TECHNISCHE UNIVERSITÄT MÜNCHEN, DEPARTMENT OF INFORMATICS
INFORMATIK IX – COMPUTER VISION GROUP

Interdisciplinary Project in Mathematics

Total Variation Segmentation Incorporating Depth Information

Nikolaus Demmel

30.09.2014

Supervisor: Prof. Dr. Daniel Cremers
Advisors: Julia Diebold, M. Sc.
Caner Hazirbas, M. Sc.

Summary

We propose a method for interactive multilabel segmentation by extending spatially-varying color distributions to additionally utilize depth information. Depth is incorporated as an additional color channel, but also used in scribble distance computation. We furthermore introduce the concept of active scribbles to improve the ability of spatially-varying color distributions to adequately describe segmentation details indicated by user scribbles close to region boundaries. The approach is implemented for parallel hardware and evaluated on images from RGB-D cameras. We show that depth information can improve the accuracy of image segmentation as well as reduce the number of required scribbles for satisfying segmentation results significantly.

Contents

1 Introduction **3**

2 Background **3**

2.1 Variational Multilabel Segmentation 3

2.2 Interactive Image Segmentation 5

2.3 Spatially Varying Color Distributions 5

2.4 Depth Input 7

3 Approach **7**

3.1 Depth for the Regularizer Weight 7

3.2 Depth as Color Channel 8

3.3 Depth for Pixel Distance 9

3.4 Active Scribbles 11

3.5 Combined Improvements 14

4 Results **14**

5 Conclusion **16**

5.1 Future Work 16

References **18**

Figure 1: Example of a color image that is to be segmented into $n = 4$ regions. Segmentation based on color values alone (center) does not yield results close to the desired segmentation (right).

1 Introduction

Image segmentation is the task of partitioning an image $\Omega \subset \mathbb{R}^2$ into disjoint sets $\Omega_1, \ldots, \Omega_n$. While some segmentation frameworks (such as graph cuts) only support binary segmentation ($n = 2$), our total variation approach allows to address the more general multilabel segmentation ($n \geq 2$).
Segmentation based on color values alone does not always yield results close to the desired goal (see Figure 1). Given the abundance of depth-sensing with cameras like the Kinect, in the following we investigate the possibility of utilizing depth information to improve image segmentation.

2 Background

2.1 Variational Multilabel Segmentation

Following Nieuwenhuis et al. [2] we formulate image segmentation as an optimization problem of the form

$$\mathscr{E}(\Omega_1, \ldots, \Omega_n) = \underbrace{\sum_{i=1}^{n} \int_{\Omega_i} f_i(x)\,\mathrm{d}x}_{\text{dataterm}} + \lambda \underbrace{\frac{1}{2} \sum_{i=1}^{n} \mathrm{Per}_g(\Omega_i)}_{\text{regularizer}}, \tag{1}$$

where a minimizer of the energy $\mathscr{E}$ constitutes the computed segmentation. This energy balances two objectives: *data fidelity* and *smoothness*. On the one hand the dataterm describes how the image should be segmented and on the other hand the regularizer ensures that the resulting segmentation has smooth boundaries. In Equation (1) $f_i(x)$ describes the dataterm for label i and pixel position x, $\mathrm{Per}_g(\Omega_i)$ is the length of the perimeter (or boundary) of region Ω_i weighted by regularizer weight g, and λ is a scalar parameter balancing dataterm and regularizer.

2.1.1 Dataterm

The dataterm $f_i(x)$ assigns to each pixel position x a cost of this pixel being labeled as region i. Small values in the dataterm correspond to a high likelihood of the pixel belonging to region i.

In our probabilistic framework we therefore choose to model the dataterm as a negative log-likelihood, specifically

$$f_i(x) = -\log\Big(\mathscr{P}\left(I(x), x \mid u(x) = i\right)\Big), \tag{2}$$

where $u(x)$ is an indicator function mapping pixel positions to assigned labels and $I(x)$ is the color value of pixel x. In particular, we model a joint probability distribution over pixel color and position, i.e. we describe for each label a spatially-varying color distribution as opposed to a globally constant color distribution that is the same for all pixels.

2.1.2 Regularizer

Since the arg-min of the dataterm usually results in noisy segmentation, a regularizer is employed to favor segmentations with smooth boundaries. We use the sum of the region boundaries Per_g in the regularizer cost (the $\frac{1}{2}$ ensures that each boundary is counted only once). A regularizer-weight $g : \Omega \to [0,1]$ allows to indicate where the segmentation should favor region boundaries (pixels with smaller weight are favored). While the overall segmentation result is largely determined by the dataterm, this weight can help to shift region boundaries by a few pixels to a "favorable" position. We use the following (typical) weight based on the color image gradient

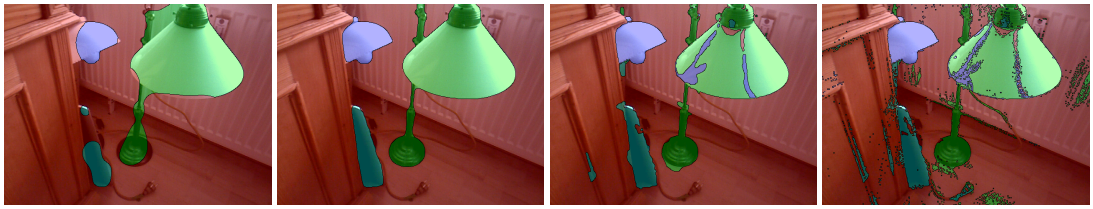
$$g(x) = \exp\Big(-\gamma\big|\nabla I(x)\big|\Big), \tag{3}$$

with parameter γ to adjust the influence of the image gradient. This choice for the weight helps to ensure that segmentation boundaries sharply align with object boundaries, since in practice those often coincide with a large color-gradient (see Figure 2).

Figure 2: Color gradient as regularizer weight.

2.1.3 Lambda

A scalar parameter λ balances between the dataterm and the regularizer such that larger λ result in smoother segmentations and smaller λ ensure that the dataterm is followed more closely (see Figure 3). In the extreme case of $\lambda = 0$ the segmentation corresponds the arg-min of the dataterm. For our examples we choose $\lambda = 10$ as a reasonable compromise.



$\lambda = 100$ $\lambda = 10$ $\lambda = 1$ $\lambda = 0$

Figure 3: Segmentation results for different values of λ.

2.1.4 Minimization via Convex Relaxation

The optimization problem in Equation (1) can be efficiently solved (near-optimal solutions for $n > 2$, which is NP-hard) by formulating it as a relaxed convex optimization problem and solving that with the primal-dual algorithm, which is efficiently implemented for parallel hardware. For details see [2].

2.2 Interactive Image Segmentation

While automatic image segmentation approaches often rely on machine learning to learn the dataterm, in our case of interactive image segmentation we assume a human user that indicates how the image should be segmented. Specifically, the user sets scribbles on pixel indicating their labeling. We model the probability distribution $\mathscr{P}(I, x \mid u(x) = i)$ based on those user scribbles.

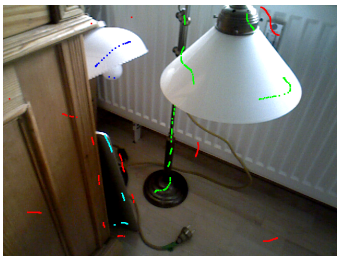


Figure 4: Example scribble input with colors indicating the $n = 4$ different labels.

2.3 Spatially Varying Color Distributions

Based on the user-provided scribbles, Nieuwenhuis et al. [2] model the joint distribution of color and position with a Parzen density estimator

$$\mathscr{P}\left(I(x), x \mid u(x) = i\right) = \frac{1}{m_i} \sum_{j=1}^{m_i} \underbrace{k_{\rho_i}\left(x - x_{ij}\right)}_{\text{distance kernel}} \underbrace{k_\sigma\left(I(x) - I(x_{ij})\right)}_{\text{color kernel}}, \tag{4}$$

with the number of scribbles m_i for region i, the pixel position x_{ij} of the jth scribble of region i, and zero-mean Gaussian kernels with appropriate norms

$$k_{\tilde{\sigma}}(\tilde{x}) = \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \exp\left(-\frac{|\tilde{x}|^2}{2\tilde{\sigma}^2}\right). \tag{5}$$

In other words the probability distribution for each label is modeled as a weighted sum over all scribbles of that label. The summands are products of kernel functions comprising a spatial kernel and a color kernel and are weighted equally with $\frac{1}{m_i}$. The spatial kernel has bandwidth ρ_i and takes as input the (Euclidean) distance to the scribble, while the color kernel has bandwidth σ and takes as input the difference of the current pixel's color value and the scribble's color value (Euclidean distance in RGB). Intuitively this means that for a scribble to contribute to the overall likelihood of a pixel x it has to be close to x (in relation to other scribbles) and it has to have a color similar to $I(x)$.
In order to address the fact that scribble positions are generally not distributed uniformly throughout the image, the bandwidth of the spatial kernel ρ_i is chosen proportional to the distance to the closest scribble $v_i = v_i(x)$ of label i.

$$\rho_i(x) = \alpha \left| x - x_{v_i} \right| \tag{6}$$

With that, for pixels close to scribbles the bandwidth is relatively small and only scribbles in the vicinity determine the color distribution, whereas for pixels far from any scribbles the bandwidth is relatively high and all scribbles have similar influence on the color distribution. Figure 5 illustrates this.

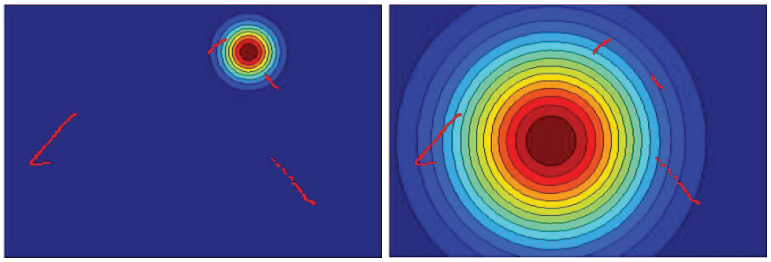


Figure 5: Bandwidth $\rho_i(x)$ is proportional to the closest scribble of label i.
(images courtesy of [2])

It can be interesting to compare spatially-varying color distributions with either global color distributions with

$$\mathscr{P}\left(I(x) \mid u(x) = i\right) = \frac{1}{m_i} \sum_{j=1}^{m_i} \underbrace{k_\sigma\left(I(x) - I(x_{ij})\right)}_{\text{color kernel}}, \tag{7}$$

or segmentation based on scribble location alone, where

$$\mathscr{P}\left(x \mid u(x) = i\right) = \frac{1}{m_i} \sum_{j=1}^{m_i} \underbrace{k_{\rho_i}\left(x - x_{ij}\right)}_{\text{distance kernel}}. \tag{8}$$

2.4 Depth Input

Depth cameras such as the Kinect provide metric depth values. However, depth values are usually not available for all pixels. We fill in the missing depth values in a preprocessing step with a colorization technique suggested by Silberman et al. [4], for which we use the implementation provided with their dataset[1]. For Kinect-like cameras the value range of the depth values $z(x)$ in meters is roughly $[0.5, 6]$. In general, for each image we normalize the actual depth range to $[0, 1]$ with

$$D(x) = \begin{cases} 1, & \text{if } z_{\max} = z_{\min} \\ \frac{z(x) - z_{\min}}{z_{\max} - z_{\min}}, & \text{otherwise.} \end{cases} \tag{9}$$

Figure 6 shows the normalized depth input for the image from Figure 1.

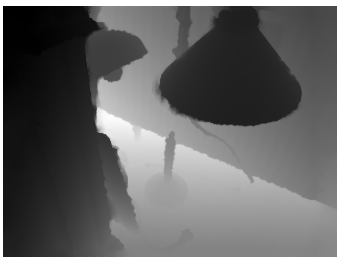


Figure 6: Normalized depth input.

3 Approach

In the following we describe how we propose to use depth information in the previously introduced model of spatially-varying color distributions.

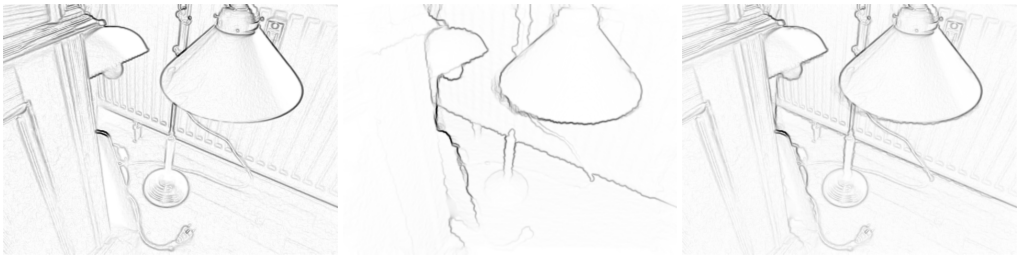
3.1 Depth for the Regularizer Weight

Similarly to the color gradient, the gradient ∇D of the normalized depth image can be used to compute regularizer weight g, for example:

$$g(x) = \exp\left(-\gamma\left(\lambda_g \left|\nabla I(x)\right| + (1 - \lambda_g)\left|\nabla D(x)\right|\right)\right), \tag{10}$$

where $\lambda_g \in [0, 1]$ allows to adjust relative influence of color and depth gradient. Figure 7 shows the regularizer weight for different values of λ_g.
One might assume the additional information from the depth gradient to be beneficial, since jumps in depth value often correspond to object boundaries. However, in practice with depth input from RGB-D cameras such as the Kinect the depth image is particularly noisy at object boundaries. Since the regularizer weight is supposed to improve segmentation specifically at

[1] `http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html`



$\lambda_g = 1$ $\lambda_g = 0$ $\lambda_g = 0.5$

Figure 7: Regularizer weight for different values of λ_g.

the boundaries we instead observe that using the depth gradient is often counter productive. We therefore suggest to only use the color gradient ($\lambda_g = 1$), unless a different depth sensor with little noise at object boundaries is used.

3.2 Depth as Color Channel

One idea to include depth information in the dataterm is interpreting the normalized depth image as an additional color channel and introducing a depth kernel in addition to the color kernel.

$$\mathscr{P}\left(I(x), D(x), x \mid u(x) = i\right) = \frac{1}{m_i} \sum_{j=1}^{m_i} \underbrace{k_{\rho_i}\left(x - x_{ij}\right)}_{\text{distance kernel}} \underbrace{k_\sigma\left(I(x) - I(x_{ij})\right)}_{\text{color kernel}} \underbrace{k_\tau\left(D(x) - D(x_{ij})\right)}_{\text{depth kernel}} \tag{11}$$

This gives rise to three variants: Equation (4) describes spatially-varying *color* distributions; replacing the color kernel with the depth kernel gives rise to spatially-varying *depth* distributions, which is analogous to color-segmentation on the depth input interpreted as a grayscale image; finally Equation (11) combines both, describing spatially-varying *color and depth* distributions. Figure 8 shows how using color and depth together improves the segmentation result over either of them.

3.2.1 Depth Kernel Bandwidth

The value of the bandwidth τ has a large influence on the segmentation result. Figure 9 shows segmentation on the normalized depth image (spatially-varying depth distributions) for different bandwidths. For this particular image small values of τ seem to be favorable. However, this image is very distinct in the depth channel. In practice we observe that for most images small τ lead to unexpected segmentation artifacts. In particular the user has to ensure that the whole range of depth values for each label is scribbled. Figure 10 shows such artifacts for small τ with spatially-varying color and depth distributions. From experimentation we conclude that a bandwidth around $\tau = 0.2$ performs best in combination with the color kernel.

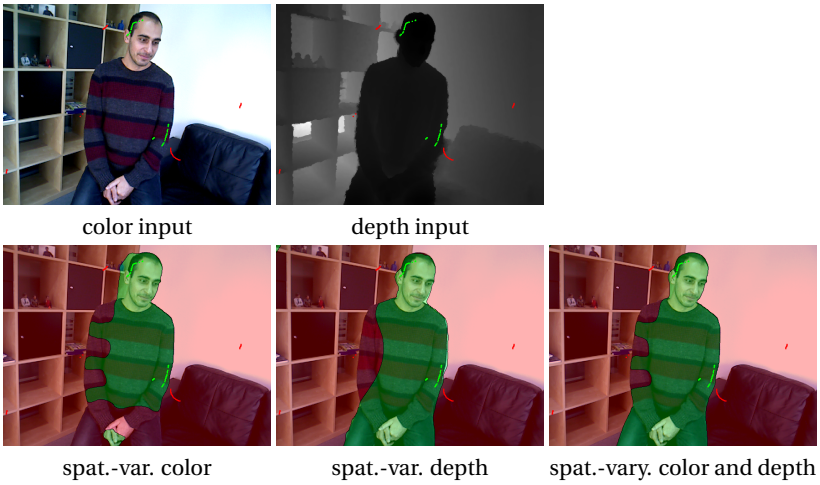


Figure 8: Segmentation results with spatially-varying *color* distributions, spatially-varying *depth* distributions, and spatially-varying *color and depth* distributions.

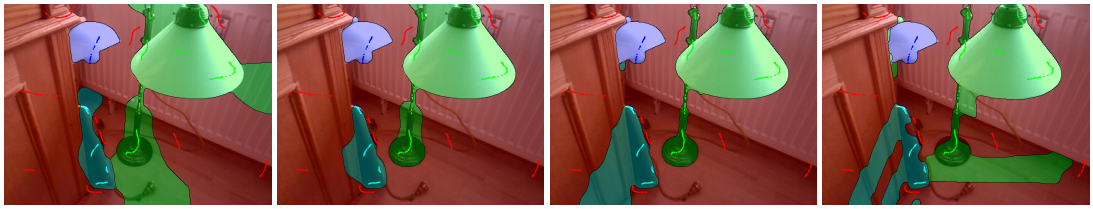
Figure 9: Segmentation with spatially-varying depth distributions for different τ.

3.3 Depth for Pixel Distance

Another idea is to utilize depth information when computing the distance kernel. Pixels close in the image are not always close in 3-dimensional space. Therefore, using the depth information to compute 3D distances between image points should benefit the segmentation as it ensures that the distance kernel reflects the real object geometry better. Consider for example a close object in front of a distant background. 3D distance computation ensures that scribbles on the background influence the object less and vice versa. This is highlighted by Figure 11.

In order to compute 3D pixel position X (in camera frame) from the pixel coordinates (u, v) and (not normalized) depth value z we can perform the inverse of the perspective transformation. Assuming a pinhole camera model

$$z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A\,X \tag{12}$$



$\tau = 0.5$ $\tau = 0.2$ $\tau = 0.05$ $\tau = 0.01$

Figure 10: Segmentation with spatially-varying color and depth distributions for different τ.

input with scribbles 2D distance back-projection 3D distance

Figure 11: The distance (indicated by a pink line) between the green scribbles on the person's head and the objects on the shelve is larger in the 3D case. Therefore those scribbles have less influence on the segmentation of the shelve.

with intrinsic matrix[2] A of the form

$$A = \begin{pmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{13}$$

we compute the 3-dimensional position vector from pixel position and depth value as

$$X = \begin{pmatrix} \frac{z(u-u_0)}{\alpha_x} \\ \frac{z(v-v_0)}{\alpha_y} \\ z \end{pmatrix}. \tag{14}$$

With that, we evaluate $k_{\rho_i}(X - X_{ij})$ instead of $k_{\rho_i}(x - x_{ij})$ and use $\rho_i(x) = \alpha\left|X - X_{v_i}\right|$, i.e. we perform all distance computation in 3D space rather than pixel space. As an alternative to the perspective back-projection we can also perform the simpler orthographic back-projection, which computes as

$$X = \begin{pmatrix} u \\ v \\ z \end{pmatrix}. \tag{15}$$

Intuitively the orthographic projection is more appropriate, since it does not spread apart pixels (and thus scribbles) in the background with large depth value. Moreover it is simpler to compute and does not require knowledge of the camera intrinsics.

[2] Typical values for a Kinect camera are $\alpha_x = \alpha_y = 525$, $u_0 = 319.5$, and $v_0 = 239.5$.

In order to choose distance kernel bandwidth parameter values (α) that are appropriate for a wide range of images we need to ensure stable value ranges and thus normalize the pixel position space. Analogous to the normalization of the depth image for the depth kernel, we ensure that the z value range of the 3D pixel positions is in range $[0,1]$. The normalization factors are chosen in the following way: For the perspective transformation we normalize all three components of X by the same scalar $z_{\max} - z_{\min}$. For the orthographic transformation we normalize the x and y component by $\max(u_{\max} - u_{\min}, v_{\max} - v_{\min})$ and the z component by $z_{\max} - z_{\min}$. Care has to be taken to handle the special (and very unlikely) case of $z_{\max} = z_{\min}$, for example by setting all z to 1.
In practice we do not notice a big difference in segmentation results between the two projections. Hence, we use the simpler orthographic projection. Figure 12 shows improved segmentation of spatially-varying color distributions using 3D distance. In the following we distinguish 2D and 3D distance computation in the spatial kernel with the terms *2D-varying* and *3D-varying*.

2D distance | 3D distance | 2D no color | 3D no color

Figure 12: Segmentation results of spatially-varying color distributions with 2D and 3D distance. The two results on the right show segmentation without color kernel, i.e. based on scribble distance alone (see Equation (8)).

3.4 Active Scribbles

In the following we highlight one issue with the distance kernel in our formulation of spatially-varying color distributions (this issue is somewhat independent from using depth). The phenomenon is related to the fact that user scribbles are typically not uniformly distributed across the image and our adaptation of bandwidth ρ_i based on the distance to the closest scribble.
Consider Equation (4) for a pixel that is very close to one of the scribbles and therefore has a relatively small distance kernel bandwidth. Most scribbles are far away in relation to the bandwidth and therefore the distance kernel for most scribbles evaluates to a small value. This in turn means that most summands in the sum of Equation (4) are small and the overall likelihood is small irrespective of the color values of the scribbles close by. The top three rows of Figure 13 illustrates one of the effects. Rows 1 and 2 respectively show normalized dataterm for the red label and arg-min-segmentation ($\lambda = 0$) based on only a 2D-distance kernel (i.e. a dataterm as in Equation (8)) for different values of α. Only the results for $\alpha = 1000$ (and in general, large α) appropriately reflect our intention with the spatial kernel: Closer scribbles should have higher influence such that the arg-min-segmentation with only the spatial kernel closely resembles a Voronoi tessellation. For small α one can see that the dataterm does not

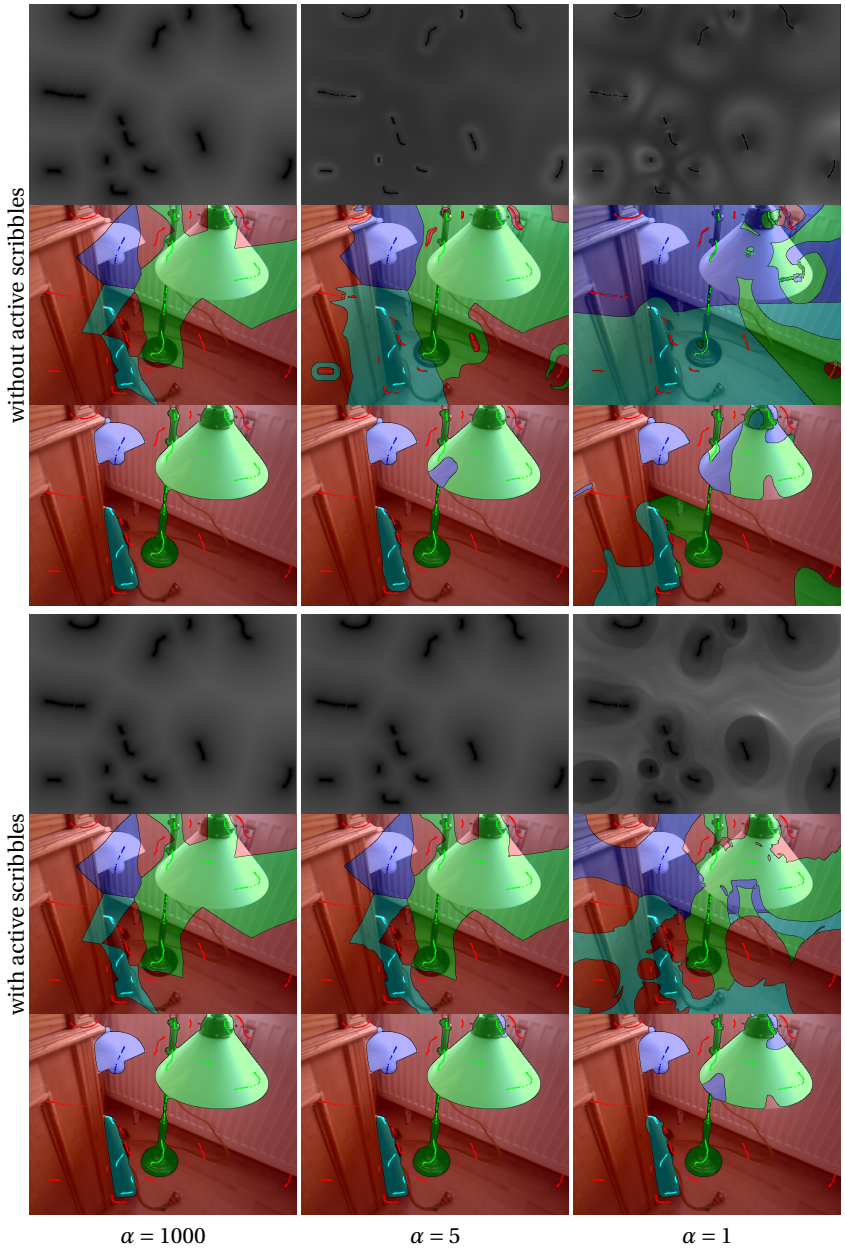
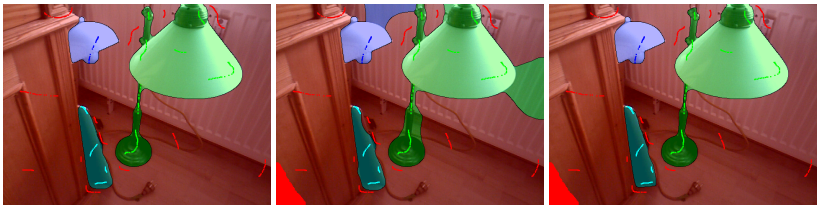


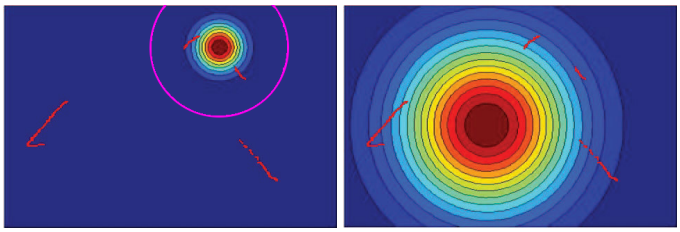
Figure 13: Results for different spatial bandwidth factors α, comparing usage of active scribbles. Row 3 and 6 show segmentation with 3D-varying color and depth distributions. To highlight artifacts in the spatial kernel for small α, row 1 and 4 show the normalized dataterm for the "red" label with only the 2D-spatial kernel (no color, no depth kernel) and row 2 and 5 show the corresponding arg-min-segmentation ($\lambda = 0$). Using active scribbles reduces the negative impact of small α to some extend.



without extra scribbles | degraded segmentation with extra scribbles | with extra scribbles and using active scribbles

Figure 14: We revisit segmentation with 3D-varying color and depth distributions from Figure 13 with $\alpha = 1000$. Adding a lot of scribbles in one part of the image degrades the segmentation result in other parts. When using active scribbles this effect is not observed and the resulting segmentation is close to the segmentation without the extra scribbles.

decrease monotonically with distance to the closest scribble as it does for large α. Row 3 in Figure 13 shows how these artifacts in the spatial kernel negatively influence segmentation with 3D-varying color and depth distributions. To sum up, unless the bandwidth factor α is chosen large (typically ≥ 1000), the distance kernel does not behave as desired.
However, a choice of large α only addresses part of the problem. Figure 14 shows the effect of introducing a large number of additional red scribbles in the corner of the image. While we would like these scribbles to have little effect on the rest of the image, they "weaken" the red dataterm throughout the image, resulting in a degraded segmentation.
In order to counteract this phenomenon we propose to use *active scribbles*. To this end, when computing the dataterm of a pixel, we consider a radius of $3\left|x - x_{v_i}\right|$ around that pixel, i.e. 3 times the distance to the closest scribble. We call all scribbles within that area *active*. In the sum in Equation (4) each scribble contributes a product of kernels that is weighted equally with $\frac{1}{m_i}$. For each label, if 80% or more scribbles are active, nothing changes. If less than 80% of all scribbles are active, we re-weight that sum such that all the active scribbles get a total of 80% of the weight. In other words we compute two separate Parzen estimates — s_a for the active and s_i for the inactive scribbles — and combine those as $0.8s_a + 0.2s_i$. Figure 15 illus-



for a pixel with short distance to the nearest scribble only some scribbles are active (indicated by pink circle) | for a pixel far away from any scribbles, all scribbles are active

Figure 15: All scribbles withing $3|x - x_{v_i}|$ are *active* and are ensured to contribute at least 80% of the overall weight in the Parzen window. On the left, the region of active scribbles is indicated by a pink circle. On the right, all scribbles are active.

trates the active scribble region. The values “3” and “80%” are chosen somewhat arbitrarily, but they have proven appropriate in practice. In particular, this choice performs better than choosing 100%, i.e. only considering the active scribbles.
This re-weighting of the sum in the Parzen estimator ensures that for pixels close to a scribble the overall likelihood is determined for the most part by the active scribbles. Figure 14 shows how using active scribbles counteracts the negative effect of a large number of additional scribbles on the segmentation result. Rows 4 – 6 of Figure 13 show that when using active scribbles the effect to too low α is mitigated, even though the bandwidth factor should still be chosen > 1.
Note that in the case of 3D distance computation we consider a sphere with radius $3\left|X - X_{v_i}\right|$ instead of a circle with radius $3\left|x - x_{v_i}\right|$ to determine the active scribbles.

3.5 Combined Improvements

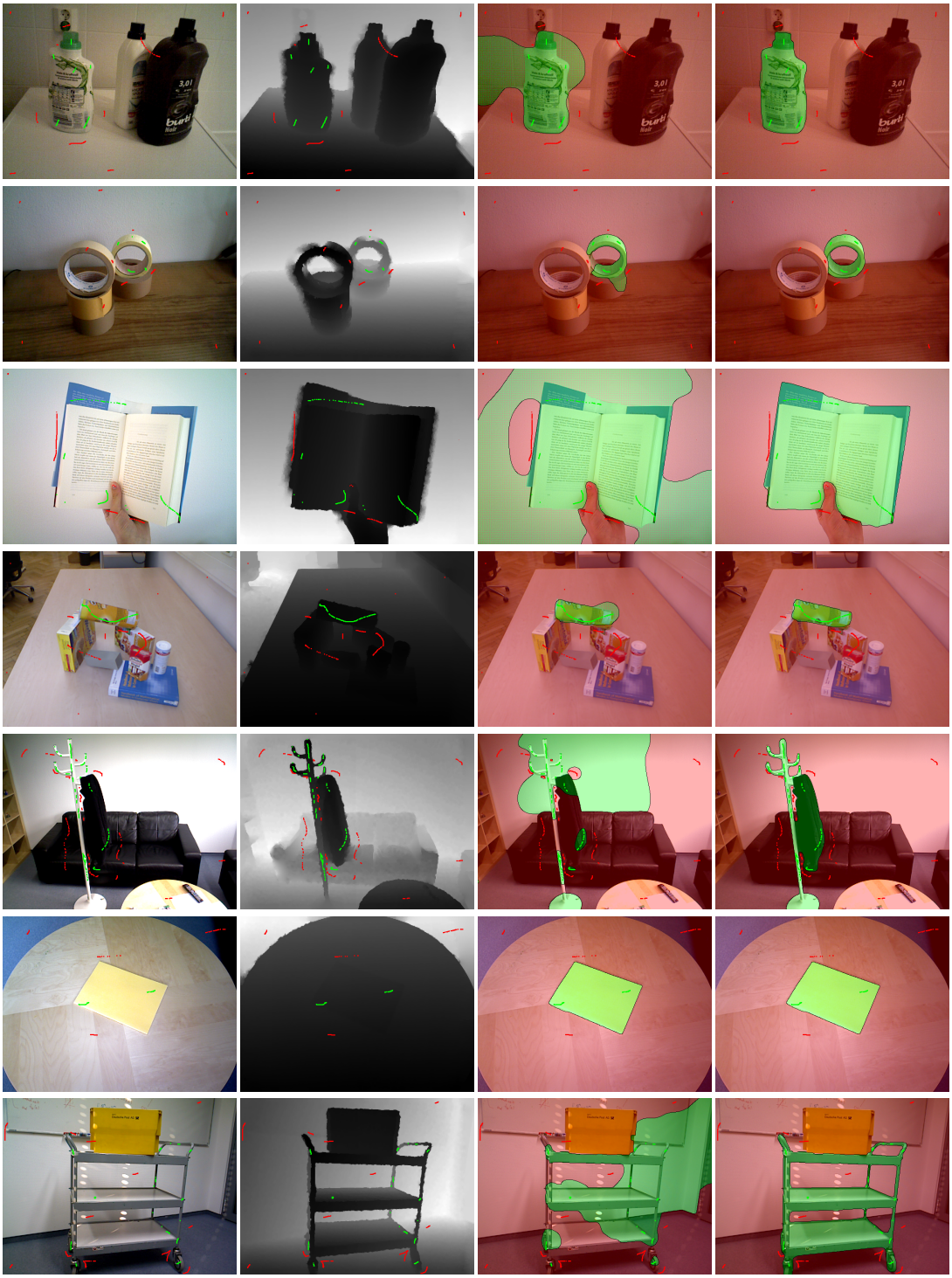
We propose to use all three suggested improvements together, i.e. computing 3D scribble distances, including a depth kernel and using active scribbles. Segmentation results are shown in Figure 16.

2D-varying depth distributions | 3D-varying color distributions | 2D-varying color distributions with active scribbles | 3D-varying color and depth distributions with active scribbles

Figure 16: Combining 3D distance computation, depth kernel, and active scribbles yields best results.

4 Results

In the following we show several images comparing for a fixed set of scribbles our segmentation results (3D-varying color and depth distributions with active scribbles) with the 2D-varying color distributions of [2].
For the examples in Figure 17 we use the parameters shown in Table 1. The value range of the RGB channels is $[0,1]$. The depth input is in meters, however for the distance computation the depth range is normalized to be in an interval of length 1 as described in Section 3.3.
Finally in Figure 18 we highlight that each aspect of the proposed approach contributes to the success of the overall segmentation.



color input | depth input | Nieuwenhuis et al. [2] | our result

Figure 17: Comparing our result of 3D-varying color and depth distributions with active scribbles to the approach from [2]. Row 4 shows and image of the OSD benchmark [3].

parameter	value	comment
λ	10	fairly strong smoothing
γ	5	fairly strong influence from gradient
λ_g	1	color-gradient only
α	1000	with active scribbles $\alpha = 10$ yields similar results
σ	0.05	slightly weaker than in [2], where $\sigma \approx 0.02$ is suggested
τ	0.2	fairly weak to avoid artifacts

Table 1: Parameters for the results presented here.

3D-varying color with active scribbles | global color and depth with active scribbles | 3D-varying color and depth | 3D-varying depth with active scribbles

Figure 18: Each segmentation result shows our approach without one aspect. In order to obtain the desired segmentation on the bottom-right of Figure 17, the combination of all components is needed.

5 Conclusion

We conclude that using depth information is very beneficial to segmentation accuracy as well as the number of scribbles needed for a satisfactory segmentation and suggest to use all three enhancements, namely the depth kernel, 3D scribble distances as well as active scribbles.

5.1 Future Work

For further investigations there are several ideas for improvements and extension.

Metric Evaluation It would be helpful to perform a quantitative evaluation of the proposed approach. Unfortunately it is not obvious how different variants should be compared. There are metrics for multi-label segmentation that asses segmentation accuracy, for example the dice score

$$\text{dice}(\bar{\Omega}_i, \Omega_i) = \frac{2|\bar{\Omega}_i \cap \Omega_i|}{|\bar{\Omega}_i| + |\Omega_i|}. \tag{16}$$

However the concrete scribble locations do influence the segmentation result heavily and for interactive segmentation other factors such as number of scribbles and time of user interaction should be taken into considerations. Moreover, depending on the application, it might also be very important that the segmentation algorithm produces not just correct overall segmentation but in particular accurate object boundaries. Additional scores specifically

for assessing performance on object boundaries should be devised. For example it might be suitable to consider the percentage of correctly segmented pixels in a k pixel band around the ground truth object boundaries, possibly weighted by distance from the boundary, such that wrong classifications further from the boundary are weighted more. We are not aware of any existing benchmarks for scribble-based interactive image segmentation including depth images.

Further Estimator Improvements A master thesis by Kehl [1] discusses several improvements of color-based spatially-varying color distributions, many of which could also be applied to our approach including depth. Among other things we might want to consider

- using other color spaces such as the Lab color space, and transforming the color space using Orthogonal Linear Discriminant Analysis (OLDA),
- using an additional texture kernel,
- using shape priors based on the distribution of scribble locations,
- automatically estimating appropriate bandwidth parameters for the different kernels.

Parameter Optimization Our segmentation algorithm has several parameters and some form of parameter optimization might help to find a set of good values. In particular the kernel bandwidth values should be considered.

Geometry and Connectivity For most real objects the depth values contain a large amount of information about the object geometry. A more sophisticated analysis on object geometry, connectivity and spatial relation could improve the segmentation further in cases where the current approach fails. This information might however not be straight forward to include in the Parzen density estimator. Examples of existing work on spatial object relation and segmentation of RGB-D images are [4, 3].

Theoretical Analysis and Alternate Formulations Additional theoretical analysis of the proposed Parzen estimator could shed light on how to improve it further. Note in particular that with the current addition of depth kernel and 3D distance computation, the estimator uses the depth value twice in a Gaussian kernel, once with a constant bandwidth τ and once with an adaptive bandwidth ρ_i, It might be helpful to analyse how these kernels interact. Moreover, the current approach expresses that pixels should have "similar color *and* depth". Different formulations could be investigated that express "similar color *or* depth". Ideas that might be explored include

$$\mathscr{P}\left(I(x), D(x), x \mid u(x) = i\right) = \frac{1}{m_i} \sum_{j=1}^{m_i} k_{\rho_i}\left(x - x_{ij}\right)\left(\lambda_c k_\sigma\left(I - I_{ij}\right) + (1-\lambda_c) k_\tau\left(D - D_{ij}\right)\right) \quad (17)$$

and

$$\mathscr{P}\left(I(x), D(x), x \mid u(x) = i\right) = \mathscr{P}\left(I(x), x \mid u(x) = i\right)\mathscr{P}\left(D(x), x \mid u(x) = i\right) \tag{18}$$

$$= \left(\frac{1}{m_i}\sum_{j=1}^{m_i} k_{\rho_i}\left(x - x_{ij}\right) k_\sigma\left(I - I_{ij}\right)\right)\left(\frac{1}{m_i}\sum_{j=1}^{m_i} k_{\rho_i}\left(x - x_{ij}\right) k_\sigma\left(D - D_{ij}\right)\right). \tag{19}$$

Improved User Experience and Alternate Input In interactive segmentation one important objective of algorithms is to minimize the user interaction time. Apart from reducing the number of scribbles required for adequate segmentation, it might be interesting to investigate how to make use of other forms of interactive input. For example, bounding boxes drawn by the user to select an area of interest are frequently used. This would firstly allow to perform segmentation only on a much smaller image region, which can be done faster. Secondly, it is more reliable, since one issue with the current approach is that sometimes the segmentation around the object of interest is good, however parts of the background far away from any object boundary are still segmented as foreground due to missing background scribbles on this particular depth or color. Thirdly, a bounding box allows us to gather information about the background distribution, e.g. by randomly sampling background scribbles from the vicinity of the bounding box. Another improvement of user interaction would be providing the user with information about where to best put scribbles. Typically some scribbles are very important for successful segmentation, while others make very little difference. If the dataterm can be computed fast enough ($\ll 1sec$), the user interface could highlight in real time image regions where the dataterm is not very distinct, meaning multiple labels have similar likelihood values.

References

[1] Wadim Kehl. *Interactive Image Segmentation Using Space-Varying Color and Texture Distributions*. Master's thesis, Technische Universität München, Munich, Germany, Jan 2013.

[2] Claudia Nieuwenhuis and Daniel Cremers. Spatially varying color distributions for interactive multilabel segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1234–1247, May 2013.

[3] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4796, Oct 2012.

[4] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, pages 746–760, Oct 2012.