Initiating GrabCut by Color Difference for Automatic Foreground Extraction of Passport Imagery

Sangüesa, Adriá Arbués; Jørgensen, Nicolai Krogh; Larsen, Christian Aagaard; Nasrollahi, Kamal; Moeslund, Thomas B.

Published in:
IEEE International Conference on Image Processing Theory, Tools and Applications

DOI (link to publication from Publisher):
10.1109/IPTA.2016.7820964

Publication date:
2016

Initiating GrabCut by Color Difference for Automatic Foreground Extraction of Passport Imagery

Adrià A. Sangüesa, Nicolai K. Jørgensen, Christian A. Larsen, Kamal Nasrollahi, and Thomas B. Moeslund
Visual Analysis of People (VAP) laboratory, Aalborg University, Denmark
e-mail: {aarbue15, njarge12, cala10}@student.aau.dk, kn@create.aau.dk

***Abstract*— Grabcut, an iterative algorithm based on Graph Cut, is a popular foreground segmentation method. However, it suffers from a main drawback: a manual interaction is required in order to start segmenting the image. In this paper, four different methods based on image pairs are used to obtain an initial extraction of the foreground. Then, the obtained initial estimation of the foreground is used as input to the GrabCut algorithm, thus avoiding the need of interaction. Moreover, this paper is focused on passport images, which require an almost pixel-perfect segmentation in order to be a valid photo. Having gathered our own dataset and generated ground truth images, promising results are obtained in terms of F1-scores, with a maximum mean of 0.975 among all the images, improving the performance of GrabCut in all cases. Some future work directions are given for those unsolved issues that were faced, such as the segmentation in hair regions or tests in a non-uniform background scenario.**

***Keywords*— Foreground Extraction, Image Pairs, GrabCut, Passport Imagery, Color Difference.**

I. Introduction

Image segmentation is a widely used technique, which groups different meaningful pixels into segments. The goal of segmentation is to simplify the amount of data to work with. A clear example is foreground extraction, in which usually, the foreground of an image is first modelled, then, the region belonging to the actual foreground is extracted by thresholding.

The goal of this paper is to design an algorithm for automatic extraction of the foreground of passport images (the head-shoulder part of the image) with high accuracy. This is useful in machines that are used for automatic capturing of passport photos. These machines need to make sure that the passport photos are following some ICAO standards, which can be measured more precisely if the face is segmented properly from the rest of the image.

As it will be explained in Section II, there are many ways to perform foreground extraction, some based on image pairs and others based on still images, where only one shot is needed. In this second case, the most popular algorithm is called GrabCut [1], which is an iterative method based on Graph Cuts [2], and was designed as an improvement of the existing image editing tools, such as intelligent scissors or magic wand. Graph Cut has been used in the literature for different purposes, such as learning how to segment humans [12] and clothes [15] or face and pose recovery [13]; furthermore, extended versions of the algorithm may make it suitable for video applications like video matting [14].

GrabCut, similar to Graph Cut, is initialized with some manual interaction: the user drags a rectangle over a region of the image where the foreground is. Then, the algorithm considers everything outside the rectangle as known background, and, from the information of these pixels and the relation between neighbours inside the rectangle, the probable contour of the foreground is found. The fact that the algorithm requires some manual interaction is a drawback: in our case, a station/machine that takes passport pictures may not have a tactile screen in order to let the users interact with it. The lack of a method to avoid this manual initialization of Graph Cut by using an initial segmentation is the main motivation of this paper. In our case, this initial segmentation will be done by combining four types of color differences, obtained with background-foreground image pairs. These color differences are: *intensity difference*, *euclidean difference*, *color distortion* and *CIEDE2000*.

The proposed system in this paper is motivated by the work of Galsgaard *et al.* [9], in which a Graph Cut based segmentation has been initiated by Circular Hough Transform (CHT) [10] for detecting piled stack of woods for estimating the volume of the stack. The CHT is a good candidate for initiating the Graph Cut in the problem context of the work of [9], because there are many circular woods in the piled stack, which provides good number of pixels for the initial estimation of the foreground. However, such a large number of circles cannot be found when working with facial images in the passport imagery (our case). Therefore, other sources of information are used, which come from different channels generated from the original image to make a rough estimation of the foreground for proper initialization of the GrabCut.

Having gathered a dataset of 23 people with a professional station for passport imagery, some combinations of the different channels of information (color differences) are presented in this paper, that will be then combined with GrabCut to avoid the need for manual interaction.

The rest of this paper is organized as follows: in Section II the state-of-the art is reviewed. Then, in Section III, the proposed system is detailed. The experimental results obtained are then reported and discussed in Section IV. Finally, in Section V the paper is concluded and some future directions

for extending this work are discussed.

II. Related Work

In this Section, different methods to perform Foreground Extraction are explained, starting with those that require two images, and finishing with the ones that only require one shot. Note that it would be also possible to do it with N images (*i.e.* video sequence), where one has the pixel history over a sequence of time. However, this was considered an expensive technique and difficult to apply in passport imagery context, so it is not covered in this paper.

One way to extract the foreground could be done by comparing two images taken under different camera settings, such as blur variations or the method proposed by Sun *et al.* [3]. This method is called Flash Cut, and extracts the foreground from a flash/no-flash image pair. This method considers the extraction as an energy minimization problem of a Markov Random Field, where four components are taken into account. First, the smoothness between adjacent pixels, which helps to differentiate flat regions from textured ones, is calculated. Then, the foreground flash contribution is computed by comparing the histograms of both images. Afterwards, the background flash term is computed, which also compensates the motion between two images; this parameter is computed by modeling the whole image as a Gaussian, which parameters are obtained after matching the SIFT descriptors [6] of both images. Finally, a color term is also introduced by clustering the flash image with 10 components, using a Gaussian Mixture Model [7]. Although the shown results are really accurate, this method has problems where the pictures are taken indoors, because the flash will also affect the background; besides, it requires a flash-illuminated image. As it will be detailed in this paper, our method is based on background-foreground image pairs, which are easier to obtain, plus no unpredictable flash behaviour is present.

The most popular method using still images is GrabCut. Notable research has been conducted in this field, such as the paper presented by Rother *et al.* [1]. The basis of GrabCut is the interaction of the user with an image with complex background: a bounding rectangle is dragged to the region of the scene where an object can be found, so all the pixels outside that region will be considered as known background. Then, the background and foreground inside the rectangle are modelled with a multivariate Gaussian Mixture Model of K components each. This clustering provides an initial segmentation. Later on, a graph is built to find a new classification of background-foreground. This graph is composed by nodes (pixels) and weights, which represent how strongly connected a pixel is to its neighborhood. Moreover, each pixel in the graph is connected to a *Source* and a *Sink* node, which weight will indicate the probability of belonging to the foreground and background respectively. An energy minimization problem is then formulated, cutting the graph in those regions where the smallest weights can be found to obtain a better segmentation. Moreover, it has to be mentioned that the algorithm used is recursive, so the estimations of the background-foreground will be repeated until the algorithm converges. GrabCut results are quite accurate while also requiring less interaction by the user. However, two drawbacks should be mentioned: GrabCut suffers from camouflage problems, which means that it does not correctly identify the foreground object when its color is similar to the background. Furthermore, while the required user interaction is minimal, it still requires interaction to initiate the algorithm.

Another way to perform foreground extraction from still images could be using prior information. In this paper, for example, we are dealing with facial images and thus prior information is known and a model could be generated like the one explained by Yuan *et al.* [5]. However, the images used in our dataset were not normalized, which means that the eyes of the person will not correspond always to the same pixels, so the model could not be built. Besides, a lot of images are needed to build a robust model.

III. The proposed system

In this Section, first segmentation using image pairs is presented then, how to combine them with the GrabCut algorithm is presented.

A. Generation of an Initial Segmentation using Image Pairs

Having the image pair for every person, four types of color differences (sources of information) are computed pixel-wise. The first color difference, which is the simplest, *Intensity Difference*, is useful in those cases were there has not been a lighting change in the time interval between the background and the foreground image. First, the images are converted into grayscale, and then, one image is subtracted from the other using an absolute value, as in Equation 1:

$$I_{dif} = |I_{fg}^{gray} - I_{bg}^{gray}| \tag{1}$$

The advantage of *Intensity Difference* is that there is almost no response in background pixels. Nevertheless, it does not deal properly with highlights (even small ones) nor camouflage (Fig. 4, second column).

The next color difference used is *euclidean difference*, which is computed in the RGB colorspace like a normal difference between vectors as shown in Equation 2:

$$\mathrm{I}_{euclidian} = \sqrt{(I_{bgR} - I_{fgR})^2 + (I_{bgG} - I_{fgG})^2 + (I_{bgB} - I_{fgB})^2} \tag{2}$$

The main drawback of *euclidean difference* is a high response in the background (false positives), at the same time that camouflage issues are not solved. However, it also

provides a strong foreground response, dealing with low intensity highlights (Fig. 4, third column).

The next method to compute the difference between two pixels is called *Color Distortion*, and it is explained by Kim *et al.* in their Codebook Algorithm [4]. First, the norm of the foreground and background images are computed, as well as the dot product between these:

$$\|I_{x_t}\|^2 = I_{fgR}^2 + I_{fgG}^2 + I_{fgB}^2$$

$$\|I_{v_i}\|^2 = I_{bgR}^2 + I_{bgG}^2 + I_{bgB}^2$$

$$\langle I_{x_t}, I_{v_i} \rangle^2 = (I_{bgR} I_{fgR} + I_{bgR} I_{fgR} + I_{bgR} I_{fgR})^2 \quad (3)$$

Then, the *Color Distortion* is computed as follows:

$$I_{p_2}^2 = \frac{\langle I_{x_t}, I_{v_i} \rangle}{\|I_{v_i}\|}$$

$$I_{colorDist} = \sqrt{\|I_{x_t}\|^2 - I_{p_2}^2} \quad (4)$$

The *Color Distortion* produces both acceptable foreground and background responses, improving the performance in camouflage regions. However, the response in the foreground is not uniform at all, producing some small holes in parts where highlights are present (Fig. 4, fourth column).

Last but not least, the *CIEDE2000* color difference is computed. This color difference is computed on the CIELab colorspace, defined by International Comission of Illumination. This space contains all the existing colors, and the equation to compute the color difference 6 takes into account that the human eye is not uniform in terms of perception. In this L*a*b* colorspace, the L* channel represents the Intensity, and both a* and b* are color-opponent axes (yellow-blue, red/magenta-green). In order to compute the *CIEDE2000*, the conversion into the L*h*c* colorspace has to be applied (as shown in equation 5), where L remains unchanged and h* and c* stand for Hue and Chroma respectively:

$$C_{ab}^* = \sqrt{a^{*2} + b^{*2}}$$

$$h_{ab}^* = arctan \frac{b^*}{a^*} \quad (5)$$

Once the L*, h*, c* components are known, the *CIEDE2000* color difference is computed as follows:

$$\mathrm{I}_{\triangle E_{00}^*} = \sqrt{\left(\frac{\triangle L'}{k_L S_L}\right)^2 + \left(\frac{\triangle C'}{k_C S_C}\right)^2 + \left(\frac{\triangle H'}{k_H S_H}\right)^2 + R_T \frac{\triangle C'}{k_C S_C} \frac{\triangle H'}{k_H S_H}} \quad (6)$$

where $\triangle L', \triangle C'$ and $\triangle H'$ are the differences between pixels in their corresponding channels, S_L, S_C and S_H are compensation terms, K_L, K_C and K_H are weighting factors that depend on the application, and R_T is the Hue rotation term. The advantage of the *CIEDE2000* color difference is the uniform response in the foreground as well as being the best of the four color differences in dealing with highlights. The disadvantages are, however, a smaller response in the foreground when comparing to other differences and, again, camouflage regions (Fig. 4, last column).

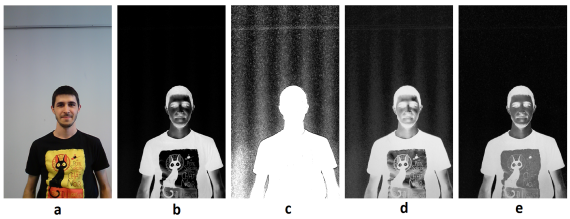

Fig. 1. (a) Original image, (b) intensity difference, (c) euclidean difference, (d) color distortion, (e) CIEDE2000 color difference.

Once all the color differences were generated, thresholding must be applied to every single channel. To find proper thresholds for each color difference channel, we decided to compute the Recall and Precision after thresholding with every single grayscale value from 0 to 255 for every single image, and then compute the mean. Generally, with a really low threshold (considering foreground everything above that value), recall is close to 1 and precision is low. The same happens with high thresholds, which produces a high precision and low recall. A way to take into account both precision and recall is to use F1-scores, as seen in equation 7:

$$F1 = 2 * \frac{P * R}{P + R} \quad (7)$$

In all cases, except in the *euclidean difference*, the threshold was set to the intersection of the Precision and Recall curves, which also intersects the F1-score curve. In the case of the *euclidean difference*, the threshold value is set to the grayscale level where the highest F1-score was obtained. These values for Intensity, Eclidean, Color Distortion, and CIEDE2000 differences are, 30, 254, 70, and 30, respectively.

At this point, knowing the ideal thresholds, four different ways to combine the color difference channels s are examined: 1) The idea behind *Parallel Fusion* is to add all the color differences together, and then, perform thresholding with the mean of the values mentioned above (30, 254, 70 and 30). 2) *Parallel Fusion with Weights* is based on the same idea than *Parallel Fusion*, but in this case, some weights are added to every channel (also while computing the threshold mean) and, at the end, the result is normalized. The thresholds used were found out empirically: six for *Intensity Difference*, one for *Euclidean Difference*, three for *Color Distortion* and four for *CIEDE2000*. The reasoning of these numbers is: *Intensity Difference* may not found the whole foreground, but it does not include false positives at all, so this color difference can be boosted (big weight). The opposite happens with *euclidean difference*, where a lot of background is included, thus a small weight is associated to that difference. Finally, both *Color Distortion* and *CIEDE2000* produce good results but also include

few false positives, so they can not be as boosted as *Intensity Difference*. Note that in this case, the final normalization would be a division by 14×4 ($(6 + 1 + 3 + 4) \times 4$ channels). 3) In the *Parallel Votes* approach the procedure is inverted: first, every channel is thresholded independently. Then, the resulting masks are combined. Finally, another thresholding is applied: if one pixel is detected as foreground by two or more masks, it will be considered as foreground, otherwise, it will be considered as background. 4) *Parallel Votes with Weights* follows the same procedure as the *Parallel Votes*, but in this case, the same weights used in the *Parallel Fusion with Weights* method are applied. This means that the resulting combination of masks will have values between 0 and 14. In this case, the final thresholding will be done based on the following criteria: if one pixel has a value above seven, it will be considered as foreground.

B. GrabCut

Once an initial segmentation is obtained and knowing how GrabCut works, the next step is to use the existing masks as input to the GrabCut algorithm. Nevertheless, it has to be first explained how the built-in version of GrabCut works in OpenCV [8]. The default option of the function requires the interaction of the user by dragging a rectangle over the region of interest. Then, the user has the control of the number of iterations, which means that the algorithm may not iterate until reaching convergence. Another feature of the built-in function is that, after a certain number of iterations, some soft (probable background / foreground) and hard (sure background / foreground) constraints can be added by drawing lines of different colors over the image. Although this feature improves the result (as it can be seen in Fig. 2), it requires even more interaction than the already existing one, so it was discarded.

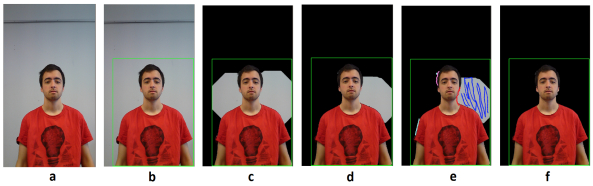

Fig. 2. (a) Original image, (b) Region of interest over the image, (c) First iteration of GrabCut, (d) Third iteration of GrabCut, (e) Manual added constraints, (f) Improved result after having introduced constraints.

Two types of results were needed: manual and automatic. On one hand, some images using the original GrabCut algorithm would help us to compare results *a posteriori*, so the existing algorithm was ran over 10 iterations for each of the 23 existing images after dragging a rectangle manually (without making any kind of touch-up). On the other hand, the region of interest was substituted with the initial segmentation (obtained with the combination of color differences) as an input of the function; most likely, this mask would provide enough constraints to the algorithm in order to obtain an improved result using the same number of iterations, as it will be seen in Section IV.

IV. Experimental Results and Discussion

In this Section, first the collected dataset will be explained, then the obtained results will be reported together with their corresponding discussion. Note that the comparison will be done both visually and numerically, with the previously mentioned metrics (precision, recall and F1-scores). The first results that will be detailed are the ones obtained with the combination of color differences. Afterwards, the results obtained with GrabCut using both a Region of Interest and masks (the ones generated from image pairs) will be discussed. This is done in order to see the effect of the inclusion of some inputs (masks) instead of a manual procedure (dragged rectangle).

A. Gathered Dataset

In order to test our algorithms, we collected our own dataset. At first, our intention was to use a benchmark database that could be used to then compare with other algorithms, but the existing databases were too perfect in terms of segmentation, *i.e.* a database with head and shoulder images with a uniform lit background and no natural light affecting the scenario (which can be unpredictable sometimes). The scenario wherein the images were taken was quite simple, with an almost uniform background plus controlled lighting; besides, the images had a resolution of 8 MP. For each person, an image pair was taken: one image only with the background of the scenario, and the next one with the person sitting in front of the camera. It was decided to take the background picture every time because lighting conditions could change and lead to wrong results.
Once the dataset was gathered, ground truth images were manually generated for every person in order to be able to compute metrics afterwards. An example is shown in Fig. 3.

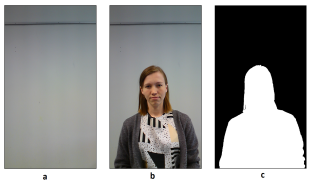

Fig. 3. (a) Background, (b) Foreground, and (c) Ground truth images.

B. Results obtained with Image Pairs

As explained before, four different methods are applied in order to obtain a final segmentation. From the visual results that can be observed in Fig. 4, the best color difference (*CIEDE2000*) is also displayed. As it can be seen, the fourth column, corresponding to the *Parallel Votes with Weights*, not only is the one that better deals with the noise in the

background, but also the one that better deals with camouflage regions (like the t-shirt of the second row).

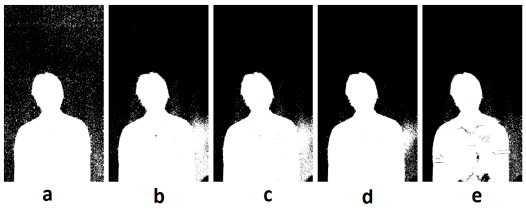

Fig. 4. (a) CIEDE2000 thresholded color difference, (b) Parallel Fusion, (c) Parallel Fusion with Weights, (d) Parallel Votes and (e) Parallel Votes with Weights.

From the metrics point of view, the thresholds explained in the previous section were used, as well as an slightly smaller and bigger values (because doing the mean of only 23 values and using empirical thresholds may not be robust at all). These results can be seen in Tables 1, 2, 3 and 4, and show that the technique of *voting* performs better than *fusion* and, in both cases, the addition of weights improves results a little bit.

This enhances the conclusion that was reached when inspecting the results visually: the method that performs better is *Parallel Voting with Weights*.

	Thresholds		
	64	*74*	*84*
Precision	0.9058	0.9298	0.9479
Recall	0.9221	0.9169	0.9129
F1-Score	0.9018	0.9119	**0.9201**

Table 1. Results obtained with the Paralell Fusion method, using the empirical threshold (74) and a bigger and smaller one.

	Thresholds		
	45	*55*	*65*
Precision	0.9669	0.9967	0.9988
Recall	0.9343	0.8926	0.8547
F1-Score	**0.9455**	0.9365	0.9145

Table 2. Results obtained with the Paralell Fusion with Weights method, using the empirical threshold (55) and a bigger and smaller one.

	20-254-60-20	*30-254-70-30*	*40-254-80-40*
Precision	0.9454	0.9588	0.9790
Recall	0.9782	0.9650	0.9450
F1-Score	0.9575	**0.9578**	**0.9578**

Table 3. Results obtained with the Paralell Votes, using the empirical thresholds (30-254-70-30) and bigger and smaller ones.

C. Results obtained with GrabCut

In this section, the performance of GrabCut using a Region of Interest will be compared to the performance of the same

	20-254-60-20	*30-254-70-30*	*40-254-80-40*
Precision	0.9658	0.9942	0.9987
Recall	0.9569	0.9308	0.8969
F1-Score	**0.9599**	**0.9599**	0.9408

Table 4. Results obtained with the Paralell Votes, using the empirical thresholds (30-254-70-30) and bigger and smaller ones.

algorithm using the previously obtained mask. The improvement of the GrabCut can be observed in Fig. 5, where the first column corresponds to the performance of the algorithm dragging a rectangle manually to the Region of Interest. As it can be seen, using this manual method, the algorithm does not get rid of all the background. Nevertheless, using the obtained masks as inputs improves the result, obtaining an almost perfect segmentation using *Parallel Votes with Weights*.

Fig. 5. (a) Original image with a ROI and (b)-(e) obtained masks from image pairs. Then, obtained results with Grabcut: (f) Region of Interest, (g) Parallel Fusion, (h) Parallel Fusion with Weights, (i) Parallel Votes, (j) Parallel Votes with Weights.

As it can be noticed, if the binary mask contains most of the foreground, it will most likely fill those camouflage regions that are really difficult to to detect by computing color differences. Another example is displayed in Fig. 6, where it can be observed that the t-shirt is completely filled when enough information is provided in the mask. This means that using a combination of GrabCut with masks improves the performance of both GrabCut and masks separately.

Fig. 6. (a)-(d): obtained masks from image pairs in RGB. (e)-(h): obtained result after applying Grabcut: (e) Parallel Fusion, (f) Parallel Fusion with Weights, (g) Paralell Votes, (h) Parallel Votes with Weights.

From the point of view of metrics, it can be seen in Table 3 that, using the combination of the mask obtained with *Parallel Votes with Weights* plus GrabCut, a F1-score mean of 0.975 is obtained among the 23 images. Some of the best and worst cases are displayed in Fig. 7.

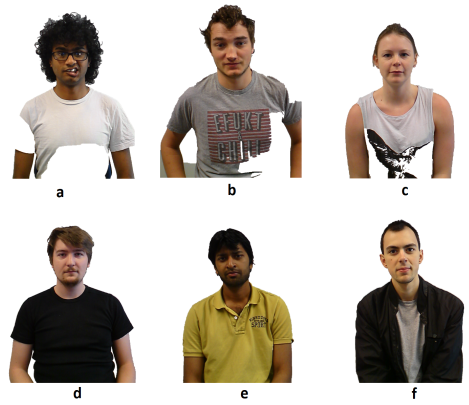

Fig. 7. (a)-(c): worst obtained results (camouflage issues), (d)-(f): some of the best obtained results.

V. Conclusions and Future Work

In this paper, a foreground extraction algorithm designed for passport imagery has been explained. This algorithm only requires 2 shots and it is an improvement of the GrabCut algorithm. From a background-foreground image pair, a first estimation of the foreground is obtained by combining 4 types of color difference: *intensity*, *euclidean*, *color distortion* and *CIEDE2000*. This combination is performed also in 4 different ways, based on *fusion* or *votes* plus adding the possibility of working with weights. The obtained results are evaluated with a F1-score metric, and these show that the *parallel votes with weights* method is the one that performs better, obtaining a score of 0.9599 among all the images. In order to improve the obtained result, the mask generated with the pair of images is used as the input of the GrabCut algorithm; results show that the performance is increased in all the 4 methods, obtaining a maximum F1-score of 0.975 in the *parallel votes with weights* method. Besides, these results are compared with the original GrabCut algorithm, where a manual initialization is required; the performance of the fully automatic method (using the mask) improves the performance of the original method at the same time that it avoids all kind of interaction.

	Precision	*Recall*	*F1-Score*
Region of Interest	0.734	0.955	0.827
Fusion	0.948	0.913	0.920
Fusion + GrabCut	0.959	0.898	0.919
Fusion with Weights	0.997	0.893	0.936
Fusion with Weights + GrabCut	0.989	0.903	0.939
Votes	0.979	0.945	0.958
Votes + GrabCut	0.977	0.946	0.959
Votes with Weights	0.994	0.931	0.960
Votes with Weights + GrabCut	0.994	0.959	**0.975**

Table 5. Comparison in terms of metrics of GrabCut, masks obtained from image pairs, and the combination of both. P stands for paralell.

Although it has been proved that our algorithm works properly in a scenario with a uniform background and controlled lighting conditions, the same method should be tested in a more challenging scenario, where natural light affects the scene and there is a moving background.
An issue that was unsolved was segmentation around hair regions, especially in those cases where there is long and curly hair, containing holes in it. Those tricky cases are difficult to detect even visually, making also the ground truth images not pixel-perfect at all, so extracting the foreground with high precision is such a difficult task. A way to solve it could be applying a specific method for hair segmentation, like the one presented by Wang *et al.* [11], which takes advantage of a Hair Occurrence Prior Probability or a Generic Hair Color Model.

References

[1] C. Rother, V. Kolmogorov and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics*, 309–314, 2004.
[2] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. *IEEE International Conference on Computer Vision, ICCV 2001*, 105–112, 2001.
[3] J. Sun, S. Kang, X. Bing, T. Zong-Ben, S. Xiaoou and Heung-Yeung. Flash cut: Foreground extraction with flash and no-flash image pairs *IEEE Conference on Computer Vision and Pattern Recognition, CVPR'07* , 1–8, 2007.
[4] K. Kim, T. H Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground–background segmentation using codebook model. *Real-time imaging, Elsevier* , 172–185, 2005.
[5] X. Yuan, F. Zhong, Y. Zhang and Q. Peng. Automatic segmentation of head-and-shoulder images by combining edge feature and shape prior. *12th International Conference on Computer-Aided Design and Computer Graphics* , 163–170, 2011.
[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* , 91-110, 2004.
[7] A. Blake, C. Rother. Interactive image segmentation using an adaptive GMMRF model. In *Proceedings of ECCV*, 428-441, 2004.
[8] OpenCV Dev Team. Miscellaneous Image Transformations - GrabCut `http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#grabcut`. Last accessed at 2016, May 23rd.
[9] B. Galsgaard, D. H. Lundtoft, I. Nikolov, K. Nasrollahi, and T. B. Moeslund. Circular Hough Transform and Local Circularity Measure for Weight Estimation of a Graph-Cut Based Wood Stack Measurement. *IEEE Winter Conference on Applications of Computer Vision*, 686-693, 2015.
[10] Smereka, Marcin, Duleba, Ignacy. Circular Object Detection Using a Modified Hough Transform. *Int. J. Appl. Math. Comput. Sci.*, 18(7): 85-91, 2008.
[11] D. Wang, S. Shan, W. Zeng, H. Zhang and X. Chen. A novel two-tier Bayesian based method for hair segmentation *16th IEEE International Conference on Image Processing (ICIP)*, 2401–2404, 2009.
[12] V. Gulshan, V. Lempitsky and A. Zisserman. Humanising GrabCut: Learning to segment humans using the Kinect. *IEEE International Conference on Computer Vision Workshops,*, 1127–1133, 2011.
[13] A. Hernandez, M. Reyes, S. Escalera and P. Radeva. Spatio-temporal grabcut human segmentation for face and pose recovery. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops,*, 33–40, 2010.
[14] D. Corrigan, A. Kokaram and S. Robinson. Video matting using motion extended grabcut. *5th European Conference on Visual Media Production,*, 1–9, 2008.
[15] M. Wang, L. Shen, and Y. Yuan. Automatic foreground extraction of clothing images based on grabcut in massive images. *IEEE International Conference on Information Science and Technology,*, 238–242, 2012.