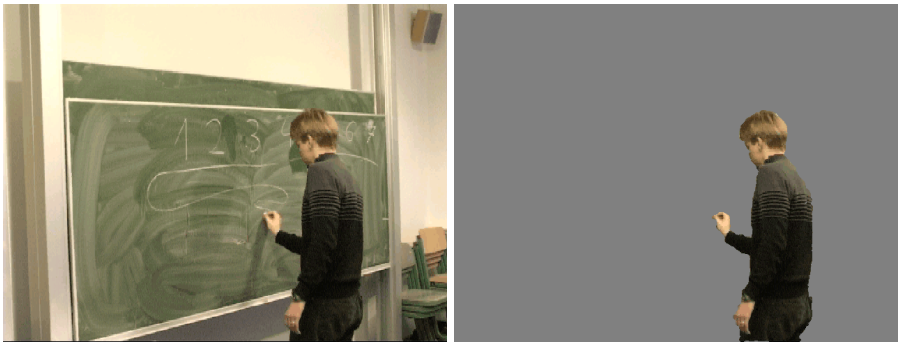
The background and foreground samples are fed into the clustering algorithms as described above. Once built-up, the clustering is only updated, when more than a quarter of the underlying background or foreground sample has changed.

Since many colors are identical in consecutive frames a hashtable allows for very efficient classification of the non-background pixels in each frame. The performance of the algorithm depends on the complexity of the background and on how often it has to be updated. In most cases, however, our current Java-based prototype implementation processes a $640 \times 480$ video at 25 frames per second. This includes a preview window and a motion JPEG compression.

As the algorithm focuses on the background it provides rotation and scaling invariant tracking of objects that still works if the camera is moved substantially. However, if the entire scene changes, the algorithm would have to learn the new background which would again take a few seconds.

## 7 Detail Refinement Brushes

For most pictures, a pixel accurate object extraction gives satisfying results. Images containing highly structured textures, like hair or fine tree branches, however, require subpixel accuracy. subpixel accuracy is also needed to remove



(a) An image from a video sequence. (b) The lecturer separated from the blackboard.

**Fig. 7.** A video application for segmentation in e-learning.

spill colors that result from motion blur or image filters that smooth borders. For this reason, a third brush is provided in addition to the foreground and background brush: the so-called detail refinement brush. Using coarse strokes, the brush is used to refine regions where satisfying results cannot be achieved automatically.

A pixel $p$ determined by its LAB color is affected by the brush in the following way. Let $f$ be the closest point to $p$ in the foreground signature $S_F$ and $b$ the closest point to $p$ in $S_B$. The orthogonal projection of $p$ onto the segment $\overline{fb}$ splits $\overline{fb}$ in a certain ratio which specifies the confidence value and hence the used alpha mask for this pixel. For sensible results, the angle spanned by $f, p, b$ must not be to small or a more suitable pair of points $f, b$ has to be found.

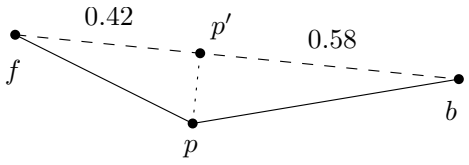
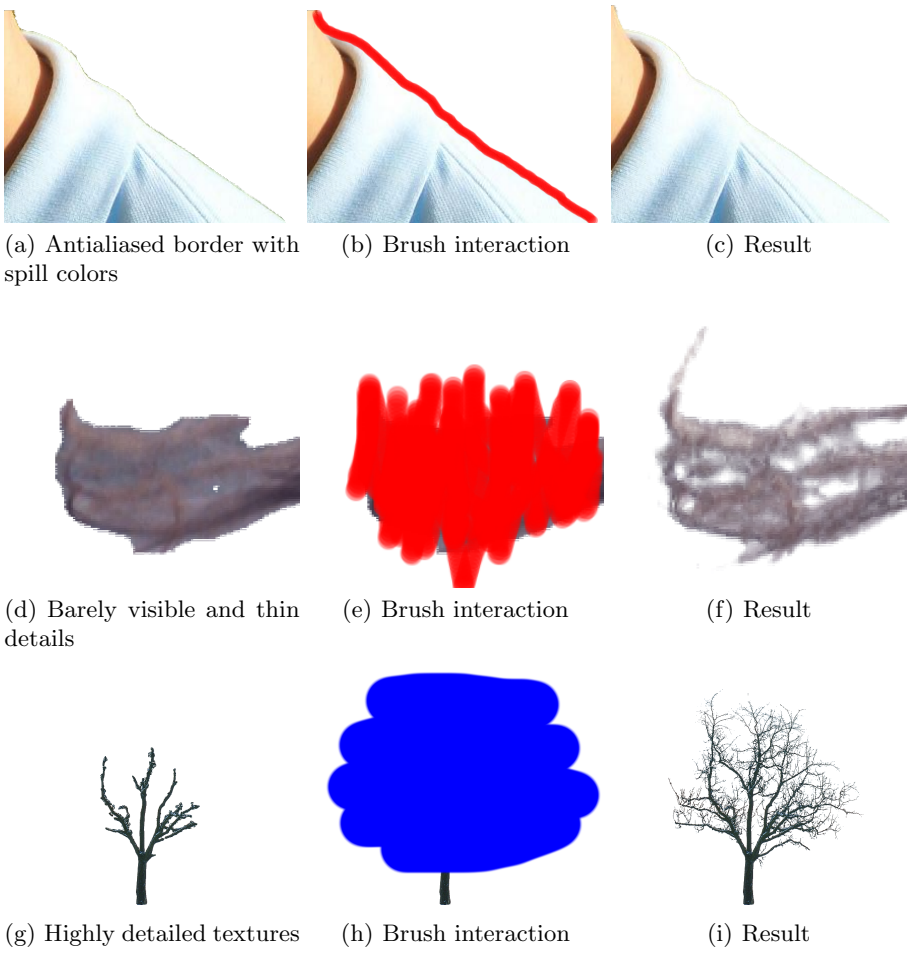



**Fig. 8.** The alpha value for a pixel is determined by the distance ratio of distance to foreground and background.

In figure 9, a, d, g show the result after automatic object extraction, figures b, e, h show the manual brush interaction, and the figures c, f, i show the refined results.



(a) Antialiased border with spill colors (b) Brush interaction (c) Result

(d) Barely visible and thin details (e) Brush interaction (f) Result

(g) Highly detailed textures (h) Brush interaction (i) Result

**Fig. 9.** Sample results for the detail refinement brush.

## 8 Conclusion and Future Work

This paper presents a runtime efficient adaptation of originally image retrieval techniques to solve foreground segmentation problems in videos and still images. In videos, the method enables scale and rotation invariant tracking of foreground and—as the background is learned—it also handles the classification of newly introduced objects.

The same simple algorithm saves users tedious manual still image segmentation work in many cases. By changing the way the confidence matrix is generated, the core of the algorithm can be used for both, still images and video. The generated color signatures can further be used to cope with highly detailed textures even with subpixel accuracy.

The presented approach can be applied to a variety of other problems where a foreground object should be tracked, extracted and/or identified.

Future enhancements may include an automatic adaption of the clustering strategy according to the color distribution of the image and a further improvement of the algorithm taking into account the first derivative of the picture. The implementation of CIELABs 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 [20].

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

## Credits

The following people have contributed to the methods described in this document:

**Gerald Friedland** has conceived the SIOX algorithm and implemented both the Java prototype of the video version and a native port in ANSI C. He developed the initial GIMP version and developed the Detail Refinement Brush and the video segmentation.

**Tobias Lenz** has conducted the experiments on the alternative clustering strategies. He also contributed to the SIOX implementation in the GIMP core.

**Kristian Jantz** has helped developing the algorithm, implemented most of the Gimp Plugin and also contributed to the GIMP core implementation.

**Raúl Rojas** is head of the E-Chalk project and inspired the development of the instructor segmentation algorithm that led to SIOX.

## References


1. Adams, J., Parulski, K., Spaulding, K.: Color processing in digital cameras. IEEE Micro **18** (1998) 20–30
2. Gibbs, S., Arapis, C., Breiteneder, C., Lalioti, V., Mostafawy, S., Speier, J.: Virtual Studios: An Overview. IEEE Multimedia **5** (1998) 18–35
3. Simon, M., Behnke, S., Rojas, R.: Robust real time color tracking. In: RoboCup 2000: Robot Soccer World Cup IV, Heidelberg, Germany, Springer (2001) 239–248
4. Haritaoglu, I., Harwood, D., Davis, L.: W4: Real-Time Surveillance of People and Their Activities. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 809–831
5. Beymer, D., McLauchlan, P., Coifman, B., Malik, J.: A Real-time Computer Vision System for Measuring Traffic Parameters. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). (1997)
6. Chien, S.Y., Huang, Y.W., Ma, S.Y., Chen, L.G.: Automatic Video Segmentation for MPEG-4 using Predictive Watersheds. In: Proceedings of IEEE International Conference on Multimedia and Expo, Tokyo, Japan (2001) 239–243
7. Zitnick, C.L., Kanade, T.: A Cooperative Algorithm for Stereo Matching and Occlusion Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 675–684
8. Rother, C., Kolmogorov, V., Blake, A.: GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts. In: Proceedings of ACM Siggraph Conference. (2004)
9. Adobe Systems, Inc: Adobe Photoshop User Guide (2002)
10. Mortensen, E., Barret, W.: Intelligent Scissors for Image Composition. In: Proceedings of ACM Siggraph Conference. (1995)
11. Chuang Y.-Y., Curless B., S.D., R., S.: A bayesian approach to digital matting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2001)
12. Corel Corproation: Knockout User Guide (2002)
13. Boykov, Y., Jolly, M.P.: Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In: Proceedings of the International Conference on Computer Vision, Vancouver, Canada (2001) 105–112
14. Rother, C., Kolmogorov, V., Blake, A.: Grabcut - interactive foreground extraction using iterated graph cuts. Proc. ACM Siggraph (2004)
15. Wyszecki, G., Stiles, W.S.: Color Science: Concepts and Methods, Quantitative Data and Formulae. John Wiley and Sons, New York, NY (1982)
16. Rubner, Y., Tomasi, C., Guibas, L.J.: The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision **40** (2000) 99–121
17. Friedland, G., Jantz, K., Rojas, R.: Siox: Simple interactive object extraction in still images. In: Proceedings of the IEEE Symposium on Multimedia (ISM2005), Irvine, California (2005)
18. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18** (1975) 509–517
19. Jantz, K., Friedland, G., Knipping, L., Rojas, R.: Image Segmentation by Uniform Color Clustering – Approach and Benchmark Results. Technical Report B-05-07, Freie Universität Berlin, Institut ür Informatik (2005)
20. Zhang, X., Farrell, J.E., Wandell, B.A.: Applications of a Spatial Extension to CIELAB. In: SPIE Electronic Imaging 97. (1997)