Converting RGB Volume Data to Scalar Fields for Segmentation Purposes

Tetyana Ivanovska[†] Lars Linsen[‡]
School of Engineering and Science
International University Bremen[*]
Bremen, Germany

ABSTRACT

Most medical scanning techniques generate scalar fields, for which a large range of segmentation algorithms exists. Some scanning techniques like cryosections, however, generate color data typically stored in *RGB* format. Since standard segmentation algorithms such as isosurface extraction, level-set and region growing methods all have their advantages and drawbacks and many extensions and specializations of the algorithms have been developed to solve specific problems, one would need to generalize all these approaches to color data to have the full range of algorithmic solutions at hand. A more viable way to proceed is to convert the color data field to a scalar field in a preprocessing step, which allows for the direct application of all above-mentioned segmentation approaches. We propose a procedure that converts color to scalar data while preserving the properties that are important for segmentation purposes. We first convert the colors from *RGB* to $L^*a^*b^*$ color space, which separates the luminance channel from the chrominance channels and distributes the chrominance with respect to human perception. Then, we cluster the colors present in the data using a number of approaches and discuss the advantages and drawbacks. In order to assign to each cluster an appropriate scalar value, we use the ideas of the recently presented *Color2Gray* algorithm and generalize it for application to volume data. The Color2Gray algorithm in its originally proposed form is too inefficient to be applied to volume data, but a restructuring of the algorithm coupled with a prior clusterization step allows us to apply the algorithm even to large volume data. We segment the resulting scalar field using standard segmentation algorithms and discuss our results in comparison to standard conversion results.

Keywords: Color-to-scalar conversion, clusterization, segmentation.

1 INTRODUCTION

Visualizing medical imaging data is one of the traditional tasks in scientific visualization. The in vivo medical scanning techniques typically generate stacks of 2D grayscale images, where the grayscale values represent, for example, the tissues' densities. From the stack of images a 3D scalar field can be reconstructed. Examples of such medical imaging techniques are CT, MRI, PET, etc. More thorough examinations can be made when using ex vivo scanning techniques. The most prominent example would be cryosections. When generating cryosections the individual slices are scanned using digital photography. Thus, the resulting images are not grayscale but colored. Typically, they are stored in *RGB* color space. Reconstruction of the volumetric data set leads to a 3D *RGB* color field.

Two main directions can be distinguished in the context of 3D medical imaging visualization, namely direct volume rendering and segmentation. While direct volume rendering displays the 3D data allowing for interactive change of the viewing parameters, segmentation extracts geometry that can be used for rendering purposes as well as for quantitative measurements and further processing. The main objective of the segmentation process is to extract boundary surfaces of certain objects, separating them from the surrounding tissues. In medical terms, one would like to extract the 3D geometry of the scanned organs. Standard segmentation algorithms include isosurface extraction, level-set methods, or region-growing approaches. These segmentation algorithms (as well as most direct volume rendering approaches) operate on 3D scalar fields. Generalization to color data is often not straihgt forward and only few attempts have been taken. Since different approaches are more or less suitable for different segmentation purposes, one would like to have the whole range of all segmentation algorithms generalized to color data. A more viable way to achieve this goal is to convert the color data into an appropriate scalar field without losing the property of being able to segment distinguishable regions.

While in the imaging community many sophisticated *RGB*-to-grayscale conversion algorithms exist, in visualization one mostly converts the *RGB* values to a color

[*]Jacobs University Bremen as of spring 2007
[†]t.ivanovska@iu-bremen.de
[‡]l.linsen@iu-bremen.de

model with a separate luminance channel and uses the luminance channel for segmentation purposes. Obviously, the creation of a 3D luminance field leads to a loss in data information. In particular, two colors with same luminance but different chrominance are mapped to the same scalar value. Thus, any automatic segmentation algorithm would fail in separating two adjacent regions that are colored with the respective two colors. Any linear mapping that projects the 3D color space onto one axis (even chrominance-based approaches or Principal Component Analysis) would suffer from this drawback. We present a procedure for converting *RGB* color fields to scalar fields under the consideration that the scalar field will be used for segmentation purposes. Thus, we would like to distinguish all significant colors (especially when they occur in adjacent regions) while maintaining luminance order for colors with similar chrominance values. Our general approach is described in Section 3.

Our conversion method is based on two steps. In a first step, we analyze the color data in terms of occurring colors, the number of occurrences, and their distribution in the color space. We use the $L^*a^*b^*$ color space, since it separates the luminance channel from the chrominance channels and distributes the chrominance values such that the Euclidean norm in the $L^*a^*b^*$ space approximately captures perceptual dissimilarity. Exploiting the color distribution in the $L^*a^*b^*$ space for a given data set, we cluster the colors into n regions, where n is the predefined cardinal number of the range of the resulting scalar field.

We use several approaches for clusterization. One of them is based on a genetic algorithm. It produces excellent results but has high computational costs and, therefore, is not applicable to larger 3D data sets. We use it to validate with it other clusterization algorithms, e. g. axes-aligned binary space splitting, k-means, c-means, and median cut. The details are described in Section 4.

In the second step, a representative of each cluster has to be mapped to an appropriate scalar value. For this mapping we use the ideas of the recently presented *Color2Gray* approach [GOTG05]. The Color2Gray algorithm uses three parameters that determine the mapping in terms of luminance and chrominance. We can set these parameters such that the mapping has the desired before-mentioned properties.

The main drawback of the Color2Gray algorithm is its asymptotic computation time that is $O(N^6)$ for a $N \times N \times N$ volume data set. This runtime does not make it practical for 3D applications. However, when restructuring the algorithm and applying it to the clustered colors instead of the original image colors, we can reduce the computation time to $O(k^2)$, where k is the number of clusters, e. g. $k = 256$ for byte-sized output. Details are given in Section 5.

In Section 6, we give results that document the efficacy of our method. In particular, we compare them to luminance-based conversion and to other clusterization techniques common in the imaging community. We apply standard segmentation techniques to illustrate the advantages of our method when coupling the conversion with segmentation procedures.

2 RELATED WORK

The standard mapping for a color-to-grayscale transformation is the projection of the colors of a color image to the gray axis in the respective color space. Thus, color is mapped to its luminance. Such a method is inadequate for the pixels with same luminance but different chrominance. Several methods have been proposed for solving the general problem of reducing an n-dimensional set of data to m dimensions, where $m < n$. Since all standard color spaces have three dimensions, $n = 3$ and $m = 1$ in our case. Principal Component Analysis (PCA) is one of such methods [Jol02]. PCA can be used for computation of an ellipsoid in color space (principal components). Color values in the image can then be projected on a (luminance) axis defined by the primary axis of this ellipsoid. The efficacy of this method depends on the color space. However, PCA can also suffer from the problem of projecting colors of different chrominance to the same position an the axis.

Another way of generating grayscale images out of color images is to match local contrast in color images [SB02]. The contrast is regarded as a gradient and then the grayscale is recovered by solving a Poisson equation. This method has difficulties with certain classes of images as the global contrast influence is avoided. Moreover, local approaches do not work for our purposes, as one color may be mapped to different scalar values in different regions.

Another recently presented method maintained the proportionality between perceived color difference and perceived luminance difference, but ignores spatial arrangement of pixels [RGW05].

The goals of the listed approaches are indeed diverse, but none of them is targeted towards a subsequent segmentation step. Our approach instead aims for the necessary properties, which are to distinguish all significant colors and to maintain luminance order for colors with comparable chrominance values. We use a generalization of the recently developed two-dimensional Color2Gray algorithm [GOTG05] coupled with a clusterization / quantization procedure.

We have analysed a number of quantization algorithms which could be used for the prior quantization of the initial image before using the Color2Gray. The objective of color quantization is displaying a full color image with a restricted set of representative colors without a significant, i. e. perceptually almost not noticeable, loss of color impression. Colors are to be approx-

imated as closely as possible when quantized. Quantization techniques consider quality criteria such as human perception, computation time, and memory requirements [Sch97]. We have investigated the following existing quantization algorithms: static color look-up algorithm using look-up tables [Hec82], popularity algorithm [Hec82], median cut algorithm [Hec82], k-means algorithm [Mac67], fuzzy c-means algorithm [Bez81].

The idea of static color look-up table algorithms is to divide the color cube into equally thick slices in each dimension. The crossproduct of these color levels can be used as the entities of the color look-up table. A significant drawback of this method are artifacts in form of edges in the resulting image. The main idea of popularity algorithms is to build a colormap by finding the K most frequently occurring colors in the original image. The colors are stored in a histogram. These K most frequently occurring colors are extracted and used as entries in the color table. The image is quantized with respect to that table. The question that remains is how to map the colors that appear in the original image but are not stored in the color table. A method that detects the most frequently used color out of the colors stored in the color table within a small neighborhood of the regarded pixel. Thus, in general, each pixel has to be tested to find the shortest distance to one of the K most frequently used color values. The main drawback of this method is that some important but "unpopular" image colors could be lost. The median-cut method was originally described by P. Heckbert [Hec82]. The idea behind it is to use each of the color in a synthesized look-up table to represent an equal number of pixels of the original image. The algorithm partitions the color space iteratively into subspaces of decreasing size. The algorithm starts with an axes-aligned bounding box that encloses all the different color values present in the original image. The "size" of the box is given by the minimum and maximum of each of the color coordinates that encloses the current box. For splitting the box one determines the dimension, in which the box will be (further) subdivided. The splitting is executed by sorting the points by increasing values in the dimension, where the current box has its largest edge and partitioning the box into two halves at the position of the median. Approximately equal numbers of points are generated on each side of the cutting plane. Splitting is applied iteratively and continued until K boxes are generated. The number K may be chosen to be the maximum number of color entries in the available colormap. The color assigned to each of the K boxes is calculated by averaging the colors of each box. The median-cut method performs well for pixels, whose colors lie in a high-density region of the color space, where repeated divisions resulted in cells of small size and, hence, small color errors. However, colors that fall in low-density regions of the color space are within large cells, where large color errors are to be expected. The main idea of the k-means algorithm is to define K centroids, one for each cluster. These centroids should be placed as far from each other as possible. Then each point from the initial data set is associated with the nearest centroid. When all the points have "their" centroids, the K centroids are recalculated as the average centers of each cluster.Then a new binding has to be done between the same data set points and the nearest new centroid. This loop is continued until no more changes are done. The idea of the fuzzy c-means method is similar to the k-means approach, but it allows one data point to belong to two or more clusters. Fuzzy partitioning is carried out through an iterative optimization of the data points membership in the clusters and the corresponding update of the cluster centers. The iteration terminates when there is no difference between the membership results with respect to some given precision. The results of the k-means and c-means approaches depend on the choice of the initial centroids. We take as initial centroids the data points that are distributed as far as possible from each other in the color space.

3 GENERAL APPROACH

We present a procedure for converting color data to a scalar field in a way that is amenable for subsequent segmentation of the volume. Any segmentation technique may be applied to the resulting scalar field.

Our conversion approach consists of three main steps. Assuming that the 3D color data is given in form of *RGB* data, we first convert the *RGB* values to a color representation in the $L^*a^*b^*$ color space. Secondly, we apply a quantization step that reduces the number of used colors in the given color data from their original number to a typically much smaller number of colors in the same color space. The reduced amount of colors is chosen with respect to the number of distinguishable output values in the to be generated scalar field. When considering a scalar field that allows us to store 1 byte information per sample point, we would set the reduced number of colors for quantization to 256. The quantization step assures that all important colors can still be distinguished after quantization. The quantization step also generates a unique mapping of each color of the original color set to one color of the reduced color set. Finally, the reduced set of colors is mapped to scalar values in a way that luminance order is preserved for colors with similar chrominance values.

Operating in *RGB* space is inadequate for our purposes, since it does not distinguish between luminance and chrominance. Several color models such as *HLS* or *HSV* have this property and are widely used for this particular reason. However, we decided to use the $L^*a^*b^*$ color space, because its Euclidean norm closely correspond to perceptual dissimilarity [Pas03]. *CIE* $L^*a^*b^*$

is the most complete color model used conventionally to describe all the colors visible to the human eye. It was developed for this specific purpose by the International Commission on Illumination or Commission Internationale d'Eclairage (CIE). The three parameters in the model represent the luminance L of the color, where $L = 0$ yields black and $L = 100$ indicates white, its position a between red and green, where negative values indicate green while positive values indicate red, and its position b between yellow and blue, where negative values indicate blue and positive values indicate yellow.

Assuming that the original data is given in *RGB* color space, we need to convert the colors to a representation in the $L^*a^*b^*$ color space. To do so, we first transform the *RGB* data to the *CIE XYZ* color space and, afterwards, convert the *XYZ* values to $L^*a^*b^*$ colors. Note that the matrix of transformation from *RGB* data to *XYZ* depends on the chosen *RGB* standard. We consider the *R*709 *RGB* standard. Hence, the three channels of the $L^*a^*b^*$ colors are computed by

$$
\begin{aligned}
L^* &= \begin{cases} 116 \cdot \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16, & \text{if } \frac{Y}{Y_n} > 0.008856 \\ 903.3 \cdot \left(\frac{Y}{Y_n}\right), & \text{otherwise} \end{cases}, \\
a^* &= 500 \cdot \left(\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} \right), \text{ and} \\
b^* &= 200 \cdot \left(\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}} \right),
\end{aligned}
$$

where X_n,Y_n,and Z_n are the values of X,Y,and Z, respectively, for a specified reference of the white, i. e. illuminant, color, and X, Y, and Z are computed by

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}
$$

For the quantization step, we first choose the number K of clusters to be generated, which is equal to the number of distinguishable values in the outputted scalar field. If the number of unique colors in the image is less than the available output values, the quantization step can be skipped. Each color defines its own cluster. However, for real data we are dealing with, this case does never occur in practice when using byte-sized output values. Since we are computing a scalar field rather than a greyscale image, we are not limited to typical image format sizes. Any number of output clusters can be chosen. An increasing number of output clusters only affects the performance of the subsequent conversion algorithm.

Given the number K of output colors, we execute some clusterization algorithm in order to generate K clusters of colors and determine the representative colors for each cluster. The clusterization algorithms we have developed are described in Section 4.

The actual quantization of the picture is performed by substitution the initial color value by the value of the representative of the cluster, to which the initial color value has been assigned. The output of this step is a color data set in $L^*a^*b^*$ color space that only makes use of a restricted number of K different colors.

Finally, K representative colors of the clusters are to be mapped to K scalar values. For this purpose we use the Color2Gray algorithm. However, we do not apply the Color2Gray algorithm directly, as the computational costs are too high for running this algorithm on standard-sized 3D data sets. Instead, we apply the mapping idea of the Color2Gray algorithm to the K representative colors of the cluster. Since the Color2Gray algorithm can be fine-tuned by three intuitive parameters, we examine, which of these parameters would be most suitable for our purposes, such that all significant parts in the data are distinguishable. Details on this last step are given in Section 5.

The final output of our processing pipeline is a scalar field of same dimensions as the initial color data field. We use the scalar values for segmentation purposes. We apply standard segmentation algorithms for a proof of concept.

4 CLUSTERIZATION

We introduce two novel clusterization techniques and compare them to the clusterization techniques described in Section 2. We came up with novel methods ourselves, as we observed that the known clusterization methods we applied did not lead to fully satisfiable results. We discovered that some important color differencies can get lost during clusterization. This observation is due to the fact that algorithms such as median cut yield only average quality results. We achieved very good results while using k-means and c-means clustering algorithms on some of our examples.

We propose two novel clusterization methods for a more adequate clustering. We first propose a genetic algorithm that generates high-quality results and can be used as a standard for other clusterization algorithms. For the genetic algorithm, we have to define the genetic material, its initialization, the update rules that are iteratively applied, and the fitness function.

The genetic material of the individual is stored in a chromosome made up of basic genes which define the physical features of the individual. A previously presented genetic clusterization algorithms [TAE98] takes the mapping of all sample colors to the K palette colors as a genetic chromosome. This choice leads to an extremely high memory consumption, as each chromosome consists of a number of genes equal to the number of unique colors in the picture. In our algorithm, we take the centers of all cluster, i. e. points in the $L^*a^*b^*$ color space, as a gene. Thus, a chromosome consists of K genes only, where K is the number of clusters.

For the initial population generation we build the median cut tree, take the cubes lying on the depth equal to $log_2(K)$, where K is the number of clusters, and determine which unique colors of the image belong to which cube. Then, we find the average centers of each cluster as it is done in the median-cut algorithm. The initial population is formed by the centers of each cube.

We define an overall fitness function F_l that characterizes each individual l by

$$F_l = \sum_{m=1}^{K} \left\{ D(\bar{c}_m, \tilde{c}_m) + \max_{i,j \in P_m} (D(i,j)) \right\},$$

where $\bar{c}_m$ is the average center of the image colors, which belong to cluster m, $\tilde{c}_m$ is the cluster center chosen by the genetic algorithm, D is the Euclidean distance function in the $L^*a^*b^*$ color space, and P_m is the set of the colors that belong to cluster m.

We minimize this function for the individuals by generating new populations using reproductions (mutations, crossovers) and comparing the fitness functions.

On the test examples the genetic algorithm achieved a clusterization that distinguishes all important colors and generates an equal distribution among the remaining colors.

Unfortunately, the computation times for the genetic algorithm tend to be rather high, such that it is not practical to apply it to larger 3D data sets. Therefore, we will use this algorithms only as a standard, to which we compare the results generated by faster clusterization algorithms. The goal was to develop a clusterization algorithm which is much faster than the genetic clusterization algorithm and, at the same time, more adequate than median cut, k-means and c-means algorithm in certain cases.

We developed the axes-aligned binary-tree partitioning approach for clusterization. Like in median cut algorithm our algorithm starts with an axes-aligned box that encloses all the different color values from the original image in the $L^*a^*b^*$ color space. The "size" of the box is given by the minimum and maximum of each of the color coordinates that enclose the box . For splitting the box we have to decide, in which dimension we want to perform the splitting step. We choose the dimension, in which the box has its longest edges. The points are sorted in this dimension. In this order we are looking for the largest gap in the splitting dimension. The partitioning of the box into two halves is done at the lower end of that largest gap. We iterate these splitting steps until all unique colors that have a representative in the given image are split. This procedure forms a binary tree with the unique colors stored in the leaves. As a result, pairs of closest colors are the pairs stored in leaves with the same parent. Note that the binary tree is typically unbalanced.

In order to determine K clusters out of the binary-tree structure, we iteratively merge leaves of the tree. In each step, we merge the two leaves, whose colors have the minimal Euclidean distance in $L^*a^*b^*$ color space. The color assigned to the new leaf is the average color of the colors stored in the leaves that have been merged. The average color c_{av} is computed by

$$c_{av} = \frac{1}{M} \cdot \sum_{i=1}^{M} c_i \,, \tag{1}$$

where M is the number of all leaves which belong to this subtree, and c_i are the colors that have been stored in the leaves. We apply this step until the number of leaves is equal to the number of cluster centers.

In general, the decision which method to take depends on the initial data.

5 COLOR-TO-SCALAR CONVERSION

The most common way of converting *RGB* images to grayscale in terms of image processing or converting an *RGB* color data set to a scalar field in terms of visualization, is to operate on the luminance. First, the image colors are converted from *RGB* to a color space with a luminance channel, for example, to the $L^*a^*b^*$ color space. Then, luminance values are taken as resulting scalar values. More sophisticated techniques for mapping the 3D color space to one particular axis have been developed, but all these methods are ineffective at preserving different colors orthogonal to the chosen axis. Often the axis is oriented close to the luminance axis such that isochromatic colors are projected to similar regions on the axis.

The Color2Gray algorithm allows to take into account both luminance and chrominance differencies in a source image and construct an appropriate grayscale image. It was introduced for the conversion of 2D images, but could be generalized to voume data.

The user can influence the output of the Color2Gray algorithm using three simple and intuitive parameters. The first parameter θ controls whether chromatic differences are mapped to increases or decreases in luminance value. The second parameter α determines how much chromatic variation is allowed to change the source luminance value. The third parameter μ sets the neighbourhood size used for chrominance estimation and luminance gradients.

The color differences between pixels in the color image are expressed as a set of signed scalar values. The differences are measured in the various channels of the $L^*a^*b^*$ color space. Thus, both luminance and chrominance differences are computed. The generation of the output in form of a grayscale version of the image is based on these differences. For each pixel i and each neighbor pixel j, the signed distance scalar δ_{ij} based on luminance and chrominance differences is computed by

$$\delta_{ij}(\alpha,\theta) = \begin{cases} \triangle L_{ij}, & \text{if } \|\triangle L_{ij}\| > \mathit{crunch}(\| \triangle \vec{C}_{ij} \|) \\ \mathit{crunch}(\| \triangle \vec{C}_{ij} \|), & \text{if } \mathit{crunch}(\| \triangle \vec{C}_{ij} \|) \cdot \vec{V}_\theta \geq 0 \\ \mathit{crunch}(- \| \triangle \vec{C}_{ij} \|), & \text{otherwise} \end{cases}$$

where L_i is the luminance of ith pixel, $\triangle L_{ij} = L_i - L_j$, $\| \triangle \vec{C}_{ij} \|$ is the Euclidean norm of the vector $\triangle \vec{C}_{ij} = (\triangle A_{ij}, \triangle B_{ij})$ with $\triangle A_{ij}$ and $\triangle B_{ij}$ being the differencies between pixels i and j in the chrominance channels a^* and b^*, respectively, $\vec{V}_\theta = (cos\theta, sin\theta)$ is a normalized vector defined by θ, and $crunch(x) = \alpha * tanh(\frac{x}{\alpha})$. For a detailed derivation of the formula, we refer to the original work on the Color2Gray algorithm [GOTG05].

Given a set of signed differencies δ_{ij} for pixel pairs (i, j) of an ordered set S, a scalar field g is computed such that g minimizes a target function $f(g)$. The target function $f(g)$ is given by

$$f(g) = \sum_{(i,j)\in S} ((g_i - g_j) - \delta_{ij})^2 \, .$$

Hence, the minimization problem can be written in the form of

$$min(f(g)) = min\left(\frac{1}{2} \sum_{(i,j)\in K} (x_i - x_j - b_{ij})^2\right) \tag{2}$$

where the x_i, x_j, and b_{ij} can easily be derived. This is a least-squares problem of the form

$$min\left(\frac{1}{2}(Ax-b)^T(Ax-b)\right) \, ,$$

which can be rearranged to

$$min\left(\frac{1}{2}x^T A^T Ax - (A^T)^T x + \frac{1}{2}b^T b\right) \, .$$

This equation is quadratic with a symmetric, positive semi-definite Hessian. Therefore, minimizing it is equivalent to satisfying the linear equation:

$$A^T Ax = A^T b \, .$$

Deriving the terms d_k on the right-hand side of the equation, we obtain

$$d_k = \left[A^T b\right]_k = \sum_{j\geq k} \delta_{kj} - \sum_{i<k} \delta_{ik} \, .$$

Because of the regular form of the Hessian, $A^T Ax = A^T b$ expands to

$$(N-1) \cdot x_k - \sum_{l\neq k} x_l = d_k$$

with A being a $N \times N$ matrix. For any two indices i and j, we obtain

$$d_i - d_j = ((N-1) \cdot x_i - x_j) - ((N-1) \cdot x_j - x_i)$$

leading to

$$x_i = \frac{d_i - d_j + N \cdot x_j}{N} \, . \tag{3}$$

For any $x_i = c$ there is exactly one solution to Equation 2, which may be obtained by taking any known minimal vector x', and shifting all of its elements by $x_i - x'_i$. So, the problem can be solved by setting $x_0 = 0$ and getting all other x_i from Equation 3. Then, the found grayscale values are shifted to be as close to the source luminances as possible.

The bottleneck of the algorithm is the calculation of the coefficients d_k. The cost of the calculations is $O(N^6)$ for a $N \times N \times N$ volume image. Using local variants of the algorithm by adjusting parameter μ is inappropriate for our purposes, as one color value should always be assigned to the same greyscale value, which is only assured by using a global version of the algorithm.

However, if we do the prior clusterization of the image, the calculations can be reduced dramatically. After the quantization of the image we are left with arrays of length K, where K is the number of generated clusters. Let the output generated by the cluster be given in form of the color values of each cluster's center stored in array *Centers*, the number of occurrences of colors from each cluster stored in array *Occurs*, and the indices that assign to each pixel of the color the appropriate cluster stored in array *Indices*.

For each cluster k we can calculate d'_k by

$$d'_k = \sum_{j\geq k} \delta_{kj} * Occurs[j] - \sum_{i<k} \delta_{ik} * Occurs[i] \tag{4}$$

The δ_{kj} and δ_{ik} are computed on the cluster colors only using the information stored in array *Centers*. The cost of this calculation is $O(K^2)$. Thus, it only depends on the typically small number of clusters and is independent of the number of pixels/voxels in the image. In particular, it does not matter, of which dimension the original image is. Our approach scales to arbitrary dimensions, as it only operates on the clusters and their centers.

Finally, for each pixel or voxel i of the original 2D or 3D image, we get the desired value d_i by determining the cluster it has been assigned to and using the respective value d'_k. Thus, we retrieve $d_i = D[Indices[i]]$, where D is the array that stores the d'_k we computed for all clusters k. These values d_i are used, as before, to compute the x_i from Equation 3.

By bringing down the computational costs for the bottleneck computation from $O(N^6)$ to $O(K^2)$, we significantly speed up the procedure, which makes it applicable to larger 3D data sets.

6 RESULTS AND DISCUSSION

Since the improvement of our approach over luminance-based conversion models can be documented best by looking at 2D images, we first want to give some examples, where we convert individual slices through a 3D color data set.

For one of the examples we use a part of a horizontal slice through the Visible Female data set[1]. The data set is obtained by taking cryosections of an entire female human body. The first image of the Figure 2 shows a red organ surrounded by a yellowish tissue. The luminance of the surrounding tissue varies, but the chrominance values of the surrounding yellow region are clearly distinguishable from the chrominance value of the red organ.

[1]Data set courtesy of the National Institute of Health.

Figure 2 shows a conversion of the slice to a greyscale image using a luminance-based approach on the left-hand side and our approach on the right-hand side. Using the luminance-based approach, the boundary of the initially red region to the left gets lost. Using our approach, the initially red region is still clearly distinguishable from the surrounding tissue.

For generating the results of our approach throughout the paper, we used the axes-aligned binary-space partitioning or median cut clusterization and the following Color2Gray parameters: parameter $\alpha = 40$ and parameter $\theta = \frac{\pi}{4}$ or $\theta = \frac{3\pi}{2}$. Parameter μ always has to chosen such that the entire image is considered as a neighborhood to obtain a global approach.

For the generation of Figure 3 we applied a segmentation algorithm to the images of Figure 2. For segmentation purposes, we used a standard approach for isosurface segmentation based on marching squares for 2D images and marching cubes [LC87] for 3D images. However, any other segmentation approaches such as other isosurface extraction methods, level-set methods, or region-growing approaches could be used instead. In Figure 3, the contours are shown in red. The figure illustrates that the segmentation algorithm was not able to segment the initially red region when applied to the image converted by the luminance-based approach on the left-hand side. When applied to the image generated with our conversion algorithm, the region can be segmented well.

In Figure 1 on the left-hand-side, we show a part of a slice of a cryosection of a Macaque monkey brain.[2] The brain slices have been digitized using high-resolution digital photography that allows to scan even very small structures up to single neurons. We pick a region of interest with different tissues. The encircled tissue of interest has a purple color and is surrounded by tissue with brownish color shades.

The next two images of Figure 1 show greyscale images after conversion using a luminance-based approach (second) and our approach (third). Again, our method manages to let the tissues distinguishable, while the boundary of the purple tissue gets lost when using the luminance-based approach, as the luminance values for the purple and the brownish tissue were about the same. The last two images of Figure 1 show the results when applying a segmentation algorithm to separate the two tissues. Segmentation after luminance-based conversion only leaves us with a segmentation of the black spots. The purple regions cannot be segmented. Segmentation after our conversion allows for segmentation of the purple tissue.

For the generation of Figure 4 we used a data set of a cancer cell that has been scanned using fluorescence microscopy[3]. The 3D cancer cell data set is used to extract the yellow regions. The results are shown in Figure 4. Our conversion approach allows us to exactly extract the yellow regions by using isosurface extraction (middle), while the segmentation run on the luminance-based converted scalar field (right) extracts much larger regions that also include many originally green and red parts.

The results document that our color-to-scalar conversion method allows us to convert data sets with neighbored isoluminant areas that can still be separated by a standard segmentation method after conversion. Obviously, when the original color data only contains colors that vary in their luminance, a luminance-based conversion produces an optimal conversion that cannot be improved by our approach.

7 CONCLUSIONS

We have presented an approach for conversion of *RGB* color data to scalar data that is amenable for subsequent segmentation of the scalar field. In particular, our conversion method does not map isoluminant colors to the same scalar value, but allows us to preserve all important colors such that they are still distinguishable by a segmentation algorithm after conversion. Moreover, the order in luminance for isochrominant colors is maintained during conversion.

Our method operates in $L^*a^*b^*$ color space and uses clusterization algorithms for quantization. Afterwards, the quantized colors are mapped to the appropriate scalar values. We have presented two novel clusterization methods based on a genetic algorithm and an axes-aligned binary space partitioning. For the final assignment of the cluster colors to the scalar values we used ideas from the Color2Gray approach. By only applying the algorithm to the clustered data, we were able to achieve computation times that also allow for the application of our methods to larger 3D data sets.

We presented results of 2D and 3D color data set that document the improvements over other approaches when converting data sets with different isoluminant colors. We have applied standard segmentation techniques to show how the subsequent segmentation improved when using our approach. Our approach allows us to apply the entire range of all segmentation techniques to color data via our conversion step. This is obviously a more viable way than generalizing all existing and useful scalar field segmentation approaches to color volume data, which may not be straightforward for many of them.

REFERENCES

[Bez81] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press - New York, 1981.

[2] Data set courtesy of Edward G. Jones, Center for Neuroscience, University of Davis, California.

[3] Data set courtesy of the Department of Medicine, University of California, San Diego.

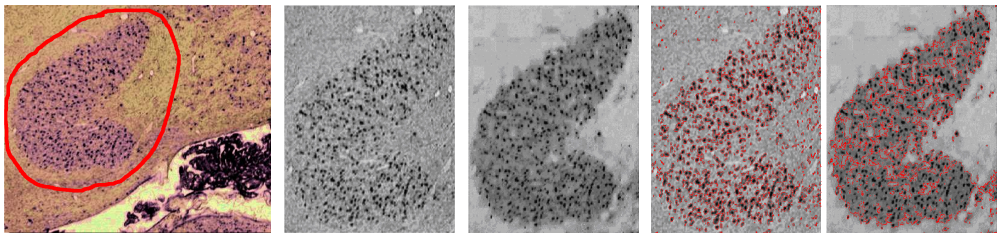

Figure 1: *First*: Cryosection of Macaque monkey brain with encircled region of interest. *Second*: Converted images using luminance-based approach. The two tissues cannot be distinguished anymore. *Third*: Converted images using our approach. The two tissues can clearly be separated by a segmentation algorithm. *Fourth*: Segmentation after luminance-based conversion cannot detect the formally purple region. *Fifth*: Segmentation after conversion with our approach allows to separate the tissues.

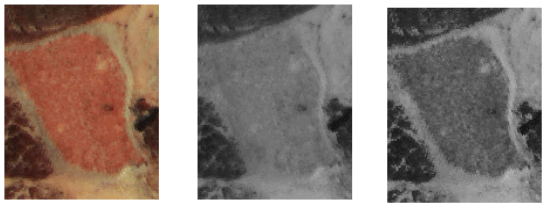

Figure 2: Comparison of converted images using luminance-based approach (middle) and our approach (right) when applied to the region of interest (left). Using the luminance-based approach, the boundary of the initially red region to the left gets lost. Using our approach, the initially red region is clearly distinguishable from the surrounding tissue.

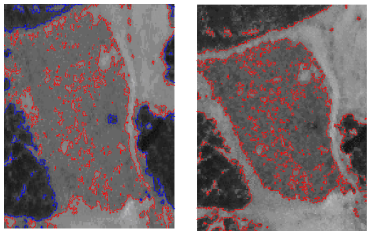

Figure 3: Comparison of segmentation of the converted images from Figure 2. For the image generated by a luminance-based conversion, the segmentation of the initially red region fails (left). For the image generated with our conversion algorithm, the region can be segmented very well (right).

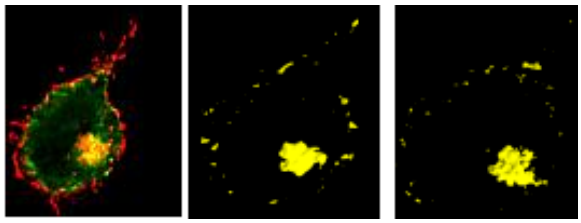

Figure 4: Fluorescence microscopy data set of a cancer cell (left). 3D segmentation of yellow regions via our color-to-scalar conversion approach (right) works well, while the segmentation after luminance-based conversion (middle) does not separate the yellow from th green and red regions completely.

[GOTG05] Amy A. Gooch, Sven C. Olsen, Jack Tumblin, and Bruce Gooch. Color2gray: salience-preserving color removal. *ACM Trans. Graph.*, 24(3):634–639, 2005.

[Hec82] P. Heckbert. Color image quantization for frame buffer display. In *Computer Graphics (Proceedings of ACM SIGGRAPH 82)*, pages 297–307, 1982.

[Jol02] I. T. Jolliffe. *Principal Component analysis*. Springer - Verlag, 2002.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, volume 21, pages 163–169, July 1987.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[Pas03] D. Pascale. *A Review of RGB Color Spaces...from xyY to R'G'B'*. Babel Color, 2003.

[RGW05] Karl Rasche, Robert Geist, and James Westall. Re-coloring images for gamuts of lower dimension. *Eurographics/Computer Graphics Forum*, 24(3), 2005.

[SB02] D. A. Socolinsky and L. B.Wolff. Multispectral image visualization through first-order fusion. *IEEE Transactions on Image Processing*, 11(8), 2002.

[Sch97] P. Scheunders. A comparison of clustering algorithms applied to color image quantization. *Pattern Recognition Letters*, 18, 1997.

[TAE98] Tolga Tagdizen, Lale Akarun, and Cem Ersoy. Color quantization with genetic algorithms1. *Signal Processing: Image Communication*, 12, 1998.