SETTING GRAPH CUT WEIGHTS FOR AUTOMATIC FOREGROUND EXTRACTION IN WOOD LOG IMAGES

Enrico Gutzeit, Stephan Ohl, Arjan Kuijper, Joerg Voskamp and Bodo Urban
Fraunhofer Institute for Computer Graphics Research IGD, Joachim-Jungius-Str. 11, 18059 Rostock, Germany

Keywords: Image segmentation, Graph cuts, Foreground extraction, Weight setting.

Abstract: The automatic extraction of foreground objects from the background is a well known problem. Much research has been done to solve the foreground/background segmentation with graph cuts. The major challenge is to determine the weights of the graph in order to obtain a good segmentation. In this paper we address this problem with a focus on the automatic segmentation of wood logs. We introduce a new solution to get information about foreground and background. This information is used to set the weights of the graph cut method. We compare four different methods to set these weights and show that the best results are obtained with our novel method, which is based on density estimation.

1 INTRODUCTION

Graph-cut based segmentation techniques are a very powerful tool in image segmentation. In interactive image analysis, e.g. in medical imaging (F. Malmberg and Borgefors, 2009), it is possible to get a good segmentation with only a few refinements. In a fully automatic system there is no possibility for refinements and so the problem of initialization of the graph-cut algorithm, mainly setting the weights for the graph, is a difficult problem. In this paper the problem is especially addressed to automatically and soundly segment wood log images.

The volume of wood and the sizes of the logs are important factors of commercial and logistic processes in timber industry. There are different methods to measure the amount of wood. Most reliable techniques are laser scanning methods or wood weighing for volume analysis. These methods, however, are mainly used in factories since they are relatively difficult to apply. In forests estimations are used often. After logs have been cut they are piled up onto stacks. The amount of cut wood is then estimated from the front side of the stack. Furthermore, the distribution of log diameters is estimated by visual judgment only. It is described by a constant because measuring the diameter of each log is impossible in practice.

Our aim is to use computer vision methods to make these front side measurements faster and more reliable. Separating the log cut surfaces from the background leads to the well known problem of binary image segmentation. A robust automatic binary segmentation allows more accurate volume estimation techniques than the manual one described briefly above. The imaging devices we use are mobile phone cameras as these are lightweight common place devices. On the other hand, their optics incorporate some trade-offs on image quality. Furthermore, we restrict ourselves to images taken from frontal positions.

The contribution of this paper is a methodology for automatic and robust foreground / background segmentation of these low quality wood log images. We use a min-cut/max-flow graph-cut algorithm in conjunction with a kd-tree accelerated density estimation. Our novel density estimation improves current approaches with respect to robustness and is relatively simple to implement, which we call KD-NN in the following. The methodology is split into two parts. First, we make use of some properties of the image structure to initiate models in color space. Second, we used them to yield the final segmentation. The graph-cut algorithm is used in both steps.

2 RELATED WORK

Our specific application has been considered in imaging barely. (Fink, 2004) describes classical vision algorithms and active contours to find the log cut surfaces. The proposed procedure is only half automatic

Gutzeit E., Ohl S., Kuijper A., Voskamp J. and Urban B. (2010).
SETTING GRAPH CUT WEIGHTS FOR AUTOMATIC FOREGROUND EXTRACTION IN WOOD LOG IMAGES.
In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 60-67
DOI: 10.5220/0002831000600067

and makes a lot of assumptions about the type of wood, the lighting conditions and the image quality. The photos of the stack of wood are taken in a controlled environment which makes the proposed algorithms useless in our scenario.

In computer vision there is a variety of approaches to tackle the binary segmentation problem (Jaehne, 2005). However, when it comes to stable and fully automatic segmentation of natural images, things are getting complicated. Normalized Cuts (Shi and Malik, 2000) are used to find a reasonable partition of the image. But these kinds of algorithms are optimized in order to solve the grouping problem in vision.

Instead, we focused on a quite popular binary segmentation framework from the fields of combinatorial optimization. Hereby, a graph-cut results from solving for the maximum flow (Boykov and Jolly, 2001). The corresponding minimum cut then describes the resulting partition of the image. There are algorithms that are optimized for grid-graphs (Boykov and Kolmogorov, 2004) and provide a feasible solution in a few seconds, even for megapixel sized images.

There are different approaches to set the weights of the graph. The weights between two adjacent vertices are set often by using some metric like the euclidian for instance. While finding a pixel to pixel distance is straight forward, existing algorithms differ in setting the weights for the remaining edges to the source and the sink node. A good and practical solution for gray scaled images is the usage of a histogram which describes the two necessary distributions for binary segmentation (Boykov and Jolly, 2001). In the case of RGB images, it is more challenging to find a model for the two distributions. In (C. Rothar and Blake, 2004) a Gaussian Mixture Model (GMM) is used for description. However, the method deals with interactive segmentation. It is derived from (Orchard and Bouman, 1991).

We implemented this GMM method and found it unstable. The method is quite prone to outliers. In interactive segmentation, this is not so important because the model can be changed quickly. But this does not meet our requirements. Instead, in our comparison we replaced the method for estimating the GMMs by the Expectation Maximization algorithm.

3 PROBLEM DISCUSSION

The volume especially the solid cubic meter of a stack of wood is simple computable via a multiplication of the area of the log cut surfaces with the depth. The depth per stack of wood is known, but not the wood surfaces. To determine the wood area from images, the stack of wood must be taken from a frontal position, whereby the log cut surfaces must be visible (see figure 2).

To obtain the area of wood the image must first transform into a real coordinate system, so that every pixel has the same area in real square meter. Secondly, a stack of wood often doesn't fit into a single image. Due to this problem more images must be taken from one stack of wood and be stitched together. Thirdly, a segmentation is required to separate the wood and non-wood pixel. How to transform the images and stitch them together is beyond the scope of this paper. Therefore, we here only address the third problem, the segmentation.

The objective is to separate automatically and soundly the log cut surfaces from different images. Log cut surfaces of a stack of wood in praxis vary in shape and color. The shape of an log cut surface seems to be a circle or an ellipse, but that is not ever given in praxis (see figure 1). Therefore a shape finding technique, e.g. ellipse fitting, cannot be applied. Furthermore the color from one stack of wood to another is different and dependent on the wood type, whereby no general color matching is appropriate.

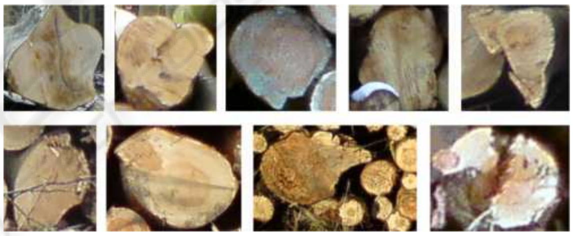


Figure 1: Logs with different shape and color.

Logs have a certain self-similarity, but in a different degree. Hence the usage of simple region based methods, e.g. watershed or split and merge, lead to the well known problem of under- or over-segmentation. In summary, there is no exact color or shape for all logs usable, but for one single stack of wood the color is mostly similar and the gradients between logs and non logs are often high.

For this reason we extract color information from the image first and use this to segment the image. To use both characteristics of local gradient and global color a graph-cut approach is used.

4 OUR APPROACH

For our approach we require some general restrictions on the image acquisition to have some context information. First, the image needs to be taken from a frontal position as mentioned earlier and the stack of wood has to be in the center. Second, the upper and lower area of the image must not contain log cut surfaces. This is realized in common praxis because a stack of wood has a limited height (see also figure 2).

Figure 2: A sample image of a stack of wood.

In the following, the image part that represents the log cut surfaces is called foreground and the remaining areas are called background. This follows the common naming conventions in graph-cut papers.

The segmentation is generally split into two main parts (see figure 3). In step one a region in the center of the input image is used to extract information about the log cut surfaces and the darker regions in between. In this step a number of properties and a first graph-cut with our novel KD-NN to set the graph weights are used.

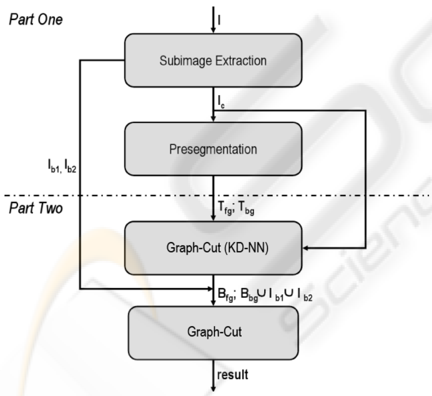


Figure 3: The two parts of the segmentation procedure.

Additionally, one subimage from the bottom and one from the top are used to gain information about the background. Both subimages are needed because they represent characteristics of quite different parts of the image, e.g. sky, forest, soil, snow or grass.

This information is used together with the background characteristics to apply graph-cut a second time to the whole input image, but this time with different weight setting algorithms for later comparison. Background characteristics means the objects in front of and behind the stack of wood. All steps are described in detail in the following sections.

4.1 Our Novel Approach for Fore- and Background Extraction

The aim of the first part of the segmentation is to find a first estimate for the foreground (log cut surfaces) and background color models. These models are necessary for the graph-cut algorithm to accurately segment the log cut surfaces. The following description is also illustrated in figure 3.

Let I denote the input image of size $m \times n$ where m is the height and n the width. We extract three subimages from I. First, we extract a subimage I_c from the center region. The other two subimages, I_{b1} and I_{b2}, are extracted from the top and bottom of I respectively. Due to the constraints in image acquisition, I_c contains log cut surfaces only and the shadowed regions in between whereas I_{b1} and I_{b2} contain regions to be classified as background. We chose I_c to be of size $\lfloor \frac{m}{3} \rfloor \times \lfloor \frac{n}{3} \rfloor$. I_{b1} and I_{b2} are horizontal bars of size $\lfloor \frac{m}{20} \rfloor \times n$. Our experiments have proven these dimensions reasonable.

Whereas the pixels of I_{b1} and I_{b2} can directly be used for the background model we need to segment I_c into foreground and background pixels. We do this by using a stable novel method which is described in detail in the following.

Segmenting I_c is much easier than the segmentation of I because the difference in luminance between the log cut surfaces and shadowed regions in between is very strong. Nevertheless, muddy logs, different types of wood and leaves are disturbing factors.

Hence, for a stable segmentation we use the intersection of two binary threshold segmentations where each of these is performed in another color space. The V-channel from the HSV color space is thresholded directly. In RGB color space we use the observation that wood surfaces often contain a strong yellow component. Therefore, we extract Y from the RGB image using the following equation pixel wise.

$$Y = \max(\min(R - B, G - B), 0) \tag{1}$$

Both channels V and Y are automatically thresholded by using the method in (Otsu, 1979). The resulting binary images are V_b and Y_b. The intersection of both

$$T_{\text{fg}} = V_b \cap Y_b \tag{2}$$

$$T_{\text{bg}} = \overline{V_b} \cap \overline{Y_b} \tag{3}$$

results in a trimap $T = (T_{\text{fg}}, T_{\text{bg}}, T_{\text{unknown}})$. T_{fg} is the foreground, T_{bg} is the background and T_{unknown} are pixels of which it is not known, if they belong to the foreground or the background. T_{unknown} is expanded further by morphological operators to ensure definitely a clean segmentation.

To get a more accurate binary segmentation of I_c a first graph-cut segmentation is used. The pixel sets T_{fg} and T_{bg} are used to build the foreground and background model. The result is a binary segmentation $B = (B_{\text{fg}}, B_{\text{bg}})$. For the final segmentation a second graph cut is applied to I. The pixel set to describe the foreground is B_{fg}. The background model is built by using the pixel set union $B_{\text{bg}} \cup I_{b1} \cup I_{b2}$.

4.2 Graph-Cut and Weight Setting

Graph based image segmentation methods represent the problem in terms of a graph $G = (V, E)$. The graph consists of a set of nodes $v \in V$ and a set of weighted edges $e \in E$ connecting them. In an image each node corresponds to a pixel which is connected to its 4 neighbors. Additionally there are two special nodes called terminals (sink and source), which represent foreground and background. Each node has a link to either of the terminals.

In the following let w_r be the weights of the edges connecting pixels and terminals and let w_b be the weights of inter pixel edges. We assume that all weights w_b, w_r for the graph are in the interval $[0,1]$. A vector in feature space (RGB color space) is denoted by $\mathbf{x}$.

For setting the w_b, we applied the following formula, whereby i and j indicate adjacent graph nodes.

$$w_b = 1 - e^{-\alpha * \|\mathbf{x}_i - \mathbf{x}_j\|_1} \quad (4)$$

We used the Manhattan Metric because it is a little faster than the Euclidian Metric and the difference in segmentation results was negligible for our images. A higher choice for the value of the free parameters α leads to more similarity between two feature vectors.

For setting the w_r to the terminal nodes we implemented four different feature space analysis methods, each of which is described in detail in the following subsections. In every case we built two models, one for the foreground and one for the background respectively. We did not introduce indices in the formulas to keep the notation uncluttered.

After having set all weights, the min-cut/max-flow algorithm from (Boykov and Kolmogorov, 2004) is applied. The results of each of the four following feature space analysis methods are presented and discussed in chapter 5.

In the following we describe four different methods to determine w_r by using the foreground pixels B_{fg} and the background pixels B_{bg}. Afterward the results will be present and compared.

4.2.1 Histogram Probability

A simple method to determine a probability is to use a histogram. We used a 3D histogram to approximate the distribution of the foreground and background over the color space. One bin of the histogram represent a probability p_h. The probability can be used to determine w_r. To reduce noise, we scale the histogram with the factor $hs \in [0,1]$, which is experimentally set and used for comparison. The number of bins per color channel in our case is calculated with $\lfloor 256 * hs \rfloor$. Another problem of histograms are peaks, which lead to many low probabilities. Peaks in our case arise from many pixel with the same color in B_{bg} or B_{fg}. To get a suitable weight the histogram is normalized first, so that the maximal bin has a probability of 1. Afterward, the weights are determined by the following equation, which get the best results in our experiments.

$$w_r = 1 - e^{-\beta * sum_h * p_h} \quad (5)$$

The sum_h indicates the sum of all bins, p_h the bin value and so the probability of the corresponding pixel and β is a free parameter.

4.2.2 K-Mean Clustering

Clustering is a common used method to grouping similar points into different clusters. K-Mean is a simple clustering algorithm and is often used to cluster the color space. In our case we are applied K-Mean to cluster the background and foreground pixels. The results are k cluster with the mean m_i, whereby $i \in \{1, \ldots, k\}$. To determine the weights different ways are possible. One possibility is to find the nearest cluster mean m_i, calculate the distance d_{min} between the pixel and m_i and directly determine the weight. Another way is to determine the weights using the average of all distances to all cluster means m_i. Additionally the amount of pixels per cluster can also be included for weight computation. We experimentally found out, that the best segmentation results are created by using the euclidian distance to one cluster mean and the free parameter γ by

$$w_r = 1 - e^{-\gamma * d_{min}} \quad (6)$$

4.2.3 Gaussian Mixture Models

A Gaussian Mixture Models (GMM) is a compact way to describe a density function. It may be seen as a generalization of K-Means Clustering. The standard way to find the mean vectors and covariance matrices of a GMM of K components is the Expectation Maximization Algorithm. To speed up learning we additionally sampled from our learning data.

When predicting from the model we can not take the density function values directly to initialize the weights of the graph. The reason therefor are very small probability values and sharply peaked gaussian components. Instead, we used it in a similar fashion like the K-Mean Clustering. We left the normalization factors out in the prediction phase, so our model reduces to

$$w_r = \sum_{k=1}^{K} \pi_k * \exp(-\frac{1}{2}(\mu_\mathbf{k} - \mathbf{x})^T \Sigma^{-1} (\mu_\mathbf{k} - \mathbf{x})). \quad (7)$$

The π_k sum up to 1. That means the weights w_r will lie in $[0,1]$. The difference to the cluster centers is anisotropic, whereby the simple K-Means approach leads to isotropic differences. We notate this method with EM-GMM in the following.

4.2.4 Our Novel Density Estimation by Nearest Neighborhood

Our novel method is based on density estimation by using a kd-tree, which we named KD-NN. To set the weights to source and sink node we use two kd-tree based models. The kd-trees contain all selected pixels and the associated values. One contains the foreground and one the background pixels. So all information is stored and used to set the weights in a later step by using NN. The used model is similar to a photon map in (Jensen, 2001). A photon map contains photons, our contains pixels. Both are used for density estimation. For fast NN search a balanced kd-tree is needed. We use a data driven kd-tree and save a pixel in relation to the color value. Each node of the tree splits the color space by one dimension, stores the position in color space and the number of pixels. We built a balanced kd-tree (Jensen, 2001). Building a kd-tree in this way is a $O(n * \log n)$ operation.

Similar to (Felzenszwalb, 2004) nearest neighborhoods are involved. The NN are used for density estimation and setting the region weights w_r. Especially for every graph node v the corresponding density in color space within a sphere is used.

The first step is to determine the density in relation to all pixels in the kd-tree. Hence, the number of pixels p_{all} in the kd-tree and the volume of the color space are used to calculate an overall density ρ_{all}. In our case, we use the *RGB* color space, where $R, G, B \in [0, 255]$, and compute the density

$$\rho_{\text{all}} = \frac{p_{\text{all}}}{255^3}. \quad (8)$$

We estimate the density for every v by using its sphere environment (see figure 4). The number of pixels within this sphere with a predefined radius r is

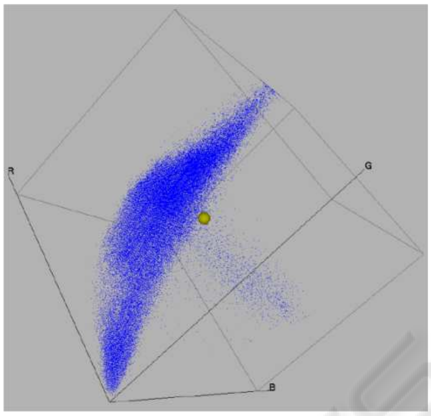


Figure 4: The pixels of the log cut surfaces are visualized as points in RGB color space. The yellow sphere demonstrates the search environment for the density estimation. Also, there is a number of outliers.

searched in the kd-tree. The NN search in a balanced kd-tree is an $O(\log(n))$ operation (Samet, 2006). The volume of the search sphere v_s is

$$v_s = \frac{3}{4} * \pi * r^3. \quad (9)$$

The density ρ_s within the sphere environment is estimated by the number of found pixels p_s. It is used for the weight calculation in the following section.

$$\rho_s = \frac{p_s}{v_s} \quad (10)$$

The setting of w_r is done by the density ρ_s. However, the number of pixels in one kd-tree and hence ρ_{all} is different and depends on the foreground and background pixels. Hence, we use the overall density ρ_{all} to determine a factor s, which is taken for weight computation.

The idea is to map the overall density ρ_{all}, which is also the mean density from the spheres in all possible positions, to a defined weight w_m. So, if the mean density is found, the weight w_b will be equal to w_m. In addition, a greater density than ρ_{all} must produce a high weight and a lower density a low weight, which all must be in the interval $[0,1]$. Therefore, the factor is determinated by the following equation.

$$w_m = 1 - e^{-\rho_{all} * s} \quad (11)$$

$$s = \frac{\ln(1 - w_m)}{-\rho_{all}} \quad (12)$$

Finally, the region weights are estimated by the density in the search sphere and the predetermined factor s by

$$w_r = e^{-\rho_s * s}. \quad (13)$$

5 RESULTS

We tested our approach on images taken by employees from forestry. Due to the image acquisition with a mobile phone, our tested input images are jpeg compressed with maximal resolution of 2048×1536 pixels. In our application we implemented the weight computation in four different ways as described before. All methods determine the edge weights w_b as specified in 4.2 but differ in the calculation of the w_r.

In our novel approach the graph-cut is processing twice. To compare the different methods the same weight setting for graph-cut in the presegmentation is used. Hence the same conditions, especially the same foreground B_{fg} and background pixels B_{bg}, are given for the second graph-cut. We generally used the best method for setting the weights our KD-NN for the graph cut segmentation as evaluated later. In figure 5 and 6 the different states of the presegmentation and the final result of the input image (figure 2) are shown. Thereby the image quality improves over the consecutive steps.

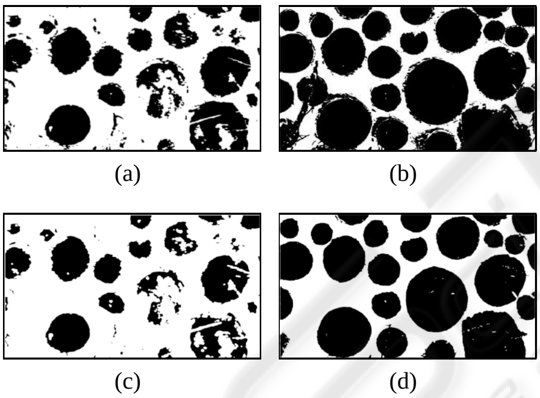


Figure 5: The different steps of the presegmented center image are shown. Image (a) shows the thresholded modified Y-channel, (b) the thresholded V-channel and (c) the intersection of (a) and (b). In Image (d) the result of the first graph cut with KD-NN is shown.

For the comparison of the different weight setting methods including our novel KD-NN from section 4.2, we performed a ground truth test. Therefore 71 very different sample images with wood logs were marked and the difference to the segmentation result were measured as shown in figure 7. We experimentally choose the best parameter to determine w_r. The weights created with the histogram were calculated with $\beta = 1000$. For K-Mean γ was set to 0.02 and for our KD-NN $w_m = 0.5$ and $r = 2.5$ were used. For all methods we used $\alpha = 0.0005$ for w_b.The simple RGB color space, where $R,G,B \in [0,255]$ was used in all methods.

Figure 6: The final segmentation of the second graph-cut run with KD-NN by using the presegmented center image, which is shown in figure 5.

We measured the difference to the ground truth by three values. The first is the percent amount of too little segmented pixels, which are notated as false negative. The second is the percent amount of too much segmented pixels and the third the sum of them, which is also the general error in the segmentation. This all is in relate to the number of pixels in the image. Furthermore we calculate the standard derivation of the values. The table 1 shows the three measured values for the different methods.

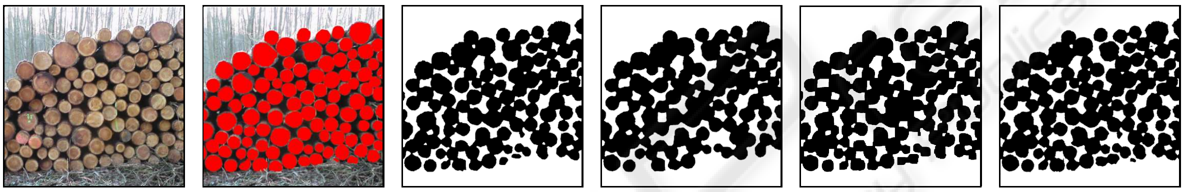
In table 1 it can be seen, that our KD-NN leads to the least general segmentation error. The K-Mean with eight clusters, whereby each cluster is initial positioned in one corner of the color cube, lead to similar results. The Histogram generally gives the worst results and the EM-GMM is a little better. K-Mean and KD-NN are the best methods for setting the weights in our application, whereby KD-NN performs slightly better.

6 CONCLUSIONS AND FUTURE WORK

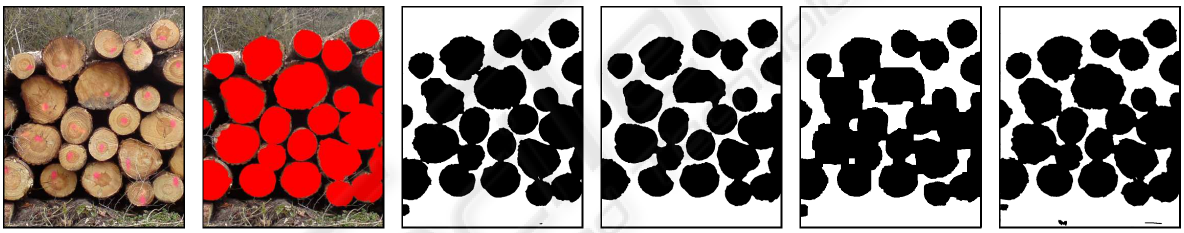
We present a novel method to segment accurately log cut surfaces in pictures taken from a stack of wood by smart phone cameras using the min-cut/maxflow framework. If certain restrictions on the image acquisition are made, the described approach is robust under different lighting conditions and cut surface colors. Robustness stems from our new, relatively simple and easy to implement density estimation. We compared our method with other approached and showed that we mostly outperformed them. Our method lead to similar results as K-Mean clustering of the color space. However, our method is faster because of the kd-tree we are using. It is also robuster against outliers, which can be a problem using K-Means cluster-

Table 1: The measured difference to the ground trues for all methods are shown here. The standard derivations are represented in brackets.

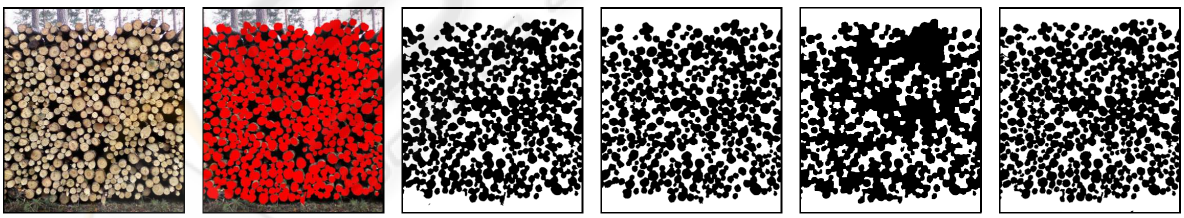
Method	number of bins	amount of cluster	false negative	false positive	false segmented
Histogram	16	-	14,62(17,04)	11,29(14,36)	25,91(14,74)
Histogram	32	-	7,8(12,33)	8,95(9,94)	16,75(12)
Histogram	64	-	3,45(5,52)	7,85(8,32)	11,3(8,23)
Histogram	128	-	2,62(3,38)	7,72(8,12)	10,34(7,57)
Histogram	256	-	2,73(2,45)	7,73(7,99)	10,46(7,2)
EM-GMM	-	2	4,28(4,44)	4,87(4,46)	9,15(4,64)
EM-GMM	-	4	4,02(4,69)	5,23(4,31)	9,25(4,92)
EM-GMM	-	8	3,72(3,91)	5,57(4,59)	9,29(4,67)
EM-GMM	-	12	3,75(3,94)	5,75(4,91)	9,5(4,87)
EM-GMM	-	16	3,63(3,87)	5,9(4,9)	9,53(4,75)
K-Mean	-	8	3,69(3,93)	4,29(3,1)	7,98(4,27)
KD-NN	-	-	2,38(3,03)	5,35(3,67)	7,73(3,94)



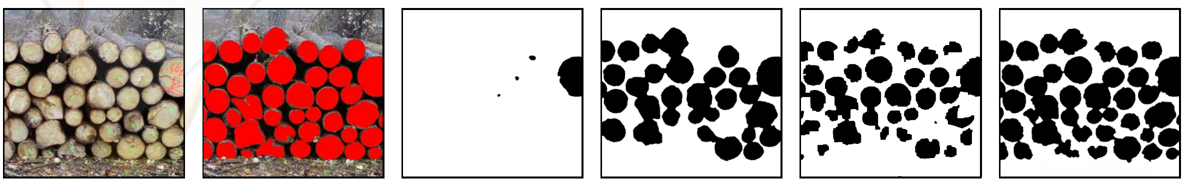
(a) Sample Image (b) Ground Truth (c) Histogram (d) K-Mean (e) EM-GMM (f) KD-NN



(g) Sample Image (h) Ground Truth (i) Histogram (j) K-Mean (k) EM-GMM (l) KD-NN



(m) Sample Image (n) Ground Truth (o) Histogram (p) K-Mean (q) EM-GMM (r) KD-NN



(s) Sample Image (t) Ground Truth (u) Histogram (v) K-Mean (w) EM-GMM (x) KD-NN

Figure 7: Segmentation of four sample images (a),(g),(m),(s), the ground truth images and the corresponding results. Thereby the best parameter were experimentally chosen for each method. For the segmentation with the histogram 32 bins per color channel were used. For the K-Mean and EM-GMM segmentation eight foreground and background clusters were applied.

ing. We used a constant search radius which works very well for our application. This radius might be need to be set slightly variable in a more general setting.

REFERENCES

Boykov, Y. and Jolly, M. P. (2001). Interactive graph cuts for optimal boundary region segmentation of object in n-d images. In *Int. C. Comput. Vision*, pages 105–112.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparision of min-cut/max-flow algorithms for energy minimation in vision. In *PAMI*, pages 1124–1137.

C. Rothar, V. K. and Blake, A. (2004). Grabcut - interactive forground extraction using iterated graph cuts. In *ACM Transactions on Graphics*, pages 309–314. ACM Press.

F. Malmberg, C. O. and Borgefors, G. (2009). Binarization of phase contrast volume images of fibrous materials - a case study. In *International Conference on Computer Vision Theory and Applications 2009*, pages 97–125.

Felzenszwalb, P. F. (2004). Efficent graph-based image segmentation. In *International Journal of Computer Vision*, pages 888–905.

Fink, F. (2004). Foto-optische erfassung der dimension von nadelrundholzabschnitten unter einsatz digitaler bildverarbeitender methoden. In *Dissertation*. Fakultaet fuer Forst- und Umweltwissenschaften der Albert-Ludwigs-Universitaet Freiburg i. Brsg.

Jaehne, B. (2005). *Digital Image Processing*. Springer Verlag, Berlin Heidelberg, 6th reviewed and extended edition edition.

Jensen, H. W. (2001). *Realistic Image Synthesis Using Photon Mapping*. The Morgan Kaufmann Series in Computer Graphics.

Orchard, M. and Bouman, C. (1991). Color quantization of images. In *IEEE Transactions on Signal Processing*, pages 2677–2690.

Otsu, N. (1979). A threshold selection method from gray-level histograms. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 62–66.

Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. The Morgan Kaufmann Series in Computer Graphics.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 888–905.