Image Segmentation by Uniform Color Clustering Approach and Benchmark Results

Gerald Friedland, Kristian Jantz, Lars Knipping, Raul Rojas

Institut für Informatik, Freie Universität Berlin
`[fland|jantz|knipping|rojas]@inf.fu-berlin.de`

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

1 Introduction

The algorithm presented here has been developed for the extraction of an instructor teaching in front of an electronic chalkboard. In the E-Chalk system, used for recording and transmitting lectures over the Internet, the board content is transmitted as vector graphics, producing thus a high quality image, while the video of the extracted lecturer is sent separately [RFKT04,FKST04,FKT04]. It is then pasted onto the board image. It is possible to dim the lecturer from opaque to semitransparent, or even transparent. This makes it possible to transmit the mimic and gestures of the lecturer in relation to the board without using either too much bandwidth or having blurry artefacts around the board strokes produced by state-of-the-art video compression. Figure 1 shows an example of such a video.

Video segmentation requires speed more than accuracy. In still image segmentation, accuracy becomes the higher priority. This article shows that our algorithm for the segmentation of the instructor is general enough to be used also for foreground extraction in still images. Section 2 first introduces related work. Section 3 then explains the algorithm and how it is used. We benchmark our results and compare them to the *GrabCut* underlying algorithm presented in [RKB04a]. Section 5 shortly summarizes the video segmentation approach before, in Section 6, a conclusion is drawn and possible future work is presented.

2 Related Work

A nicely written summary and discussion of most of several foreground extraction methods can be found in [RKB04b]. In the following, some approaches for still image segmentation are only briefly described. The most popular tool to extract foreground is *Magic Wand* [Ado02]. *Magic Wand* starts with a small user-specified region. The region grows through connected pixels such that all selected pixels fall within some adjustable tolerance of the color statistics of the specified region. The methods works good for images that contain very few color, such as comic strips. For natural images, finding the correct tolerance threshold is often cumbersome. A satisfactory segmentation is seldom achieved. *Knockout* is a proprietary plugin for *Photoshop* [Cor02] that is, like the approach presented here, driven from a *trimap* (see Section 3.1). According to [CYYR01] the results are sometimes similar, sometimes of less quality than Bayes matting. Bayes Matting also gets a trimap as input and tries to compute alpha values over the unknown region. A disadvantage is that the user must specify a lot of shape information for the algorithm to work properly. *Grabcut* [RKB04b] relies on *Graph Cut* [BJ01] and [RKB04a]. The idea is to build a graph where each pixel is a node with outgoing edges to each of the 8 pixel's neighbors. The edges are weighted such that a max-flow/min-cut problem solves the segmentation. The user only provides the region of interest.

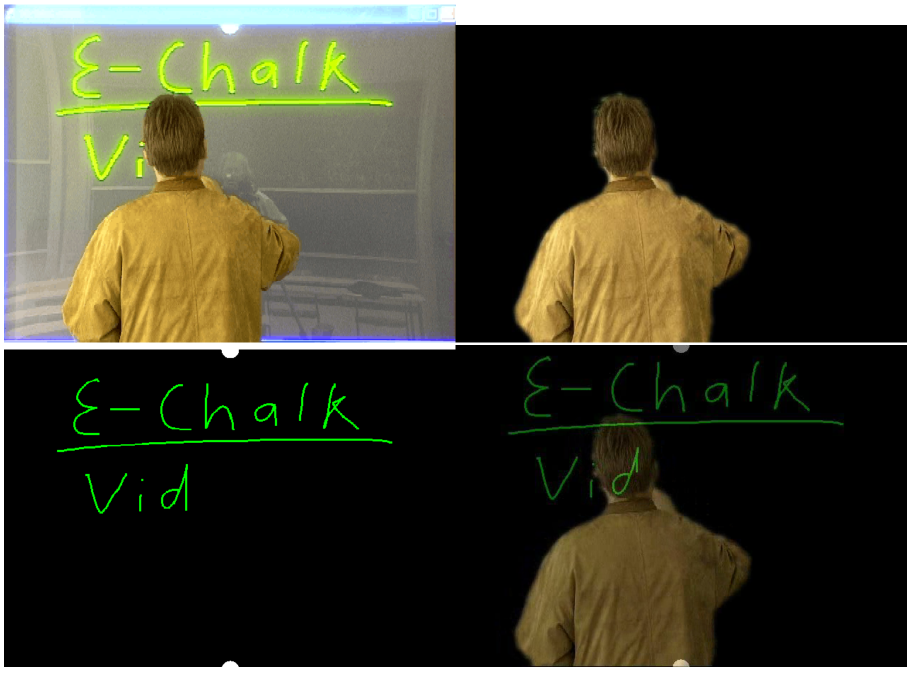


Fig. 1. A video of the lecturer is recorded (above left), the instructor is segmented (above right) and superimposed semi-transparently on the vector based board data (below left) and replayed together using MPEG 4 (below right).

Grabcut also includes manual post processing tools called background brush, foreground brush, and matting brush to smooth borders or re-edit classification errors manually. The disadvantage is its complicated data structure that requires high computational effort. At the time of writing this article, there was no program available to test the performance of the tool, nor are there sufficient information to compare any benchmark results. In this article only automatic classification without manual post-editing is considered. Benchmark comparison is done using the results in [RKB04a].

3 The Algorithm

The algorithm described here is to solve the task of foreground extraction in a given image. We define foreground to be a single, spatially connected object that is of interest to the user. The rest of the image is considered background. The user has to specify at least a superset of the foreground and the algorithm is to return an image that does not contain any background.

3.1 Input

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

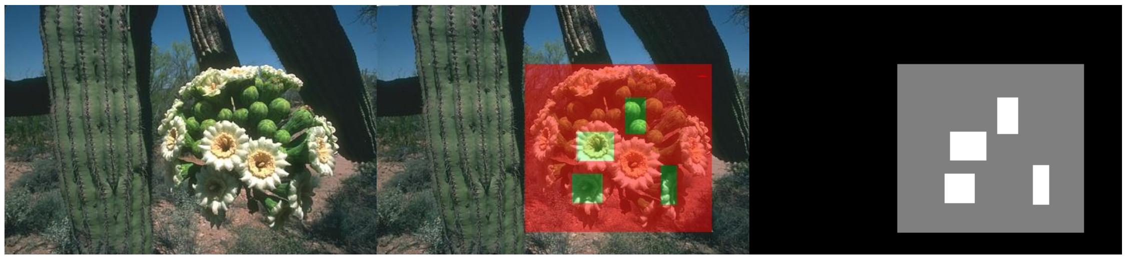


Fig. 2. The original image, a user provided rectangular selection (red: region of interest, green: known foreground), and the corresponding trimap (black: known background, gray: unknown, white: known foreground).

3.2 Conversion to CIELAB

The first step of the algorithm is to convert the entire image in CIELAB color space. This color space was explicitely designed as a perceptually uniform color space. It is based on the opponent-colors theory of color vision, which says that two colors cannot be both green and red at the same time, nor blue and yellow at the same time. As a result, single values can be used to describe the red/green and the yellow/blue attributes. When a color is expressed in CIELAB, L defines lightness, a denotes the red/green value and b the yellow/blue value. In the algorithm described here, the standard observer and the D65 reference white (see [WS82]) is used as an approximation to all possible color and lightning conditions that might appear in an image. CIELAB may still not be the optimal color space and the aforementioned assumption clearly leads to problems but in practice, the Euclidean distance between two colors in this space better approximates a perceptually uniform measure for color differences than in any other color space, like YUV, HSI, or RGB. Refer to Section 4.6 for a short discussion on the limits and issues of using this color space.

The biggest disadvantage of using CIELAB is the computational costs involved for the conversion from the image color space (usually RGB or YUV). To reduce the computational cost, we experimented with two approaches: Using a hash table to lookup already converted colors and using a lookup table filled with pre calculated values to approximate the cubic roots appearing in the conversion formula. These appeared to be the efficiency bottleneck. Section 4.5 looks at the trade-off between memory consumption and segmentation accuracy.

3.3 Color Segmentation

The segmentation method was adapted from [RTG00] who describes the use of color signatures and the Earth Mover's Distance for image retrieval. The idea behind our approach is to create a kind of color signature of the known background and use it to classify the pixels in the image into those belonging to the signature and those not belonging to it. The known background sample is clustered into equally sized clusters because in LAB space specifying a cluster size means assuming to specify a certain perceptual accuracy. To do this efficiently, we use the modified two-stage k-d tree [Ben75] algorithm described in [RTG00], where the splitting rule is to simply divide the given interval into two equally sized subintervals (instead of splitting the sample set at its median). In the first phase, approximate clusters are found by building up the tree and stopping when an interval at a node has become smaller than the allowed cluster diameter. At this point, clusters may be split in several nodes. In the second stage of the algorithm, nodes that belong to several clusters are recombined. To do this, another k-d tree clustering is performed using just the cluster centroids from the first phase. We use different cluster sizes of for the L, a, and b axes. The default is 0.66 for L, 1.25 for A and 2.50 for the B axis. The values can be set by the user according to the perceived color diversity in each of the axes. For efficiency reasons, clusters that contain less than 1 ‰ of the pixels of the entire background sample are removed.

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

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

3.4 Post Processing

The remaining task is to eliminate the background colors appearing in the foreground. By above definition, the foreground must be a unique, spatially connected region. In addition to several smoothing and an erosion/dilation step, a breadth-first-search on the confidence matrix is performed to identify all spatially connected regions that were classified as foreground. We assume that the biggest region is the one if user interest and eliminate all other regions[1]. The user can specify a smoothness factor to specify 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 example hair or clouds. The values of the confidence matrix are directly used as transparency factors (also known as α-values) for each corresponding pixel. The default value for the smoothing factor is 6. Figure 3 shows the result for the running example image directly after color segmentation and after post-processing.

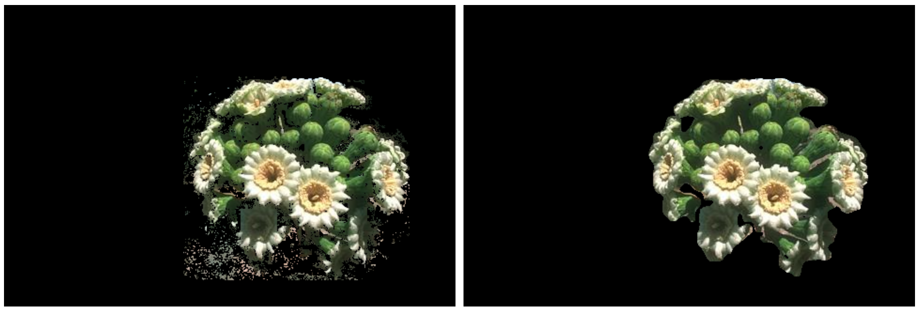


Fig. 3. The result of the color classification (left) and after post processing (right).

4 Benchmarking Results

4.1 Data Set

[RKB04a] presents a database of 50 images plus the corresponding ground truth that may be used to benchmark foreground extraction approaches. The benchmark dataset is available on the Internet [Mic04] and includes also 20 images from the Berkeley Image Segmentation Benchmark Database [MFTM01]. In addition to the ground truth and images the database also contains user specified trimaps. These trimaps, however, are not optimal inputs for the algorithm presented

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

here because their known foreground is not always a representative color sample of the entire foreground. Furthermore, creating such a trimap for the user would be too cumbersome, as it already contains a lot of shape information. The benchmark would then not represent the results a user could obtain. For this reason, the authors created an additional set of trimaps better suited to test the approach. The authors asked a non-involved user to draw appropriate rectangles for the region of interest and known foreground in each of the images. These trimaps may still be suboptimal but it is assumed here that they represent the typical input of a user. Using a rough free hand selection instead of a rectangular area, for example, would improve the segmentation result of those images where the smallest possible rectangle already covers almost the entire picture. For the benchmark, the default values for the smoothness factor and the cluster granularity were used. Figure 4 shows an example of an image with both types of trimaps and the ground truth.

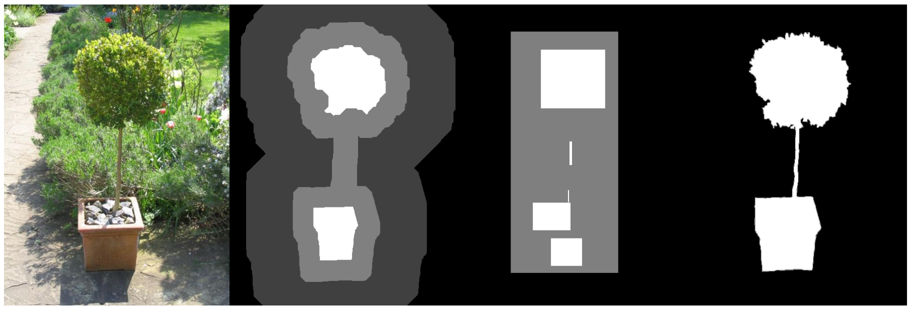


Fig. 4. From left to right: The original image, the lasso selection, the trimap by a user, and the ground truth

4.2 Error Measure

Given a perceptual accurate error measurement for foreground extraction approaches reduces the entire task to minimize the error function. Unfortunately it is difficult, maybe impossible, to create a general error measure. Because we want to create comparable results, we stick to the error measurement defined in [RKB04a][2]. The segmentation error rate is defined as:

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

In low contrast regions a true boundary is just not observed. This results in the ground truth database also containing unclassified pixels. For comparibility these pixels are excluded from the number of misclassified pixels as in [RKB04a].

4.3 Results

Figure 5 presents the error rates when applying trimaps provided by the database (lasso selection style). The average error is 9.1 %.[3] As already mentioned, the lasso selections are not optimal for the segmentation algorithm presented here. Figure 6 shows the result for the additional set of trimaps based on rectangular user selections. The detailed results are shown in Table A.1. The average error is 4.3 %. The best case average error rate on the database for the *GrabCut* underlying algorithm is reported as 7.9 %[RKB04a].[4] Using another trimap for classification results in another

[2] The authors chose comparsion with this article because the solutions presented there are commonly considered to be very succesful methods for foreground extraction.

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

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

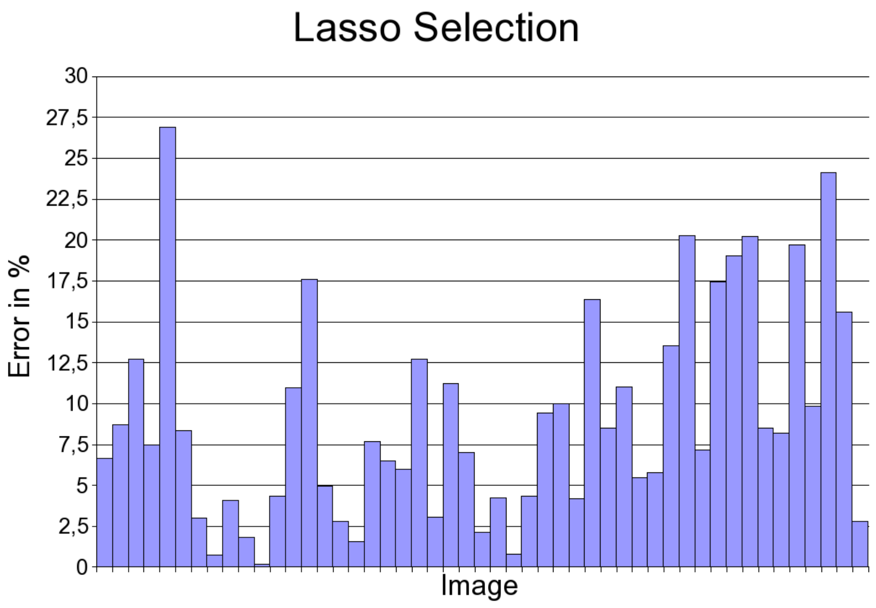


Fig. 5. Error rates of segmenting the benchmark images using the lasso trimaps.

total number of pixels to classify. One could object that a higher total number of pixels to classify may contain more pixels that are easier to classify and thus may beautify the error rate, because the critical pixels are not focused. 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 equal chance to be misclassified no matter in what region of the image it is located. Having more pixels to classify is therefore an even harder test. The segmentations subjectively appears much better when the additional trimap is used.

4.4 CIELAB vs. YUV vs. HSI vs. RGB

In order to test the impact of using CIELAB as underlying color space, the algorithm was also applied to the benchmark images using YUV, HSI, and RGB. Otherwise the algorithm remained completely unchanged. CIELAB proves to be better than all other color spaces. Although YUV comes close in average, CIELAB shows a significantly smaller worst-case error. Figure 7 shows the detailed results and Table 1 summarizes the average and worst-case results.

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

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

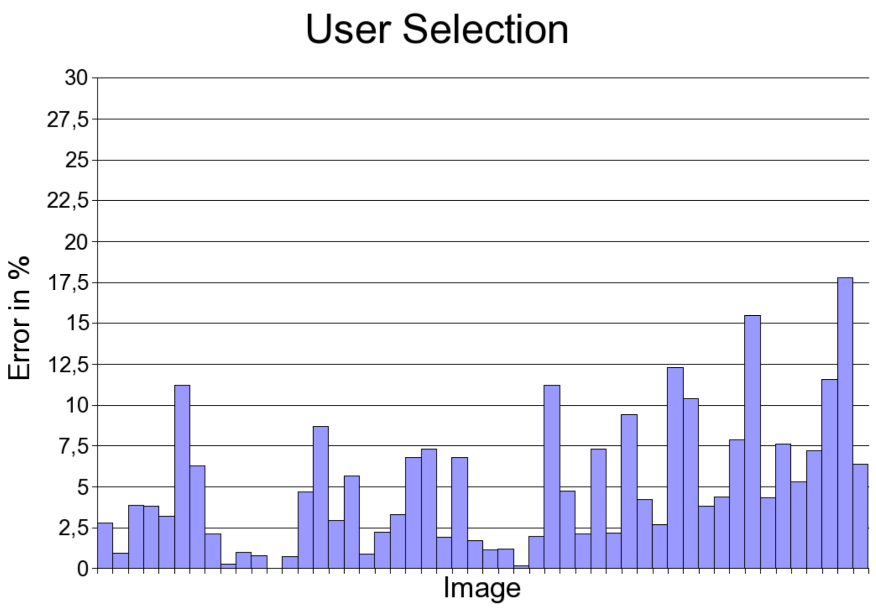


Fig. 6. Error rates of segmenting the benchmark images using the rectangular selections trimap. See Appendix for details.

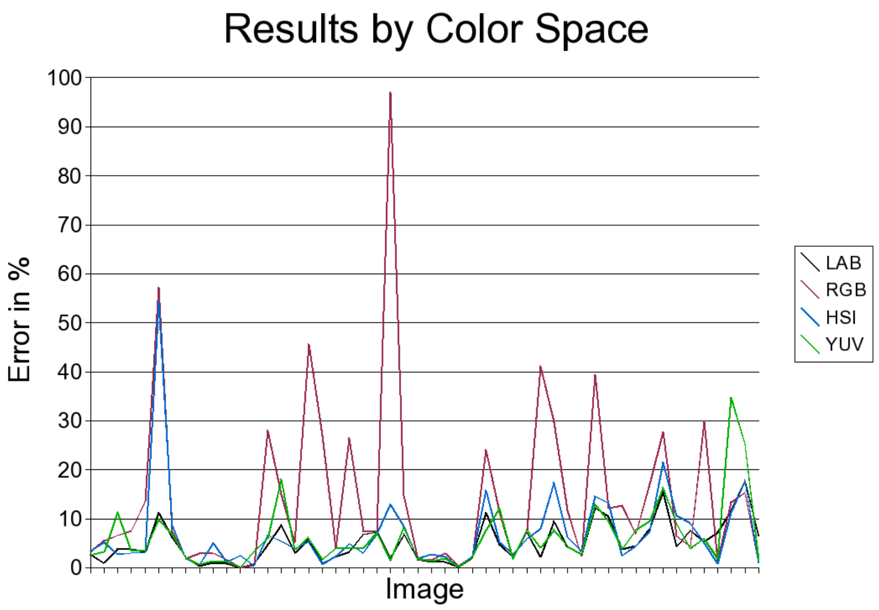


Fig. 7. Error rates of segmenting the benchmark images in different color spaces.

4.5 Exact CIELAB vs CIELAB approximation

As described in Section 3.2 an approximation of CIELAB was used to improve speed. Table 2 shows the classification error for different approximation granularities. The granularity defines the number of lookup values within a domain interval of size 1 for the cubic root table. The table's domain is $[0, 100]$, i.e. the number of entries is $100 \times$granularity. The approximation error was measured as root mean square error over one million random pixels against the average benchmark error in percent. The measured speed up gained for the conversion is about a factor between 10 and 30 depending on the machine[5]. For the purpose of error reduction versus table size a non-linear distribution of interpolation points would yield better results but then the lookup itself would be complicated and thus slower.

Granularity	RMS Error	Benchmark Error
10	35.26 %	35.3 %
100	3.24 %	6.2 %
1000	0.32 %	5.1 %
10000	0.03 %	4.6 %

Table 2. CIELAB conversion approximation accuracy versus classification error.

4.6 Strengths and Weaknesses

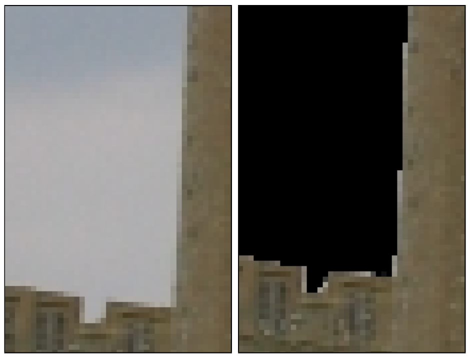


Fig. 8. An example of the accuracy of the segmentation in a middle-contrast region (left: original image, right: segmented image).

The benchmark shows that the presented algorithm performs well on a number of difficult pictures where it is even difficult to construct an accurate ground truth. If the contrast is good, the segmented border is accurate to a pixel, see Figure 8 for an example. The classification copes well with noise although the computation needs considerably more time for noisy input. Figure 9 shows the result of classifying a noise image. However, looking at the resulting pictures also discloses some weaknesses. The segmentation depends heavily on the user provided trimap. The user must select a region of interest that does contain the whole foreground object. Failing to do so will give unusable 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. It is not possible to extract multiple objects at once. For example, extracting multiple clouds from a sky

[5] Tested on a few Windows and Linux PCs with JRE 1.4.

requires several steps[6]. If two very similar objects exist on the picture, where one of them is to be considered foreground, the segmentation mostly gives bad results. The reason is, that most of the colors of the foreground are then considered background because they exist on the second object. The only workaround is to include both objects in the region of interest and to provide good foreground samples. Still, this method may fail when the unwanted similar object is bigger than the wanted one. Foreground objects that are connected together with objects of the same color structure (for example, two people embracing each other) are almost impossible to segmentate using the approach. Most of the misclassified pixels in the benchmark result from objects that are close to the foreground object, both in color structure and in location. The same reason counts for shadows and reflections. Still another problem is the use of the standard observer and the D65 reference white. Pictures photographed with different illumination conditions are segmentated poorly. Especially underwater scenes are awkward to segmentate, because of the natural color quantization underwater [Ric00]. For these pictures, a different model would have to be used[7].

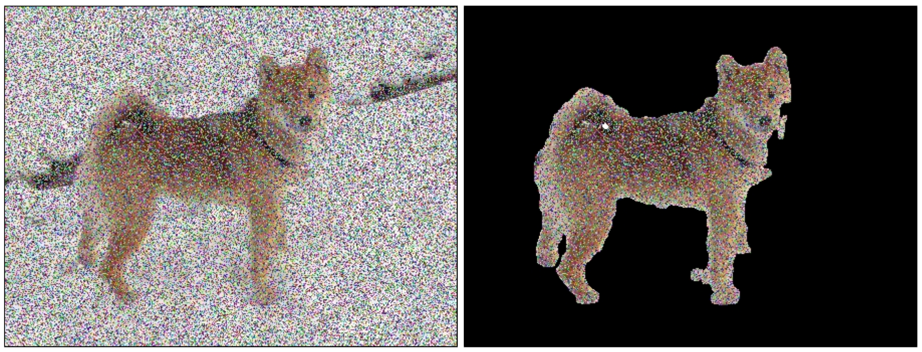


Fig. 9. A fairly high signal-to-noise ratio has only little effect on the segmentation (left: original image, right: segmented dog).

5 Video Segmentation

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

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

To distinguish noise from real movements, the following simple but general model is used. Given two measurements m_1 and m_2 of the same object with each measurement having a maximum deviation e of the real world due to noise or other factors, it is clear that the maximum possible deviation between m_1 and m_2 is $2e$. Given several consecutive frames, we estimate e to find out which pixels changed due to noise and which pixels changed due to real movement. To achieve this, the color changes of each pixel over a time period $h_{(x,y)}$ (where x and y specify pixel coordinates) is recorded. It is assumed that during this interval, the minimal change should be one that is caused by noise. The frame is then divided into 16 sub-frames and the changes in each sub-frame

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

[7] In the case of underwater photography, this model would have to depend on the depth where the picture was taken

are accumulated. Under the assumption, that at least one of these sub-frames was not touched by any foreground object, $2e$ is estimated to be the average variation of the sub-frame with the minimal sum. Then those pixels of the current frame are joined with the background sample that during this history period $h_{(x,y)}$ did not change more than our estimated $2e$. The history period $h_{(x,y)}$ is initialized with one second and is continously increased for pixels that are seldom classified as background, to avoid that a still-standing instructor is added to the background buffer. Figure 10 shows some examples of reconstructed backgrounds. It normally takes several seconds, until enough pixels could be collected to form a representative subset of the background. The background sample buffer is organized as an ageing FIFO queue.

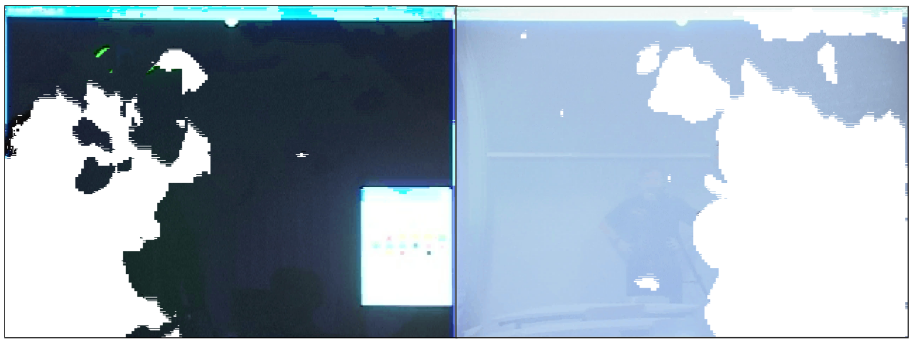


Fig. 10. For extracting foreground in a video, the trimap has to be reconstructed from motion statistics. The images show known background reconstructed over several frames. The white regions constitute the unknown region.

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

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

The performance of the algorithm depends on the complexity of the background and on how often it has to be updated. The algorithm was applied to segmentate an instructor standing in front of an electronic chalk board [RFKT04,FKST04]. Using these segmentation videos, the current Java-based prototype implementation processes a 640×480 video at 6 frames per second. This includes a preview window and a motion JPEG compression. A 320×240 video can be processed at 14 frames per second on a standard 3 GHz PC. This rate can be dramatically increased, by utilizing the SIMD multimedia instruction sets of modern CPUs.

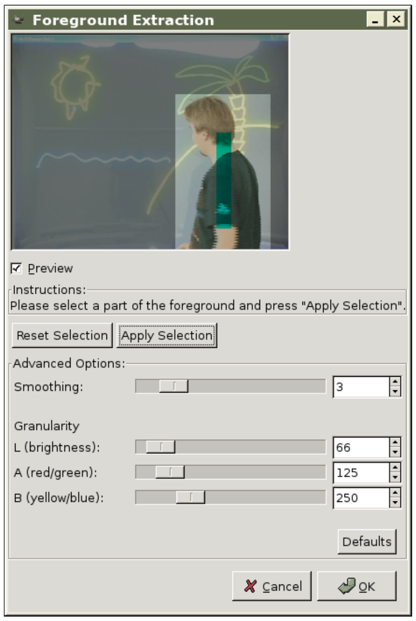
As the algorithm focuses on the background it provides rotation and scaling invariant tracking of any biggest moving object.

6 Conclusion and Future Work

The article presented a color classification algorithm that can be used for foreground extraction in images as well as in videos. The advantage of the algorithm is that its central data structure is efficient and not spatially bounded to a certain picture, like a graph spanned between pixels. Once build-up, the structure can be reused for subsequential frames in a video. Benchmark results

show, that an implicit use of the spatial information provided in the image using region growing suffices to compete with approaches that use this information explicitely by spanning graphs over pixels. Demonstration videos and images can be seen at:
`http://kazan.inf.fu-berlin.de/echalk/Segmentation/`

A usable implementation of the algorithm will be made available as a plugin for GIMP [GIM05], see Figure 11 for a screen shot of the current prototype. Future enhancements may include an automatic finding of the cluster sizes according to the color distribution of the image and a further improvement of the classification speed. Different observers and illumination models may improve segmentation of underwater scenes, space images, or pictures taken at night. The authors are also experimenting with the integration of color distribution based methods and with using the SCIELAB space [ZEW97].



☑ Preview

Instructions:
Please select a part of the foreground and press "Apply Selection".

Advanced Options:
Smoothing: 3

Granularity
L (brightness): 66
A (red/green): 125
B (yellow/blue): 250

Fig. 11. A screenshot of the Gimp plugin that is currently under development.

Credits

The following people are currently contributing to the segmentation approach:

Gerald Friedland has conceived the algorithm and implemented both the Java prototype of the video version and a native port in ANSI C.
Kristian Jantz has helped developing the algorithm, conducted the benchmark, and implemented most of the Gimp plugin.
Lars Knipping has created the optimized CIELAB conversion methods in Java and a small test framework for them.
Raúl Rojas is head of the E-Chalk project and inspired the development of the instructor segmentation algorithm.

References

Ado02. Adobe Systems, Inc. Adobe Photoshop User Guide, 2002.

Ben75. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

BJ01. Yuri Boykov and Marie-Pierre Jolly. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In *Proceedings of the International Conference on Computer Vision*, pages 105–112, Vancouver, Canada, July 2001.

Cor02. Corel Corproation. Knockout User Guide, 2002.

CYYR01. Salesin D. Chuang Y.-Y., Curless B. and Szelinski R. A bayesian approach to digital matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

FKST04. Gerald Friedland, Lars Knipping, Joachim Schulte, and Ernesto Tapia. E-Chalk: A Lecture Recording System using the Chalkboard Metaphor. *International Journal of Interactive Technology and Smart Education*, 1(1), February 2004.

FKT04. Gerald Friedland, Lars Knipping, and Ernesto Tapia. Web Based Lectures Produced by AI Supported Classroom Teaching. *International Journal of Artificial Intelligence Tools*, 13(2), 2004.

GIM05. GIMP Team. The GNU Image Manipulation Program. http://www.gimp.org, 2005.

MFTM01. D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

Mic04. Microsoft Research. Microsoft Foreground Extraction Benchmark Dataset. http://www.research.microsoft.com/vision/cambridge/segmentation/, 2004.

RFKT04. Raúl Rojas, Gerald Friedland, Lars Knipping, and Ernesto Tapia. Teaching With an Intelligent Electronic Chalkboard. In *Proceedings of ACM Multimedia 2004, Workshop on Effective Telepresence*, pages 16–23, New York, New York, USA, October 2004.

Ric00. Drew Richardson. *Adventures in Diving Manual.* International PADI, Inc, Rancho Santa Margarita, CA, 2000.

RKB04a. C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *Proc. ACM Siggraph*, 2004.

RKB04b. Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts. In *Proceedings of ACM Siggraph Conference*, August 2004.

RTG00. Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

WS82. G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* John Wiley and Sons, New York, NY, 1982.

ZEW97. Xuemei Zhang, Joyce E.Farrell, and Brian A. Wandell. Applications of a Spatial Extension to CIELAB. In *SPIE Electronic Imaging 97*, 1997.

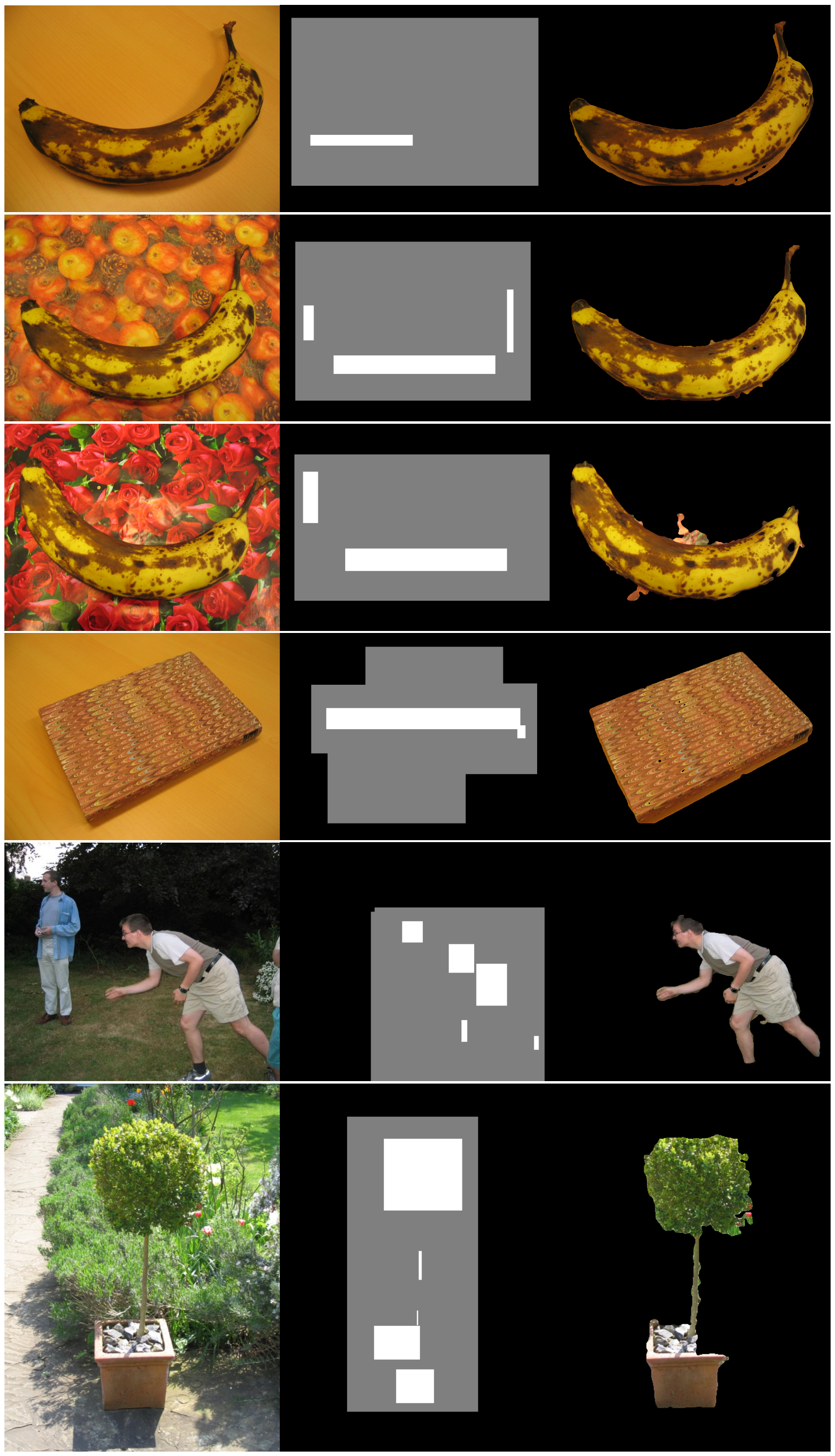
A Detailed Benchmark Results

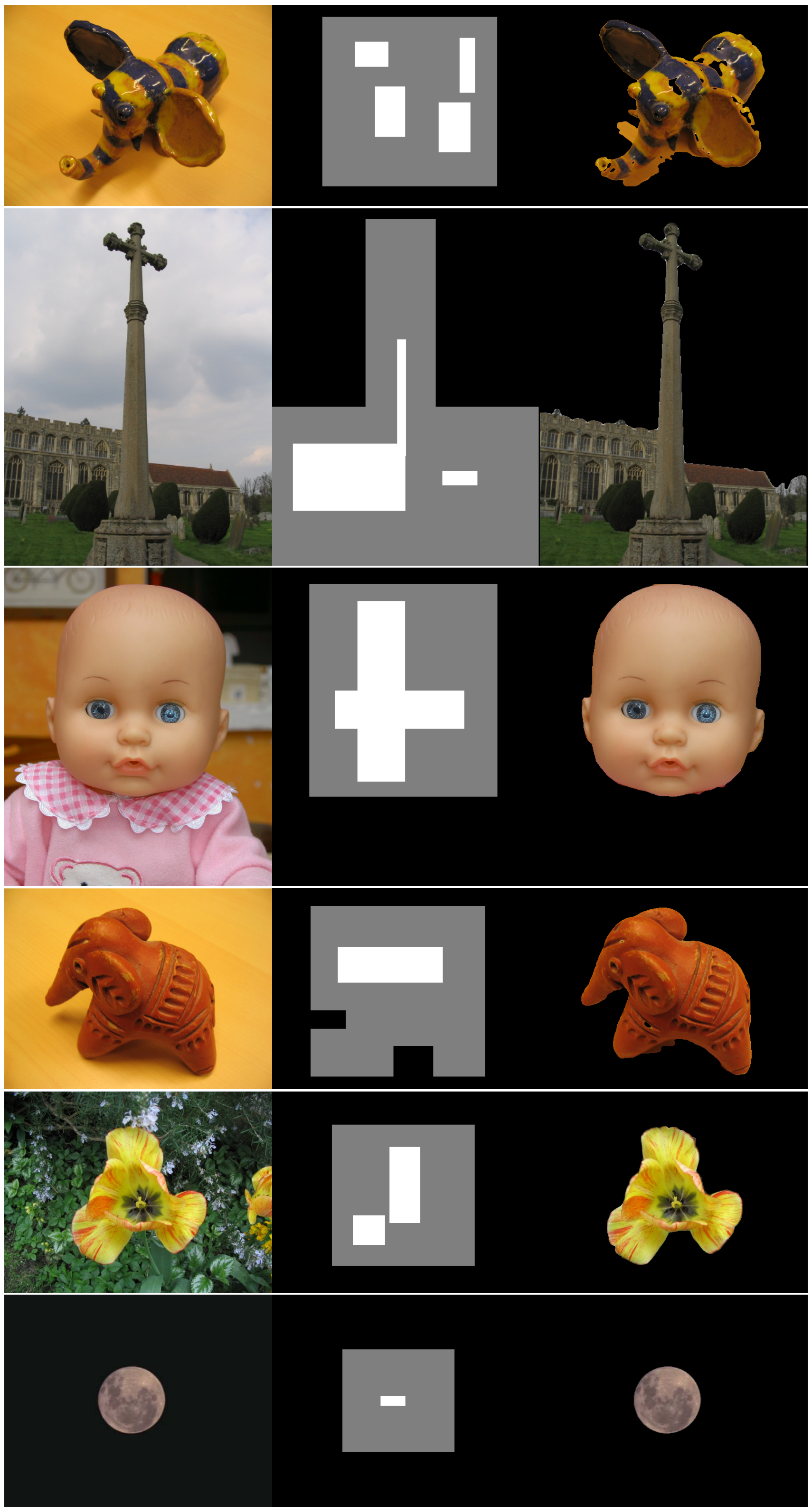
A.1 Numerical results

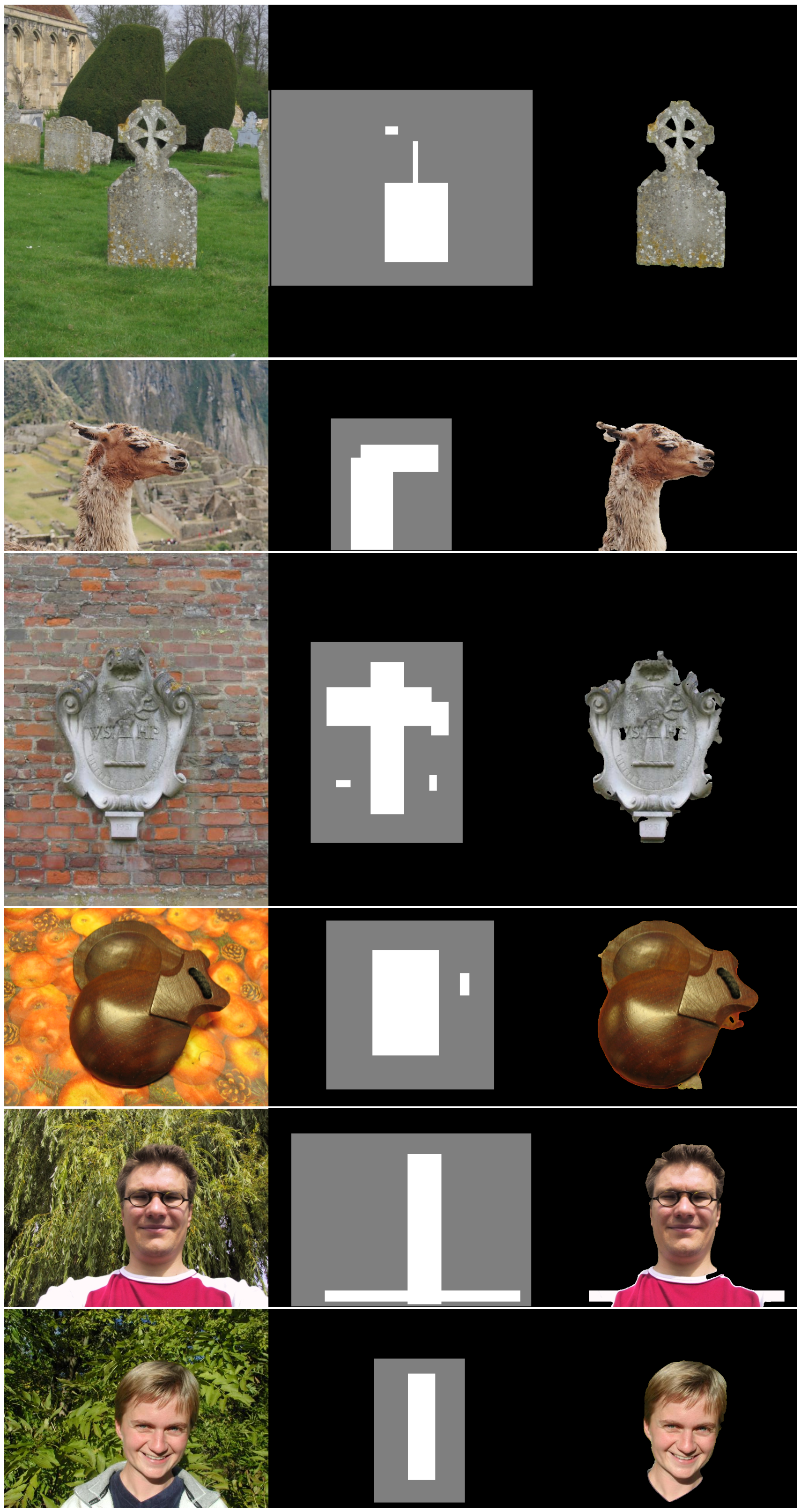
Image Name	Pixels to classify	Wrong Pixels	Error
banana1.bmp	217336	6036	2.7773 %
banana2.bmp	181541	1669	0.9194 %
banana3.bmp	177310	6873	3.8763 %
book.bmp	149236	5685	3.8094 %
bool.jpg	97781	3143	3.2143 %
bush.jpg	80140	8987	11.2141 %
ceramic.bmp	141541	8931	6.310 %
cross.jpg	131367	2790	2.1238 %
doll.bmp	84100	232	0.2759 %
elefant.bmp	138369	1362	0.9843 %
flower.jpg	84638	676	0.7987 %
fullmoon.bmp	30609	5	0.0163 %
grave.jpg	133324	984	0.7380 %
llama.bmp	39243	1833	4.6709 %
memorial.jpg	63443	5526	8.7101 %
music.JPG	123759	3649	2.9485 %
person1.jpg	178285	10079	5.6533 %
person2.bmp	52214	467	0.8944 %
person3.jpg	48819	1095	2.2430 %
person4.jpg	65989	2169	3.2870 %
person5.jpg	27659	1874	6.7753 %
person6.jpg	57223	4172	7.2908 %
person7.jpg	33783	645	1.9092 %
person8.bmp	63632	4325	6.7969 %
scissors.JPG	183373	3128	1.7058 %
sheep.jpg	17477	204	1.1672 %
stone1.JPG	63949	773	1.2088 %
stone2.JPG	113080	186	0.1645 %
teddy.jpg	47677	947	1.9863 %
tennis.jpg	46474	5216	11.2235 %
106024.jpg	30888	1472	4.7656 %
124084.jpg	94731	1999	2.1102 %
153077.jpg	85225	6241	7.3230 %
153093.jpg	71508	1555	2.1746 %
181079.jpg	74573	7023	9.4176 %
189080.jpg	81215	3438	4.2332 %
208001.jpg	54619	1463	2.6786 %
209070.jpg	43280	5314	12.2782 %
21077.jpg	17425	1815	10.4161 %
227092.jpg	64448	2473	3.8372 %
24077.jpg	66354	2918	4.3976 %
271008.jpg	58207	4583	7.8736 %
304074.jpg	19732	3054	15.4774 %
326038.jpg	43581	1883	4.3207 %
37073.jpg	44911	3429	7.6351 %
376043.jpg	56738	3005	5.296 %
388016.jpg	61026	4388	7.1903 %
65019.jpg	34973	4054	11.5918 %
69020.jpg	79050	14054	17.7786 %
86016.jpg	30495	1945	6.3780 %
Total:	3986350	169767	4.2587

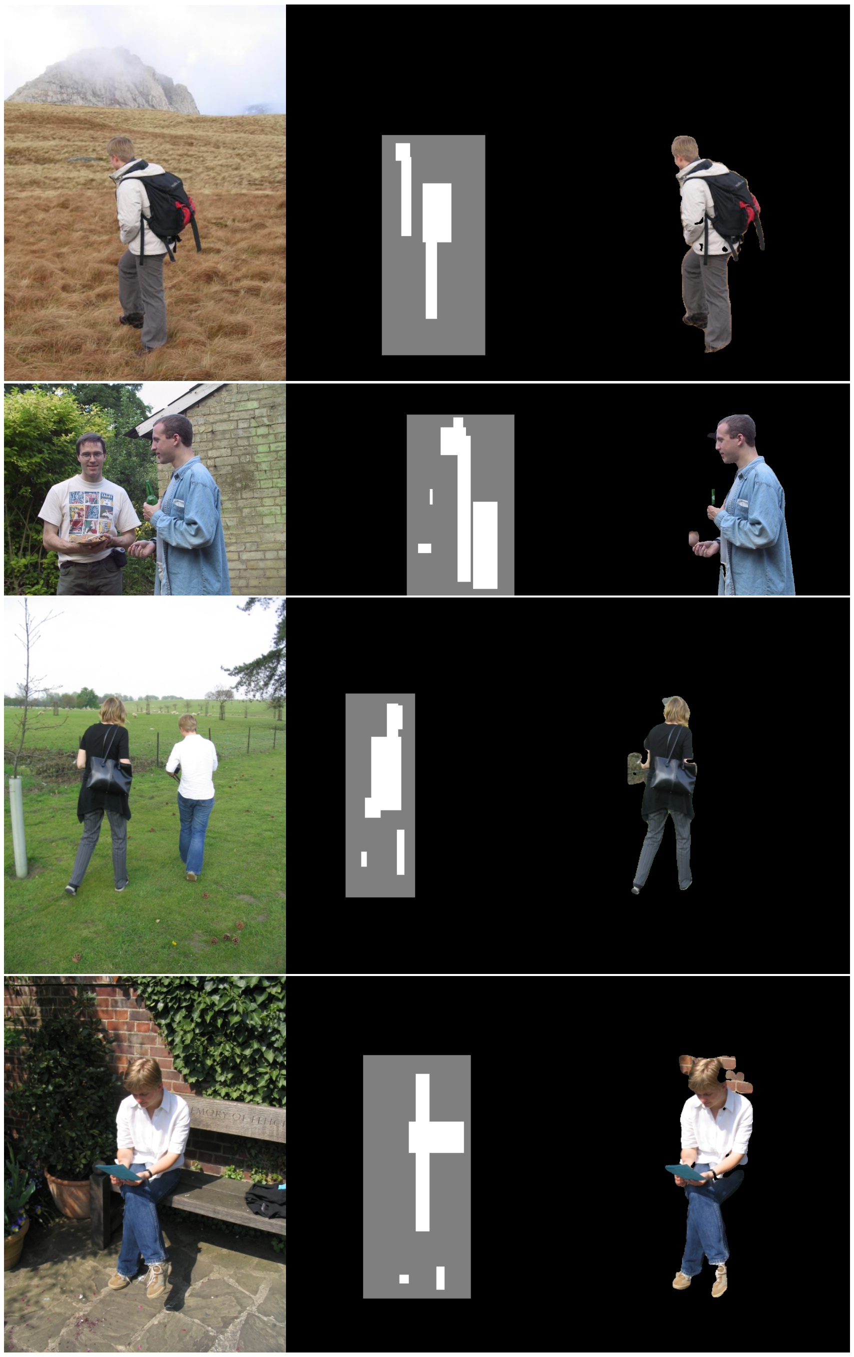
A.2 Images

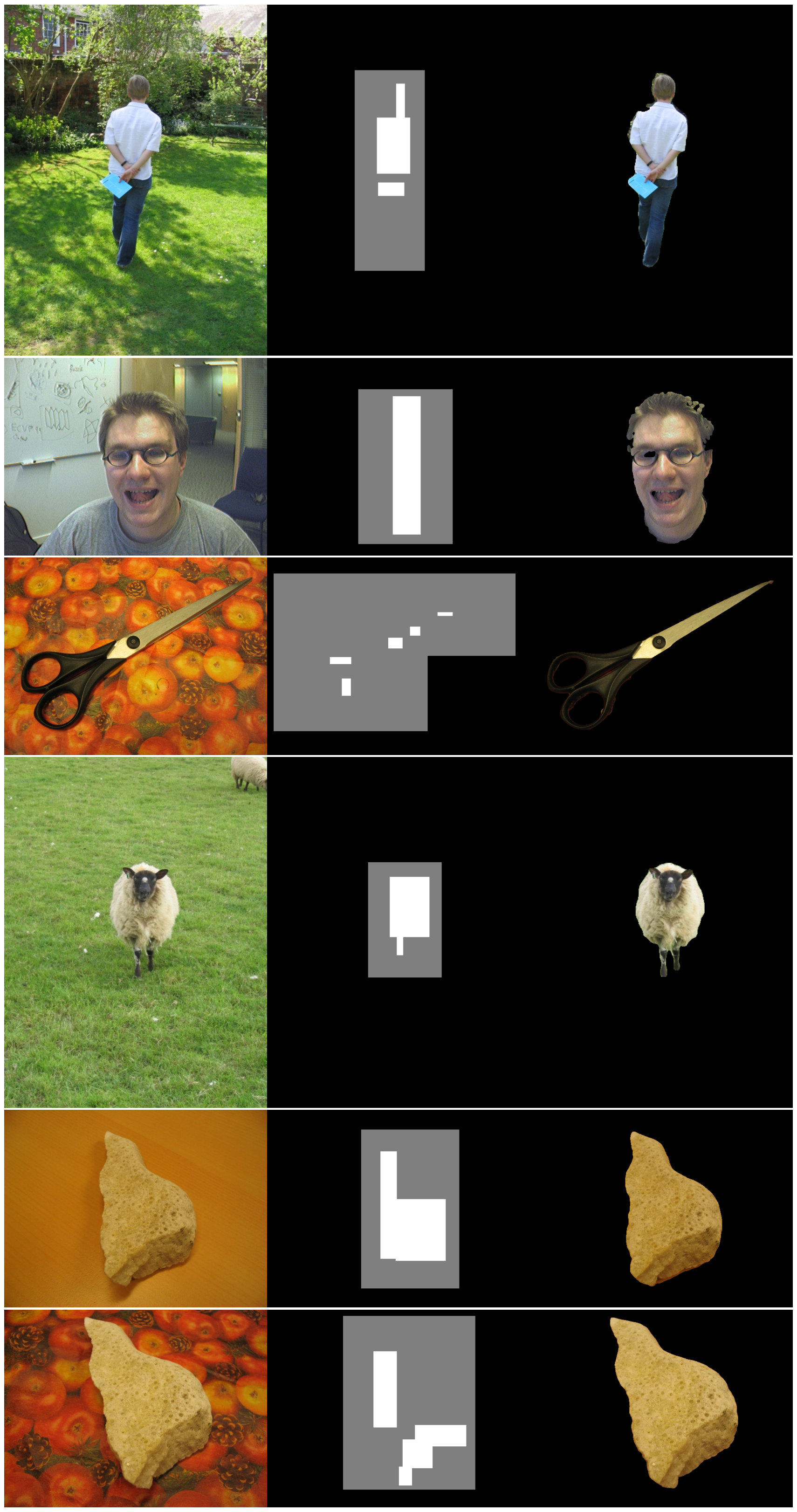
Below are the 50 benchmark images, along with the rectangle trimaps, and the segmentation result.

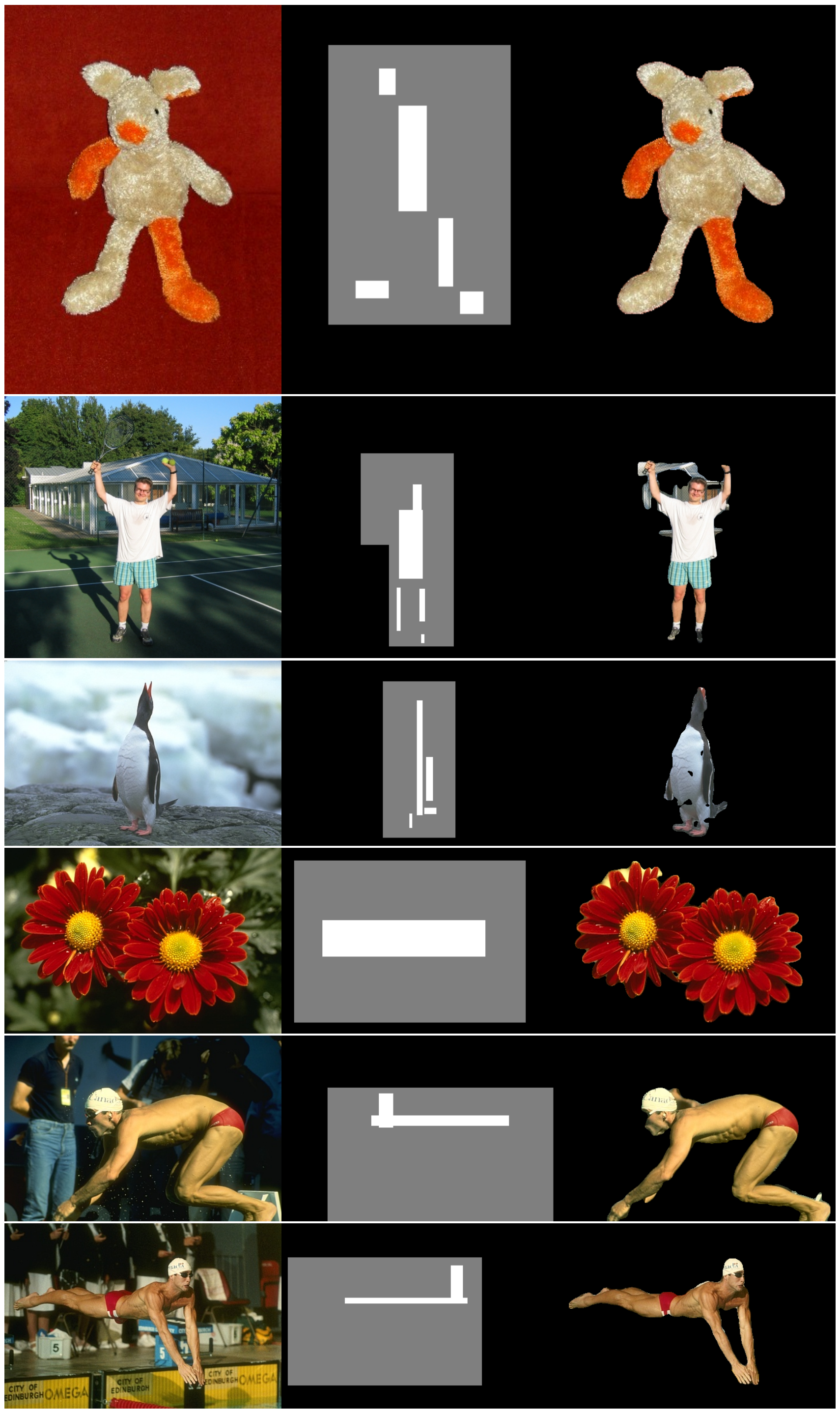











5
CITY OF EDINBURGH
OMEGA