Chapter 10

Generalizing the Instructor Extraction

A variation of the instructor video segmentation approach presented in the previous chapter can also be applied to a variety of other problems where a foreground object has to be extracted in an image or video. This generalization of the instructor extraction approach solves many of the open problems described in the last chapter. I have released it as an open-source segmentation framework under the name *Simple Interactive Object Extraction (SIOX)* [48]. SIOX has been integrated into several open-source image and video manipulation applications. This chapter 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. Further information can be found in [Friedland et al., 2005b], [Friedland et al., 2005e], [Friedland et al., 2006a], and [Friedland et al., 2006b].

10.1 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 [Mortensen and Barrett, 1999] 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.

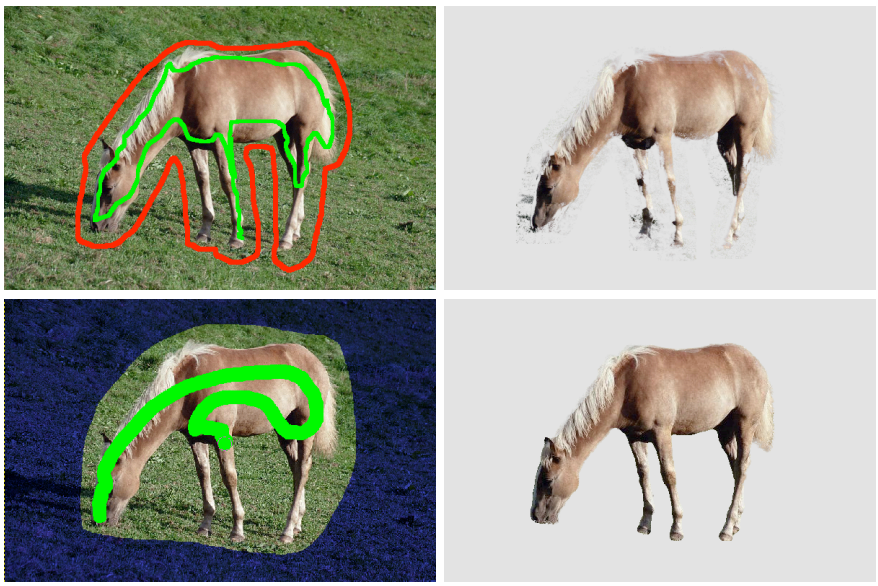


Figure 10.1: A sample comparison (horse image from [14]) between Corel's Knockout 2 and SIOX as implemented in GIMP (see Section 10.3). Upper row: Knockout 2 requires the user to specify the outer region (red) and an inner region (green). The tool then tries to classify the unknown pixels between the two strokes. Kockout'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 segmentation result is shown in the right picture.

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 [Chuang Y.-Y. and R., 2001] gets a shrinked 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 [Corel Corporation, 2002]. According to [Chuang Y.-Y. and R., 2001] 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 [Trinkwalder, 2006]. Figure 10.1 shows a comparison of interaction and results between SIOX and Knockout 2.

GrowCut [Vezhnevets and Konouchine, 2005] 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 [Rother et al., 2004] is a two step approach. The first step is automatic segmentation that relies on the work of *Graph Cut* [Boykov and Jolly, 2001]. 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 Figure 10.14).

GrabCut was extended by [Li et al., 2005] and [Wang et al., 2005] who present semi-automatic video cut-out tools that are far from being realtime capable. To accelerate the interactive refinement, [Wang et al., 2005] 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.

10.2 Algorithm Description

Like the instructor extraction approach in Chapter 9, SIOX separates foreground objects from background based on their color characteristics. Consequently, it requires color images and 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 [Adams et al., 1998]. 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.

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

Figure 10.2: The original image (source [Martin et al., 2001]), a user-provided rectangular selection (red: region of interest, green: known foreground), and the corresponding confidence matrix (black: known background; gray: unknown; white: known foreground). Figure 10.4 shows the segmentation result.

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.

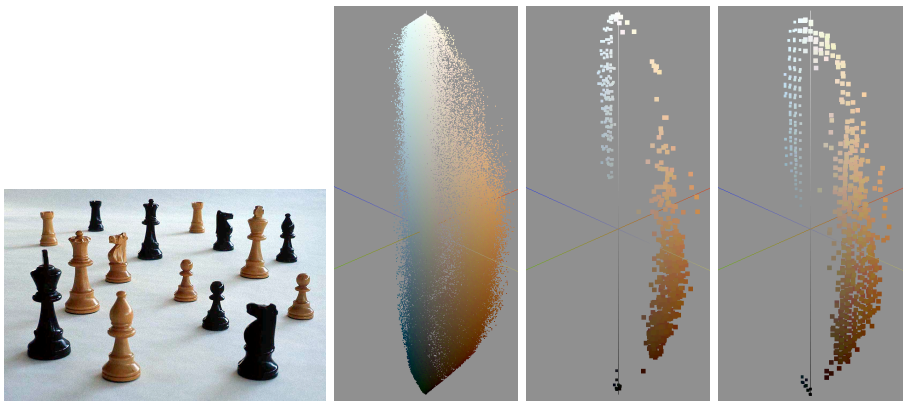
4. Find the connected components with high confidence in M_o which are large enough or correspond to user markings. Set the values of all other connected components to 0.

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 Section 10.5. Figure 10.2 shows a sample input for the algorithm and the corresponding confidence matrix.

10.2.1 Construction of Color Signatures

As already defined in the previous chapter, a color signature is a set of representative colors, not necessarily a subset of the input colors. A color signature



(a) Original image (b) All colors (c) KD-tree signature (d) Array signature

Figure 10.3: In (b), all colors from (a) are visualized as points in CIELAB space, (c) shows the color signature resulting from the tree clustering algorithm, (d) shows the signature from the faster array-based algorithm described in [Friedland et al., 2006b]. The array-based clustering is faster, but the segmentation result is worse.

is constructed by clustering a set of pixels into equally-sized clusters. The centroids of the clusters are defined to be the representative colors.

The algorithm for creating a color signature from a set of pixels has already been described in Section 9.5.3: 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 Section 10.7, 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 Section 10.7). As already observed by [Rubner et al., 2000], 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 [Friedland et al., 2006b]. 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

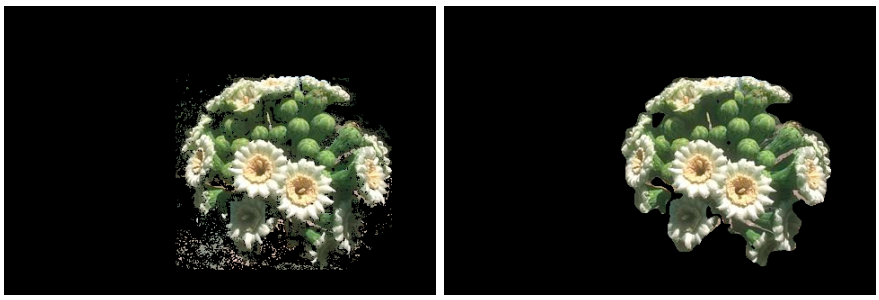


Figure 10.4: The result of the color classification before (left) and after post-processing (right). The original image and the user selection is shown in Figure 10.2.

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 Figure 10.3. 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 10.7).

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 $R = S_1 \backslash S_2$ 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).

10.2.2 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 a practical one: In image editing tools it is usually easier to erase wrongly classified foreground than to reconstruct wrongly classified background. However, in natural images, such as photographs, this case has a very low probability.

10.2.3 Post-processing

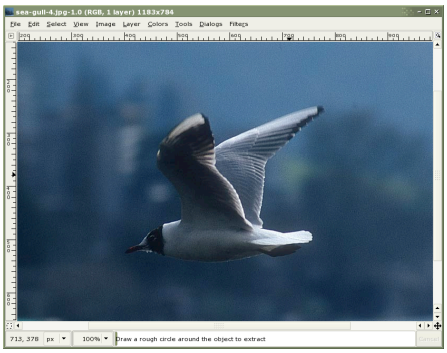
As already explained in Section 9.5.4, the pure foreground/background classification based on the distances to the color signatures will usually select some individual pixels in the background with a foreground color and vice versa, resulting in tiny holes in the foreground object. Again, the wrongly classified background pixels are eliminated by a standard "erode" filter operation while the tiny holes are filled by a standard "dilate" operation [Gonzalez and Woods, 2002] 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 10.4 shows a sample result before and after post-processing.

10.3 Segmentation of Still Images

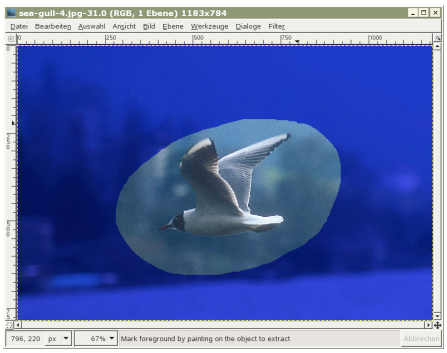
For still image object extraction, the user specifies the known background and known foreground regions manually. In the following, the user-specified regions are called *trimap*. As explained above, the known foreground is optional, but it improves the robustness of the segmentation. To provide this 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) [21]. Figure 10.5 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 (Figure 10.5 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.

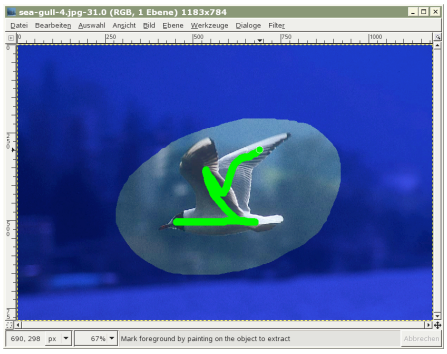
The user then uses a foreground brush to mark representative foreground regions (Figure 10.5 c). Internally, this input is mapped into a confidence matrix, where each element of the matrix corresponds to a pixel in the image. The values of the elements lie in the interval $[0, 1]$ where a value of 0 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 (Figure 10.5 d). Pressing the "Enter" key results in the creation of the final selection



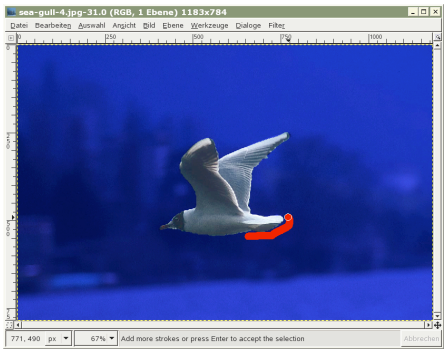
(a) User loads an image and chooses the foreground extraction tool...



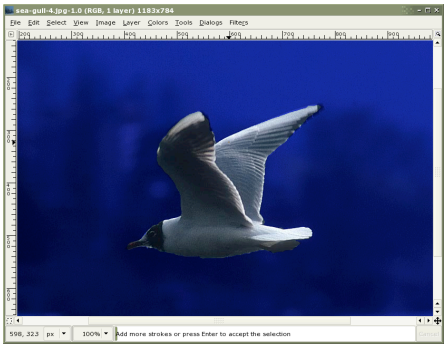
(b) selects region of interest...



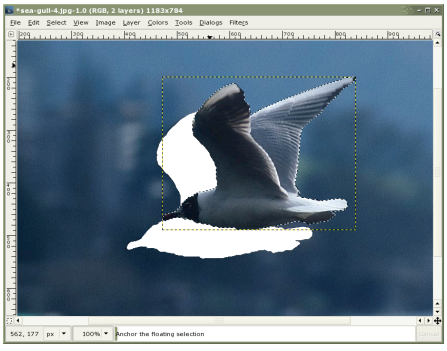
(c) specifies representative foreground regions...



(d) optionally refines the result...



(e) and is provided with a tight selection.



(f) The object can now be handled independently.

Figure 10.5: 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 [14]).

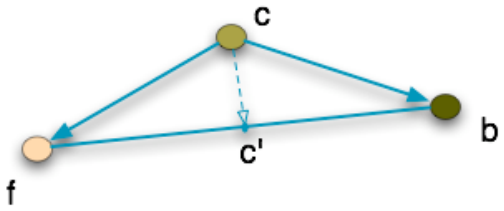


Figure 10.6: An illustration of the basic idea of the Detail Refinement Brush: Spill colors can be detected by the ratio of the distances to the closest representative foreground color f and the closest representative background color b.

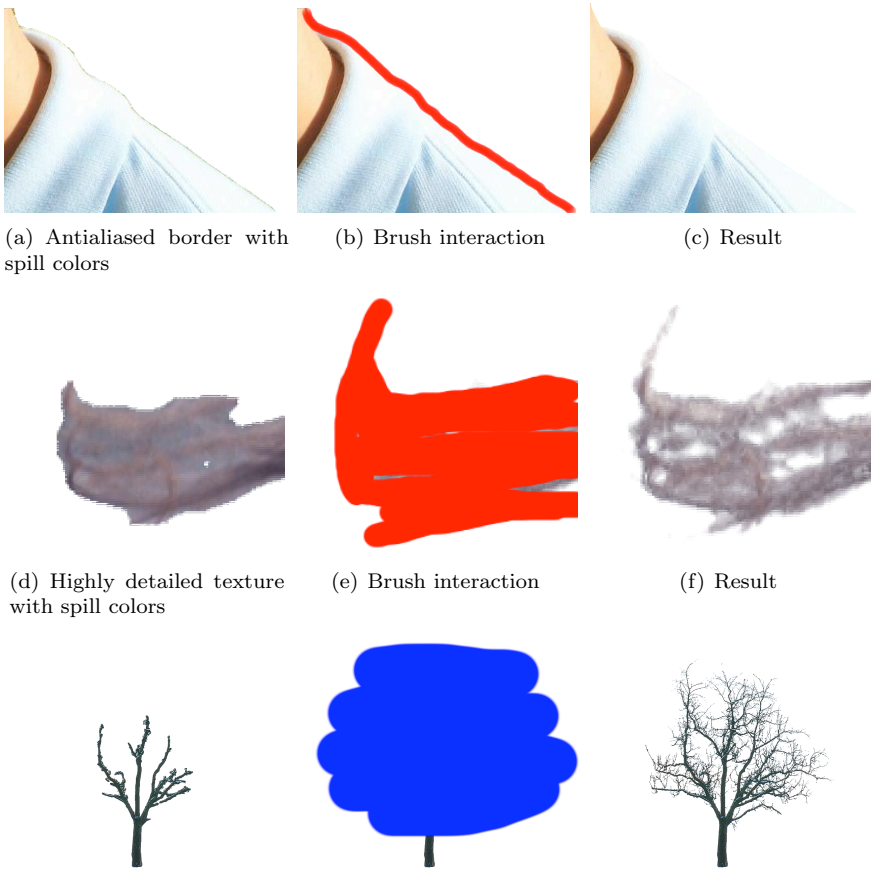
mask (Figure 10.5 e). The object can then be manipulated independently (Figure 10.5 f).

10.4 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 10.6 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 $\overline{fb}$. The point c' splits $\overline{fb}$ into the two segments $\overline{fc'}$ and $\overline{c'b}$. Checking if the ratio $\frac{\|\overline{fc'}\|}{\|\overline{c'b}\|}$ 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.

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



(a) Antialiased border with spill colors

(b) Brush interaction

(c) Result

(d) Highly detailed texture with spill colors

(e) Brush interaction

(f) Result

(g) Highly detailed texture where too many pixels were classified as background

(h) Brush interaction

(i) Result

Figure 10.7: Sample results for the Detail Refinement Brush. The image in the left column show the result after SIOX, the middle column shows the user interaction with the DRB, and the right column the results. Spill colors can be removed in subtract mode (red), pixel omissions in highly textured regions can be repaired using add mode (blue).

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 representiative color in foreground and background signature, respectively):

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

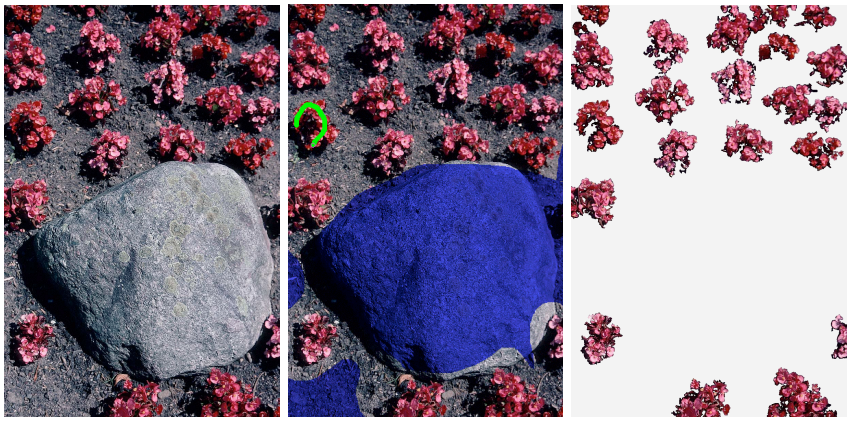


Figure 10.8: Simultaneously extracting multiple objects of the same color structure using SIOX can save time. From left to right: Original image (source: [14]), user selection (blue: known background, green: known foreground), and final result.

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. Figure 10.7 a, d, g show some sample results after automatic object extraction. The manual brush interaction illustrates Figure 10.7 b, e, h. The refined results are shown in Figure 10.7 c, f, i.

I 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.

10.5 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 Section 10.2: 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 10.8 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 Section 10.1), 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.

10.6 Video Segmentation

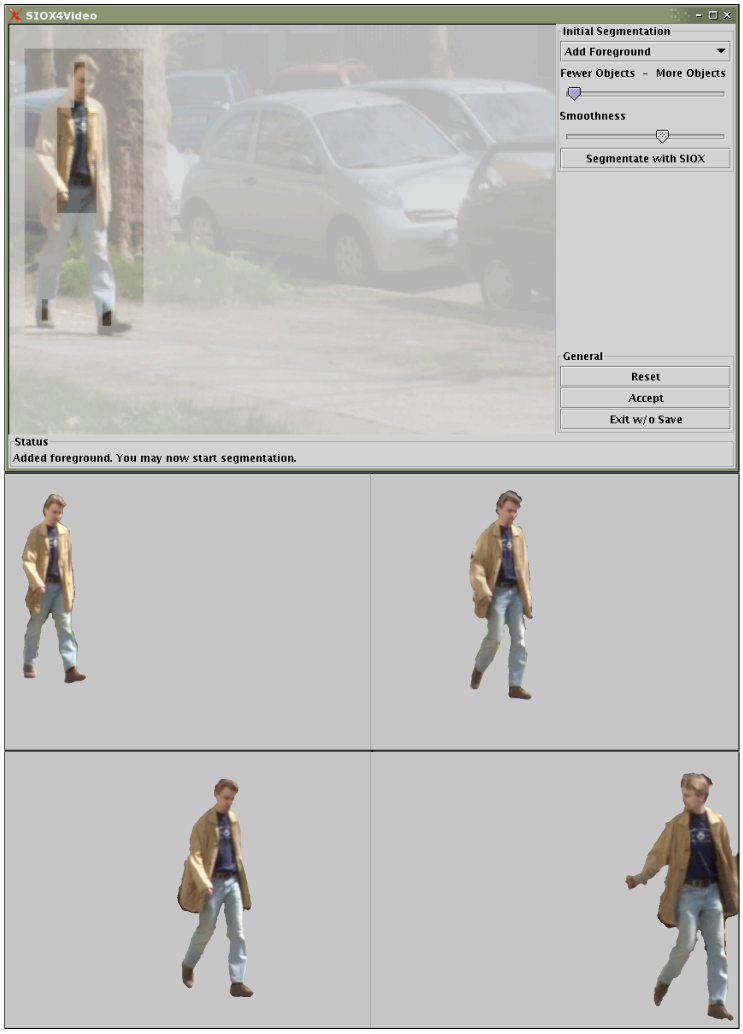


Figure 10.9: The user specifies known foreground and known background for the first frame in a scene (above). SIOX segments it and reuses the color signatures for automatic segmentation of subsequent frames (below).

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 Section 10.3, 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

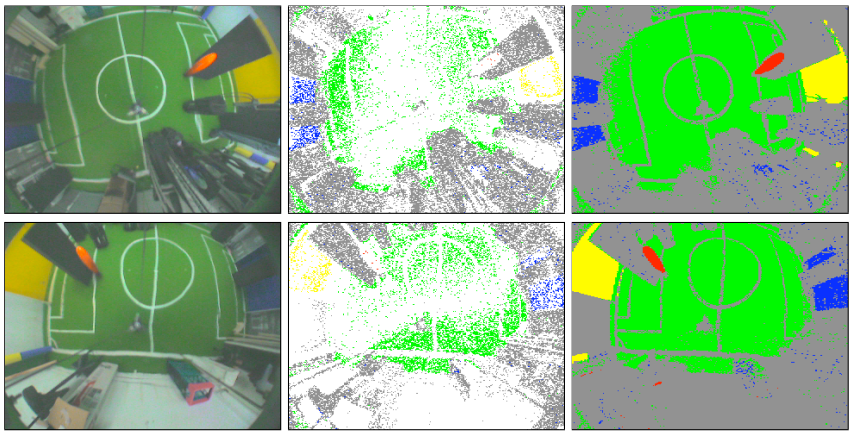


Figure 10.10: Original video images (left), color classification using the traditional strategy by [Gunnarsson et al., 2005] (middle), and result using SIOX (right). The gray regions define known background, the white regions are unclassified, all other colors mark a certain object. No post-processing step is applied because the result is already sufficient for the subsequent robot-control processing steps.

new manual selection has to be performed. Fortunately, many heuristics exist for scene change detection (see for example [Zabih et al., 1995, Aoki et al., 1996, Feng et al., 2005]). I 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 30 frames per second. Figure 10.9 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. A specialized approach used for instructor segmentation is described in Section 9.5.2. A hardware-based approach is presented in Chapter 11. Of course, the instructor can also be extracted using a manual specification of foreground and background samples. In fact, this provides a more robust and motion-independent lecturer extraction.

Of course, multi-object extraction as described in Section 10.5 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* [57] 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 [Gunnarsson et al., 2005] or automatically [Mayer et al., 2002] selecting representative regions of each

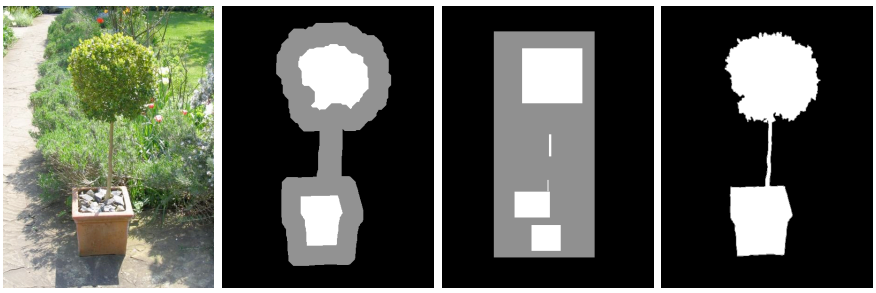


Figure 10.11: From left to right: The original image from the benchmark, the original lasso selection as provided by [35], a user-specified trimap used for benchmarking SIOX, and the ground-truth provided in the benchmark dataset. SIOX has been benchmarked with the user-specified trimaps because they are more realistic and the lasso trimaps are not suitable for SIOX because they do not select representative colors.

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. [Gunnarsson et al., 2005] 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 (which may change over time), 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 classification. 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_1, goal_2, ball, playing field, robots, and residual objects.

Figure 10.10 illustrates the difference in the results between the method described by [Gunnarsson et al., 2005] 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.

10.7 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. Until now, this dissertation has done likewise. 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

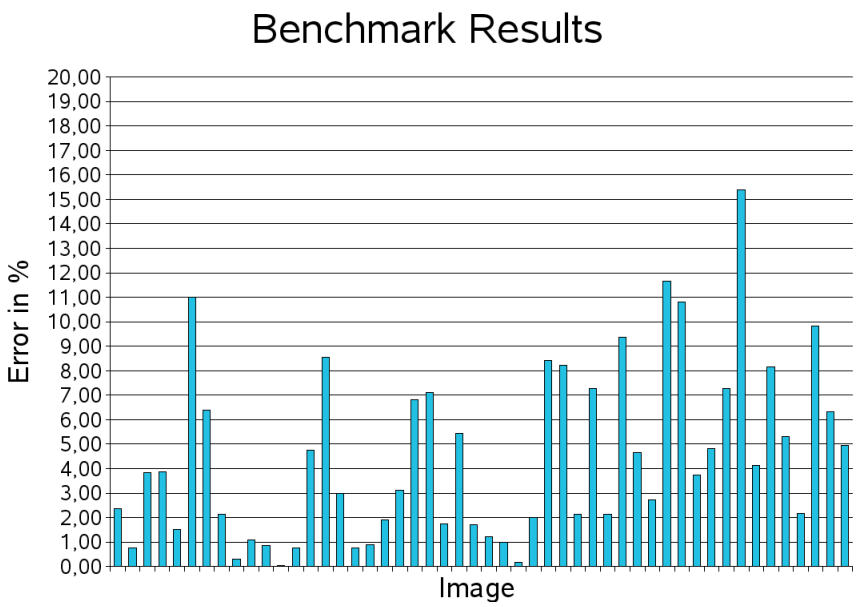


Figure 10.12: Per-image error measurement from applying SIOX on the benchmark dataset provided by [35]. Please refer to the text for a detailed description.

are not mathematically well defined. In practice, there is almost no image or video-processing method that does not fail in certain special cases. As already discussed in Section 9.2.2, 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 my experiments to provide evidence that SIOX gives satisfying results for a large set of images and is therefore a generalization of the instructor extraction problem. However, it is impossible to provide an undeniable proof because object extraction is not yet mathematically definable.

10.7.1 Benchmarking and Tuning of Thresholds

In [Blake et al., 2004] 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 [35] and also includes 20 images from the Berkeley Image Segmentation Benchmark Database [Martin et al., 2001]. The data set contains color images, a pixel-accurate ground truth, and user-specified trimaps. I chose comparison with this database because the solutions presented in [Blake et al., 2004] 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, I created an additional set of trimaps better suited

Color Space	Worst Image Error	Total Error
CIELAB	15.4 %	3.6 %
RGB	97.0 %	11.8 %
HSI	54.4 %	5.2 %
YUV	34.7 %	4.74 %

Table 10.1: Average and worst-case classification results for different color spaces. Details of the experiment are explained in the text.

for testing the approach. I 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 10.11 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 I want to create comparable results, I stick to the error measurement defined in [Blake et al., 2004], 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 Section 10.4). This results in the ground truth database containing unclassified pixels. For comparability, these pixels are excluded from the number of misclassified pixels as in [Blake et al., 2004].

As discussed in Sections 9.5.3 and 10.2.1, 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. I tuned the parameters manually and used a genetic algorithm [47] to verify the result. As already mentioned in Sections 9.5.3 and 10.2.1, the triple $(0.64, 1.28, 2.56)$ for the box width in 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 10.12 shows the result for the additional set of trimaps based on rectangular user selections[1]. The overall error is 3.59 %

[1] Throughout this dissertation, the benchmark images are always listed in the same order.

Abstraction	Worst Image Error	Total Error
Algorithm as proposed	15.4 %	3.6 %
No abstraction at all	40.2 %	8.8 %
No relevance threshold	44.3 %	9.9 %

Table 10.2: 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.

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. Appendix G presents the benchmark images along with their segmentation results. Using the alternative clustering method described in Section 10.2.1, the error is 4.21 %.

The best-case average error rate on the database for the GrabCut underlying algorithm is reported as 7.9 % [Blake et al., 2004].[2] 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.

10.7.2 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.

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 10.1 summarizes the average and worst-case results. Of course, a small worst-case error is very important for a generic image manipulation tool.

[2] At the time of writing of this dissertation, a per-image error measurement has not been published.

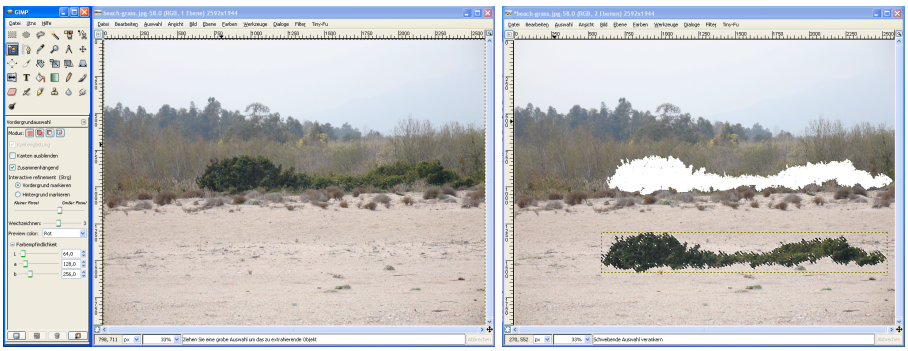


Figure 10.13: Cutting out an object with a fairly complex shape from this high-resolution image (2592 × 1944 pixels) takes about 6 seconds in GIMP v2.3.9 (Pentium 4 3 GHz, 2 GB RAM). Further refinement steps would take about 2-3 seconds per interaction.

Need for Abstraction

Section 9.5.3 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 10.2 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.

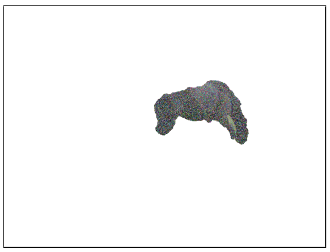
Without the clustering performed to create the color signatures, the segmentation is not only several orders of magnitude slower because more comparisons have to be made, the result is also worse. If the unknown pixels are directly compared with each pixel of the background and foreground sample, the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

As described in Section 9.5.3, 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 %).

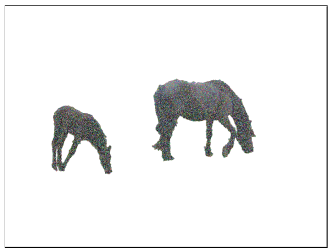
10.7.3 Other Means of Evaluation

The benchmark provides some evidence of the robustness of the SIOX algorithm, expecially 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 manip-

(a) Original



(b) GrabCut



(c) SIOX

Figure 10.14: Extracting multiple objects from a noisy image (source: [14]). Graph-Cut-based approaches like [Rother et al., 2004] are usually only capable of extracting one object at a time and have difficulty segmenting noisy images.

ulation program GIMP. In February 2006, an early GIMP implementation of SIOX was tested by the editorial staff of *c't magazine* [Trinkwalder, 2006]. 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 (see Appendix A for a list of particular names) and the extraction of the object from the 5 megapixel photograph shown in Figure 10.13 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.

Figure 10.15: If color signatures overlap heavily, the result is bad segmentation. The original (taken from [14]) 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).

- 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.

10.8 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

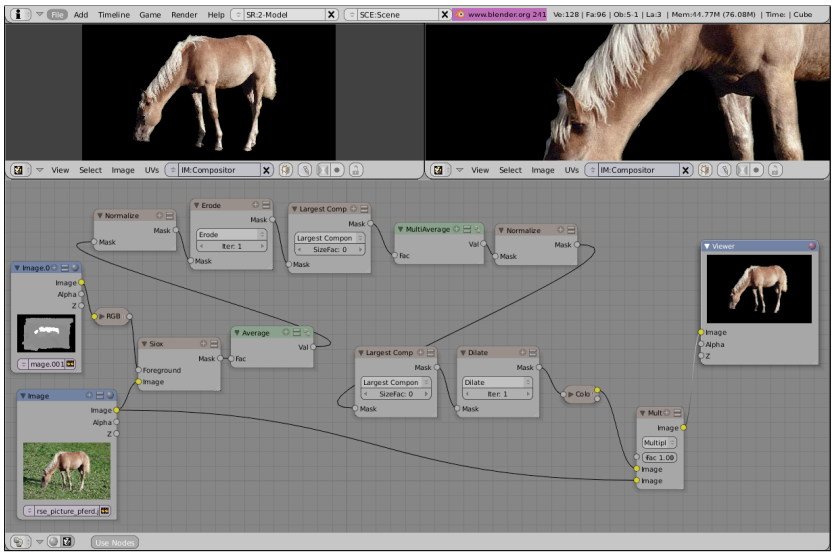


Figure 10.16: The SIOX algorithm is currently being adopted for several open-source applications. This screenshot was provided by Brecht van Lommel and shows a SIOX implementation in Blender.

input. Figure 10.14 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 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.

Still another problem is the use of the standard observer and the D65 reference white. Pictures taken with different illumination conditions are segmented poorly. Especially underwater scenes are awkward to segment, because of the natural color quantization underwater [Richardson, 2000]. For these pictures, a different model would have to be used[3]. As already explained in Chapter 9, the

[3]In the case of underwater photography, this model would have to depend on the depth

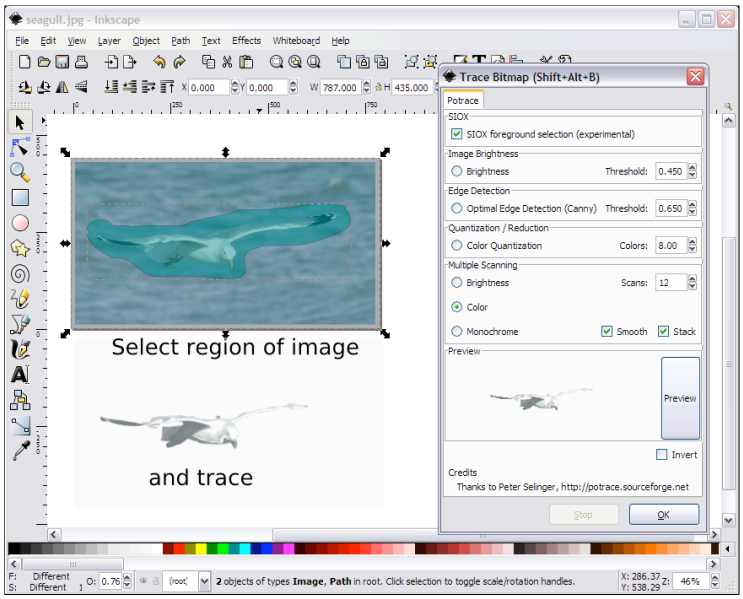


Figure 10.17: This image is a screenshot of a running Inkscape v0.44pre3 provided by Bob Jamison. Inkscape provides SIOX in combination with bitmap tracing. This allows users to vectorize only certain objects in an image.

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 Figure 10.15. 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.

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. I also experimented with the integration of color-distribution-based methods and with the SCIELAB space [Zhang et al., 1997]. Automatically applying the DRB to detected 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.

where the picture was taken.

10.9 Conclusion

This chapter illustrated that a generalization of E-Chalk's instructor extraction approach solves problems that currently cannot be addressed by state-of-the-art solutions, mostly relying on graph-cut algorithms. Using graph-cut-based approaches, recovering the boundary of an object can be impractical and sometimes even impossible. Highly detailed textures, such as hair or trees with branches, prevent object extraction from being reduced to finding a simple cut. Especially in the case of videos, there may be fuzzy edges, for example due to interlace effects, noise, or motion blur. 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. 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 into practical use in GIMP, an open-source image manipulation program.

Since the release of an open-source reference implementation in Java [48], implementations of SIOX are currently also being integrated into Krita (part of KOffice) [29], Inkscape [27], and Blender [71]. Figures 10.16 and 10.17 show preliminary screenshots from Blender and Inkscape. The next chapter will present yet another application of the algorithm, namely the use of SIOX in combination with a 3D camera.