ROI-SEG: Unsupervised Color Segmentation by Combining Differently Focused Sub Results

Michael Donoser and Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology
{donoser,bischof}@icg.tu-graz.ac.at

Abstract

This paper presents a novel unsupervised color segmentation scheme named ROI-SEG, which is based on the main idea of combining a set of different sub-segmentation results. We propose an efficient algorithm to compute sub-segmentations by an integral image approach for calculating Bhattacharyya distances and a modified version of the Maximally Stable Extremal Region (MSER) detector. The sub-segmentation algorithm gets a region-of-interest (ROI) as input and detects connected regions having similar color appearance as the ROI. We further introduce a method to identify ROIs representing the predominant color and texture regions of an image. Passing each of the identified ROIs to the sub-segmentation algorithm provides a set of different segmentations, which are then combined by analyzing a local quality criterion. The entire approach is fully unsupervised and does not need a priori information about the image scene. The method is compared to state-of-the-art algorithms on the Berkeley image database, where it shows competitive results at reduced computational costs.

1. Introduction

The problem of segmentation is to partition an image into a set of non-overlapping regions. Traditionally, segmentation is formulated as bottom-up process, where no high-level knowledge about the image scene is incorporated into the algorithm. An overview of color segmentation algorithms can be found e. g. in [9]. Bottom-up approaches identify regions in the input image only based on low-level cues, like color or texture. Although such segmentation results can be achieved in an efficient way, they often do not match manual segmentations.

To overcome the limitations of low-level cues based approaches, there has recently been much interest on top down algorithms e. g. by Vasconcelos *et al*. [19], or on simultaneous combination of top-down and bottom-up approaches e. g. by Levin and Weiss [11]. But there are still many applications where a priori information is hard to obtain or is not available at all and thus, have to rely on an efficient and as good as possible bottom-up segmentation.

In this work, we present a fully unsupervised color segmentation scheme named ROI-SEG, which does not need any learning process or a priori information. The underlying idea is to compute the final segmentation as a combination of differently focused sub-segmentations of the same input image. A combinatorial approach was recently also used for segmentation by Pichel *et al*. [16] and Micusik and Hanbury [14]. Both algorithms either suffer from long computation times or from insufficient quality of the sub-segmentations and the final combination step. To overcome this problem, we propose an algorithm to calculate sub-segmentations in a fast and fully automatic way and an approach to efficiently combine the results by analyzing a segmentation quality criterion.

The rest of the paper is organized as follows. Section 2 introduces the novel sub-segmentation concept, which needs a region-of-interest (ROI) as input. The algorithm is based on efficient calculation of Bhattacharyya distances by an integral image approach combined with a modified version of the Maximally Stable Extremal Region (MSER) detector. The sub-segmentation algorithm returns a set of connected regions which all have approximately the same color distribution as the input ROI. The final segmentation result is a combination of n sub-segmentations, where each is calculated for one of n different ROIs. For unsupervised segmentation all input ROIs have to be found automatically. Section 3 introduces an efficient concept to identify ROIs representing the predominant color and texture regions in a color image. Section 4 describes the combination process which analyzes a Bhattacharyya distance based quality measure. The entire ROI-SEG framework is illustrated in Figure 1. Finally, Section 5 compares segmentation results of the ROI-SEG algorithm on images of the Berkeley segmentation database to state-of-the-art color segmentation algorithms.

1-4244-1180-7/07/$25.00 ©2007 IEEE

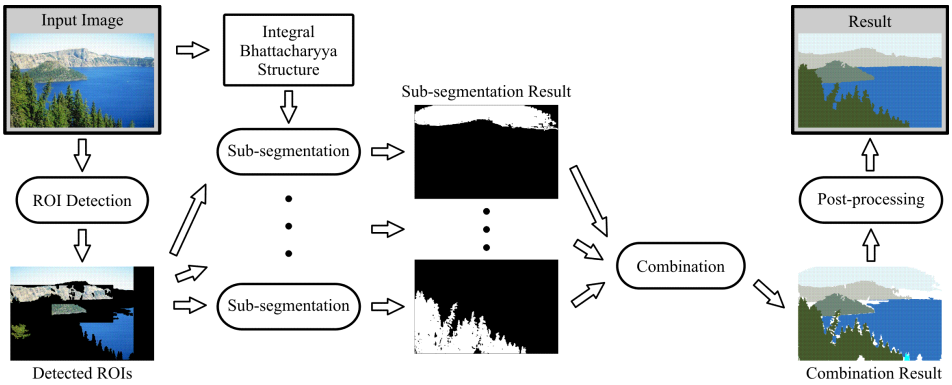


Figure 1: Illustration of unsupervised color segmentation concept. The final segmentation is a combination of sub-segmentation results, where each is based on an automatically identified region-of-interest (ROI).

2. Sub-segmentation concept

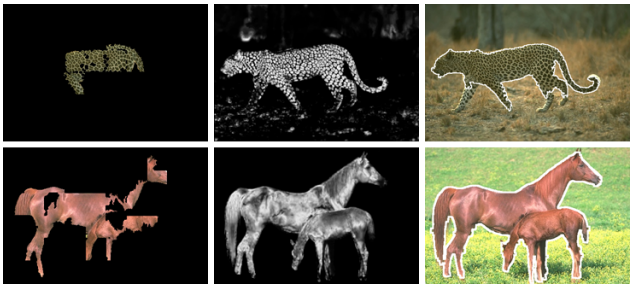
The main part of our unsupervised segmentation concept is an algorithm, which allows detection of a set of connected regions in a color image, based on a previously defined region-of-interest (ROI). The detected regions all have approximately the same color distribution as the input ROI.

The sub-segmentation algorithm, which gets an arbitrarily shaped ROI and a color image as input, can be roughly divided into four subsequent steps. First, the image is converted into the CIE Luv color space in order to have an isotropic feature space. Second, the color distribution of the ROI is modeled by a Gaussian Mixture Model (GMM), which is initialized by the results of a Mean Shift algorithm, as described in Section 2.1. Third, all color pixels are ordered by calculating Bhattacharyya distance values for each pixel. This is done by an efficient integral image based approach, which is described in Section 2.2. Finally, Section 2.3 shows how the ordered pixel values are passed to a modified version of the Maximally Stable Extremal Region (MSER) detector to compute the final result – a set of connected regions, which approximately have the same color appearance as the input ROI. Figure 2 illustrates the concept described, where (a) shows input ROIs, (b) calculated Bhattacharyya distances mapped to a grayscale range and (c) segmentation results.

2.1. Estimating ROI color distribution

The color distribution of the pixels within the input ROI is characterized by a Gaussian Mixture Model (GMM) in the Luv feature space. For a three-dimensional Luv feature vector $\vec{x}$ the mixture density for the GMM model Θ is defined as

$$p(\vec{x}, \Theta) = \sum_{c=1}^{C} \omega_c \mathrm{N}\left(\vec{\mu}_c, \vec{\Sigma}_c\right). \qquad (1)$$



(a) Region-of-interest (b) Bhattacharyya (c) Segmentation result

Figure 2: Illustration of sub-segmentation algorithm. Based on a ROI as input (a), Bhattacharyya distances are calculated (b) and passed to a modified MSER detection algorithm, which results in a set of connected regions (c).

Thus, the density is a weighted linear combination of C multivariate Gaussian distributions $N(\vec{\mu}_c, \vec{\Sigma}_c)$, where $\vec{\mu}_c$ is the 3×1 mean vector and $\vec{\Sigma}_c$ is the 3×3 covariance matrix of the c-th component. The weights ω_c describe the proportion of the c-th component and fulfill $\sum_{c=1}^{C} \omega_c = 1$. Thus, the parameter set λ of the GMM model Θ consists of $\lambda = \{\omega_c, \vec{\mu}_c, \vec{\Sigma}_c\}$, with $c = 1 \ldots C$. The parameters of the model are estimated as maximum likelihood parameters using the Expectation–Maximization (EM) algorithm [6]. The EM algorithm is an iterative approach, which refines the parameters of the model to monotonically increase the likelihood for the observed feature vectors.

One challenge of representing the probability distribution of the Luv feature vectors by a Gaussian Mixture Model is to find the number of components C and to initialize the EM algorithm. In this work, we use an idea motivated by Cho *et al.* [2] to determine the number of components automatically by a Mean Shift algorithm designed for detecting the number of significant clusters in the feature space.

The Mean Shift algorithm proposed by Comaniciu and Meer [4] is a non parametric kernel-based density estimation technique. In general, the Mean Shift concept enables clustering of a set of data-points into C different clusters, without prior knowledge of the number of clusters. It is based on the non parametric density estimator at the d-dimensional feature vector $\vec{x}$ in the feature space, which can be obtained with a kernel $K(\vec{x})$ and a window size h by

$$f(\vec{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\left\| \frac{\vec{x} - \vec{x}_i}{h} \right\|^2 \right), \tag{2}$$

where $\vec{x}_i$ are the n feature vectors of the data set. Calculating the gradient of the density estimator shows that the so-called Mean Shift defined by

$$m(\vec{x}) = \frac{\sum_{i=1}^{n} \vec{x}_i K\left(\vec{x} - \vec{x}_i\right)}{\sum_{i=1}^{n} K\left(\vec{x} - \vec{x}_i\right)}, \tag{3}$$

points toward the direction of the maximum increase in the density. The main part of the algorithm is to move the window iteratively by

$$\vec{x}^{t+1} = \vec{x}^t + m(\vec{x}^t). \tag{4}$$

It is guaranteed that the shift converges to a point where the gradient of the underlying density function is zero. These points are the detected cluster centers of the distribution and represent an estimated mean value for the cluster.

We are not interested in the mean shift clusters itself. We just run the Mean Shift procedure to find the stationary points of the density estimates, which are the modes of the distribution. These modes serve as initialization of the EM algorithm to find the maximum likelihood parameters of the GMM model.

2.2. Definition of ordering relationship

The next step is to order the pixels of the input color image. Therefore, for every pixel a unique distance value $\overline{\beta}$ between a single Gaussian distribution fitted to the Luv values within a $x \times y$ window around the pixel and the Gaussian Mixture Model (GMM) of the ROI, as obtained by the step described in Section 2.1, is calculated. While the single Gaussian distributions have to be recalculated for all of the windows located on every pixel, the GMM of the ROI stays the same during the entire computation.

The comparison between the GMM and the single Gaussian distributions is based on calculating Bhattacharyya distances [1]. The Bhattacharyya distance β compares two d-dimensional Gaussian distributions $N_1 = \{\vec{\mu}_1, \vec{\Sigma}_1\}$ and $N_2 = \{\vec{\mu}_2, \vec{\Sigma}_2\}$ by

$$\begin{aligned} \beta\left(N_1, N_2\right) = & \frac{1}{2} \ln \frac{\left| \frac{\vec{\Sigma}_1 + \vec{\Sigma}_2}{2} \right|}{\sqrt{\left|\vec{\Sigma}_1\right| \left|\vec{\Sigma}_2\right|}} + \quad \frac{1}{8} (\vec{\mu}_2 - \vec{\mu}_1)^t \\ & \left[\frac{\vec{\Sigma}_1 + \vec{\Sigma}_2}{2} \right]^{-1} (\vec{\mu}_2 - \vec{\mu}_1). \end{aligned} \tag{5}$$

The Bhattacharyya distance allows a quantitative statement about the discriminability of two Gaussian distributions by the Bayes decision rule. Therefore, for our analysis in the three-dimensional CIE Luv feature space, the Bhattacharyya distance represents a quantitative statement if two color distributions are likely to be equal or not.

To be able to compare a single Gaussian distribution to a GMM, C different Bhattacharyya distance values to every component of the GMM have to be calculated. Then, the final distance value $\overline{\beta}$ is computed by

$$\overline{\beta} = \sum_{c=1}^{C} \omega_c \, \beta\left(N_c, N_w\right), \tag{6}$$

where ω_c is the c-th GMM weight, $N_c = \{\vec{\mu}_c, \vec{\Sigma}_c\}$ denotes the c-th component of the GMM and $N_w = \{\vec{\mu}_w, \vec{\Sigma}_w\}$ is the single Gaussian fitted to the window pixels.

To be able to calculate a Bhattacharyya distance for every pixel, the window has to be moved all over the image. For every window location the mean and the covariance matrix of the corresponding Luv values within the window have to be computed which is a time-consuming process. Therefore, we introduce an adaption of the Summed-Area-Table (SAT) approach to efficiently calculate the Bhattacharyya distances. The SAT idea was originally proposed for texture mapping and brought back to the computer vision community by Viola and Jones [20] as integral image.

In general, the integral image $Int(r, c)$ is defined for a gray scale input image $I(x, y)$ by

$$Int(r, c) = \sum_{x \leq r, y \leq c} I(x, y), \tag{7}$$

as the sum of all pixel values inside the rectangle bounded by the upper left corner.

Tuzel *et al.* [18] proposed an efficient method to calculate covariances based on the integral image concept. They build d integral images P_i for the sum of the values for each dimension and $d * (d + 1)/2$ images Q_{ij} for each product between the values of any two dimensions, where d is the number of dimensions of the feature space.

Thus, the integral images P_i, with $i = 1 \ldots d$, are defined by

$$P_i(r, c) = \sum_{x \leq r, y \leq c} I_i(x, y), \tag{8}$$

where I_i is the value of the i-th dimension of the input image. The product of the values of any two dimensions Q_{ij}, with $i, j = 1 \dots d$, is calculated by

$$Q_{ij}(r,c) = \sum_{x \leq r, y \leq c} I_i(x,y) * I_j(x,y). \quad (9)$$

Based on the three integral images P_i the mean vector $\vec{\mu} = \{\mu_1, \dots \mu_d\}$ for any window defined by the upper left and the lower right coordinates p^{ul} and p^{lr} can be calculated by

$$\mu_i \left(p^{ul} : p^{lr}\right) = P_i^{ul} + P_i^{lr} - P_i^{ur} - P_i^{ll}, \quad (10)$$

where P_i^{ul} is the value of P_i at the upper left coordinate of the window and P_i^{lr}, P_i^{ur} and P_i^{ll} denote the values for the lower right, upper right and lower left coordinates, respectively.

The covariance matrix $\vec{\Sigma}(p^{ul} : p^{lr})$ for the same window can be calculated by

$$Q_{sub}(i,j) = Q_{ij}^{ul} + Q_{ij}^{lr} - Q_{ij}^{ur} - Q_{ij}^{ll}, \quad (11)$$

$$P_{sub}(i) = P_i^{ul} + P_i^{lr} - P_i^{ur} - P_i^{ll}, \quad (12)$$

$$\vec{\Sigma}(p^{ul} : p^{lr}) = \frac{1}{m-1}\left(\vec{Q}_{sub} - \frac{1}{m}\vec{P}_{sub}\vec{P}_{sub}{}^T\right), \quad (13)$$

where m is the number of pixels within the window. Thus, for calculation of the Bhattacharyya distances, nine different integral images are needed. Please note, that these integral images only have to be calculated once, even if we want to perform comparisons to different GMMs, as described in Section 4.

2.3. MSER detection

The last step of the sub-segmentation concept is to detect a set of connected regions by analyzing the calculated Bhattacharyya distances. This step is based on a modified version of the Maximally Stable Extremal Region (MSER) concept from Matas *et al*. [13]. The MSER detector is one of the best interest region detectors in computer vision as e. g. shown by Mikolajczyk *et al*. [15].

The underlying idea is to apply the MSER detector to the Bhattacharyya distances which allows the integration of color information into the detection process. In general, MSERs are defined for connected, weighted graphs, i. e. for gray scale images. To make the MSER detection applicable for color images, a unique ordering of the color pixels has to be provided. We propose to use the Bhattacharyya distances to define the ordering and to use them as weights in the MSER detection algorithm.

MSERs are detected by analyzing the levels L_ω of a connected, weighted graph. A specific level L_ω contains the set of nodes that have a weight above a given threshold ω. The groups of connected nodes within each level are called extremal regions R_i, because they all fulfill

$$\forall p \in R_i \, , \forall q \in boundary(R_i) \rightarrow W(p) \geq W(q), \quad (14)$$

where p and q are pixels within the input image, $W(p)$ is the corresponding weight and $boundary(R_i)$ is the set of all boundary pixels of the region. In contrast to the original MSER detector we work on continuous valued weights – the Bhattacharyya distances – which we have to discretize. This is done by constraining the analysis on levels at constant distance steps Δ_{Bhatt}. Then, MSERs are detected as those extremal regions R_i within the levels L_ω with $\omega = i * \Delta_{Bhatt}$ and $i \in [0, 1, 2 \dots]$, that have a local stability minimum Ψ defined by

$$\Psi(R_i) = \frac{|R_j^{\omega - \eta\Delta_{Bhatt}}| - |R_k^{\omega + \eta\Delta_{Bhatt}}|}{|R_i^\omega|}, \quad (15)$$

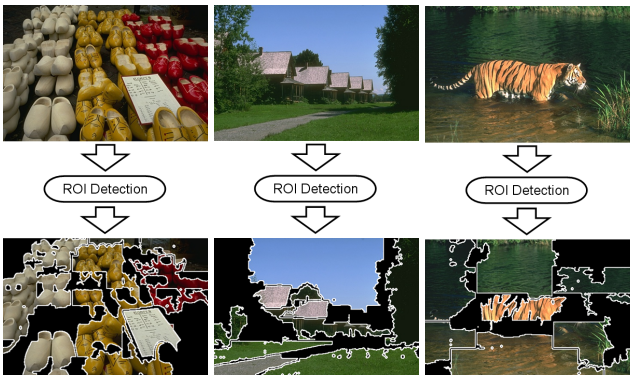
where $|.|$ denotes the cardinality, R_i^ω is a region which is obtained in a level at weight ω and η is a stability range parameter. $R_j^{\omega - \eta\Delta_{Bhatt}}$ and $R_k^{\omega + \eta\Delta_{Bhatt}}$ are the extremal regions that can be found at the same location within the levels at weights $\omega - \eta\Delta_{Bhatt}$ and $\omega + \eta\Delta_{Bhatt}$. Thus, the extremal regions with the highest stability (MSERs) are those regions which have approximately the same size over a level range of $2\eta\Delta_{Bhatt}$.

MSERs can be calculated in a very efficient way by using the component tree as data structure as shown by Donoser and Bischof [8]. In addition, we are only interested in detecting MSERs at very low weights, i. e. regions that have a color distribution that is approximately similar to the one of the input ROI. Thus, constraining the search on low weights makes the MSER computation even faster.

3. Automatic ROI detection

The concept presented in Section 2 enables the detection of connected regions based on a previously defined region-of-interest (ROI). Such a ROI can be initialized manually, e. g. if we are interested in a specific region of the image. Of course, for unsupervised segmentation a set of input ROIs has to be provided automatically. This section introduces a method to detect ROIs representing the predominant color and texture distributions of a color image.

Please note, that this is not a critical step of the unsupervised concept and that any other algorithm can be used for the automatic ROI detection. The number of detected ROIs only influences the computational time and has almost no effect on the final quality of the results, because bad initializations will be sorted out in the combination step, as described in Section 4. Because of the efficient design of the



(a) Nine ROIs (b) Seven ROIs (c) Eight ROIs

Figure 3: Automatic ROI detection results.

other parts of the concept, we propose a fast method which allows detection of a low number of ROIs in short time.

The first step of the ROI detection concept is to partition the input color image into n equal sized blocks. For each of these blocks a K-Means algorithm is applied in the five dimensional space consisting of the three Luv values and the two spatial coordinates of each pixel. To achieve consistent results the K-Means is always initialized by the the first k data points and the ordering of the data points is fixed. The clustering result is mapped back to the image and holes are filled. The result of this step is a segmentation of the input image into $n * K$ clusters. Different values for K influence the ROI detection quality, but experimental evaluation showed that on average $K = 3$ yields the best results. The ROIs are then identified by applying a region merging approach to the results of the K-Means algorithm. The merging step is done by analyzing the $n * K$ mean Luv values of all clusters by Mean Shift clustering, as described in Section 2.1. The Mean Shift clusters the regions into $m \leq n * K$ new clusters, thus, some regions are merged. These m detected clusters finally represent an estimate of the predominant color and texture distributions within the input image. The regions identified are then eroded to avoid border effects, and the smaller ones are removed.

The result of the ROI detection algorithm is a set of connected regions. As one can see from the examples shown in Figure 3, the predominant color distributions are found quite well.

4. Combination of sub-segmentation results

The sub-segmentation concept presented in Section 2 is applied to all of the ROIs detected with the algorithm presented in Section 3. Each input ROI provides a segmentation of the input image into a number of connected regions. As last step of our concept, these results have to be combined into the final segmentation result.

To be able to combine the sub-segmentation results, we define a local criterion, which evaluates the segmentation quality for each detected region in every result. We propose to analyze the corresponding Bhattacharyya distances to evaluate the quality of each segmentation. The lower the Bhattacharyya values the more similar the compared distributions are, which is a sign of good segmentation quality. Thus, the mean Bhattacharyya distance of all pixels within a detected region is used to distinguish between a high and a low quality result.

We calculate the mean Bhattacharyya value for every detected region within each sub-segmentation result. Then we assign the label of the region with the locally lowest mean Bhattacharyya distance to every pixel of the input image, which provides a first segmentation result. Some regions will still be unassigned, as can be seen in Figure 1 and some post-processing is necessary. If the unassigned region is bigger than a fixed threshold one can assume that this area covers a non–initialized ROI of the image. Then, another sub-segmentation is performed and added to the set of sub-results. All other regions are assigned to those region in its neighborhood with the most similar mean Luv value.

5. Experimental evaluation

Our ROI-SEG algorithm is benchmarked against state-of-the-art color segmentation algorithms based on images from the Berkeley database [12]. Since our algorithm does not include a priori information about the image scene or any learning algorithms, we only use the 200 images of the test data set. Figure 4 shows segmentation results of our algorithm for 15 selected images. The images chosen are the same as in [14] in order to allow a direct comparison to one of the state-of-the-art methods.

For quantitative comparison of segmentation results obtained with different algorithms we analyze the Global Consistency Error (GCE). The GCE requires calculation of a Local Refinement Error (LRE) for every pixel p_i by

$$LRE(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \backslash R(S_2, p_i)|}{|R(S_1, p_i)|}, \qquad (16)$$

where S_1 and S_2 are two different segmentation results, p_i is one pixel of the image, $|.|$ denotes cardinality, $R(S_j, p_i)$ is the connected region to which pixel p_i belongs to in segmentation S_j and $R(S_1, p_i) \backslash R(S_2, p_i)$ is the difference between two regions. The GCE is a single value and is calculated by

$$GCE(S_1, S_2) = \frac{1}{m} \min \left\{ \sum LRE(S_1, S_2), \sum LRE(S_2, S_1) \right\}, \qquad (17)$$

where m is the number of pixels in the image. The GCE allows to make a quantitative statement about the similarity between two segmentation results S_1 and S_2.

Figure 4: Segmentation results of our algorithm on images of Berkeley database. The same images as in [14] are chosen, to allow a direct comparison to one of the state-of-the-art algorithms.

Based on the GCE we have compared our results to six state-of-the-art color segmentation algorithms: the JSEG algorithm from Deng and Manjunath (*Jseg*) [7], the Mean Shift method (*Mshift*) from Comaniciu and Meer [3], the standard normalized cut algorithm (*Ncuts*) from Shi and Malik [17], the multi-scale normalized cut approach (*Mscuts*) from Cour *et al*. [5], the seeded graph cuts method (*Seed*) from Micusik and Hanbury [14] and the pixel affinity based method (*Affin*) from Fowlkes *et al*. [10]. All results were calculated with publicly available implementations, with the exception of *Affin* and *Seed*, where the results are taken directly from the paper [14].

At least five human segmentations, which represent the ground truth for this benchmark, are available for each of the 200 images from the Berkeley database. In the first step, for every algorithm 200 mean GCE values denoted as $\overline{GCE}_n$, with $n = 1 \ldots 200$, were calculated by

$$\overline{GCE}_n = \frac{1}{H_n} \sum_{i=1}^{H_n} GCE\left(Seg_n, Hum_n^i\right), \qquad (18)$$

where H_n is the number of available human segmentations for the n-th image, Seg_n is the segmentation result of one of the seven algorithm and Hum_n^i is the i-th human segmentation of the n-th image. Then, for each algorithm an overall quality measure $\overline{GCE}$ was calculated by

$$\overline{GCE} = \frac{1}{200} \sum_{n=1}^{200} \overline{GCE}_n. \qquad (19)$$

To be able to measure the consistency of the human segmentations itself, we computed GCE values of all possible combinations between all available human segmentations for every image, which additionally allows calculation of an overall mean $\overline{GCE}$ for the human results. The obtained value was 0.0783, which is an indication for good consistency of the human segmentations.

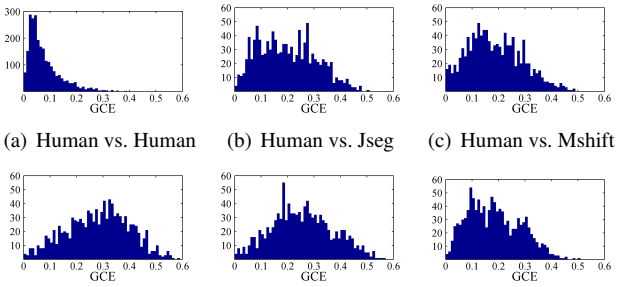
Table 1 summarizes the comparison of the aforementioned algorithms using the quality measure $\overline{GCE}$. As one can see, *Ncuts* and *Mscuts* exhibit the highest values indicating poorer performance compared to the other algorithms, while our method yields the lowest value. In addition, our algorithm provides on average 18 regions per segmentation, which is close to the human average.

To further compare the performance of different algorithms, many publications, as e. g. [14] calculate the $GCEs$ to all available human segmentations and summarize these values in a histogram, as shown in Figure 5. Again, it seems that the algorithms approximately perform the same, except the *Ncuts* and the *Mscuts* algorithms, their histograms are located more at the right side.

The histograms shown in Figure 5 do not allow a deeper insight into divergences between segmentation results obtained with different methods, since the correspondence between the calculated GCE and the input image is lost. In order to overcome this weakness, we compare our approach directly to other algorithms using scatter plots, as shown in Figure 6. Each point in these plots represents one of the 200 input images, where the x-value is the $\overline{GCE}_n$ of our method and the y-value is the $\overline{GCE}_n$ of the algorithm currently compared. Points on the diagonal line $y = x$ indicate identical deviation from the ground truth for both algorithms, while the distance to the line is a measure for the disagreement between the obtained segmentation results.

	Human	*Jseg*	*Mshift*	*Ncuts*
GCE	Reference	0.1996	0.1870	0.2811
# regions	17	36	20	7
	Mscuts	*Affin*	*Seed*	***Our***
GCE	0.2557	0.214	0.209	**0.1840**
# regions	7	13	4	**18**

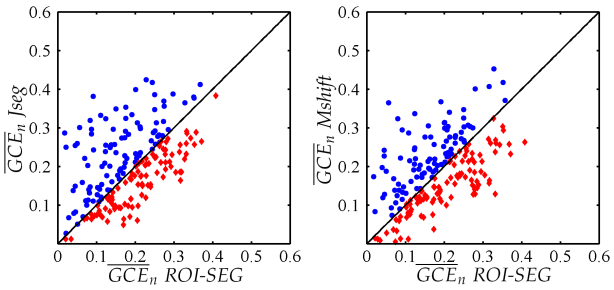
Table 1: Comparison of state-of-the art methods. It shows the quality measure $\overline{GCE}$ and the average number of detected regions. The $\overline{GCE}$ values for the *Affin* and *Seed* method are taken directly from [14].



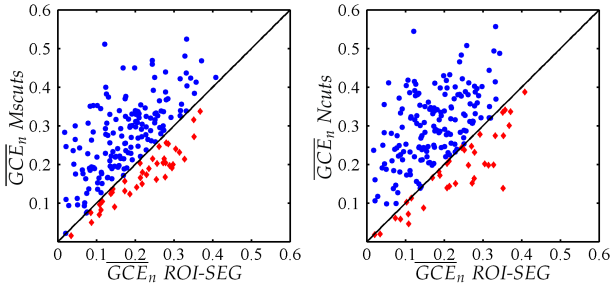
(a) Human vs. Human (b) Human vs. Jseg (c) Human vs. Mshift

(d) Human vs. Ncuts (e) Human vs. Mscuts (f) Human vs. Our

Figure 5: Comparison of state-of-the-art algorithms by histograms showing the GCE distribution computed on 200 Berkeley images.



(a) Jseg vs. ROI-SEG (b) Mshift vs. ROI-SEG



(c) Mscuts vs. ROI-SEG (d) Ncuts vs. ROI-SEG

Figure 6: Comparison of segmentation performance using GCE scatter plots. Points above the diagonal line represent images, where ROI-SEG performs better. This discrimination is emphasized by different colors in the scatter plots.

Points above the diagonal line represent images, where our algorithm performs better, whereas points below indicate superior performance of the other algorithm. This discrimination is emphasized by different colors in the scatter plots. As can be seen in Figures 6(c) and 6(d), the two normalized cut based approaches perform inferior to our approach, since most of the points are above the diagonal line. Although the *Jseg* algorithm has a almost similar $\overline{GCE}$, the scatter plot in Figure 6(a) reveals that the points above the diagonal line are on average farther away from the line than the points below. This is an indication that for some images *Jseg* performs distinctively poorer as our approach. *Mshift* in Figure 6(b) achieved almost similar performance as our algorithm. In order to gain a deeper insight into the main differences between these two algorithms, it is essential to identify images depicting scenes which cause different, respectively the same, levels of performance. The required information is available in the scatter plots, because every point can be matched to the corresponding image. Therefore, the scatter plot from Figure 6(b) is analyzed in more detail in Figure 7, which shows the corresponding images of key data points located in the four corners of the plot.

The data points in the lower left corner of Figure 7 represent images, where both algorithms achieved almost similar agreement with the ground truth. The three key images shown, reveal that these are images, which can be easily segmented in distinct regions. The upper right corner contains images, where both results exhibit a large difference to the ground truth. These are images with hard-to-segment, complex scenes. On the other hand, the upper left and the lower right images are the most interesting ones, because they cause a large difference in the performance of the two algorithms. As can be seen in Figure 6(b), there is a larger number of data points in the upper left corner compared to the lower right, which indicates that there are more images, where our algorithm performs distinctively better. But in contrast, for the easy-to-segment images in the lower left corner, the *Mshift* mostly performs better, because its detected region boundaries are more accurate than ours.

To sum up, our ROI-SEG algorithm shows competitive results to state-of-the-art segmentation algorithms. Additionally, because of the efficient design of the individual concepts our algorithm provides results in short computation times, outperforming the other algorithms. The detection of the input ROIs is done within a second and every sub-segmentation needs about half a second for the Berkeley images. Thus, segmentation results can be achieved in a few seconds.

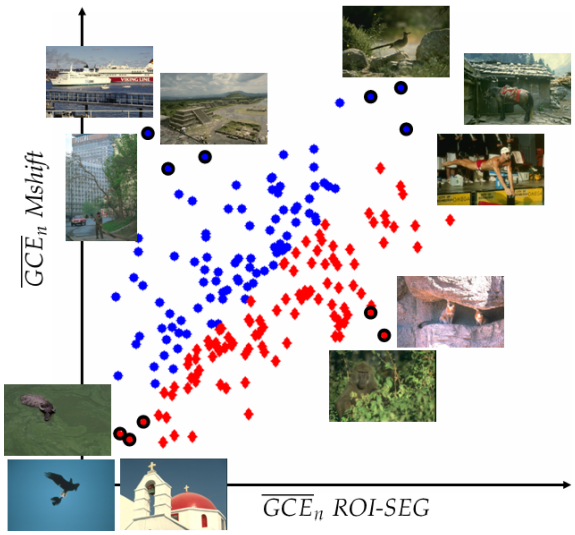


Figure 7: $\overline{GCE}_n$ scatter plot of Mean Shift versus the proposed algorithm including key images.

6. Conclusion

This paper introduced a novel unsupervised segmentation scheme named ROI-SEG, which is based on the main idea of combining a set of differently focused sub-segmentation results. Because of the efficient design of the individual parts of the concept, segmentation results can be achieved within a few seconds. Experimental evaluation on the Berkeley image database showed visually appealing results. In addition, a quantitative evaluation proved, that ROI-SEG can keep up with state-of-the-art algorithms and is calculated with reduced computational costs.

Acknowledgements Part of this work has been carried out as part of the K-plus Competence center ADVANCED COMPUTER VISION funded under the K plus program. Additionally, the authors gratefully acknowledge financial support from Mondi Business Paper Hausmening, Mondi Packaging Frantschach, M-Real Hallein, Norske Skog Bruck, Sappi Gratkorn, SCA Laakirchen, UPM Steyrermuehl, Voith Paper and the Austrian Industrial Research Promotion Fund FFG.

References

[1] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943.

[2] W. Cho, J. Park, M. Lee, and S. Park. Unsupervised color image segmentation using mean shift and deterministic annealing EM. *Proc. of Conf. on Computational Science and Its Applications*, pages 867–876, 2005.

[3] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *Proc. of Conf. on Comp. Vision and Pattern Rec.*, pages 750–755, 1997.

[4] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[5] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *Proc. of Conf. on Comp. Vision and Pattern Rec.*, pages 1124–1131, 2005.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[7] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

[8] M. Donoser and H. Bischof. Efficient maximally stable extremal region (MSER) tracking. In *Proc. of Conf. on Comp. Vision and Pattern Rec.*, pages 553–560, 2006.

[9] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2002.

[10] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: combining patch-based and gradient-based approaches. In *Proc. of Conf. on Comp. Vision and Pattern Rec.*, pages 54–61, 2003.

[11] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *Proc. of European Conf. on Comp. Vision*, 2006.

[12] 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. of 8th Int. Conf. on Comp. Vision*, pages 416–423, 2001.

[13] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. of British Machine Vision Conf.*, pages 384–393, 2002.

[14] B. Micusik and A. Hanbury. Automatic image segmentation by positioning a seed. In *Proc. of the European Conf. on Comp. Vision*, volume 2, pages 468–480, 2006.

[15] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. Journal of Comp. Vision*, 65(1-2):43–72, 2005.

[16] J. C. Pichel, D. E. Singh, and F. F. Rivera. Image segmentation based on merging of sub-optimal segmentations. *Pattern Recognition Letters*, 27(10):1105–1116, 2006.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[18] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *Proc. of European Conf. on Comp. Vision*, pages 589–600, 2006.

[19] M. Vasconcelos, N. Vasconcelos, and G. Carneiro. Weakly supervised top-down image segmentation. In *Proc. of Conf. on Comp. Vision and Pattern Rec.*, pages 1001–1006, 2006.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of Conf. on Comp. Vision and Pattern Rec.*, pages 511–518, 2001.