OBJECT CUT AND PASTE IN IMAGES AND VIDEOS

GERALD FRIEDLAND
International Computer Science Institute
1947 Center Street, Suite 600
94704-1198 Berkeley, CA, USA
fractor@icsi.berkeley.edu

KRISTIAN JANTZ*, TOBIAS LENZ†, FABIAN WIESEL‡
and RAÚL ROJAS§
Institut für Informatik, Freie Universität Berlin
Takustr 9, 14195 Berlin, Germany
*jantz@inf.fu-berlin.de
†lenz@inf.fu-berlin.de
‡wiesel@inf.fu-berlin.de
§rojas@inf.fu-berlin.de

Most image and video editing applications implement only a set of low-level operations, such as linear and non-linear filters, scaling, and simple rectangular cropping. A skilled user is required to combine these traditional tools creatively to perform a useful task. This article presents an example of an approach that takes into account the semantics of the image and is therefore more powerful than traditional tools. Using a model of the foreground and the background of a given image, the SIOX tool (Simple Interactive Object Extraction) allows a user to cut out a certain object from an image or video with very little user interaction. The core of the algorithm is a color clustering approach that has been derived from image retrieval techniques and evaluated against a benchmark proposed by Microsoft Research. The method allows for sub-pixel accuracy, allows video segmentation in realtime (640 × 480 × 25), and is noise robust. SIOX has recently been integrated into the development versions of several open source image and video editing applications such as GIMP, Blender, and Inkscape.

Keywords: Object extraction; image and video processing.

1. Image Manipulation as a Set of Low Level Operations

While producing photographs or video clips has gotten much simpler in recent years, editing them is still a task for professionals. Digital cameras are able to generate hundreds of pictures containing millions of pixels in a few seconds. For the vast majority of people, however, the only option is viewing them because most image manipulation software still focuses on providing low-level tools.

Even after several decades of image processing and computer vision research, popular photo editing software such as Adobe Photoshop, Corel Draw, Corel
Paintshop Pro or others put things like brush patterns, sharpening filters, or histogram equalization on the top of their feature list. Without doubt, these tools are an important part of the functionality of every image manipulation program. However, editing a photograph by tuning the components of a convolution filter matrix should not be required for end-users. They are accustomed to creating images in one click, without knowing anything about the difficult processes that are going on when they press the button on the camera. It is hard enough for many users to connect the camera to a PC using to retrieve the pictures, sort them somehow and then maybe print them or burn them on a cdrom. When it comes to small corrections of the photos, people are forced to work with applications that are fully loaded with low-level tools, that directly expose technical details to the user. For example, somebody who wants to copy and paste an object is forced to know vocabulary like alpha channel, layer, or path. The reason for this is that low-level tools are just not able to incorporate the semantics of the content of the picture.

This article presents our approach for object extraction in still images and videos, called SIOX (Simple Interactive Object Extraction). The tool incorporates a model of the content (here background and foreground) defined by the user, that is also refined during the several computational steps. Even though computers may never be able to “know” what the object is, i.e. build a semantic model of the picture, the user is able to work together with the computer on approximating it. This greatly simplifies separating an object from its background. SIOX has been released as an open-source segmentation framework and has subsequently been integrated into several open-source image and video manipulation applications. This article describes SIOX, compares it with related approaches, and presents some details of the integration of the method as a cut-out tool in different image and video manipulation applications.

2 The State of the Art

Many popular image manipulation programs contain semi-automatic object extraction tools. The most popular tool for extracting foreground semi-automatically in image manipulation programs is Magic Wand. Magic Wand starts with a small user-specified region. The algorithm then performs a region growing by absorbing connected pixels such that all selected pixels fall within some user-adjustable tolerance of the color statistics of the specified region. For natural images, finding the correct tolerance threshold is often problematic. The methods works well for images which contain few colors, such as drawings. For “natural” images that contain many colors, such as photographs, the results are unusable or the interaction required is far from being feasible. In practice, it is better to use non-automatic tools, for example a path tool, to extract an object from a photograph by hand rather than to use Magic Wand.

Intelligent Scissors [16] can be used to select contiguous areas of similar color in a fashion similar to Magic Wand. Intelligent Scissors creates a selection boundary
by assisting the user to create a set of connected line segments around the object. Clicking with the mouse creates nodes that are joined using curve shapes that attempt to follow color weights. Although the method works even with sub-pixel accuracy, a satisfactory segmentation is only achieved with very simple photographs that have clear edges.

Bayes Matting [4] gets a shrunked shape of the object and a subset of the background as input. The user uses a brush to coarsely redraw the shape of the input with the brush stroke having to contain both foreground and background. The algorithm then tries to compute opacity values over the pixels marked with the brush. The main disadvantage is that for complicated objects the user must specify quite detailed shape information for the algorithm to work properly. Knockout-2 is a proprietary plug-in for Photoshop [5]. According to [4] the results are sometimes similar, sometimes of less quality than Bayes Matting. Adobe’s Photoshop contains a tool called Extract. It requires a little less user interaction. Instead of two strokes, only one thick brush strokes has to be drawn by the user, which has to cover the edge of the object. Extract gives similar results to Knockout-2 [23]. Figure 1 shows a comparison of interaction and results between SIOX and Knockout 2.

Fig. 1. A comparison (horse image from [6]) between Corel’s Knockout 2 and SIOX as implemented in GIMP (see Section 4). Upper row: Knockout 2 requires the specification of an outer (red) and an inner region (green). The tool then tries to classify the unknown pixels between the two strokes. Knockout’s output is shown in the right picture. Bottom row: SIOX requires the selection of a region of interest and optionally a very coarse grain specification of known foreground. The output is shown in the right picture.
GrowCut \[24\] is a very recent algorithm based on a cellular automaton. The classification of a pixel is partly determined by the classification of its neighbors. Doing this over many iterations, the selection will become more and more stable. Due to the large number of iterations required, this process takes more than a minute even for moderate complex screen-resolution images that are far from the high resolutions of modern digital cameras for example.

Grabcut \[19\] is a two step approach. The first step is automatic segmentation that relies on the work of Graph Cut \[3\]. The second step is manual post-editing. The idea of automatic classification is reduced to building a graph where each pixel is a node with outgoing edges to each of the 8 neighboring pixels. The edges are weighted such that a max-flow/min-cut problem computes the segmentation. The user only selects 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. In terms of robustness, 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 therefore fails for highly detailed regions and noisy pictures (compare Fig. 14).

GrabCut was extended by \[12\] and \[25\] who present semi-automatic video cut-out tools that are far from being realtime capable. To accelerate the interactive refinement, \[25\] cluster the pixels by a hierarchical mean shift into 2D regions, which in turn, are combined by motion estimation to 3D regions. After a manual specification of known background, the GrabCut algorithm generates a contour, which is further refined by reconstructing a 3D-contour mesh. The whole process is reported to be computationally quite expensive: more than 22 seconds per frame.

3. Algorithm Description

The core idea behind the approach presented here is based on the notion of a color signature. A color signature models an image or part of an image by its representative colors. This abstraction technique is frequently used in different variants in image retrieval applications, where color signatures are used to compare patterns representing images, see for example \[17\]. SIOX requires a color image and works under the assumption that the foreground objects are sufficiently perceptually different from the background. Fortunately, digital cameras typically try to optimize color variance resulting in perceptual dissimilarity of different objects \[1\].

Of course, there is no unique definition of “foreground” or “object” because the semantics ultimately depends on the understanding of the individual who is perceiving the image. Inside the scope of the algorithm, SIOX defines foreground to be a set of spatially connected pixels that are “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. However, during the course of this text, even our own definition will change slightly.
The input for the SIOX algorithm is a color image or a video-frame in CIELAB space and an initial confidence matrix $M_i$. A confidence matrix is a matrix of the same dimensions as the image. Each element of the matrix contains a floating point number that lies in the interval $[0, 1]$ and corresponds to one pixel in the image. A value of 0 means the corresponding image pixel belongs to the background, a value of 1 means the corresponding image pixel belongs to the foreground. Any value between 0 and 1 describes a certain tendency that the corresponding pixel belongs to either foreground or background, with 0.5 expressing no tendency. In the following, confidence values of 1 mean known foreground, values of 0 known background, and values of 0.5 unknown. This notion of a confidence matrix has a few advantages. The confidence matrix along with the original picture can easily be passed between different processing steps or be serialized. Its elements can easily be interpreted as probabilities or as values of a gray-scale image. The latter interpretation allows to apply standard image operations, such as convolutions or morphological operators, without having to touch the original picture. The confidence values can directly be mapped to transparency values. The input confidence matrix for SIOX may contain known foreground, known background, and unknown elements (defined by the confidence values 1.0, 0.0, and 0.5). It must, however, at least contain known background. $M_i$ is either specified by the user or generated by an automatic classifier. The steps of the algorithm are as follows:

(1) Create color signatures $S_B$ and $S_F$. $S_B$ represents the specified known background and $S_F$ the known foreground (either it has been specified or the signature is calculated as a difference signature between the signature of the entire image and $S_B$).

(2) Classify each unknown pixel of the image as foreground or background using a nearest-neighbor search in $S_F$ and $S_B$. This produces a new confidence matrix $M_o$.

(3) Filter out noise by applying erode/dilate and blur on the matrix $M_o$ to remove artifacts and optionally close holes up to a specific size.

(4) Find the connected components with high confidence in $M_o$ which are large enough or correspond to user markings. Set the values of all other connected components to 0.

(5) Apply the confidence matrix $M_o$ to the image. This is usually done by mapping the elements of $M_o$ directly to the transparency values of the pixels contained in the image.

As defined above, the algorithm computes a separation of one (possibly disconnected) foreground object from the background. However, a straightforward extension of the algorithm also allows for multi-labeling, that is the separation of the image into several different objects and background. An example of such an approach is described in Sec. 6. Figure 2 shows a sample input for the algorithm and the corresponding confidence matrix.
3.1. Construction of color signatures

A color signature is a set of representative colors, not necessarily a subset of the input colors. A color signature is constructed by clustering a set of pixels into equally-sized clusters [20]. The centroids of the clusters are defined to be the representative colors.

Given a set of color pixels, all colors are regarded as points in a $d$-dimensional color space. This color space is subdivided recursively, starting with the whole space. In step $i$, the points in the current box $B$ of the subdivision are projected onto the axis $a$ along dimension $i \mod d$. The two extreme projections $p, q$ are determined, and if $\|p - q\|$ is larger than a given threshold $l_{i \mod d}$, $B$ is split into two with a plane orthogonal to $a$ at $\frac{p + q}{2}$. This is done until all boxes have at least one dimension that is smaller than the threshold for that dimension. As described in Sec. 8, the triple (0.64, 1.28, 2.56) for the box width in dimension $i$ was found to be a good set of threshold values using genetic algorithms.

In a second pass, all center points of the boxes resulting from pass one are taken and are used as input points for the same algorithm. To improve noise robustness, only the center points of such boxes $B$ are considered that contain at least $t$ points for a fixed threshold $t$. These points are representative points and therefore become part of the signature. A good value for $t$ is the number of specified pixels divided by 1000 (again, see Sec. 8). As already observed by [20], this clustering method produces a good distribution and a representative signature with few points.

For applications where speed is a major issue and quality a minor problem, such as high-resolution video segmentation with high frame rates, an alternative color segmentation algorithm can be used that is much simpler and almost an order of magnitude faster [7]. The dynamic splitting rule from the kd-trees is then exchanged by a fixed discretization of the color space. This can be realized as a simple three-dimensional array. This 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 entire signature.

The signatures resulting from clustering typically contain only a few hundred points or less, which makes the subsequent steps very fast. To compare different
clustering techniques, one can look at the clusters they create as shown in Fig. 3. The discretized CIELAB space yields a very regular signature compared to the kd-tree approach due to its array implementation which allows further geometric optimizations. However, this clustering gives slightly worse segmentation results (see Sec. 8).

A color signature is built for the set of pixels having confidence 0 and another one is built for the pixels of confidence 1. If the confidence matrix does not contain any pixels with confidence 1, the foreground signature is found by color signature subtraction which is defined as follows. Two color signatures $S_1$ and $S_2$ are subtracted into a resulting signature $R = S_1 \setminus S_2$ by comparing the representative colors contained in $S_1$ and $S_2$ using the Euclidean distance. For each element in $S_2$, the element in $S_1$ with minimum distance is marked. $R$ is a subset of $S_1$ that contains only those representative colors of $S_1$ that have not been marked. $S_2$ must not contain more elements than $S_1$. In order to build a foreground signature when only known background is given, the background signature is subtracted from the signature of the entire image (which has always the same or a higher cardinality because of the constant box dimension thresholds).

### 3.2. Classification of unknown pixels

The pixels with confidence value 0.5 are classified using nearest neighbor search. If the Euclidian distance of a pixel’s color is closer to an element of the foreground signature than to all elements of the background signature, it is classified as foreground, otherwise it is classified as background. If a color has equal minimal distances to both signatures, the pixel is considered foreground. The reason for this is
3.3. Post-processing

The pure foreground/background classification based on the distances to the color signatures will usually select some individual pixels in the background with a foreground color and vice versa, resulting in tiny holes in the foreground object. Again, the wrongly classified background pixels are eliminated by a standard “erode” filter operation while the tiny holes are filled by a standard “dilate” operation directly performed on the confidence matrix. A breadth-first search on the confidence matrix is performed to identify all spatially connected regions that were classified as foreground. Either the biggest region or all regions with an area greater than a threshold are considered the final foreground object(s). The user can specify a smoothness factor to define how much smoothing should be applied to the confidence matrix. More smoothing reduces small classification errors. Less smoothing is appropriate for high-frequency object boundaries, for clouds or drawings. The values of the confidence matrix are directly used as transparency values (also known as α values) for each corresponding pixel. Figure 4 shows a sample result before and after post-processing.

4. Segmentation of Still Images

For still image object extraction, the user specifies the known background and known foreground regions manually. In the following, the user-specified regions are called trimap. As explained above, the known foreground is optional, but it improves the robustness of the segmentation. To provide this information, the user makes several selections with the mouse. The outer region of the first selected area specifies the known background while the inner region defines a superset of the foreground,
i.e., the unknown region. Using additional selections, the user may specify one or more known foreground regions or additional background regions to refine the region of interest. Internally, the trimap is mapped into a confidence matrix.

Using this interaction style, SIOX has been integrated into the core of the open-source project GIMP (GNU Image Manipulation Program) [8]. Figure 5 shows the user interaction necessary to create the initial confidence matrix as implemented in GIMP version 2.3.9. A freehand selection tool is used to specify the region of interest (Figs. 5(a) and (b)). It contains all foreground objects to be extracted and as few background as possible. The pixels outside the region of interest form the known background while the inner region defines a superset of the foreground, i.e., the unknown region. The known background is visualized as dark area.

The user then uses a foreground brush to mark representative foreground regions (Fig. 5(c)). 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. The selection can be refined by either adding further foreground markings or by adding background markings using the background brush (Fig. 5(d)). Pressing the “Enter” key results in the creation of the final selection mask (Fig. 5(e)). The object can then be manipulated independently (Fig. 5(f)).

5. Sub-Pixel Refinement

In most cases, a pixel-accurate object extraction gives satisfying results. Sometimes, however, a single pixel contains parts of the foreground as well as parts of the background. The resulting color of the pixel is a mixture of the foreground and the background. For this reason, images containing highly structured textures, such as hair or fine tree branches, look sloppy if they are classified only with pixel resolution. Sub-pixel accuracy is also needed to remove spill colors that result from motion blur or image filters that smooth borders.

Fortunately, color signatures provide an adequate model, and a simple extension of the algorithm allows to cope with this issue. Figure 6 illustrates the idea. Let $c$ be a certain CIELAB color in the picture. Let $f$ be the closest representative color to $c$ in the foreground signature and $b$ the closest representative color to $c$ in the background signature. Let $c'$ be the orthogonal projection of $c$ onto the segment $fb$. The point $c'$ splits $fb$ into the two segments $fc'$ and $c'b$. Checking if the ratio $\frac{\|fc\|}{\|c'b\|}$ comes closer to 1 than a threshold $t$ allows to detect whether a color $c$ is likely to be a mixture between foreground and background. In other words, if the Euclidian distances between $c$ and $f$ and between $c$ and $b$ are very similar or equal, then $c$ is assumed to be a mixture between foreground and background. Of course, for sensible results, the angle spanned by $f, c, b$ must not be too small, or a more suitable pair of points $f, b$ has to be found.
(a) User loads an image and chooses the foreground extraction tool . . .
(b) selects region of interest . . .

(c) specifies representative foreground regions . . .
(d) optionally refines the result . . .

(e) and is provided with a tight selection.
(f) The object can now be handled independently.

Fig. 5. User interaction to provide initial confidence matrix in the image-editing program GIMP. SIOX has been integrated as a core feature in GIMP since version 2.3.3 (seagull image from [6]).
However, this method fails for colors that are inherently a mixture of many colors, for example white. Although their nearest neighbor clearly classifies them as part of the background or foreground, these colors are very often inherently detected as mixture colors because there are also many close representatives in the antagonist signature. Another question is how to set the threshold $t$, i.e., how to define “close enough to 1”. These two questions make it hard to implement a full-automatic solution that would allow for the detection of mixture colors. In practice, a semi-automatic solution was favored and was called Detail Refinement Brush (DRB). The Detail Refinement Brush is offered to the user as a simple interactive drawing tool. Using coarse strokes, the brush is used to refine regions where the results achieved by the automatic, pixel-accurate segmentation are not satisfying. With the user specifying the regions to search for mixed colors, the risk that a wrong detection destroys already approved segmentation results is lowered. In addition to the brush, the user is provided with a slider that enables the adjustment of the threshold $t$. The brush has two different modes: add and subtract. Add re-adds wrongly classified foreground. Subtract is used to remove spill colors at borders or from highly detailed textures. The brush affects the confidence $conf(p)$ of a pixel $p$ with color $c$ in the following way: ($f$ and $b$ being the closest colors in foreground and background signature, respectively):

$$\begin{align*}
conf(p) &= \begin{cases} 
1 - \min \left( \frac{\|c-f\|}{\|c-b\|}, 1 \right) & \text{in subtract mode} \\
\min \left( \frac{\|c-b\|}{\|c-f\|}, 1 \right) & \text{in add mode}
\end{cases}
\end{align*}$$

If $conf(p)$ is smaller or equal to the user-defined threshold, the pixel confidence value is directly mapped to the opacity value of the given pixel. Figures 7(a), (d), (g) show some sample results after automatic object extraction. The manual brush interaction illustrates Figs. 7(b), (e), (h). The refined results are shown in Figs. 7(c), (f), (i).
We also experimented with complete removal of the background tone from the pixel’s color but this turned out to be too aggressive. The perceptual result of mixing two or more colors is non-linear and an “un-mixing” would require a more accurate color model.

6. Extraction of Multiple Objects

The extraction of multiple objects (also often referred to as multi-labeling) of uniform color structure and size has already been described in Sec. 2: Instead of defining the biggest connected component as the final result in the post-processing step, one allows for multiple objects that have at least a certain size. In practice, this
can easily be implemented by providing the user with a checkbox that disables or enables multi-object extraction and a slider that allows to adjust the minimum allowed object size. In order to facilitate usage, the SIOX implementation in GIMP provides only a checkbox: If the extraction of several objects is enabled, all those connected foreground components are considered objects of interest that have at least one quarter of the size of the biggest connected component. Figure 8 shows some examples of the extraction of multiple objects with similar color structures.

Of course, it is also possible to extract multiple objects of different color structure using repeated extractions of single objects. Graph-based segmentation approaches, such as Grabcut (see Sec. 2), have to rely on repeated extractions in order to implement multi-labeling because they seek a minimal-cut. Using SIOX, the extraction of multiple objects of differing color structure in a single step only requires the creation of a color signature for each object. An example where the extraction of several objects in a single step is desirable is given in the next section.

7. Video Segmentation

For object extraction in videos, the confidence matrix can be either specified by the user or can be learned from motion statistics. If the matrix is specified by the user, the approach is similar to the one described in Sec. 4, with the exception that color signatures can be reused in consecutive frames. Since many colors are identical in consecutive frames, a hash table allows for very efficient classification of the non-background pixels in each frame. However, when the displayed scene changes too much, the segmentation has to be stopped and a new manual selection has to be performed. Fortunately, many heuristics exist for scene change detection.
Fig. 9. The user specifies known foreground and known background for the first frame in a scene (above). SIOX segments it and reuses the color signatures for automatic segmentation of subsequent frames (below).

We experimented with observing the hit/miss rate of the hash table for each frame, which results in a robust detection of most scene changes.

If background and foreground signatures have been build, the current Java reference implementation easily processes a 640 × 480 video at 25 frames per second. Figure 9 shows a sample object extraction in a video. The videos were extracted
using a manual specification of known background and foreground in the first frame. For fully automatic object extraction, any known method may be used that is able to provide at least a subset of the background and preferably also a subset of the foreground so that color signatures can be computed without manual interaction.

Of course, multi-object extraction as described in Sec. 6 is also possible in videos. The following experiment presents a practical video segmentation application where the extraction of several objects in a single step is desirable: object tracking in robotic soccer. Robocup [22] is a competition of autonomous robots playing soccer in a color-coded environment. Each class of objects seen by the robots is associated with a unique color. A robot’s vision relies on the classification to identify and discriminate various objects on the field which in turn is very important for its behavior and finally for the success of the entire soccer team. The canonical approach used by many Robocup participants is to perform a color calibration by either manually [9] or automatically [14] selecting representative regions of each object in several video frames and feed them into a classifier. A look-up table is then built in which each color is associated with a class [9] proposes to fill the color table by a computational intensive process that automatically identifies regions of the various objects by shape and marks their colors correspondingly. This is done in non-time-critical moments so that during the actual game, a simple look-up suffices to classify each object.

The perceived color, however, depends on several factors such as lighting conditions, camera settings, and shadows or reflections cast by surrounding objects. Filling the look-up table exclusively with marked colors only yields satisfactory results when several dozens of frames have been processed this way. However, the abstraction mechanism provided by color signatures allows for a satisfying classification even with a small initial region specification. Using either the user-selected regions or an automatic output of a geometric pre-classifier, a color signature is generated for each object class: goal, ball, field, robots, and everything else.

Figure 10 illustrates the difference in the results between the method described by [9] and SIOX. In the first frame, sample regions were assigned to their respective classes manually. The experiment indicates that SIOX is also useful for real-time tracking of multiple objects, at least in an environment where the colors are deliberately chosen to be easily distinguishable.

8. Evaluation

Unfortunately, showing that a certain image or video-processing method works is often reduced to publishing results achieved on a small pre-selected set of sample pictures. Even though this often suffices to demonstrate that a certain idea may be applicable for a special problem domain, this kind of “proof by example” does not guarantee that the image or video-processing approach yields satisfying results in general. On the other hand, proving mathematically that a certain image-processing method works is often impossible because both the problem and the output are
not mathematically well defined. In practice, there is almost no image or video-processing method that does not fail in certain special cases. This is especially a problem for object extraction methods. Because the processing of the human brain that enables vision is not yet understood researchers are forced to rely on heuristic approaches.

This section presents our experiments to provide evidence that SIOX gives satisfying results for a large set of images. However, it is impossible to provide an undeniable proof because object extraction is not yet mathematically definable.

8.1. Benchmarking and tuning of thresholds

In [2] a database of 50 images plus the corresponding ground truth to be used for benchmarking foreground extraction approaches is presented. The benchmark data set is available on the Internet [15] and also includes 20 images from the Berkeley Image Segmentation Benchmark Database [13]. The data set contains color images, a pixel-accurate ground truth, and user-specified trimaps. We chose comparison with this database because the solutions presented in [2] are commonly considered to be very successful methods for foreground extraction.

The trimaps, however, are not optimal inputs for the algorithm presented here because the specified 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 quite detailed shape information. The benchmark therefore does not represent the results a user could provide. For this reason, we created an additional set of trimaps better suited for testing the
Fig. 11. From left to right: The original image from the benchmark, the original lasso selection as provided by [15], a user-specified trimap used for benchmarking SIOX, and the ground-truth provided in the benchmark dataset. SIOX has been benchmarked with the user-specified trimaps because they are more realistic and the lasso trimaps are not suitable for SIOX because they do not select representative colors.

We asked several independent users 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 freehand 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. Figure 11 shows an example of an image in correspondence with both types of trimaps and the ground truth.

Unfortunately, it is difficult, maybe impossible, to create a generally valid error measure. Assuming such a perceptually accurate error measure for foreground extraction approaches would exist. The entire task would be reduced to minimizing this error function. Because we want to create comparable results, we stick to the error measurement defined in [2], which is defined as:

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

In low-contrast regions, a true boundary cannot be observed using pixel-accurate segmentation (see Sec. 5). This results in the ground truth database containing unclassified pixels. For comparability, these pixels are excluded from the number of misclassified pixels as in [2].

As discussed in Sec. 3.1, the results of the SIOX algorithm depend on the setting of the thresholds for the box width dimensions for each cluster in CIELAB space as well as the abstraction threshold for the removal of clusters that contain too few pixels. Therefore, the purpose of running SIOX on the benchmark data set is two-fold: Besides comparing the segmentation results with other algorithms, the benchmark also helps to find the optimal values for the four parameters. We tuned the parameters manually and used a genetic algorithm to verify the result. As already mentioned in Sec. 3.1, the triple (0.64, 1.28, 2.56) for the box width in
Fig. 12. Per-image error measurement from applying SIOX on the benchmark dataset provided by [15]. Please refer to the text for a detailed description.

1. dimension $i$ seems to be optimal for the cluster size. The best value for the abstraction threshold seems to be 0.01, i.e. the number of specified pixels divided by 1000. The results presented in the following were generated using these values for the parameters.

If only the background signature is given and the foreground signature has to be calculated by color signature subtraction, the overall error is 11.32%. The overall error when applying the lasso trimaps provided by the database is 8.75%. As already mentioned, the lasso selections are not optimal for the segmentation algorithm presented here. Figure 12 shows the result for the additional set of trimaps based on rectangular user selections. The overall error is 3.59% and the segmentations subjectively appear much better. This indicates that the robustness of the algorithm is significantly improved with the user providing a foreground sample. Using the alternative clustering method described in Sec. 3.1, the error is 4.21%.

The best-case average error rate on the database for the GrabCut underlying algorithm is reported as 7.9%[2].\textsuperscript{a} 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

\textsuperscript{a}At the time of writing of this article, a per-image error measurement has not been published.
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.

8.2. Testing assumptions

The benchmark offers the possibility to check some of the basic assumptions underlying SIOX presented in the preceding chapters. The following experiments have been conducted to provide evidence that the keystones of the theoretical derivation for the SIOX algorithm hold.

8.2.1. CIELAB vs RGB vs HSI vs YUV

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. The parameters were again tuned using genetic algorithms. Otherwise the algorithm remained completely unchanged. Although there is no guarantee that the genetic algorithm found the optimal constants in each case (and did not get stuck in a local minimum of the fitness function), looking at both the average error and also the worst-case error reveals a clear evidence. CIELAB proves to be better than all other color spaces. Although YUV comes close on average, CIELAB shows a significantly smaller worst-case error. 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.

8.2.2. Need for abstraction

Section 3.1 discusses that color signatures provide an important means of abstraction. Without this abstraction, noise and outliers make segmentation difficult. On the other hand, too much abstraction makes segmentation impossible. The right trade-off between abstraction and accuracy is tuned with the constants that have been found as described in the previous section. Table 2 shows the results of two benchmark experiments to provide evidence that the abstraction mechanism provided by color signatures does not only improve speed but also improves accuracy.

Table 1. Average and worst-case classification results for different color spaces. Details of the experiment are explained in the text.

<table>
<thead>
<tr>
<th>Color Space</th>
<th>Worst Image Error</th>
<th>Total Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIELAB</td>
<td>15.4%</td>
<td>3.6%</td>
</tr>
<tr>
<td>RGB</td>
<td>97.0%</td>
<td>11.8%</td>
</tr>
<tr>
<td>HSI</td>
<td>54.4%</td>
<td>5.2%</td>
</tr>
<tr>
<td>YUV</td>
<td>34.7%</td>
<td>4.74%</td>
</tr>
</tbody>
</table>
Table 2. Using color signatures as an abstraction mechanism does not only improve the speed of the segmentation, it also improves the results. The details of the experiment are explained in the text.

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Worst Image Error</th>
<th>Totalx Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm as proposed</td>
<td>15.4%</td>
<td>3.6%</td>
</tr>
<tr>
<td>No abstraction at all</td>
<td>40.2%</td>
<td>8.8%</td>
</tr>
<tr>
<td>No relevance threshold</td>
<td>44.3%</td>
<td>9.9%</td>
</tr>
</tbody>
</table>

Without the clustering performed to create the color signatures, the segmentation is not only several orders of magnitude slower because more comparisons have to be made, the result is also worse. If the unknown pixels are directly compared with each pixel of the background and foreground sample, the resulting classification error more than doubles to 8.8% (worst result: 40.2%).

As described in Sec. 3.1, an abstraction threshold removes clusters that represent only very few pixels in the picture. This is especially useful for removing a few wrongly specified known foreground or known background pixels. These appear frequently in human-generated trimaps. If the clustering is performed for creating the representative colors but clusters that contain only a few pixels are not removed, the classification error increases to 9.9% (worst result: 44.3%).

8.3. Other means of evaluation

The benchmark provides some evidence of the robustness of the SIOX algorithm, especially because the pictures were not selected by me. However, the results are only partly significant because the images in the data set did not contain any images with highly detailed textures where sub-pixel accuracy would be a requirement. Furthermore, these regions were excluded from the test. The benchmark did not test interactive refinement by the user and it did not contain any noisy or blurry images. Finally, videos where not part of the data set. SIOX has been implemented into the core of the open-source image manipulation program GIMP. In February 2006, an early GIMP implementation of SIOX was tested by the editorial staff of the very popular *c’́t magazine* in Germany [23]. The magazine compared SIOX to GrabCut and KnockOut. Although the tested implementation did not yet include the Detail Refinement Brush and multiple object extraction, the magazine positively mentioned SIOX’s capability to extract objects with highly complex shapes. The main concern of the magazine was that, compared to the two other commercial tools, the implementation had not yet been optimized for speed and the extraction of an object from a 5 megapixel image took too long to process. Since the publication of the article, many technical optimizations had been done by the open-source community and the extraction of the object from the 5 megapixel photograph shown in Fig. 13 takes about 6 seconds on a 3-GHz Pentium 4. Further refinement steps, which can partially reuse already calculated data, take about 2–3 seconds per interaction.
The inclusion of the algorithm into GIMP also resulted in a huge amount of user feedback, mainly in newsgroups, blogs, and mailing lists. This feedback allows to extract a few rules of thumb for as to which image properties increase the chance of an instantly perfect segmentation result:

- The better the foreground object is distinguishable from the background, the easier the segmentation.
- The better the foreground and background selections, the better the segmentation result. The user must make sure the entire object is inside the region of interest and the foreground samples are representative and do not contain any background. Ideally, the foreground samples should contain all the colors that the foreground object contains. Of course, finding such a set of samples is often cumbersome or even impossible. For animals and persons the following rules of thumb seem to hold:
  - Animals: The user should select at least the face, a large part of the body, and every special feature that a specific animal might have.
  - Human beings: The user should select a part of the face, the hair, and different parts of the clothes he or she wears. Skin is difficult to extract because of reflections, so as much skin as possible should be selected.
- High color variance: With the color spectrum being wide, the chance that background and foreground share the same colors is decreased. If the color spectrum of an image is very narrow, colors are shared by different objects.
- Good contrast: If the object boundaries are unclear, segmentation is often not accurate and manual refinement is required. The higher the number of mixture colors, the lower the chance for fast and accurate segmentation.
A summarizing rule of thumb could be deduced stating that if a picture was taken to show a particular object — like a portrait foto of a human being, an animal, or any other particular item — SIOX will most probably be able to extract it.

9. Limits of the Approach

The benchmark as well as the experiences of many users indicate that the presented algorithm performs well on a number of difficult pictures where it is even difficult to construct an accurate ground truth. The classification copes well with noise although the computation needs considerably more time for noisy input. Figure 14 shows the result of classifying a noisy image with SIOX and using a graph-cut-based algorithm. However, looking at the resulting pictures also discloses some weaknesses.

The segmentation depends heavily on the user provided trimap. The user has to select a region of interest that does contain the whole foreground object. Failing to do so will give unsatisfying results. Difficult images require a wise selection of representative foreground. Therefore, the user must have at least a little knowledge

![Original](image1)

(a) Original

![GrabCut](image2)

(b) GrabCut

![SIOX](image3)

(c) SIOX

Fig. 14. Extracting multiple objects from a noisy image (source: [6]). Graph-Cut-based approaches like [19] are usually only capable of extracting one object at a time and have difficulty segmenting noisy images.
of what could be representative. If two very similar objects exist on the picture, where 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 many similar colors 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 wanted one. When SIOX is integrated into an image manipulation application this problem can be avoided easily by combining SIOX with other operations, for example by cropping the image prior to using SIOX. Foreground objects that are connected with objects of the same color structure (for example, two people embracing each other) are almost impossible to extract using SIOX. 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 applies to shadows and reflections.

Pictures taken with non-standard illumination conditions are segmented poorly. Especially underwater scenes are awkward to segment, because of the natural color quantization underwater [18]. For these pictures, a different model would have to be used. The most critical drawback of the approach is color dependence. Although many photos are well separable by color, the algorithm cannot deal well 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 can be seen in Fig. 15. Gray-scale pictures or pictures that have already been color quantized give bad results (for example GIF images or videos encoded with a codec that performs color reduction). Although SIOX also works for drawings, the postprocessing steps blur their edges. Computer-created drawings with a few colors are better segmented by using Magic Wand.

Fig. 15. If color signatures overlap heavily, the result is bad segmentation. The original (taken from [6]) has very smooth transitions making it hard even for a human to find the exact boundaries (left). The automatic segmentation result has clearly visible dents and holes which have to be eliminated by user interaction (right).

In the case of underwater photography, this model would have to depend on the depth where the picture was taken.
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 CIELAB’s different observers and illumination models may improve segmentation of underwater scenes, space images, or pictures taken at night. We also experimented with the integration of color-distribution-based methods and with the SCIELAB space [27]. Automatically applying the DRB to detect spill-color regions on the boundary is a matter of further research, but in the end there will always be cases where a computer cannot distinguish between detail and noise without additional user interaction or information about the content.

10. Conclusion

This paper presents a runtime efficient adaptation of techniques originally used in image retrieval to solve foreground segmentation problems in video and still images. In video, the method enables scale and rotation invariant tracking of foreground objects.

By changing the way the confidence matrix is generated, the core of the algorithm can be used for a variety of applications. The generated color signatures can further be used to cope with highly detailed textures even with sub-pixel accuracy.

Fig. 16. The SIOX algorithm is currently being adopted for several open-source applications. This screenshot was provided by Brecht van Lommel and shows a SIOX implementation in Blender.
The presented approach can be applied to a variety of other problems where a foreground object should be tracked, extracted, and/or identified. SIOX has already been put to practical use in GIMP, an open-source image manipulation program. Since the release of an open-source reference implementation in Java [21], implementations of SIOX are currently also being integrated into Krita (part of KOffice) [11], Inkscape [10], and Blender [26]. Figures 16 and 17 show preliminary screenshots from Blender and Inkscape.

We hope that in the future more tools will take advantage of human-defined and refined semantic information so that users are freed from having to rely on low-level operations. It is our hope that the work described in this article will be seen as an example and have an impact on the future development of multimedia content creation and editing systems.

Acknowledgements

We want to thank Christian Zick for creating many test and demo material. Lars Knipping polished up some code in the SIOX reference implementation. We also want to thank the open source community for their support with ideas and code. Most noticeably, we want to thank Sven Neumann for his work on the integration and optimization of the SIOX implementation in GIMP.
References


