Diplomarbeit

Piecewise smooth deconvolving segmentation

Katharina Philipp

11. April 2013

Technische Universität Dresden
Fakultät Informatik
Institut für Künstliche Intelligenz
Professur Intelligente Systeme

Max-Planck-Institut für molekulare Zellbiologie und Genetik

Betreuender Hochschullehrer:	Dr. Dmitrij Schlesinger
Betreuer:	Dr. Ivo Sbalzarini
	Dr. Janick Cardinale

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig erstellt und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Dresden, den 11. April 2013

Katharina Philipp

Thanks and Acknowledgements

First of all, I want to thank my supervisors, Dr. Schlesinger, Dr. Sbalzarini and Dr. Cardinale.

I am very grateful to Dr. Schlesinger for filling in as an associate professor in the Intelligent Systems group and thus giving me the opportunity to study in the field of image processing. I enjoyed his lectures as they helped me further my understanding of image segmentation.

Without Dr. Sbalzarini and Dr. Cardinale, this work would not exist. I will always be grateful to Dr. Sbalzarini for answering my request for an interesting topic so quickly. Thanks to him, I was able to work on this research topic, visit Zurich, take part in my very first conference, and meet the wonderful MOSAIC group. I thank Janick Cardinale, who was always available for advice and encouragement and quickly answered when I had problems. His knowledge and insight in image segmentation in general and the Region Competition Framework in detail helped me a lot. I also want to thank the MPI-CBG for giving me the opportunity to work in their institute and to meet so many people from all around the world. I greatly enjoyed my foray into the world of science. A thank you also goes toward the MOSAIC group for welcoming me so warmly. I enjoyed the opportunity to work alongside you.

Furthermore, I want to thank my parents for always supporting me, not just financially but also morally when I was at my wit's end. Much of my thanks also goes to my friends, especially Annegret and Maria. They all listened patiently when I had troubles and reminded me that I also had a life outside the diploma thesis.

Above all, I want to thank Marcus, who helped me out in so many ways, as an administrator when something did not work, as a fellow programmer when I could not find a bug, and, last but not least, eased my work load by doing a lot of my chores.

Thank you.

Contents

List of Figures

1 Introduction

Image segmentation is a computer vision technique to separate an input image into several non–overlapping regions. The pixels within a region belong semantically together. For example, in a microscopy image each region might represent a cell.

With the result of a segmentation, each pixel can be assigned to a region. In a discrete setting that result is a label image which identifies each region by a number. This labeling is then used to further analyze the image. For instance, in cell classification, shape and mean color of each region can be used for cell–type classification.

Therefore, the result depends not only on the segmentation algorithm but also on the definition that declares which pixels belong together. Such a model might declare that neighboring pixels are in one region if they have a similar gray value. Or it might not depend on the image at all and declare that the region's boundaries should be as smooth as possible. There are many different kinds of models. In this work I describe and implement a piecewise smooth deconvolving model for the *Region Competition framework* [CPS12].

A piecewise smooth deconvolving model combines image segmentation with image restoration. The interaction between both improves the result of each process. Image restoration increases the quality of an image by reducing noise and blur. The latter originates from out–of–focus light and light scattering during the image acquisition process. In microscopy, this effect leads to seemingly thicker objects and vague shapes as depicted in Figure 1.1. A piecewise smooth segmentation without deconvolution fails to correctly delineate the true objects. Therefore, in this work, I combine both.

I estimate the true image by restoring the image. This restoration is done with the help of the label image and the image model. Unlike piecewise constant deconvolution, in which the intensities are constant within one region, piecewise smooth deconvolution allows smooth intensity variations in the regions. With the estimated true image, I simulate the image formation in the microscope and compare the result to the input image. The comparison gives an estimation how well the current estimated image fits the true image. Additionally, I can also see how well the current label image matches the image's objects because the estimated image is calculated from the label image.

Image segmentation tasks are often formulated as optimization tasks in which the image–formation model is incorporated into an energy function. In mathematical optimization, the energy function represents the energy of the system modeled and we want to know the parameters that minimize or maximize this function. In image segmentation the labeling is one such parameter and the energy function describes how well the current regions and the image model fit the input image.

The *Region Competition framework* also regards segmentation as an optimization task. It is a discrete segmentation algorithm that separates the energy function from the optimization. The energy model is not explicitly implemented but can be chosen

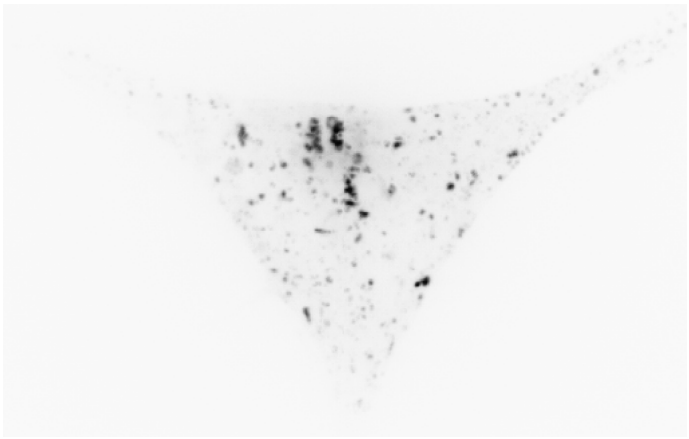


Figure 1.1: Example of a convolved fluorescence microscopy image showing endosomes in live HER911 cells. A piecewise constant deconvolution has difficulties with the piecewise smooth background that in some areas the background is brighter than some of the endosomes. The figure has been inverted for illustration purposes. Source: Prof. Urs Greber, University of Zurich, and Dr. Christoph Burckhardt, Harvard University

as an argument for the algorithm. Several energy terms are already available, e.g., piecewise constant with different noise models, piecewise smooth with different noise models, piecewise constant deconvolution, cut metrics, Kybic–Kratky curvature flow, moment based shape priors [Car13]. In this work, I add another energy model — piecewise smooth deconvolution.

The remainder of this thesis is structured as follows. In Chapter 2, I describe the existing framework and the theoretical background of the different energy models I refer to. In Chapter 3, I first explain the theory of my approach and then my implementation within the *Region Competition framework*. In the evaluation chapter I compare my model to other existing ones. I conclude this work with a summary and suggestions for future work and improvements.

2 Theoretical Background

In this chapter I describe the background in which this thesis is set. I start with a brief literature overview to position this work. Afterwards, I explain the theoretical background of a piecewise smooth deconvolving energy model. In order to give an idea on the features the model should have I explain the *Region Competition framework* before I focus on the model itself. In Section 2.3 and 2.4, I concentrate on piecewise smooth segmentation and deconvolution separately and in Section 2.5 I detail their combination. I conclude this chapter with the design decisions that I inferred from the previous analyses.

2.1 Related Work

In this thesis, I explore the combination of image segmentation and restoration in a piecewise smooth setting. Especially microscopy images can be so blurred that both tasks are needed to achieve proper results in which the cells or often also their inner structure have been correctly segmented. However, both processes are actually closely related to each other and the result of one would improve the other. Paul et al. reflected on their similarities in [CPS13]. Whereas in image segmentation, the objects are explicitly modeled and expressed through their boundaries, image restoration assumes the effect of the image's objects in the underlying mathematical structures. If the object's position and boundary were known, an image restoration algorithm could better account for the intensity jumps that occur at objects' edges. On the other hand, a segmentation algorithm cannot segment the objects correctly if they are blurred. The image would have to be restored first by inverting the blur. In order to use both effects, a piecewise smooth convolved model combines both.

Image segmentation separates an input image into several non–overlapping regions to further process the result, for example, for object classification. In order to segment an image we need the object's description — the model — and an algorithm. The model is a mathematical, abstract description of the image's objects. It is often formulated into an energy functional, whose minimum is then the energy state in which the boundaries represent the optimal partition of the image.

The *Mumford–Shah functional* [MS88] and *Bayes' theorem* [GG84] and their variations are primarily used to describe objects. Mumford and Shah assumed that within the objects, intensities only vary slightly. Edges, however, are indicated by intensity jumps. Hence, they use piecewise smooth functions to describe the image.

Geman and Geman solve the segmentation problem statistically [GG84]. In order to find the correct edge set $\mathcal{C}$ given the input image I, they try to maximize the posterior probability $p(\mathcal{C}|I)$. Brox and Cremers showed that both approaches are closely related

and that the *Mumford–Shah functional* is an approximation of the Bayesian posterior maximization in the case of local statistics [BC07].

There are many variations of the two models. One example is the addition of prior knowledge about the objects' shape by Cremers et al [CKS02]. Restricting the piecewise smooth functions of the *Mumford–Shah functional* to be piecewise constant is called the *Cartoon Limit* and was introduced by Mumford and Shah [MS88]. If the *Cartoon Limit* is additionally constrained to two regions, it is known as the *Chan–Vese model* [CV01]. Sandberg et al. further extend it to account for texture [SCV02].

There have been several approaches to combine the *Cartoon Limit* with image restoration. Jung et al. use the level set to solve the resulting functional and study the effect of Sobolev gradients [JCS+09]. Helmuth et al. view the combination from the other perspective and integrate piecewise constant segmentation into *Constrained Iterative algorithms* that are often used in image restoration [HS09]. Paul et al. introduce a model based on generalized linear models and Bregman divergences [CPS13]. They use piecewise constant image segmentation and restoration as an example application. All these works are based on a piecewise constant model in which the intensities do not change within the objects. However, microscopy images are often heterogeneous and require a piecewise smooth segmentation.

In this thesis, I extend the piecewise smooth *Mumford–Shah functional* to account for blurring. In [KTCW02], Kim et al. first introduced a convolution with a blur kernel into the piecewise smooth *Mumford–Shah functional*. In their work, the blur kernel has to be known. Later, Bar et al. and Zheng et al. included additional terms to learn the blur kernel during the joint segmentation / restoration [BSK04] [ZH06]. They all use variational algorithms to solve their energy models.

Kim et al. use *level sets* to represent the boundaries. A level set separates the image into a region above and a region below the level set. The topology, however, is not fixed. Therefore, as long as regions do not touch, it is possible to segment an arbitrary number of similar objects. But in many images, that is not a safe assumption since objects often touch. Using $log_2(M)$ level sets, it is possible to segment up to M regions [VC02]. In this case, an upper bound for the number of regions has to be defined initially. Furthermore, it leaves empty region statistically undefined even though these regions are still considered in the overall system of equations [BW04]. The *Region Competition framework* restricts the topology and defines regions as face–connected components. In return, the framework does not make any assumptions on the number of regions and their characteristics as it dynamically updates both.

Bar et al. and Zheng et al. do not actually segment an image, but rather follow Mumford and Shah's original idea and produce an edge map. In a segmentation, the edges surround the regions, thus, regions cannot overlap. Mumford and Shah's edge map, however, does not have this constraint on edges. It is possible for edges to just end in a region without hitting another edge. Their algorithms are primarily used for image restoration.

2.2 Region Competition framework

The *Region Competition framework* by Cardinale et al. [CPS12] is a discrete algorithm that regards segmentation as an optimization task. It tries to minimize the following functional:

$$\mathcal{E} = \mathcal{E}_{data} + \lambda \mathcal{E}_{length} + \alpha \mathcal{E}_{merge}. \tag{2.1}$$

The first two terms are the standard formulation of an energy term used in image segmentation. They are derived from the Bayesian Model:

$$p(\mathcal{C}, \theta | I) = \frac{p(I|\mathcal{C}, \theta) \cdot p(\mathcal{C}, \theta)}{p(I)}. \tag{2.2}$$

From the image I we want to retrieve the image's objects that are described by their contour $\mathcal{C}$ and their description given by the parameter set θ. That is most often done by maximizing the posterior $p(\mathcal{C}, \theta|I)$. Instead of solving the maximization problem, it is preferred to turn it into a minimization problem and solve the negative logarithm of Equation 2.2. The logarithm transforms above multiplication of likelihood $p(I|\mathcal{C}, \theta)$ and prior $p(\mathcal{C}, \theta)$ into the addition of energy terms seen in Equation 2.1.

The data term $\mathcal{E}_{data}$, also called external energy, is derived from the likelihood. It represents the probability that image I is recorded from a scene with the objects described by $\mathcal{C}$ and θ. It is often characterized by a distance between the image I and an image estimated from $\mathcal{C}$ and θ.

The probability of the segmentation a–priori is given by the prior $p(\mathcal{C}, \theta)$. It fulfills the same role as the length term that is also called internal energy. Both introduce prior knowledge about the image objects into the overall energy. For instance, it might be used to punish curvature to ensure that the objects do not have sharp corners.

The evidence $p(I)$ is omitted because it does not affect the segmentation of the objects. In Equation 2.2, it is used only for normalization purposes.

The third term, the merge term $\mathcal{E}_{merge}$, allows two adjacent region i and j to merge if they are similar enough. The similarity is measured with the Kullback–Leibler divergence of the empirical distributions of the regions and their merging:

$$\mathcal{E}_{merge} = \sum_{(i,j)>0 : X_i \sim X_j} H\left[D_{KL}(P_{X_i} || P_{X_i \cup X_j}) + D_{KL}(P_{X_j} || P_{X_i \cup X_j}) - \theta \right]. \tag{2.3}$$

$H(\cdot)$ is the Heaviside distribution that restricts the calculations to the regions involved. X_i and X_j are the adjacent foreground regions that want to merge. The adjacency is indicated by $X_i \sim X_j$. In the equation, P_X is the empirical intensity distribution of X; that is P_{X_i} is the empirical distribution of region i, while $P_{X_i \cup X_j}$ is the distribution for the merged region. The parameter θ is used to quantify similarity. The merge term is only favorable to the overall energy if both regions are similar enough to their merged region so that the sum is smaller than θ.

The main feature of the framework is that the number of regions and their characteristics are unknown beforehand. Because the number of regions has to be estimated in addition to their characteristics and boundaries, further regularization is needed in order

to identify the desired regions. For example, additional information is needed to decide whether two regions at different positions in the image but with identical statistics are actually one region or not. Since the algorithm is designed to work locally rather than globally, these two regions should get different labels.

Cardinale et al. use the discrete mesh topology of the image by defining that foreground regions are face–connected components using a 4–neighborhood in 2D and a 6–neighborhood in 3D. The background region has no topological restrictions. Due to the definition, region A and C in Figure 2.1 are different regions even if they would have the same statistical properties. The algorithm could merge these two regions only if the label of pixel 2d or 3c switches to either A or C.

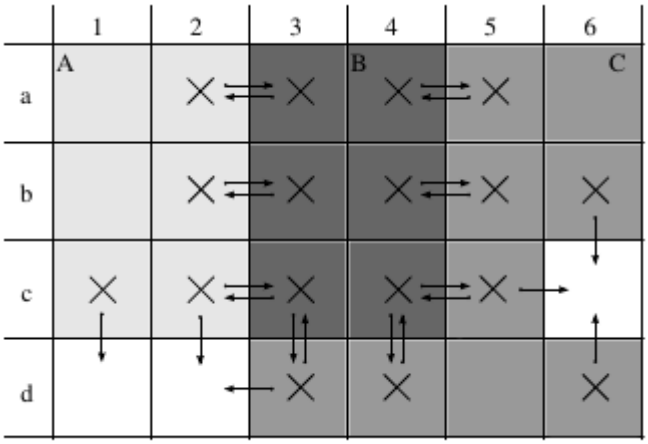


Figure 2.1: Example of a label image with three foreground regions A, B, C and a white background region. The particles are tagged with ×. The arrows symbolize possible contour movements. Source: [CPS12]

In order to reach a local minimum, the algorithm iterates through the following steps after initialization:

- Calculate the energy difference of the current pixel for the case that it would belong to another region. Only regions in the immediate neighborhood of the corresponding pixel are candidates.
- Decide how to move the boundaries in this iteration, so that the overall energy of the image is minimized. However, not all moves can be performed. Only valid moves are executed in order to prevent violations of the topological constraint that regions are face–connected components.
- Detect and execute changes in the topology, such as merges, splits or the creation of holes. These changes occur only in foreground regions since the background region has no constraints on its topology.

Label changes can occur only on the region boundaries. In order to speed up the calculations the energy is evaluated and, if possible, calculated only at these boundaries.

This inner contour is represented by particles. A particle is a zero–dimensional and indivisible object that stores several properties, such as position, label, energy, and relevant neighborhood information. Hence, if I use the term *particles*, I refer to the pixels on the inner contour whose labels can change. The term *pixels* refers to the actual pixels of the whole image.

Cardinale et al. designed this framework to efficiently segment images. Therefore, they heavily depend on local methods to calculate the different energy terms. FOr most energy models implemented, that is possible and, therefore, disconnects the time–complexity of the algorithm from the image size. In the case of piecewise constant deconvolution that is not possible as an imagewide convolution is required each iteration.

Furthermore, the local approach allows the particles to be processed in parallel. Because much of the current performance gain in computing is due to parallel cores, parallelized computations would further increase the algorithm's performance. Therefore, an implementation of a piecewise smooth deconvolving energy model in the *Region Competition framework* should use only local information and refrain from writing global variables.

2.3 Piecewise Smooth Segmentation

The principal idea of piecewise smooth segmentation is that the gray values do not change abruptly within an object but only do so at the object's boundaries. Mumford and Shah [MS88] developed a functional based upon this assumption, originally with the intent to use it for image denoising. In order to estimate the restored image $I_{est}(\vec{x})$ of an image function $I(\vec{x})$, they solve the following minimization task:

$$F^{MS}(I_{est}, \mathcal{C}) = \underbrace{\int_{\Omega} (I_{est}(\vec{x}) - I(\vec{x}))^2 \, d\vec{x}}_{fidelity} + \underbrace{\mu \int_{\Omega \setminus \mathcal{C}} (\nabla I_{est}(\vec{x}))^2 \, d\vec{x}}_{smoothness} + \underbrace{\lambda \cdot length(\mathcal{C})}_{internal} \rightarrow \min_{I_{est}, \mathcal{C}} . \tag{2.4}$$

The *Mumford–Shah functional* has the same background as the first two energy terms in Equation 2.1. The fidelity and the smoothness term from Equation 2.4 represent the likelihood; the internal term is the prior.

The first term in Equation 2.4 forces the restored image I_{est} to be as similar as possible to I by computing a L^2–norm between both images. The term is minimal if both images are equal. The second term only influences the values within the regions and not at the borders. It punishes large intensity variations and is used to enforce the piecewise smooth feature of the model. Taken alone, these two terms would segment the image into many small homogeneous regions, possibly one region for each pixel. Therefore, the third term is used to regularize this functional by penalizing the length of the contours. The parameters λ and μ are used to tune the influence of each term.

The *Mumford–Shah functional* is solvable but it is difficult to solve it efficiently. Trying to minimize the functional with variational methods leads to numerical errors at the boundaries $\mathcal{C}$ because the integral of the smoothing term is restricted to a part of the image. Therefore, Ambrosio and Tortorelli [AT90] suggest to replace the contour set with an indicator function that becomes 1 at the boundary and drops to 0 with

increasing distance. After replacing $\mathcal{C}$ in Equation 2.4 with the indicator function and restructuring it into a system of elliptical equations, the *Mumford–Shah functional* can then be solved, for instance, by an alternating minimization scheme [CS05].

Other authors (e. g., in [TYW01] and [VC02]) follow a similar idea by using a Heaviside function to indicate regions and a Dirac function to indicate the contour. The equation is then solved numerically with level sets. However, level sets are restricted to only two regions – foreground and background. In order to overcome this restriction, multiple level sets can be used, but this is computationally expensive. Furthermore, it leaves empty region statistically undefined even though these regions are still considered in the overall system of equations [BW04].

In both cases, large systems of equations have to be solved, usually $|\Omega| \times |\Omega|$ systems with $|\Omega|$ being the number of pixels in the image. As one of the features of the *Region Competition framework* is that the energy calculations are only required locally, a solution with variational methods is computationally too expensive.

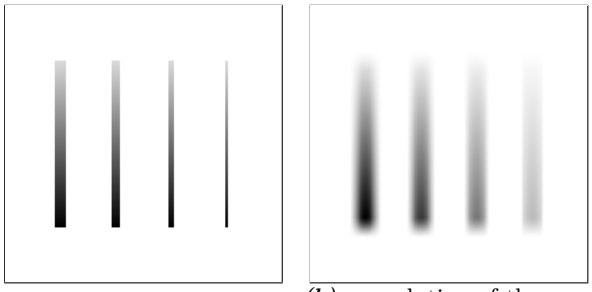
Furthermore, only a part of the *Mumford–Shah functional* needs to be solved, because the piecewise smooth energy model is supposed to be part of the *Region Competition framework*. A comparison of Equation 2.1 and 2.4 shows the similarity between them. However, the *Region Competition framework* evaluates each term in Equation 2.1 separately. The internal term of the *Mumford–Shah functional* is the same as $\mathcal{E}_{length}$. Therefore, the internal term can be omitted. Additionally, the energy term is evaluated in every iteration during which the contour is fixed. Hence, within an iteration, I only need to minimize with regard to the true image I_{est} and not the contour $\mathcal{C}$ as well. For this framework, Equation 2.4 can thus be simplified to:

$$\mathcal{E}_{data}(I_{est}) = \int_{\Omega} (I_{est}(\vec{x}) - I(\vec{x}))^2 \, d\vec{x} + \mu \int_{\Omega \setminus \mathcal{C}} (\nabla I_{est}(\vec{x}))^2 \, d\vec{x}. \tag{2.5}$$

To avoid solving the problem over the entire image, I concentrated on local solutions. Brox and Cremers studied the connection between minimizing the *Mumford–Shah functional* and methods based on Bayesian statistics in [BC07]. They found that it is possible to approximate Equation 2.5 with a maximum a-posteriori estimation in a Bayesian setting based on a local Gaussian kernel function with a fixed standard deviation of $\sigma = \sqrt{0.5}$ and a radius of $\rho = \sqrt{2\lambda}$. In [CPS12], Cardinale et al. further approximate this solution by cutting the low–energy part of the Gaussian kernel. Additionally, they replace the high–energy part with a local mean and thereby neglect the Gaussian weights for a faster calculation. Following from this, the external energy in the *Region Competition framework* is the squared distance between the gray value at the current pixel $\vec{x}$ and the local mean:

$$\mathcal{E}_{data}^{PS} = \sum_{i=0}^{M-1} \sum_{\vec{x} \in X_i} \left(\sum_{\vec{y} \in X_i \cap S_{\vec{x}}^{r_{mean}}} \frac{I(\vec{y})}{|X_i \cap S_{\vec{x}}^{r_{mean}}|} - I(\vec{x}) \right)^2. \tag{2.6}$$

The mean is calculated over a hypersphere S that has a radius r_{mean} and is centered at $\vec{x}$. The local mean takes only pixels with the same label i into account. It resembles



(a) artificial scene

(b) convolution of the scene in Figure (a)

Figure 2.2: An example of how the distortion of a microscope effects a scene. The intensities were reverted for illustration purposes.

the nearest neighbor smoother called *Running Mean* that is used in regression [FKL07] with the exception that it respects the region's boundaries.

Another possibility would be to approximate the image with locally defined smooth functions. There are a number of different bases that can be used like radial basis functions or splines [FKL07]. Buhmann analyzed the usage and properties for interpolation and approximation using radial basis functions [Buh03]. The basis function's values are defined by their distance to their node. The combination of all bases form a smoothed function of I. For piecewise smoothness, the behavior at the borders has to be decided. For instance, just like in the *Running Mean* implementation from Cardinale et al., it is possible to only use the nodes with the same label in order to get piecewise smooth functions. The smoothness is then defined by the number of nodes used. However, Buhmann postulates that they are better used for scattered data and recommends using tensorproduct B–Splines for meshed data, which is the data class images belong to.

B–Splines have a simple univariate form and, more importantly, a small compact support [dB90]. Just like with radial basis functions, a boundary behavior has to be defined. Contrary to radial basis functions, all pixels are used as nodes and the overall smoothness is defined by the basis function's smoothness. However, they are computationally more expensive than the previously mentioned *Running Mean*.

2.4 Deconvolution

According to the pinhole camera model, a sharp image means that the picture has to be shot using a infinitely small hole. In practice, however, an infinitely small hole would lead to black images. In real cameras, lenses are used to focus the light as a compromise between the need for light and sharp images, but that still leads to aberrations at the boundaries of objects. The same occurs in microscopes where lenses are additionally used to magnify small objects. Figure 2.2 shows an example of the effect of this distortion. Mathematically, this blurring is the result of a convolution with a kernel function:

$$I_{true} * PSF + \varepsilon = I_{seen} \tag{2.7}$$

In fluorescence microscopy, this kernel is commonly called point spread function (PSF) and I will use this designation for the remainder of this thesis.

Deconvolution is the inverse process to get the true image I_{true} from the seen image I_{seen}. Unfortunately, this is not a well–posed problem. Well–posedness was defined by Jacques Hadamard [Had02] and means that a unique solution exists which does not change much if the initial condition varies slightly. If a problem is not well–posed (or ill–posed) then the problem is either not defined well enough and no unique solution exists or the problem is very sensitive to noise.

Deconvolution is either blind or non–blind. In blind deconvolution the kernel is unknown. This introduces an additional difficulty, namely, it is unclear whether a gray value change is part of the actual scene or because of the acquisition process. Therefore, additional restrictions and assumptions have to be made.

Knowing the point spread function makes the problem significantly easier. But it still remains ill–posed. Convolution is a multiplication in Fourier space, hence deconvolution is a division in Fourier space. However, this amplifies the noise ε because the kernel has frequencies close to zero. Therefore, a deconvolution with known kernel is still an ill–posed problem, because just a small addition of noise can lead to very different results.

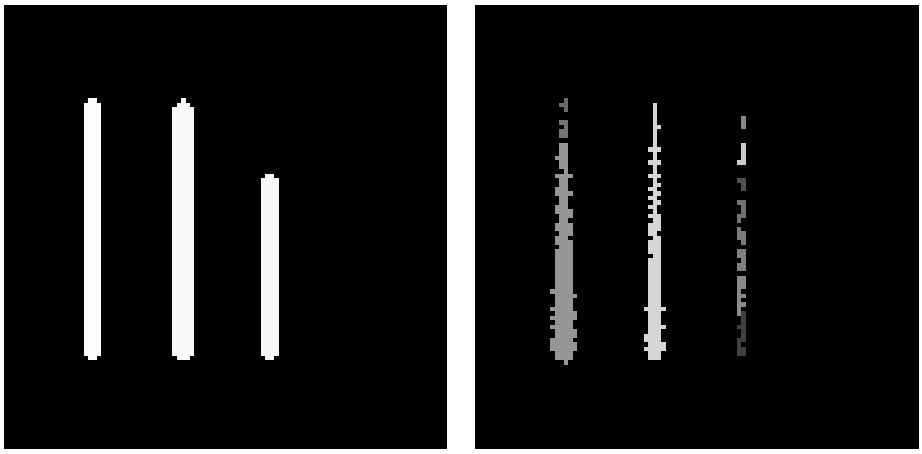
When the PSF is known, we have the advantage of knowing when the true image is found. Because, in the absence of noise, the convolved true image is the same as the seen image. This is used by *Constrained Iterative algorithms* [Jan05] [WSSB13]. These algorithms commonly start with a first estimation of the true image, often the seen image itself, and then iterate through the following steps:

- Calculate an error criterion by comparing the convolved estimated image to the seen image, e. g., $(I_{est} * PSF - I_{seen})^2$.

- Improve the estimated image by calculating a correction with the help of the error criterion.

Due to the ill–posedness of the problem, the estimation step is constrained to produce semantically possible images.

In [HS09], Helmuth and Sbalzarini suggest to use the model information and the resulting label image from segmentation to stabilize the deconvolution. In addition, this improves the segmentation results as further knowledge about the image–formation process is included. The *Deconvolving Active Contours algorithm* [HS09] is a *Constrained Iterative algorithm* with restrictions that expect piecewise constant objects and limit oscillating edges. The true intensities of the objects are estimated with the label image from the previous iteration. Because of that, a classical deconvolution can be avoided. Solely convolution is used in the forward model $I_{est} * PSF$ that simulates the image acquisition process. The piecewise constant deconvolution model, which is already implemented in the *Region Competition framework*, uses the same concept.



(a) piecewise smooth segmentation ***(b)*** piecewise constant deconvolution

Figure 2.3: Segmentation results of a piecewise smooth segmentation ($\lambda = 0.005$, $\alpha = 0.1$, $\beta = 0.002^{\dagger}$, $r_{mean} = 3$) and a piecewise constant deconvolution ($\lambda = 0.047$, $\alpha = 0.1$, $\beta = 0.1^{\dagger}$). Additionally to the columns, result (a) also segments the blurring and discounts low intensity regions as background whereas the piecewise constant deconvolution result in Figure (b) inverts the blur but splits due to the gradually changing intensities. The thinnest line is difficult to detect due to length regularization. This line is also the darkest, because it is most effected by the convolution.

2.5 Piecewise Smooth Deconvolving Segmentation

Both of the implemented models — piecewise smooth without deconvolution and piecewise constant deconvolution — have their shortcomings. The piecewise smooth segmentation without deconvolution fails at heavily distorted images that often occur in microscopy images because it interprets the blur as part of the object. Furthermore, it has problems segmenting low intensity objects in which the convolution causes an additional intensity flux and further dims the object. The piecewise constant deconvolution model on the other hand interprets all blur as part of the image–formation process and reverses some of it. But due to the varying intensities within the object, a piecewise constant model may split the object into several regions. Figure 2.3 shows exemplary segmentation results of the image shown in Figure 2.2(b) to illustrate the problems of the two models. In order to improve results in blurred microscopy images I combine piecewise smooth segmentation with deconvolution.

Kim et al., Bar et al., and Zheng et al. have combined these two before in [KTCW02], [BSK04], and [ZH06] respectively. All of them extend the *Mumford–Shah functional* by convolving the approximation with the point spread function PSF before comparing it

[†] β is a parameter weighting an outward balloon flow. This balloon flow presses the contour outwards according the gray values in the input image I. Cardinale et al. added it to realize a greater flexibility in regard to the initial label image [CPS12].

to the image:

$$F^{MS}(I_{est}, \mathcal{C}) = \int_{\Omega} (PSF * I_{est}(\vec{x}) - I(\vec{x}))^2 \, d\vec{x} + \lambda \int_{\Omega \setminus \mathcal{C}} (\nabla I_{est}(\vec{x}))^2 \, d\vec{x} + \mu \cdot length(\mathcal{C}) \rightarrow \min_{I_{est}, \mathcal{C}} . \tag{2.8}$$

Deconvolution alone is ill–posed. However, the smoothing of the *Mumford–Shah functional* works just like a Tikhonov [TA77] or a Total Variation [ROF92] regularization, hence no further adjustments have to be made in order to ensure the solution's stability.

Initially, Kim et al. introduced this equation and solved it using level sets. While Kim et al. assumes the kernel function to be known, Bar et al. and Zheng et al. enhance the approach from Kim et al. to also cover blind deconvolution and to make the solution more robust with regard to the initialization. They use the concept presented in [AT90] to represent the edge image and to solve Equation 2.4. But just like before for the *Mumford–Shah functional* alone, they solve a large system of equations in each iteration in order to minimize 2.8.

Because I prefer a solution that works locally, and thus avoids unnecessary energy difference computations, I propose to use the same approach as it is used in [HS09]. However, unlike the algorithm by Helmuth et al., the *Constrained Iterative algorithm* assumes that the image objects are piecewise smooth instead of piecewise constant.

2.6 Summary

All in all, I follow the idea of the already existing piecewise constant deconvolution for the algorithm in order to achieve a piecewise smooth deconvolving model. It is also a *Constrained Iterative algorithm* and I use the following external energy term as an error criterion that will be minimized:

$$\varepsilon_{data}^{PSDec} = \sum_{\vec{x} \in \Omega} (I_{est}(\vec{x}) * PSF - I(\vec{x}))^2 . \tag{2.9}$$

I_{est} is a piecewise smooth estimation of the true image I_{true} from Equation 2.7. There are several ways to compute I_{est} from I as I explained in section 2.3. To avoid calculating huge systems of equations, I decided for the Gaussian approach by Brox and Cremers [BC07] with the additional approximations from Cardinale et al. [CPS12]. While approximation with locally defined functions would have also been a solution, they are computationally more expensive and less founded.

Therefore, at first the algorithm starts with an initial label image. From then on, each iteration starts by calculating I_{est}. With that image, the *Region Competition framework* will evaluate each energy term from Equation 2.1 separately and then decide how to move the boundaries $\mathcal{C}$. At the end, the algorithm stops in a local optimum. The label image is the segmentation result that returns the indicated objects. The image I_{est} shows the deconvolution result, that is the restored version of the input image I.

3 Implementation

In this chapter, I further detail the algorithm and how I implemented it. In Section 3.2 I concentrate on the generation of the piecewise smooth function I_{est} from Equation 2.9 as it influenced many of the design decisions later on. In Section 3.3 I explain the general structure of the implementation and in the following section I describe the changes I made to the the already existing framework. From the beginning, the main problem had not been just whether a piecewise smooth deconvolving model is possible, but also if it is possible in an efficient way. Hence, at the end of this chapter, I describe the additions I made for performance reasons.

3.1 Goals

Before I started designing the implementation of the energy model I wrote a list of goals:

1. quality of the results
2. performance
3. adherence to the characteristics of the *Region Competition framework*
4. simple expandability and adaptability

This list aided me when I faced design decisions and also later when I evaluated my results.

At the end, I want to segment blurred microscopy images with the help of this energy model. My result should be a label image, in which the image objects are accurately tagged, and a restored image that shows me the actual scene. Intensity changes within objects should not change the segmentation result. Intensity changes due to blurring, on the other hand, should be discarded during the estimation of the restored image I_{est}. That is my primary goal.

However, speed is a factor that I cannot dismiss completely. The already implemented energy models are very fast. Hence, a speed loss is only acceptable if I equally gain something in quality. But sometimes, much can be gained by risking a slight quality loss if the performance win is very high. Therefore, the performance goal follows right after quality. During implementation, I often needed to weight these two goals against each other.

The *Region Competition framework* was implemented without a specific energy model in mind. The data term and the length term of Equation 2.1 can be chosen with an argument. The framework works locally on particles and, thus, a large part of the image can be ignored. Furthermore, the framework was built with parallel computing

in mind. Much of the current performance gain in computing is due to parallel cores, hence processing the particles in parallel can give the *Region Competition framework* quite a performance boost. I designed the algorithm with these features in mind.

Simple adaptability is a goal that should be set as a priority for every program. No non–trivial program is perfect and this one is not likely to be an exception. The implementation has to be understandable either to myself at a later time or to other developers by following coding style standards and documenting the code. The *Region Competition framework* uses the Insight Segmentation and Registration Toolkit [ITK02]. Therefore, I follow the ITK coding style that the main ITK developers ask of other developers to adhere to [ITK05].

3.2 Intensity Estimation

In order to calculate the error criterion in Equation 2.9, I need to estimate the true image I_{true} from Equation 2.7. In 2.9, this estimated image is referred to as I_{est}. I continue to use this notation in this chapter.

Per definition, I_{est} has to be a piecewise smooth function. However, it is not possible to simply approximate the given image I by solving the *Mumford–Shah functional* or its shortened form from Equation 2.5. The convolution leads to an intensity flux from high–intensity to low–intensity regions. A simple approximation would not show the same intensity differences that were in the original scene I_{true}. Therefore, aside from calculating a piecewise smooth function, I also have to estimate the true intensities.

My goal is to estimate an unknown piecewise smooth function I_{est} from a known function I. The unknown function I_{est} should equal the true scene I_{true} from Equation 2.7 in the case of a correctly labeled image. Therefore, I want to minimize $(I_{est} * PSF - I)^2$. That can be done by iteratively modifying I_{est} until convergence is reached. But, for performance reasons, I prefer to just iterate once. The general intensity estimation is thus structured as follows:

1. Generate a first estimation of I_{est}, which for easier understanding I name I_{est}^{-1}.
2. Calculate correction factors with the aid of the original image I, the beforehand calculated estimated image I_{est}^{-1} and its convolution.
3. Apply the correction factors to calculate I_{est} that then can be used in Equation 2.9.

The two remaining questions are where to introduce the piecewise smoothness and how to calculate the correction factors. As decision aid, I simulated the intensity estimation process with each possibility on the region shown in Figure 3.1. I varied the region's intensities, its intensity gradient and its size. The results I show in Appendix A are exemplary for the overall results.

I simulated the segmentation process of a smaller initial region growing outwards, which represents a standard case in the *Region Competition framework*. A frequently used initialization process assumes the foreground regions to be brighter than the background and places the initial foreground regions at the brightest spots. During the segmentation, these regions then grow outwards.

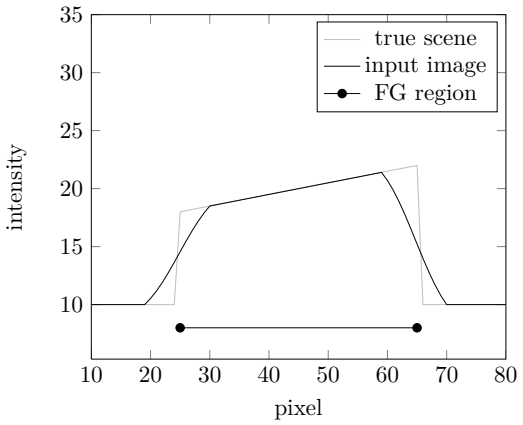


Figure 3.1: Illustration of the 1D image I use in the following simulations. I varied the region's intensities, its intensity gradient and its size.

The original image is blurred with a Gaussian kernel with a diameter of 13 pixels and a variance of 2. Furthermore, I added a Gaussian noise using a signal–to–noise ratio from 2 to 15. The distance between the true boundary and the current one is marked with *offset*. Depending on the region size, I chose a maximal offset between -3 to -10. A negative offset means I assume the current region to be within the real foreground region.

I do not evaluate the difference over the whole foreground region. Instead I concentrate on the particles at the region's border because within the *Region Competition framework*, the energy difference is also evaluated only at the region's boundaries.

With the *offset* I simulate a segmentation process in which I start with a smaller region that slowly grows outwards. This growth stops at the true region's boundaries. The main reason is that outside the true region, the estimated intensity would be compared to the background intensity. As the estimation is influenced by its neighbors, it would be higher than the background intensity and there would be a significant difference.

However, that difference is not unwanted. If the estimation process would perfectly fit the true intensities regardless of a correct segmentation, then the segmentation process would be stopped while the boundaries are at a wrong position. Furthermore, the piecewise smooth deconvolving model overcomes intensity jumps by smoothing the underlying intensities according to its smoothing parameter r_{mean}.

Therefore, if the boundary overshoots the true segmentation, I do not want the estimated intensity to be as close as possible to the true intensity. Instead, there should be an error akin to the smoothing I want. However, the error that I want there to be is difficult to quantify. Therefore, during the simulation, I only use negative offsets.

At the end of each simulation, I evaluate at each particle how well each option minimizes $(I_{est} - I_{true})^2$. In the result figures A.1 to A.6, the left column shows the mean of the squared distance, while the right column shows the standard deviation.

3.2.1 Adding Piecewise Smoothness

There are several options with which the piecewise smooth characteristic of the resulting function can be ensured:

- Generate a piecewise smooth first estimation I_{est}^{-1}. In that case, the piecewise smoothness has to be guaranteed during the other steps of the intensity estimation. Calculating the correction factors and applying them must not destroy this characteristic.

- Add the piecewise smooth characteristic afterwards by approximating the finished estimated image I_{est} depending on the label image.

- Adapt the whole process and use piecewise constant patches similar to the piecewise smooth segmentation in the *Region Competition framework*. There, for each particle the mean from each region is estimated in a local patch around it. The entirety of these patches make the piecewise smooth approximation. It assumes that within each patch the intensities are constant. Instead of having to estimate intensity values for each pixel around the particle, I just need to calculate a few intensities depending on the number of regions in the patch. Thus, this estimation procedure requires a smaller support than the other two solutions.

3.2.2 Calculation of Correction Factors

In order to calculate the correction factors I use the following relationships:

I know that the input image I is the convolved and noisy version of I_{true}. The latter is unknown. I want to calculate an image I_{est} that is to be as similar as possible to I_{true}.

In addition, I know the effect of a convolution with the known PSF from my first estimation I_{est}^{-1} and its convolution $J = I_{est}^{-1} * PSF$. Thus, I can infer the relationship between the first intensity estimation I_{est}^{-1} and the corrected intensity estimation I_{est} from their convolved versions J and I. The standard way to analyze such a relationship is regression.

In linear regression, the relationship is defined by two correction factors α and β [FKL07]:

$$I = \alpha + \beta \cdot J \rightarrow I_{est} = \alpha + \beta \cdot I_{est}^{-1} \tag{3.1}$$

However, the relationship is described very general and does not use further information about the image's layout. To also consider the regions, a multidimensional linear regression can be used. A multidimensional regression has one parameter per region stored in the correction vector $\vec{w}$ [FKL07]:

$$\vec{I} = \vec{J} \cdot \vec{w} \rightarrow \vec{I}_{est} = \vec{I}_{est}^{-1} \cdot \vec{w} \tag{3.2}$$

The vectors $\vec{I}$, $\vec{J}$, $\vec{I}_{true}$ and $\vec{I}_{est}$ contain the pixels surrounding the current particle sorted by their region. For instance if two regions are present, then $\vec{I} = (I_0, I_1)^T$ with I_0 being all the pixels belonging to the background region 0 and I_1 being all the pixels from

Region 1. The correction vector can then be estimated with the following equation:

$$\vec{w} = (J^T \cdot J)^{-1} \cdot J^T \cdot I \tag{3.3}$$

According to the Gauss–Markov theorem, this estimator is unbiased and matches the expectations with a minimum of variance [FKL07].

An alternative to using regression are the methods Cardinale applied to calculate the region intensities in his piecewise constant deconvolving model [Car13]. For the background region, he uses the difference introduced by the blurring to estimate the intensities:

$$I_{est} = median(I - (J - I_{est}^{-1})) \tag{3.4}$$

Because the intensities are needed in a piecewise constant scenario in which just one intensity value per region is required, he uses the median to extract one value from all the region's pixels. At the same time, the median discards outliers that can occur due to incorrect region estimation.

For the foreground regions, he uses the following ratio:

$$\frac{I}{J} = \frac{I_{est}}{I_{est}^{-1}} \tag{3.5}$$

For further stability, he subtracts the beforehand calculated background intensity from each value. However, as can be seen in Figure A.1, it does not have the same influence in a piecewise smooth setting. He uses the median of the ratio $\frac{I}{J}$ for the same reasons that I explained previously. In a piecewise smooth setting, the median is not necessarily required. It is possible to estimate the intensity pixelwise, with the mean of the correction factors, and with the median of the correction factors.

3.2.3 Simulation Results

All of the following simulation results are presented in Appendix A.

I started with the calculation of the correction factors. As a general structure, I used the first option proposed in Section 3.2.1: First, the algorithm calculates a piecewise smooth approximation of I as a first approximation of the intensities. Then, these intensities are corrected.

In Figure A.4, I show the results of these methods. Additionally, I also compare Equation 3.4 and 3.5 using a pixelwise correction, the mean and the median of the surrounding correction factors in Figure A.2 and A.3. In each case, the radius of the hypersphere for the *Running Mean* from Equation 2.6 is used as the neighborhood in which the mean or median is calculated.

All in all, aside from using the correction factors pixelwise, all the options are similar. Using Equation 3.4 with a mean instead of a median showed the best results during the simulations.

Using this result, I continue to decide the general structure, that is when and how introduce the piecewise smooth characteristic. Figure A.5 compares the various options presented in Section 3.2.1 using $mean(I - (J - I_{est}^{-1}))$ to estimate the intensity.

	`ps`	`psDec` with $mean(I - (J - I_{est}^{-1}))$	`psDec` with $median\left(\frac{I}{J}\right)$
Circle	1.03%	1.29%	0.1%
4 Lines	3.01%	3.75%	1.64%

Table 3.1: Comparison of three segmentations using the *Region Competition framework* with a piecewise smooth non–deconvolving and two versions of the piecewise smooth deconvolving energy model. The table shows the percentage of wrongly labeled pixels. For the experiments I used the setting described in Section 4.1. The piecewise smooth segmentation without deconvolution and the piecewise smooth deconvolution with $median(I/J)$ use the parameters given in the evaluation section. For the piecewise smooth deconvolution using the $mean(I - (J - I_{est}^{-1}))$, I use the following parameters. Circle: $\lambda = 0.00009$, $\alpha = 0.1$, $\beta = 0.0002^{\dagger}$, $r_{mean} = 5$; 4 Lines: $\lambda = 0.003$, $\alpha = 0.1$, $\beta = 0.00003^{\dagger}$, $r_{mean} = 3$.

The third option — constant region intensities in patches around the particles — fails because J is calculated from an image with less difference between the regions. The convolution is done over a small patch, in which one half is the foreground mean and the other is the background mean. However, the wrongly labeled foreground pixel raise the background's mean higher, while in the other methods, the function would eventually drop to the actual background. Furthermore, the convolution is only done over a small patch, whereas in the other methods, the convolution also considers all the pixel that influence the convolution.

The other two options are very similar, so I compared their complexity. If I calculate a piecewise smooth approximation beforehand, the intensity estimation will first calculate a *Running Mean* and then calculate $mean(I - (J - I_{est}^{-1}))$ for the particle.

On the other hand, if I estimate the intensities first, I use the input image I as a first estimation for the image's intensities. The true intensities are calculated with $mean(I - (J - I))$. Afterwards, the piecewise smooth feature is added. However, I do not need to add the piecewise smoothness separately because it is already a part of the intensity estimation process as the mean of the correction factors is directly used as intensity.

However, when I implemented it in the *Region Competition framework*, the results were unsatisfactory. It has troubles segmenting areas with a low intensity and mislabels more pixels than a piecewise smooth segmentation. Table 3.1 shows the number of wrongly labeled pixels for two example images for the piecewise smooth model without deconvolution and two versions of the piecewise smooth deconvolving energy model. While the median of the ratio showed less promise during the simulations, it gave better results in the *Region Competition framework*.

I corrupted the convolved true scene with Gaussian noise. However, as can be seen in Figure A.3, both the mean and the median of the ratio sufficiently smooth the noise. While the mean is better in very noisy images, the median deals better with outliers.

The latter is especially helpful, when the boundary moves outside the actual fore-

[†] β is a parameter weighting an outward balloon flow. This balloon flow presses the contour outwards according the gray values in the input image I. Cardinale et al. added it to realize a greater flexibility in regard to the initial label image [CPS12].

ground region, which is a case I did not cover during the simulation. In this case, I want the estimated intensities to be different from the true intensities. Outside the true foreground region, the mean is more strongly influenced by the low intensities of the background than the median is. Hence, I decided to use the median of the ratio as a correction factor and estimate the intensities thusly:

$$I_{est} = I_{est}^{-1} \cdot median\left(\frac{I}{J}\right) \tag{3.6}$$

Figure A.6 are the results of the general structure simulation using the median of the ratio as a correction factor. Of all three options, the first is the best, provided the piecewise smooth characteristic can be ensured. The second suffers a little bit from the added approximation and underestimates the true intensities. Therefore, I decided for the first option and, first, approximate the input image I according the label image to get a piecewise smooth approximation from I. From the approximation, I then estimate the true intensities.

Using this algorithm, I need to ensure that the piecewise smoothness does not get lost and that each operation passes it along. The division might lead to numerical errors if a pixel in J is zero. In this case I divide through the smallest, non–negative number above zero instead of zero. I use the `itk::DivideImageFilter` to quickly calculate the correction factors elementwise and this filter already implements exactly the solutions I suggested above. Therefore, the resulting function is piecewise smooth.

The median operates only within the regions and, therefore, on smooth functions. Furthermore, it can be used as a smoother [FKL07]. The piecewise smooth characteristic is thus passed on to the elementwise multiplication. This operation also ensures piecewise smoothness as long as its factors are piecewise smooth.

In summary, I calculate the intensities I_{est} from Equation 2.9 with the following algorithm:

1. Calculate a piecewise smooth approximation of the input image I using the *Running Mean* from Cardinale et al. [CPS12] that I described in Section 2.3.

2. Convolve the previously calculated first estimation to get J.

3. Use both intermediate results to solve Equation 3.6 for each pixel.

3.3 General Structure

The *Region Competition framework* uses interfaces to ensure a standard communication between the main algorithm and the different energy models. The interfaces defines the following methods:

- `EvaluateEnergyDifference()`: This function is called for each particle with its current label and a possible candidate label. The function returns the particle's energy difference that a label switch would cause. It also evaluates $\mathcal{E}_{merge}$ and returns whether or not the two involved regions are similar.

- `PrepareEnergyCalculation()`: This function is invoked once in the beginning of the algorithm and then never again. It can be used to initialize member variables for later use. These preparations cannot be done in the constructor of an energy class, because at that time, the input image I and the label image are not yet initialized.
- `PrepareEnergyCalculationForEachIteration()`: Unlike the previous function, this function is called once in the beginning of every iteration. It can be used for actions that have to be done once in every iteration or that should be done over the whole image. An example is the intensity estimation in a piecewise constant setting because the model is based on the statistics of the entire regions.
- `AddPoint()`, `DeletePoint()`: These functions are called if a pixel is added to or removed from a region. It is used to update global region statistics. For instance, the framework uses them if two regions merge or split.
- `SwitchPoint()`: This function is very similar to the previous ones except that it is called if a particle's label is switched. Unlike `AddPoint()` and `DeletePoint()` it knows both labels — its previous and its new one. Therefore, this class can be used to update local statistics at a particle. `AddPoint()` and `DeletePoint()` have to be used more carefully, because they claim that the current particle either belonged or belongs to the background, even if that is not the case.

If a function is not implemented, then its default from the interface is called. The interface also gives access to the input image I, the current label image and some global statistics like the region mean, but not, for instance, to a list with all the current particles.

`EvaluateEnergyDifference()` has to return two pieces of information: the current particle's energy difference and whether the two regions should merge or not.

When a particle switches its label, not only its own intensity will change, but also all the intensities and convolved intensities within the support of the point spread function PSF. Hence, the energy is not just evaluated at the particle, but also within the PSF support, and `EvaluateEnergyDifference()` returns the result of the following equation:

$$\begin{aligned} \Delta\mathcal{E}_{data}^{PSDec} &= \sum_{\vec{y}\in S_{\vec{x}}^{r_{PSF}}} \mathcal{E}_{new}(\vec{y}) - \mathcal{E}_{old}(\vec{y}) \\ \mathcal{E}_{old}(\vec{y}) &= \left(J_i(\vec{y}) - I(\vec{y})\right)^2 \\ \mathcal{E}_{new}(\vec{y}) &= \Big(\left(J_i(\vec{y}) - \left(I_{est_i}(\vec{x}) - I_{est_j}(\vec{x})\right)\right) * PSF - I(\vec{y})\Big)^2 . \end{aligned} \tag{3.7}$$

Similar to $S_{\vec{x}}^{r_{mean}}$ from Equation 2.6. $S_{\vec{x}}^{r_{PSF}}$ is a hypersphere around the current particle $\vec{x}$ with the radius r_{PSF} that equals the radius of the PSF support. i and j are region labels and the particle switches from region i to region j. To calculate the energy difference, I need the intensity at the particle $\vec{x}$ for both possible regions and the convolved intensities within the support of the kernel function PSF of the current state.

I detail the process to compute the intensities in Section 3.2. The following algorithm extends it to include the calculation of the convolved intensities $J_i(\vec{y})$:

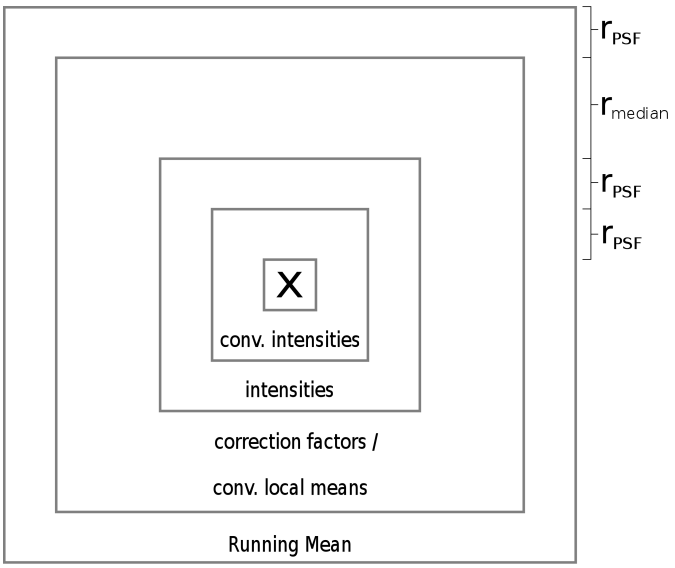


Figure 3.2: Illustration showing the support areas in which the intensity estimation process needs intermediate information. Nearly each previous result needs to be calculated in a slightly larger area to ensure accuracy. For instance, the *Running Mean* has to be calculated in an area around the current particle x with the radius $3 \cdot r_{PSF} + r_{median}$.

1. Calculate the piecewise smooth estimation of I within the area with the radius of $3 \cdot r_{PSF} + r_{median}$ around the particle

2. Convolve the piecewise smooth estimation within the same area, which gives a correct convolution in the area with the radius $2 \cdot r_{PSF} + r_{median}$

3. Calculate the correction factors as was described in Section 3.2.2. As a *Running Median* is executed, the correction factors are accurate within a radius of $2 \cdot r_{PSF}$.

4. Calculate the intensities using the median–filtered correction factors and the first piecewise smooth estimation.

5. Convolve the intensities. Because a convolution adds and multiplies values within the kernel support, the result is now only correct within the hypersphere $S_{\vec{x}}^{r_{PSF}}$.

As one can see, nearly each previous result needs to be calculated in a slightly larger area to ensure accuracy. Figure 3.2 shows an example that illustrates this problem. For the recalculation of the intensity, this issue is only slightly smaller. I just need one intensity value, but that erases only the last convolution step. The overall complexity and area size remains.

This large area provides quite a problem in designing an algorithm. At first, I wanted to present an algorithm that is completely parallelizable and only works locally. However, if each particle is evaluated separately, a lot of data is calculated over and over again because of the overlap that I illustrate in Figure 3.3.

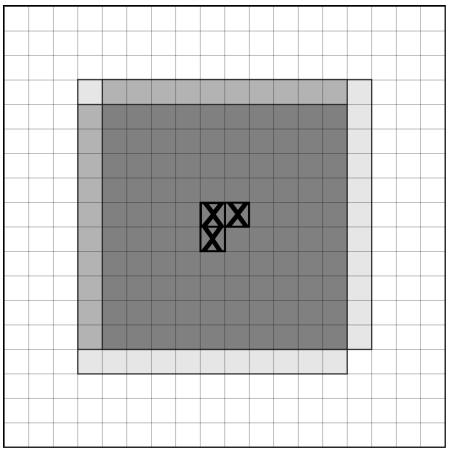


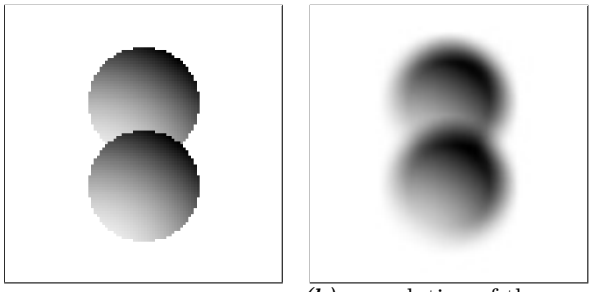
Figure 3.3: Example showing three particles and the maximal support area in which they need to calculate the *Running Mean*. The areas overlap strongly. The darker the gray color, the more often the mean at this pixel has to be calculated.

Therefore, I tried to reuse as much as possible by storing current information. While I retained the local character of the algorithm, it cost me the parallelization capability. With a hash set, I remember which information is still accurate. When a particle changes its label, I update it. However, a label switch inevitably affects all the pixels that previously depended on it for their values. The effect is small especially for the pixels further away. But particles are clustered border pixels that move along a line together. Therefore, the affected areas of the different particles most likely overlap over many pixels, leading to error accumulation. This error also grows in each iteration. In its sum, the error cannot be ignored, so the affected pixels have to be recalculated eventually.

There are two possibilities to address this. One option is to just remove the indices of the affected pixels and to recalculate the values if they are needed again. However, that would lead to the same overlap problem mentioned previously. I would gain a lot in complexity and little in speed in comparison to the first proposed algorithm.

The other possibility is to use `SwitchPoint()` to not just update the current particle but also its surrounding pixels. At the end of the iteration, this approach will ensure that all of the intermediate results are completely up–to–date again. It also removes the need for the hash set and the correctness checks. Unlike the hashset–solution, this algorithm continues to be semi–parallelizable. During the evaluation step, the algorithm is parallelizable but not in the update stage. In comparison to the first proposed algorithm, this version mostly gains speed because it calculates the area only around changes and not at every evaluated particle. It usually lowers the number of particles calculated to half or more, even at the start when most particles change. However, many pixels' values are still calculated repeatedly.

The one possibility to completely avoid all the overlap is to avoid the particlewise calculations and instead make all computations over the entire image. It means to give up the local character. But for instance, for Figure 1.1, which is a 512×386 image, it is



(a) artificial scene

(b) convolution of the scene in Figure (a)

Figure 3.4: An example in which the global similarity measure from Equation 2.3 fails to keep the two regions separated. Globally, their statistics are similar, but locally they are not. Thus, the similarity has to be calculated locally. Figure (b) shows the convolved scene. Due to the convolution with the point spread function, an intensity flux further lessens the differences between the regions.
The intensities were inverted for illustration purposes.

about twice as fast as updating per particle.

All in all, each possibility has its disadvantage. Updating using `SwitchPoint()` fails at the early iterations, because many information are recalculated time after time. However, updating at the beginning of every iteration using `PrepareEnergyCalculationForEachIteration()` is very slow at the end when only a small number of particles change.

Therefore, for the sake of speed, I combine both. When many particles change or when regions merge or split, I update every iteration. At later iterations, when the sum of the possible overlapping pixels is smaller than the image, I update at each particle.

3.4 Modifications of the Region Competition framework

In order to accommodate above algorithm within the *Region Competition framework*, only a few changes had to be made to the framework. The energy class is handled alongside the others, likewise the parameters for the energy class. I extended the `energy_ext_name` parameter that decides which external energy term is used. *psDec* has to be chosen if the framework should use this piecewise smooth deconvolving energy model. I have not added a new parameter for the median radius. Instead I use the same radius as the mean.

I also adapted the merge term $\mathcal{E}_{merge}$. Originally, the Kullback–Leibler divergences are calculated over the whole region as I explained in Equation 2.3. However, that would fail in case of piecewise smooth energy models. Figure 3.4(a) shows two overlapping circles with slowly changing intensities. Globally, both region statistics are similar, but locally they are not. For their own piecewise smooth energy model, Cardinale et al. suggest to solve Equation 2.3 not over the global regions, but only over the parts covered by the hypersphere $S_{\vec{x}}^{r_{mean}}$ [CPS12]. This is the area in which the local mean is computed.

I propose the same for the piecewise smooth convolved energy model. However, the

convolution evens out the intensity differences at the border as a comparison between 3.4(a) and its convolved version in Figure 3.4(b) shows. As a result, instead of calculating the Kullback–Leibler divergences on the input image I, I calculate it on the intensity image I_{est}. The intensity image is an estimation of the true scene before the image was blurred and the intensity differences are reinforced.

3.5 Performance Modifications

The current implementation needs about 22 seconds for an image 100×100 pixels. The time was averaged over the segmentations of the example images shown in Figure 4.1 using the setting and the parameters described in Section 4.1. As a comparison, a piecewise smooth segmentation without deconvolution in the *Region Competition framework* needs in average 0.3 seconds and a piecewise constant deconvolution needs about 1.6 seconds. For a 100×100 image, the current implementation is about 14 times slower than the piecewise constant deconvolution and about 80 times slower than the piecewise smooth model without deconvolution. In this section, I evaluate some ideas to enhance the performance.

The most complex operation is the estimation of the intensities and their convolution. These calculations are done once in every iteration for every pixel. The intensity is also recalculated with nearly the same procedure once for every particle per iteration. While the image–wide calculation becomes very expensive for larger images, my first implementation still needed about 1000 milliseconds to recalculate the intensity for a particle.

In order to avoid this recalculation altogether, I propose to interpolate the intensities from the surrounding values. During the calculation of $I_{est_j}(\vec{x})$, all the intensities of the current state are known. For the interpolation, I look at each neighbor and gather the intensities whose pixel belong the region j. Then, I take the median of them to reduce the numbers of intensities to one. I use the median instead of the mean for the same reason as during the calculation of the correction factors in Section 3.2.2. It is possible to extend the neighborhood, if better quality is desired. However, I use this modification primarily to enhance the performance and, therefore, I decided to infer $I_{est_j}(\vec{x})$ only from its immediate face–connected neighbors. The error that is introduced by this simple interpolation does not have a huge influence on the overall result because it is just a temporary result. Figure 3.5 shows the difference between a correctly estimated intensity and an interpolated intensity using the simulation setting from Section 3.2. Unlike the other simulations, in which I compared the estimated intensities to the true intensities, I now compare two estimated intensities. Therefore, I do not restrict myself to the area within the object. Once the boundaries grow outside the true region, however, the effect that the estimated intensities drops to the background intensity can be seen. There is only a noticable difference in this area. The interpolation uses old intensity values and, thus, drops slower to the background intensity than explicitely calculating does. Within the *Region Competition framework*, this difference shows up as two wrongly labeled pixels when using interpolation. The number was averaged over all example images. In one example, using interpolation actually

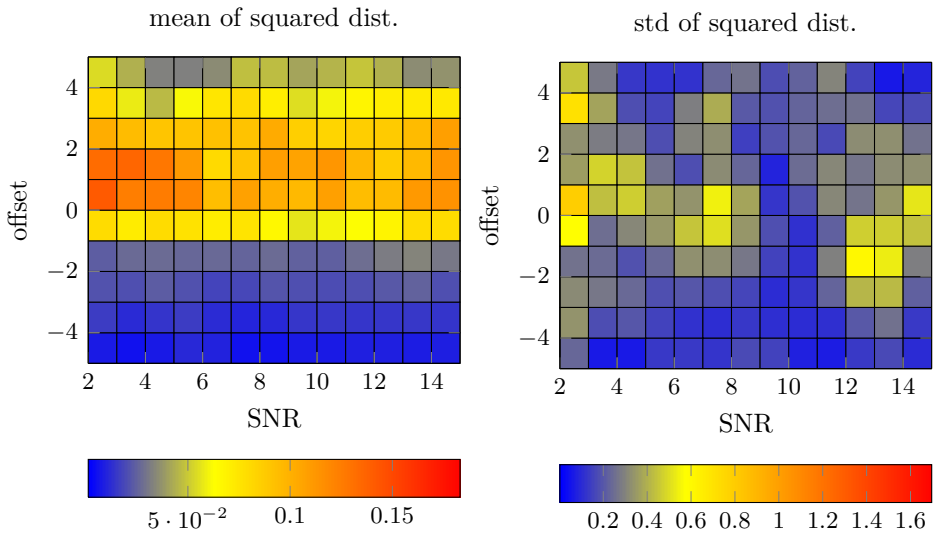


Figure 3.5: Quality evaluation of using interpolation. Using the simulation setting from Section 3.2, I compare using interpolation against explicitely calculating the intensity at the particle using the candidate label. The shown results were made on a 50 pixel image. The foreground region starts at a gray value of 30. The foreground's gray values change with a gradient of 0.1.

showed a slightly better result. However, using interpolation enhances the performance. Previously, `EvaluateEnergyDifference()` needed about 1000 milliseconds per particle. Using interpolation, this reduces to under 10 milliseconds per particle.

With growing image size, the image–wide calculations become more and more expensive. Figure 3.6 shows how much each step approximately needs of the overall time. The proportions may change depending on the mean / median radius, the kernel size and the image size.

The function that calculates the local median first collects all correction factors within the median radius. Then, it retrieves the median using `std::nth_element`. This function returns the n–th element as if the entire list was sorted without actually sorting it. Each step has a time–complexity of $\mathcal{O}\left(|\Omega| \cdot r_{median}^{d}\right)$ with $|\Omega|$ being the number of pixels in the image and d being the image's dimension. There is not much that can be done to speed up this step.

The mean calculations also have a time–complexity of $\mathcal{O}\left(|\Omega| \cdot r_{mean}^{d}\right)$. They can be sped up using the information of the pixel's neighbors. If the neighbor belongs to the same region, then the mean is computed over nearly the same area as its neighbor's. These two areas differ only at time pixels that lie further away. Only these differences need to be added or subtracted to the neighbor's mean. I illustrate this in Figure 3.7. The difference masks can be calculated a–priori, which reduces the overall time–complexity of the mean calculations to $\mathcal{O}(|\Omega|)$. However, experiments showed that it is not fast but actually twice slower independently of image size and mean radius. For the implementation, I use the `itk::NeighborhoodIterator` to iterate over the whole image,

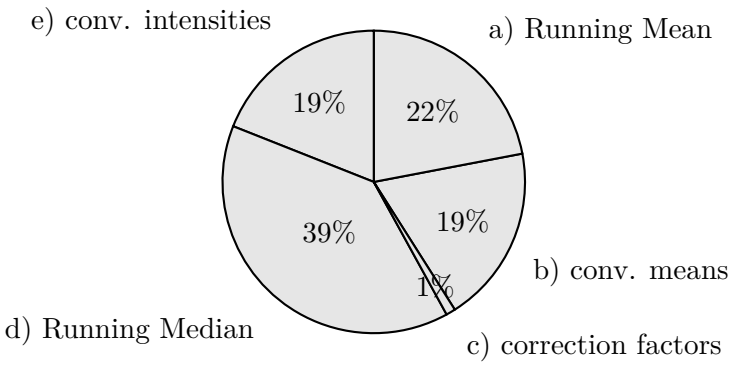


Figure 3.6: Pie chart showing the proportion of each part of the intensity estimation process. The entire process calculated over the whole image needs about 0.05 seconds for a 100×100 image.

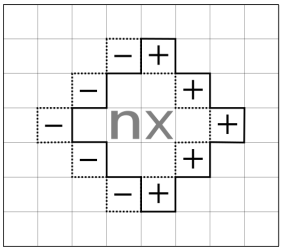


Figure 3.7: Illustration showing the mean areas of two neighboring pixels. They are nearly identical with the exception of some pixels at the edge of the areas. These differences are indicated by $-$ and $+$. The current particle is x. The mean is stored by the sum of gray values and the number of pixels that lie in the mean area and share the same label. In order to calculate its mean, it is possible to take the values from the neighbor n, provided they share the same label. Then, only the pixels indicated by $-$ have to be subtracted, while the new pixels indicated by $+$ have to be added.

while simultaneously having access to the neighbors. I assume that the performance loss results from the implementation of the NeighborhoodIterator.

The local mean is also computed during the intensity recalculation. There, I assume that the current particle belongs to a neighboring region instead. If I recalculate the changed local means from scratch, the time–complexity is $\mathcal{O}\left(r_{mean}^{2d}\right)$. However, I know the surrounding values and I can reuse these results. I traverse the hypersphere and update each pixel according to its label by adding or subtracting the current particle. Thus, I only need to calculate the local mean for the particle itself and not for every surrounding pixel as well. I also use this process when I update the intensities at every changed particle using `SwitchPoint()`. In both cases, the time–complexity reduces to $\mathcal{O}\left(r_{mean}^{d}\right)$.

For the convolutions, it is faster to multiply the kernel in the Fourier Space. Instead of a time–complexity of $\mathcal{O}\left(|\Omega| \cdot r_{PSF}^2\right)$, a convolution using the Fast Fourier transformation has a complexity of $\mathcal{O}\left(|\Omega| log |\Omega|\right)$. However, this only helps the performance if it is used

on large areas because the transformation to Fourier Space needs time as well. Therefore, updating the values on the particles and the intensity recalculation still use the original convolution.

Just like during the energy evaluation, it would be nice if I could avoid some or all of the steps in Figure 3.6. I cannot use the interpolation again, because the intensity values are not temporary during the update stage of the algorithm. The interpolation error would accumulate with each iteration.

Cardinale et al. reuse the old intensities and their convolution to calculate the intensities of the current iteration instead of always starting at the beginning as I do [CPS12]. With the same method, I would avoid the *Running Mean* and its convolution. However, their algorithm is a piecewise constant deconvolution. The mean value and the intensity are valid for an entire region and an error has less influence. In a piecewise smooth setting, in which I evaluate the intensities at each pixel, I cannot simply reuse the old values. If a particle changes its label, it would continue to use the old intensity value to calculate its current intensity. The label switch would not affect the correction factor so much, because I use the median over the whole area. However, it changes I_{est}^{-1} from Equation 3.6. The resulting intensity would always be too high or too low depending on the intensity difference between the two regions in the area. Furthermore, the error accumulates over time. In the end, I decided to refrain from recycling the old intensities and instead always estimate the intensities from scratch.

In segmentation algorithms that span many iterations, it is a standard procedure to enhance performance by skipping calculation steps and move the borders for a while first.

The question is when to skip and when not to. In its standard setting, the algorithm initializes its regions with small circles on bright spots. Therefore, at first the region border has to be moved into the rough vicinity of where the actual boundaries are. It is a general movement in the right direction, so skipping a few iterations is possible. At the end of the algorithm, however, the details for the boundaries are calculated. In order to ensure quality, it is better to calculate all the required values.

This can be realized with *Simulated Annealing* [RN03]. In *Simulated Annealing*, an energy level slowly sinks with time. I use the energy as the probability with which an iteration is skipped. I do not need to implement my own version of *Simulated Annealing* as the *Region Competition framework* already uses it to lessen the amount of particles that are allowed to change bit by bit. Instead I count the number of evaluated and changed particles and use its ratio as an energy for the *Simulated Annealing*. For further control, I parameterized it with `energy_psdec_intensity_rate`. If it is set to 0, then the intensities and their convolution are calculated in every iteration.

As I explain in Section 3.3, I switch to updating only on changed particles if that is faster. This switch occurs at the end, when only a few particles still change their labels. At this point, my algorithm primarily adapts to details in the image or oscillates between two possible states. In order to get the details, I need an exact estimation of the intensities. Therefore, I do not just rely on the *Simulated Annealing* to lower the energy, but completely stop skipping iterations when I update particlewise.

From these suggestions, I implemented all but the reuse of old intensities. Updating the mean using old values or calculating the convolutions using the Fast Fourier Trans-

formations speed up the program without any cost to the quality. Interpolation and skipping iterations changes the result though. For interpolation, the error is only slight, because the error is very small and does not accumulate over time.

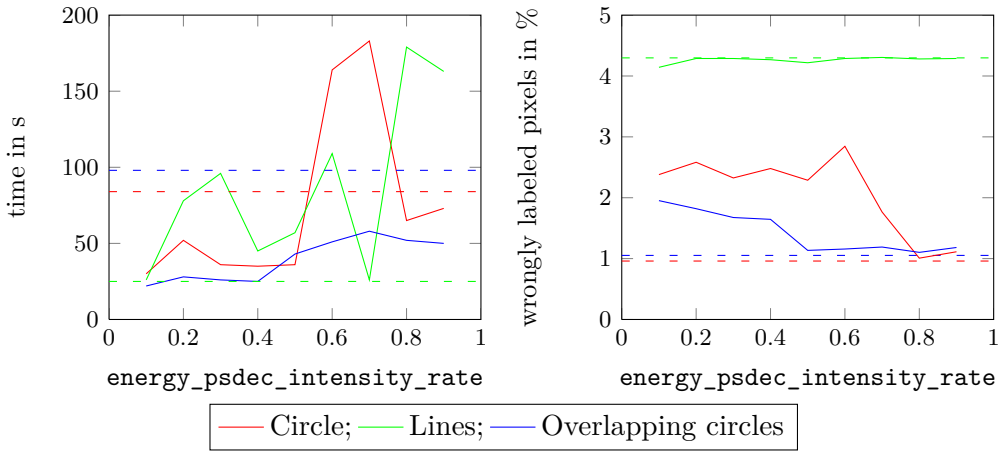


Figure 3.8: Evaluation of skipping imagewise intensity estimation every few iterations. The different colors represent the different example images shown in Figure 4.1. The dashed lines represent the time and the quality when all values are updated at every iteration.
All results were calculated on 200×200 images without noise. I used the following parameters. Circle: $\lambda = 0.00005$, $\alpha = 0.1$, $\beta = 0.007^{\dagger}$, $r_{mean} = 10$; Lines: $\lambda = 0.0003$, $\alpha = 0.1$, $\beta = 0.00006^{\dagger}$, $r_{mean} = 6$; Overlapping Circles: $\lambda = 0.01$, $\alpha = 0.1$, $\beta = 0.07^{\dagger}$, $r_{mean} = 10$

However, it is very difficult to quantify the overall gain or loss if I skip the intensity calculations a few iterations. Figure 3.8 compares the performance and the quality while I skip the intensity calculations in more and more iterations. If the intensities are calculated less, then the blurring due to the convolution is not correctly inverted but rather shown as a piecewise smooth change in the intensity image. The algorithm tries to correct this in the later iterations, but the border is often not close enough to the true border. Furthermore, the *Simulated Annealing* energy is too low to still allow that much change. This effort also increases the amount of iterations and, therefore, the overall time. All in all, this idea is not stable enough to easily evaluate the parameter's influence, so I did not use it for my further results.

Using the modifications proposed in this section, the implementation of the piecewise smooth deconvolving model is six times faster than the original version. A more detailed evaluation can be found in Section 4.3.

[†] β is a parameter weighting an outward balloon flow. This balloon flow presses the contour outwards according the gray values in the input image I. Cardinale et al. added it to realize a greater flexibility in regard to the initial label image [CPS12].

4 Evaluation

In this chapter, I evaluate my implementation of a piecewise smooth deconvolving energy model. At first, I describe the setting I used for the evaluation. I evaluate the energy model and its implementation with regard to the goals I set in Section 3.1. In order to assess the quality of the results, I examine image segmentation and image restoration separately. In Section 4.3, I evaluate the performance using the modifications I explain in Section 3.5 *Performance Modifications*. At last, I take a look at the implementation itself and analyze how well the implementation of the piecewise smooth deconvolving energy model fits into the *Region Competition framework*.

To assess the quality of an implementation of an energy model, it is usually compared with other implementations of the same model. Unfortunately, that was not possible as I had no copy of the programs from Kim et al. [KTCW02], Bar et al. [BSK04], or Zheng et al. [ZH06]. Their respective papers also showed little quantitative data to which I could compare my implementation and the authors did not respond to my inquiries for source code. A complete reimplementation would have been outside of the scope of this thesis.

Therefore, I compared the piecewise smooth deconvolving energy model with the models it is composed of. For the image segmentation, I use the implementations from the *Region Competition framework*. For further comparisons of the *Region Competition framework* to other algorithms using the same models, I refer to [CPS12] and [Car13].

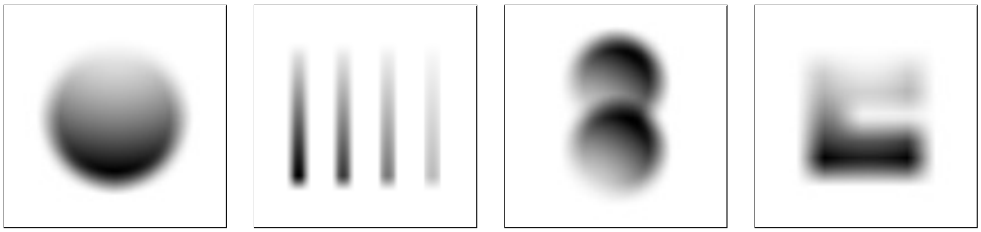
The reconstructed images I attain with a piecewise smooth energy model cannot be used for comparison in image restoration. It sees the blurring as a smooth change and not as something that should be reversed. Therefore, for image restoration, I compare the resulting intensity images to the results of the piecewise constant deconvolution. Additionally, I also compare it to the results of the Wiener Deconvolution Filter, which is a standard deconvolution method used in Image Processing. I use the ITK class `itk::WienerDeconvolutionFilter`.

4.1 Design

For the quality evaluation, the label images and the reconstructed images are calculated from artificial 100×100 pixel images that can be seen in Figure 4.1. Additionally, I distorted the images with Gaussian noise using the SNR levels 2, 5, 10, and 15. The signal–to–noise ratio is described by:

$$SNR = \frac{\sigma_{signal}}{\sigma_{noise}} \tag{4.1}$$

For the algorithms that are part of the *Region Competition framework*, I choose the optimal parameters for the input images without noise. In this evaluation, *optimal*



(a) Circle *(b)* Lines *(c)* Overlapping Circles *(d)* Overturned U

Figure 4.1: Artificial example images. I used a different point spread function for each image that can be seen in Table 4.1. The intensities were inverted for illustration purposes.

	diameter in pixel	σ
Circle	11	5
Lines	7	3
Overlapping Circles	7	3
Overturned U	13	6

Table 4.1: Table showing the different parameters for the Gaussian kernel I used as a point spread function.

means a minimum of wrongly labeled pixels. I reuse the parameters for the noisy images as the energy difference is calculated as a squared distance. In image processing, the squared distance is usually used to counterbalance Gaussian noise. For the Wiener Deconvolution Filter I can set the noise variance. I calculate it at the same time I add the noise during the image's creation.

More than 20 parameters can be set to adjust the *Region Competition framework*. The whole list is described in [Car13]. I only vary three:

- λ (`energy_coeff_length`): This parameter is used to weight the length term $\mathcal{E}_{length}$ (see Equation 2.1). I used the same length term from Kybic and Krátký [KK09] for all experiments. The higher the parameter, the smoother I want the boundary to be. However, it also shortens the boundary and therefore shrinks the region.

- β (`energy_coeff_balloon`): This parameter sets the influence of an outward balloon flow. This additional energy is used by Cardinale et al. to counteract the length term for local algorithms.
  Initially, the *Region Competition framework* starts with many small regions at the brightest spots and the border should move outwards to cover the whole object. However, in large objects, the initial local data term for piecewise smooth algorithms is close to zero. At the beginning, the length term drives the segmentation and the region shrinks. In these cases, β has to be set higher than λ to enlarge the region until the region's borders come close to the true boundaries in which the data term starts to have an influence.

	psDec	ps	pcDec
Circle	$\lambda = 0.0003$, $\beta = 0.0008$, $r_{mean} = 5$	$\lambda = 0.005$, $\beta = 0.05$, $r_{mean} = 7$	$\lambda = 0.06$, $\beta = 0.78$
Lines	$\lambda = 0.002$, $\beta = 0.0005$, $r_{mean} = 3$	$\lambda = 0.005$, $\beta = 0.002$, $r_{mean} = 3$	$\lambda = 0.047$, $\beta = 0.1$
Overlapping Circles	$\lambda = 0.004$, $\beta = 0.02$, $r_{mean} = 3$	$\lambda = 0.0007$, $\beta = 0.01$, $r_{mean} = 3$	$\lambda = 0.5$, $\beta = 0.9$
Overturned U	$\lambda = 0.001$, $\beta = 0.02$, $r_{mean} = 6$	$\lambda = 0.004$, $\beta = 0.02$, $r_{mean} = 6$	$\lambda = 0.2$, $\beta = 0.3$
Embryos	$\lambda = 0.4$, $\beta = 0.5$, $r_{mean} = 17$	$\lambda = 0.09$, $\beta = 0.1$, $r_{mean} = 17$	$\lambda = 0.1$, $\beta = 0$
Endosomes	$\lambda = 0.005$, $\beta = 0$, $r_{mean} = 8$	$\lambda = 0.007$, $\beta = 0$, $r_{mean} = 8$	$\lambda = 0.08$, $\beta = 0$

Table 4.2: Table showing the different parameters I used for the results shown in Appendix B.

A positive β moves the border along high–intensity regions of I while a negative β does the reverse. A negative β moves the boundary outwards if the intensities in I are dark.

- r_{mean} (`energy_local_window_radius`): This is the radius of the hypersphere in which I calculate the *Running Mean* for the first piecewise smooth estimation I_{est}^{-1} (see Equation 2.6) and the *Running Median* of the correction factors (see Equation 3.6). It directly influences the smoothness of the estimated intensity function and its convolution. A large radius causes both to be more stiff and, therefore, filters more details.
  The radius r_{mean} should have at least the same value as r_{PSF}. Due to the convolution with the PSF, the intensities average out and detail information gets lost. If the radius r_mean is smaller than r_{PSF}, then the intensity estimation process does not have enough information to recover good approximation values.

Table 4.2 shows the optimal parameters I used for the quality evaluation. For all images, I used 0.1 as the merging parameter α, except if otherwise stated.

To show the usability in real images, I also segmented two microscopy images. However, no ground truth exists for these. Therefore it is hard to quantify how good the different algorithms perform in comparison and I use them primarily as examples.

In order to evaluate the second goal – speed – I use the same artificial images as before. However, this time I do not vary the noise but rather the image's size. The convolution kernel grows in proportion and, therefore, also the radius of the mean and

the median for the piecewise smooth energy models.

As I am not interested in the segmentation results here, I do not calculate an optimal set of parameters again but instead settle for a local optimum.

The overall runtime heavily depends on the number of iterations the algorithm needs until convergence. Therefore, it is not enough just to give an overall time. Instead in addition to the maximum number of iterations, I use $\frac{time}{iterations}$ to get the average time an iteration needed to compare the speed of the different algorithms for energy evaluation. Since the Wiener Deconvolution Filter is not an iterative algorithm, it will not be compared with the others.

For all computations I use an Intel Core i7-3770 with 8 cores at 3.4 GHz with 8MB L3 cache. The *Region Competition framework* itself is single–threaded, so I only used one core at a time. It is possible to run the framework using the GPU in order to parallelize the particle's processing but I did not use this feature.

4.2 Quality

4.2.1 Image Segmentation

Figure 4.2 shows the percentage of wrongly labeled pixels per image and noise levels. In most cases, the piecewise smooth deconvolving model leads to a better segmentation than the piecewise smooth non–deconvolving and the piecewise constant deconvolving model.

Without deconvolution, the piecewise smooth energy model has trouble dealing with the blurring. The segmentation overshoots the true boundaries. Additionally, as the object's corners are blurred as well, it is unable to segment sharp corners in convolved images. That is further limited by the high length term that is needed to constrain the segmentation to the actual foreground.

The piecewise smooth energy model also has more problems to distinguish two close, blurred objects. At the boundaries, they are locally similar and, thus, the algorithm merges the two regions. The third column of Figure B.5 shows the piecewise smooth non–deconvolving segmentation of such a case.

In addition, the piecewise smooth energy model struggles with noise. For a high noise level, such as $SNR = 2$, the region splinters into many small regions, sometimes even in the background. However, the origin of this problem lies with the evaluation design I described in Section 4.1. I set the smoothing parameter r_{mean} by minimizing the error in an image without noise. It is possible to attain a better segmentation with a higher smoothing parameter that would counteract the noise more strongly.

The piecewise constant deconvolution inverts the blur. However, it assumes regions to have a constant intensity. Unlike the piecewise smooth models, it does not calculate the energy difference using local informations, but rather estimates an intensity for each region. The model is less affected by the noise due to this global character. However, it is also the main source of its shortcomings.

A piecewise constant segmentation, regardless of deconvolution or not, fails to distinguish the overlapping circles in Figure B.5 as the circles share similar global statistics. Even worse, a segmentation of a inhomogeneous image leads to fragmented regions if it

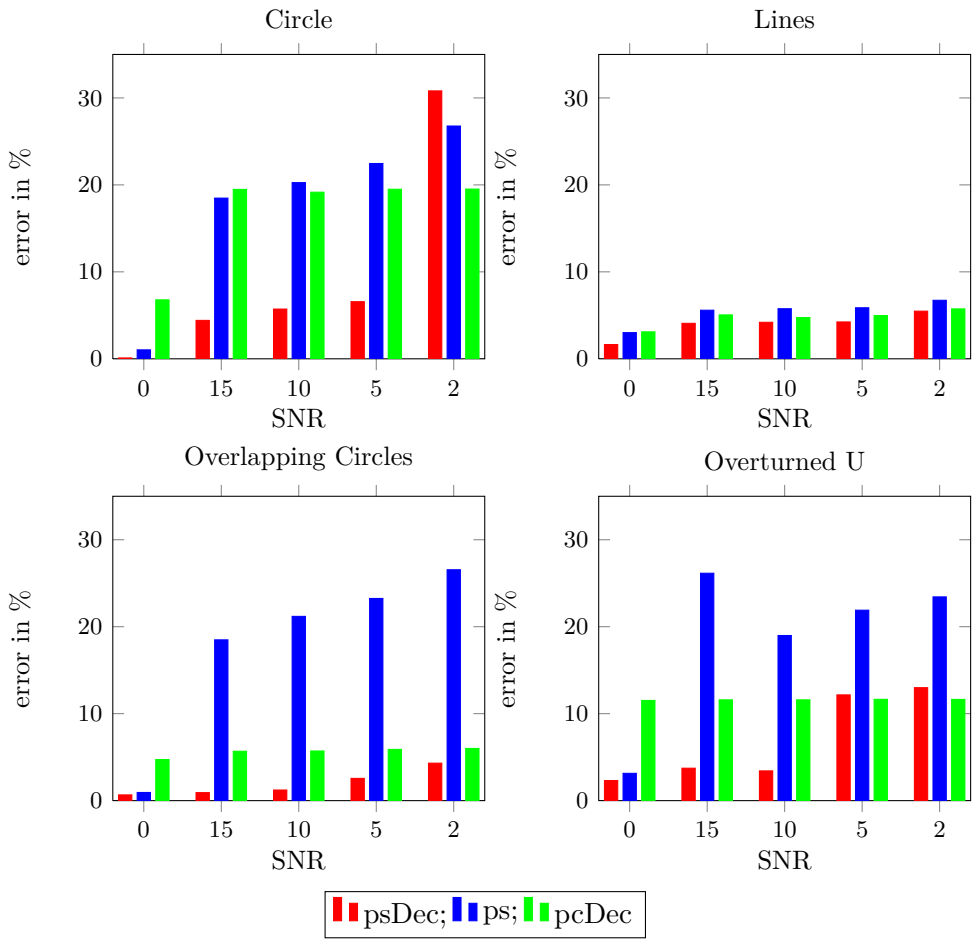


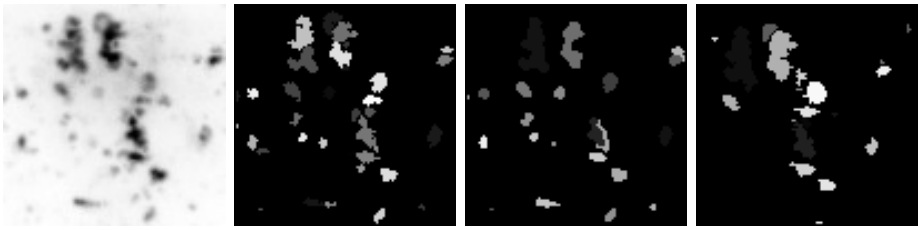
Figure 4.2: Bar plots showing the percentage of mislabeled pixels over the per example image and noise level. For the segmentation, I used the parameters from Table 4.2. The resulting label images can be seen in Appendix B.

depends on global statistics. A good example of this can be seen in Figure B.3. However, the latter can be countered with the balloon flow energy. While the piecewise constant deconvolving energy model does not require the balloon flow for its actual purpose, it helps with the segmentation nonetheless. A high balloon flow pushes the boundaries outwards and, thus, keeps the region together. For the piecewise constant deconvolving model, the balloon flow parameter β is especially high in example images with wide regions, such as the images containing circles.

The piecewise smooth deconvolving energy model inverts the blur, just as the piecewise constant deconvolution, but uses local information to calculate the energy difference. Hence, it is able to deal with inhomogeneous objects. The mislabeled pixels mostly occur in regions with sharp borders, where the curvature energy has a larger influence, and in regions with low intensities. Especially, the latter is amplified by noise. The reason has nothing to do with the low intensity, but rather its similarity to

the background.

Similar to the piecewise smooth non–deconvolving model, the problem with the noise also originates from the evaluation design. A higher smoothing parameter would segment the noisy images better. Nevertheless, it is less affected than its non–deconvolving counterpart, because the energy difference is calculated over the convolved intensities and not just the local mean. The intensity estimation contains several smoothing operations, such as the *Running Mean*, the *Running Median* and the convolutions, and each operation further reduces the noise.



(a) input image *(b)* `psDec` *(c)* `ps` *(d)* `pcDec`

Figure 4.3: Segmentation results of endosomes. This example is extracted from the introductory example from Figure 1.1. The parameters I used can be found in Table 4.2. The figures has been inverted for illustration purposes.

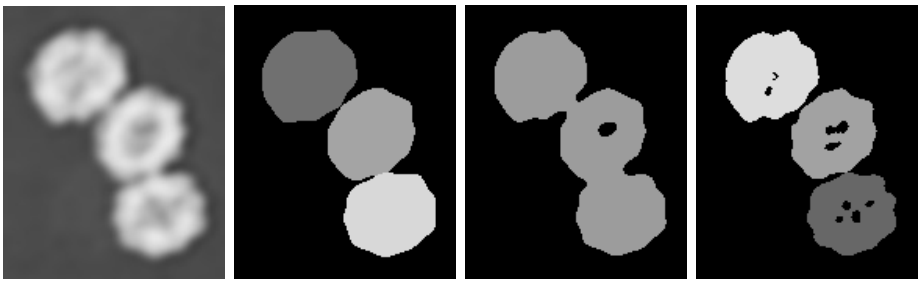
Figure 4.3 shows the segmentation results of a part of the introductory example. It shows a large number of endosomes that lie closely together. Due to the blurring, it is difficult not only to separate which endosomes belong together, but also how the endosomes actually look like. I have no ground truth for this and not enough microscopy experience, so I am not sure what the best segmentation is and how many endosomes there are.

The piecewise constant deconvolution is troubled by the piecewise smooth character of the background. In the areas, where the background is brighter (e.g., close to the top of the image), the algorithm segments the endosomes more loosely. In other areas, on the other hand, where the foreground objects are rather dark and not significantly brighter than the bright background area, the piecewise constant deconvolution is unable to segment these objects. If the parameter were set in order to label these objects as well, then it would also segment the bright background area.

Similar to its problem with the overlapping circles, the piecewise smooth segmentation without deconvolution cannot separate the various endosomes. However, it finds the darker endosomes that the piecewise constant deconvolution missed.

In pure numbers, the piecewise smooth deconvolution counted the most endosomes. However, especially for the endosomes in the upper part of the image, I assume that there are even more objects than the seven my implementation counts.

Figure 4.4 shows another real–life example. It shows three embryos that lie closely together. Originally, the image was not blurred. I convolved it with the estimated point spread function from the previous example. For a human, it is still easy to see the three different embryos. However, due to the blurring, the embryos seem very similar in those areas where they touch.



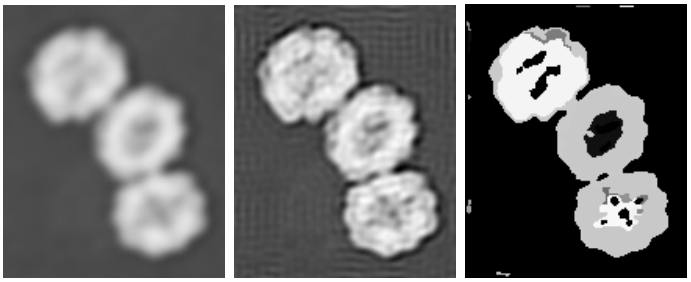
(a) conv. input image *(b)* `psDec` *(c)* `ps` *(d)* `pcDec`

Figure 4.4: Image segmentation results of embryos. The original image did not contain any blurring. I convolved it with the PSF from the endosome example. The parameters I used can be found in Table 4.2. Source: Fiji Sample Image *Embryos* [Fij]

The piecewise smooth segmentation completely fails for this example. The embryos always merge. Changing the merge parameter α does not help either as it prevents, for instance, the initial left region of the embryo to merge with the right region. Only forbidding merging and setting the initial regions appropriately might solve the problem.

The piecewise constant deconvolving model can separate the different regions, but has problems with the inner, darker part. However, setting the balloon flow higher in order to segment the inner part as well, merges the embryos again.

The piecewise deconvolving model can completely segment all three embryos.



(a) input image *(b)* deconvolution *(c)* segmentation

Figure 4.5: Segmentation results of restored embryos. For this segmentation, I first reconstructed the image using a Wiener Deconvolution Filter and then segmented the result with a piecewise smooth non–deconvolving energy model. Nonetheless, the framework was unable to discern the three embryos.

This example is perfect to test whether a deconvolution followed by a segmentation shows the same results as the combined version, because the Wiener Deconvolution Filter inverts the blur without amplifying noise. Therefore, I first reconstruct the image using a Wiener Deconvolution Filter and then segment the result with the piecewise smooth non–deconvolving energy model. The results can be seen in Figure 4.5. Nonetheless, I was still unable to separate the three embryos without splitting them in many regions or segmenting the artifacts introduced by the Wiener Deconvolution Filter.

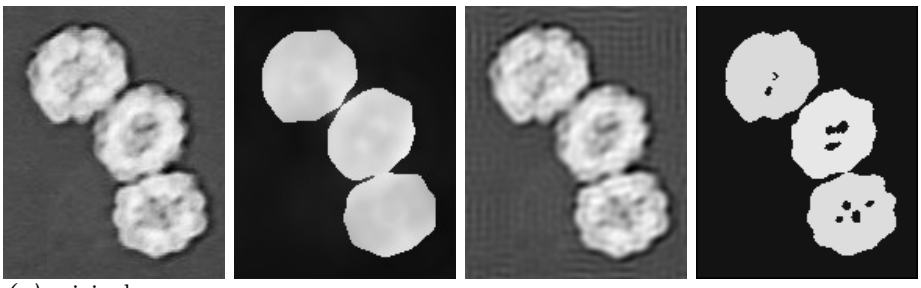
4.2.2 Image Restoration

The restored images from the artificial example images can be found in Appendix B.

While for non–noisy images, the Wiener Deconvolution is indisputable better, it clearly fails once noise is involved. All Wiener deconvolutions were made with the ground truth noise variance. Nonetheless, the noise is amplified so much that the underlying objects are nearly invisible as Figure B.2, B.4, B.6, and B.8 show.

For the deconvolving models of the *Region Competition framework*, I optimized the parameters with regard to the segmentation. An optimization that minimizes the squared distance between the reconstructed image and the ground truth might result in a better reconstruction of the input image. This is especially true for the piecewise constant images because it uses the same intensity over the whole region. An optimization geared to reconstruction would split the objects into many small regions that in sum would be the reconstructed piecewise smooth object.

Regarding the piecewise smooth deconvolving model, the optimization would show a similar result for the artificial example images irrespective of its focus. The quality of the restoration results crucially depend on the segmentation as the restored images of the overturned U show in Figure B.8. In this example, the segmentation fails. In the same areas, where pixels are mislabeled, the intensities are also not enough restored. However, for the other images, it even accounts for the noise surprisingly well. The accuracy of the segmentation shows a greater influence than the noise.



(a) original, non–blurred image

(b) `psDec`

(c) `Wiener`

(d) `pcDec`

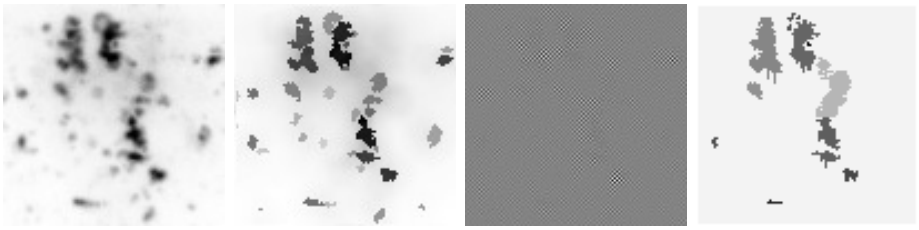
Figure 4.6: Image restoration results of embryos. The original image did not contain any blurring. I convolved it with the PSF from the endosome example. The parameters I used can be found in Table 4.2. Source: Fiji Sample Image *Embryos* [Fij]

Figure 4.6 shows the restored results of the embryo image. The Wiener Deconvolution Filter gives the best results for this example. It only introduces some ringing artifacts but otherwise sharpens the boundary and the inner structures of the embryos.

Since the objects are inhomogeneous, the piecewise constant deconvolution is unable to recover the inner structure. A large number of constant patches are needed to invert the blur. The segmentation mislabeled the inner structure of the embryos. Therefore, they share the same intensity as the background.

During the segmentation evaluation, the piecewise smooth algorithm labeled each

embryo correctly. However, while this helps to accurately invert the blur at the edges, it has the opposite effect on the structures inside the embryos. They are smoothed even further. If the inner structure is the area of interest, then the parameter should be chosen with regard to segmenting the inner structure correct.



(a) input image *(b)* `psDec` *(c)* `Wiener` *(d)* `pcDec`

Figure 4.7: Image restoration results of endosomes. This example is extracted from the introductory example from Figure 1.1. The parameters I used can be found in Table 4.2. The figures has been inverted for illustration purposes.

Figure 4.7 shows the restored images of the previously introduced endosomes example. As before during the segmentation, I do not have a ground truth and little experience in microscopy, which makes a comparison difficult.

The endosomes seem to have little internal structure. Therefore, both convolving energy models of the *Region Competition framework* have the same chance to restore the input image accurately. It also means that each restoration capability solely depends on the intensity estimation capability and the segmentation result. The piecewise smooth deconvolution gives a better estimation of the true scene because the segmentation process captured more of the endosomes than the piecewise constant deconvolution.

In this example, the Wiener Deconvolution Filter fails completely. For some reason, probably underlying noise, the Wiener Deconvolution Filter introduces strong artifacts in form of a chess pattern. An iterative algorithm that would prevent the intensity values from becoming negative would fair better.

4.3 Performance

For the performance evaluation, I use the modifications I propose in Section 3.5 with two exceptions. I do not calculate the mean via its neighbors as it turns out to be twice slower than the original implementation. I also do not skip any iterations to avoid the error in quality even though it would lower the mean time of each iteration.

Figure 4.8 shows how long each iteration needs in average as well how often the framework iterated until convergence. While the number of iterations grow uniformly with the image size, the mean time per iteration, unfortunately, does not. I also ran calculations over 800×800 images and they showed the same trend that can already be seen for the sizes in Figure 4.8: the piecewise smooth non–deconvolving model need about 0.2 seconds per iteration, the piecewise smooth deconvolution 0.6 and the piecewise smooth deconvolution 20s. Because of the large number, I decided against portraying the last

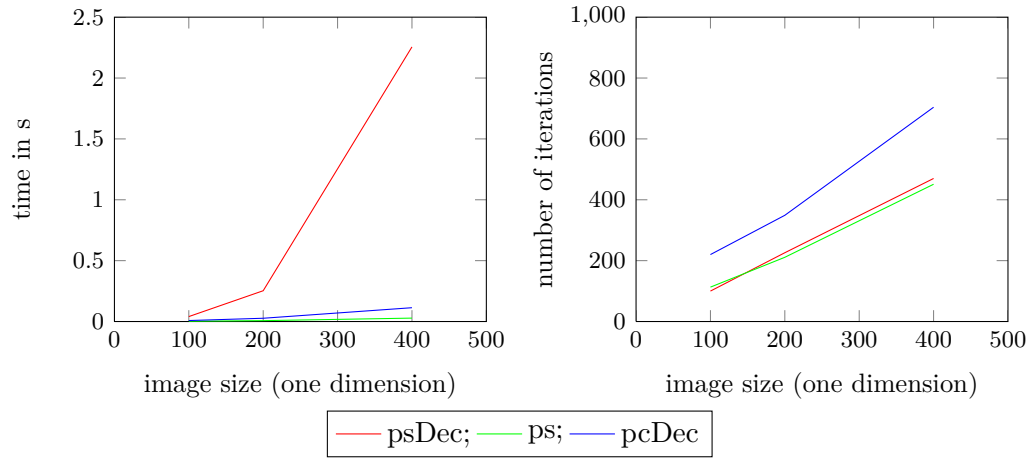


Figure 4.8: Performance results. The left image shows the mean time each iterations needs, whereas the right image shows the maximum number of iterations. The results were measured on the example images shown in Figure 4.1 and then averaged over them.

values in Figure 4.8.

The performance loss for higher images can be explained with the implementation's time–complexity. The current implementation of the piecewise smooth deconvolution has a time–complexity of $\mathcal{O}\left(|\Omega| \cdot r_{median}^d + |\Gamma| \cdot r_{PSF}^d\right)$ with with $|\Gamma|$ being the number of particles. However, that is an overly simplified version. Alongside the *Running Median*, the intensity estimation also requires the local mean of each pixel and two convolutions. The complexity is more like $\mathcal{O}\left(|\Omega| \cdot (r_{mean}^d + 2|\Omega| log|\Omega| + 2r_{median}^d) + |\Gamma| r_{PSF}^d\right)$. For the piecewise smooth deconvolution, the growth in complexity from the 100×100 to the 400×400 image is not just 4^d but rather $4^d \cdot 4^d$, because the radii of the mean, median and the PSF grew alongside the image.

As a comparison, the piecewise smooth segmentation without deconvolution only runs its own version of the *Running Mean* [CPS12], which has a time–complexity of $\mathcal{O}\left(|\Gamma| \cdot r_{mean}^d\right)$. The piecewise constant deconvolution also convolves globally, however, the mean and the median operations it applies to estimate the intensities are performed once per region and not once per pixel. The piecewise constant deconvolution has a complexity of $\mathcal{O}\left(|\Omega| log|\Omega| + |\Gamma| \cdot r_{PSF}^d\right)$

Therefore, the performance cannot be really compared to the other two energy models. It is impossible to attain the same performance as the other two energy models due to the sheer number of operations that are needed to estimate the true scene. A performance comparison with another piecewise smooth deconvolving model would be more appropriate but was, unfortunately, not possible as I explained in the beginning of this chapter.

4.4 Implementation

In this section, I analyze the implementation itself with regard to the third goal I set in Section 3.1. In order to fully realize its potential, the implementation should make use of the features of the *Region Competition framework*.

The *Region Competition framework* evaluates the energy terms from Equation 2.1 locally at the particles. Thus, the energy should also be calculated locally in order to disconnect the time–complexity from the image size. However, as I explained in Section 3.3, the calculation of accurate intensities and its convolutions requires a large support area. These areas severely overlap and many intermediate values are calculated repeatedly. Therefore, all of the values are computed globally after all. While this is faster when many particles are changed, it looses its advantage if only a few particles are changed. Then, I switch back to the local approach and update at every changed particle.

Another advantage of the framework's particle approach is its parallelization capability. If the energy is implemented accordingly, it is possible to process each particle in parallel. However, the current algorithm is only semi–parallelizable. I estimate the intensities and their convolution during the update phase. This data is later needed to calculate the energy difference, hence I store them in images. However, the writing step induces serialization.

The evaluation phase, on the other hand, only reads information and does not write anything. Each particle can be processed in parallel. Originally, this would have been a huge gain as it is rather expensive to calculate the altered intensity of a label switch for every particle at every iteration. However, I was able to cut the time cost by interpolating the new intensity, so the overall gain would not be that high. For example, in a 200×200 image, the evaluation stage needed approximately 0.02 seconds per iteration, while the update stage required around 0.55 seconds per iteration.

5 Conclusion and Future Work

5.1 Conclusion

As I have shown in the previous chapters, it is possible to combine piecewise smooth segmentation with image restoration. With this energy model, it is possible to segment inhomogeneous, blurred images that often occur in biology and medicine. The algorithm is able to deal with different levels of Gaussian noise.

However, while the quality looks promising, the algorithm is very slow, especially if either the point spread function or the image grows. In practice, the quality gain has to be weighted against this performance loss, especially in consideration of the parameter search. A global minimum with a piecewise smooth segmentation or a piecewise constant deconvolution might show better results as the local minimum of the piecewise smooth deconvolution.

But, while with the right parameter, the segmentation might cover the object correctly by chance, both approaches will fail with blurred objects that lie closely together or even overlap each other. As the embryo example from Figure 4.5 showed, even a previous deconvolution with another algorithm might not help.

Regarding the image restoration, the quality of the result depends crucially on the intensity estimation capability and the segmentation result. As the results in Figure 4.6 shows, the inner structures are not restored. Therefore, the model is not suited as an image restoration algorithm, because only the segmented object's edges will be well restored. The rest is even more blurred. The algorithms from Bar et al. and Zheng et al. that use edge maps instead of regions show better results regarding image restoration [BSK04] [ZH06].

5.2 Future Work

As I discussed in the previous chapters, the current algorithm is far from perfect. There are several topics and implementation ideas that would be interesting to analyze further but were outside of the scope for this thesis.

At the moment, the energy difference is calculated by the squared distance of the convolved intensity image and the input image. In image processing, the squared distance is used to counteract Gaussian noise. However, there are other noise types as well, such as Poisson noise. Within the *Region Competition framework*, Cardinale et al. usually covered the Gaussian noise, using a more refined model that also includes the variance, and Poisson noise. For real–life application, the variations for these two noise models should also be implemented.

This algorithm expects the point spread function to be known. In practice, that

is usually not the case. While it is possible to estimate the convolution kernel from small, known structures, the resulting point spread function is then only an estimation and, therefore, another possible source of error. For most of my experiments, I blurred the images by myself and, thus, knew the correct kernel. A more detailed analysis of the effect of a wrong point spread functions on the image restoration and the image segmentation should be done.

In Section 3.2, the intensity estimation using the the $mean(I-(J-I)$ showed better results than the $median(I/J)$. However, implemented within the *Region Competition framework*, the latter showed better results, even though $mean(I-(J-I)$ estimates the true intensities more accurately. I do not know the cause for this yet.

It is possible that the simulations were not sufficient enough. They only covered the segmentation process within the region. But at the boundaries, it is possible that the region's border is pushed further. An algorithm using an intensity estimation process that adapts the intensity too quickly would think that the local estimation fits very well to the true image and stop at the wrong position.

In addition, the simulations were done on a 1D image. I did not analyze the effect that an additional dimension can have on the intensity estimation. Furthermore, the intensity estimation is just one part within the entire *Region Competition framework*. It is difficult to estimate the effect of the other energies or the interpolation of the intensity in case of a switched label.

All in all, an implementation using the $mean(I-(J-I)$ is less complex and therefore, much faster. The correction factors can be calculated a prior and the mean of the correction factors automatically add a piecewise smooth characteristic. Hence, each iteration one convolution and the *Running median* can be omitted. A deeper analysis of the cause for this discrepancy might be beneficial to improve the performance of the current implementation.

Furthermore, while writing this thesis, I had another idea on how the convolving piecewise constant patches could be implemented as well. Instead of calculating the mean per region in the patch, it is better to compute the correction factors on the patch's pixels directly. Then, in order to downsize the number of intensities to one, it is possible to take the median or mean of the patch's intensities. Unlike before, however, that is only done once per region in the patch and not once per pixel in the patch.

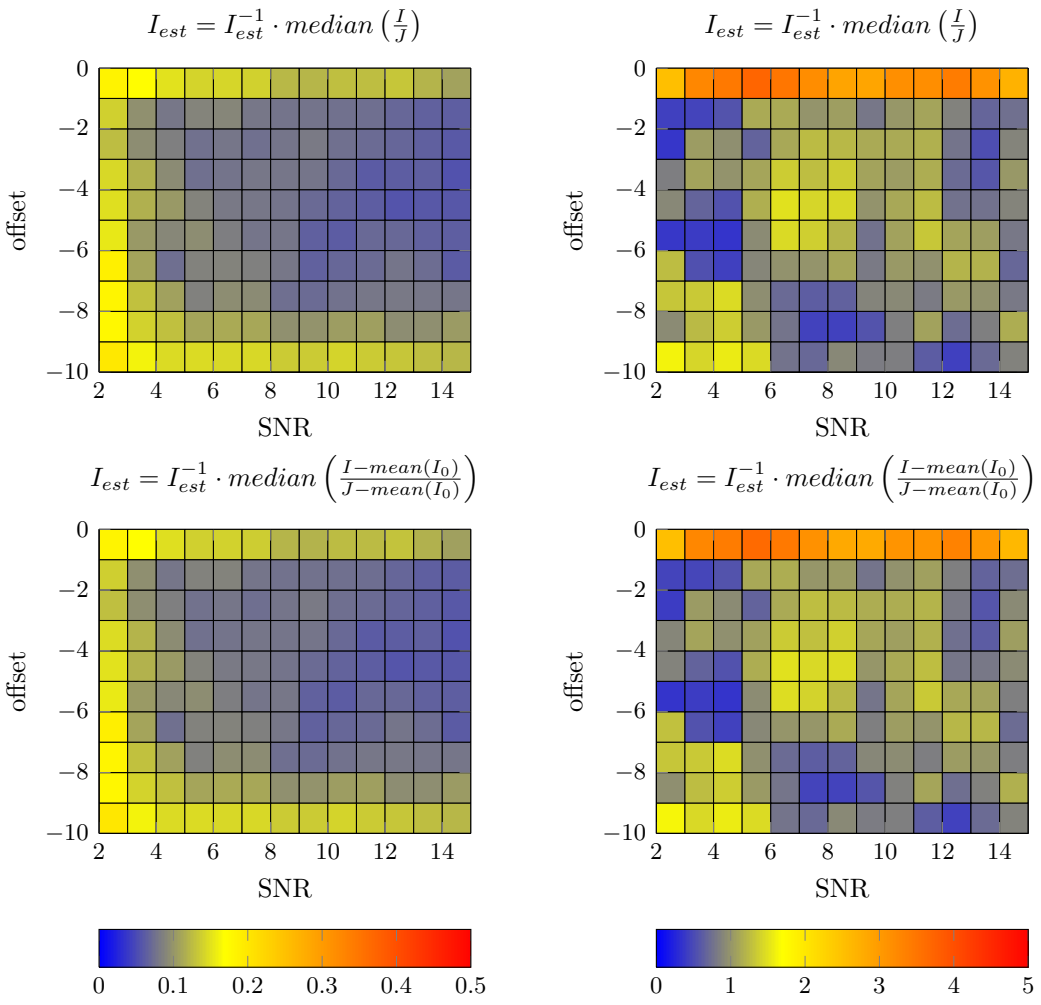
The piecewise smooth characteristic is realized with the entirety of all the patches. Therefore, no prior smoothing has to be done. The intensities can be directly estimated from the input image I and its convolution. The correction factors for the whole image can be calculated during the initialization phase.

For the comparison the the input image, the convolved intensities are calculated over a small patch of the size of the convolution kernel. If the radius of the patch, i.e., the smoothness parameter r_{mean}, is twice as large as the radius of the convolution kernel, then the small support of the convolution will have no effect on the accuracy of the result. However, in most experiments, the radii of the convolution kernel and the *Running Mean* were similar, so the small support might introduce an additional error. Unlike the other algorithms, this approach calculates everything on the small patch. Therefore, a local and fully parallelizable approach would be possible. A further study of the gain and loss of this approach might be interesting.

Appendix A

Simulation Results

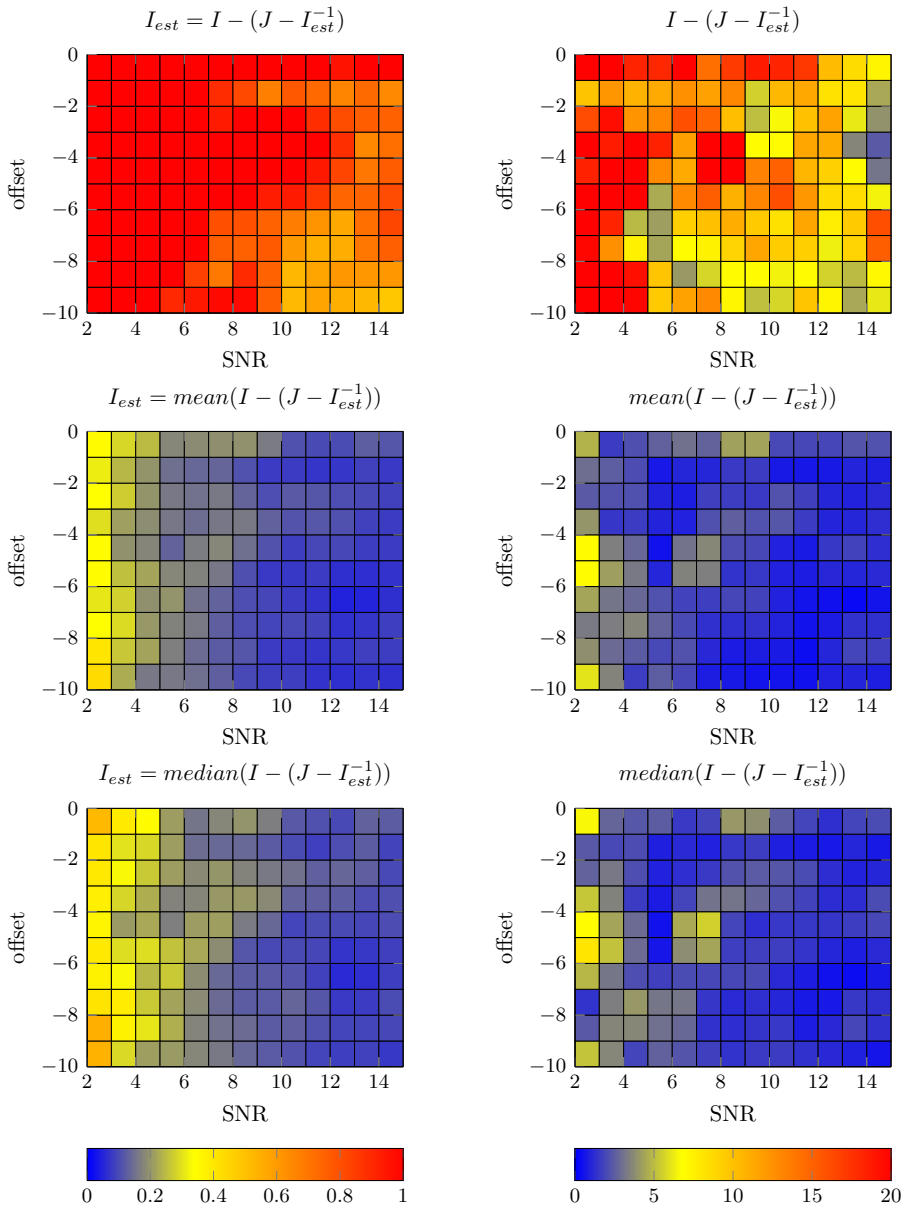
The following chapter contains the simulation results from Section 3.2:



(a) mean of $(I_{est} - I_{true})^2$

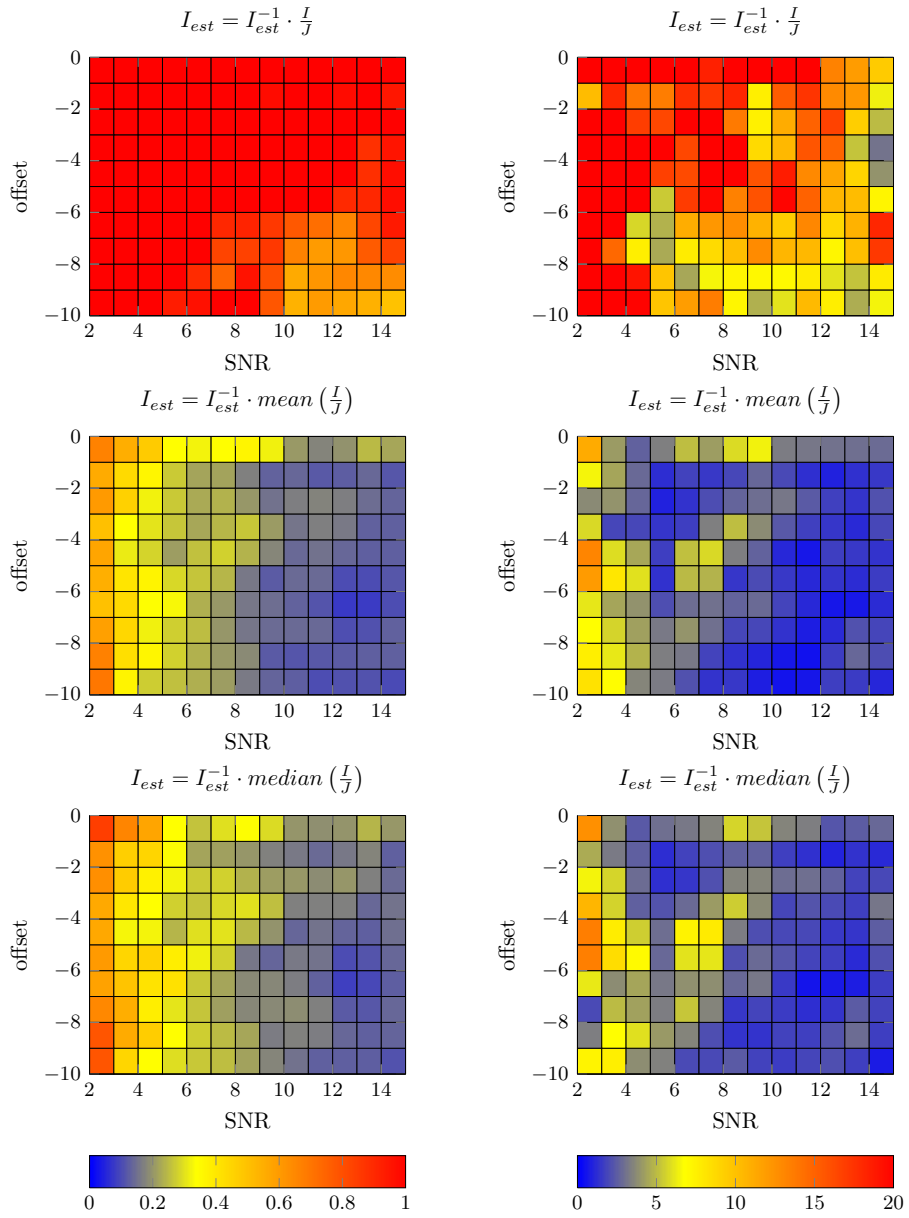
(b) standard deviation of $(I_{est} - I_{true})^2$

Figure A.1: Results showing the background influence. The median of the correction factors was chosen to estimate the intensities. The corresponding equation can be seen in Equation 3.6. The shown results were made on a 30 pixel image. The foreground region starts at a gray value of 20, while the background has a constant intensity of 10. The foreground's gray values change with a gradient of 0.5. This is just an exemplary result but using other correction factors or other variables showed similar results.



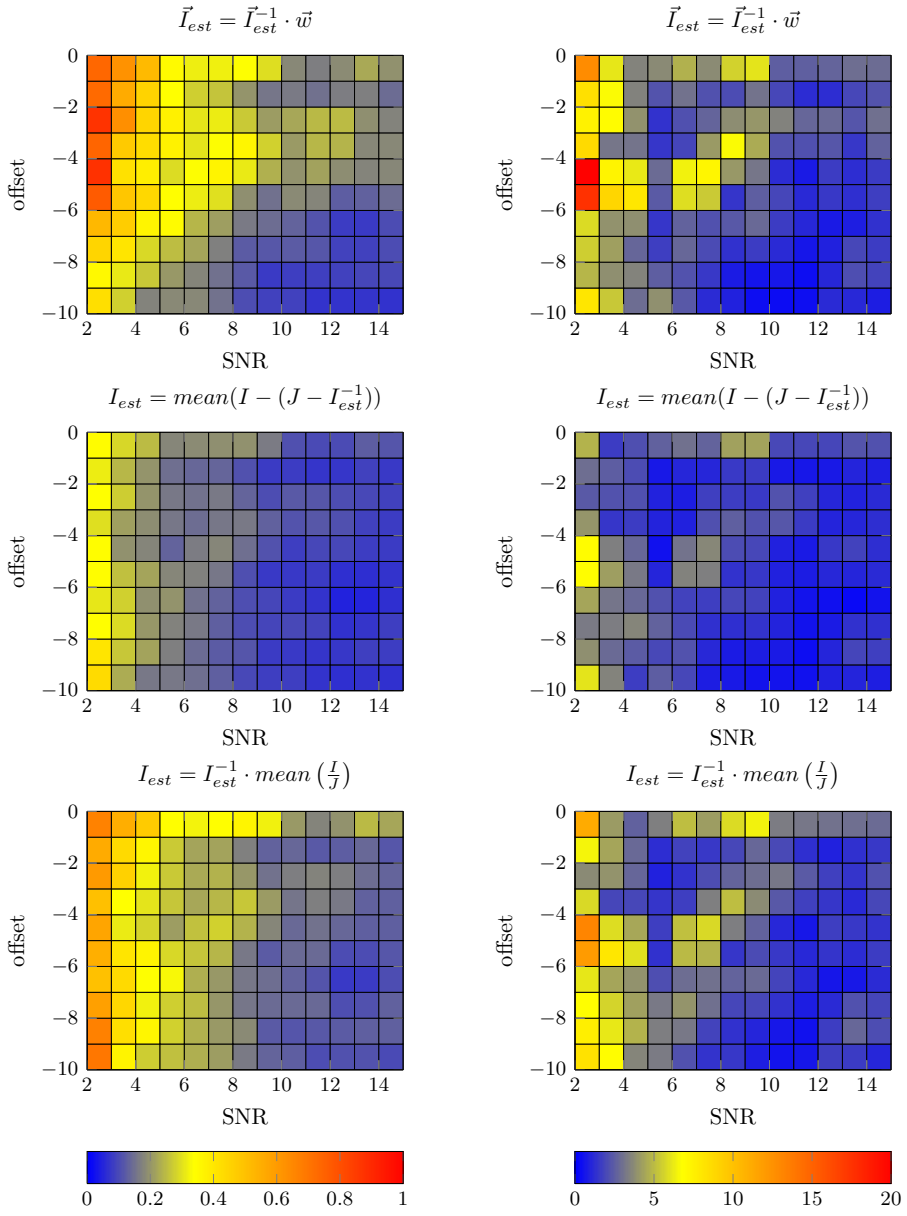
(a) mean of $(I_{est} - I_{true})^2$ ***(b)*** standard deviation of $(I_{est} - I_{true})^2$

Figure A.2: Simulation results using three different variations of Equation 3.4 to estimate the intensities. For each pixel, the correction factor $I - (J - I_{est}^{-1})$ is calculated. Then, the first variation uses each of these correction factors directly as a new intensity. The second uses the mean and the third the median of the surrounding pixel's correction factors.
The shown results were made on a 50 pixel image. The foreground region starts at a gray value of 30. The foreground's gray values change with a gradient of 0.1.



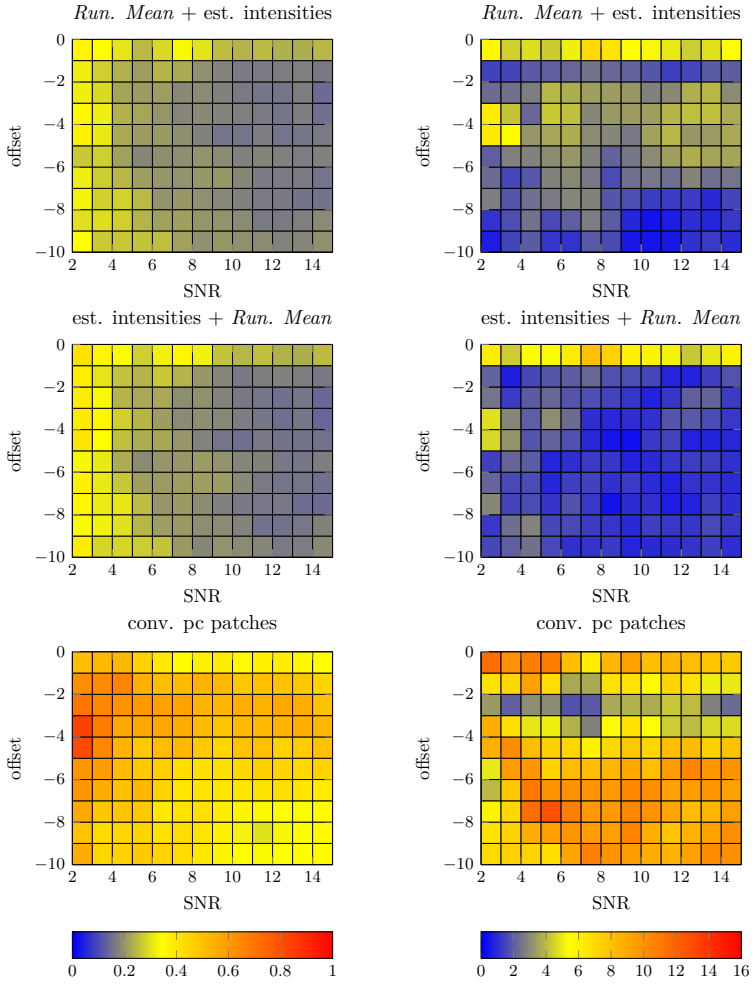
(a) mean of $(I_{est} - I_{true})^2$ ***(b)*** standard deviation of $(I_{est} - I_{true})^2$

Figure A.3: Simulation results using three different variations of Equation 3.5 to estimate the intensities. For each pixel, the correction factor I/J is calculated. The ratio is then multiplied by the first intensity estimation I_{est}^{-1}. The first variation calculates the multiplication pixelwise. The second and third use the surrounding correction factors to estimate a more robust intensity. The second row calculates the mean over the surrounding correction factors while the thir row uses the median.
The shown results were made on a 50 pixel image. The foreground region starts at a gray value of 30. The foreground's gray values change with a gradient of 0.1.



(a) mean of $(I_{est} - I_{true})^2$ ***(b)*** standard deviation of $(I_{est} - I_{true})^2$

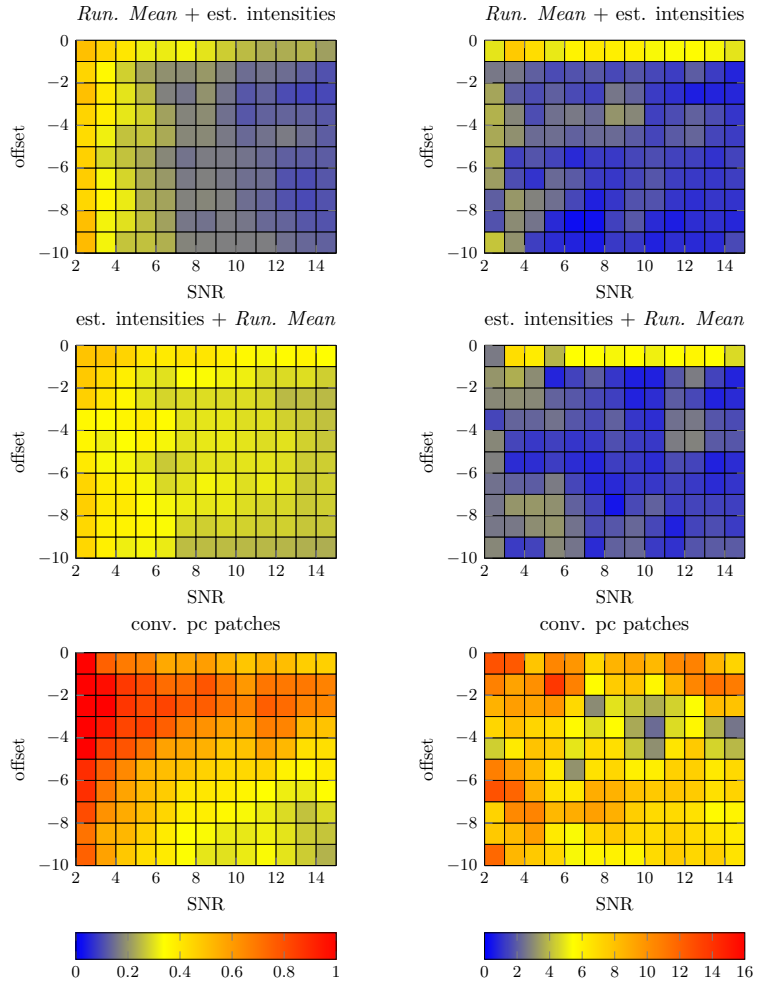
Figure A.4: Simulation result comparing the three options detailed in Section 3.2.2. The first row shows the result of the multidimensional regression. The second and third results were already shown in Figure A.2 and A.3. The results in the second row were made using $I - (J - I_{est}^{-1})$ as a correction factor whereas the third row shows the results with the ratio I/J. As there are several variations of these two methods, I picked the best of each option to compare it to each other. In both cases, it is the mean of the correction factors.
The shown results were made on a 50 pixel image. The foreground region starts at a gray value of 30. The foreground's gray values change with a gradient of 0.1.



(a) mean of $(I_{est} - I_{true})^2$

(b) standard deviation of $(I_{est} - I_{true})^2$

Figure A.5: Comparison of the three different intensity estimation algorithms I explained in Section 3.2.1. The current results were made on a 30 pixel image. The foreground region starts at a gray value of 20. Its gradient is 0.5. The intensities were estimated with the local mean of $I - (J - I_{est}^{-1})$. The further details are described in Section 3.2.
The first row shows how well the intensities are reconstructed if the piecewise smooth approximation is calculated prior to the intensity estimation. The second row shows the reverse case, that is, if the intensities are first estimated and upon the result the piecewise smoothness is added. As the addition of the piecewise smoothness is the last step of the intensity estimation, I leave out the additional smoothing. The third option is the piecewise constant patch. It assumes that piecewise smoothness is created by a large number of smaller constant patches. In order to reconstruct a piecewise constant intensity on the patch, I first calculate the mean of each region within the patch and then use the region's mean to estimate the true intensity of each region.



(a) mean of $(I_{est} - I_{true})^2$

(b) standard deviation of $(I_{est} - I_{true})^2$

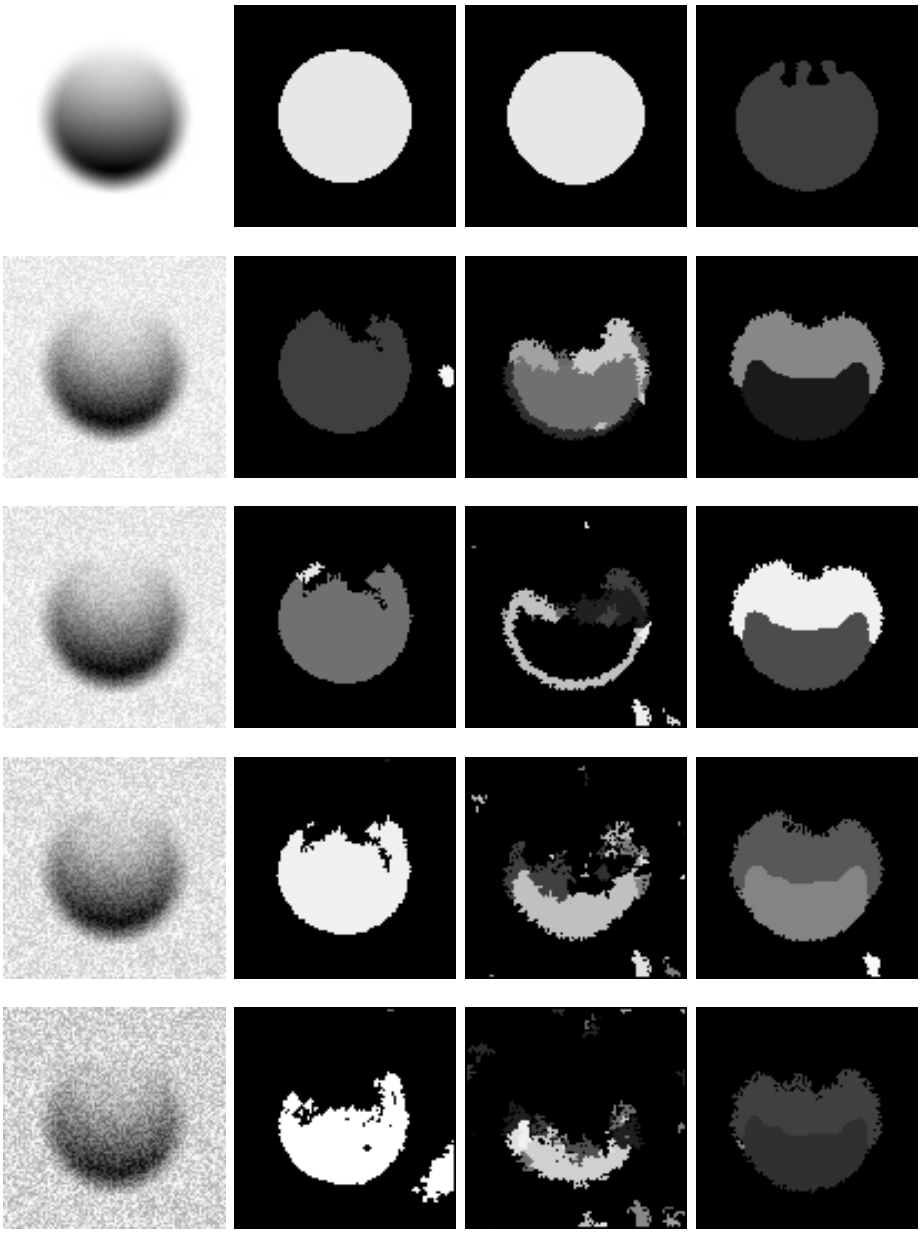
Figure A.6: Comparison of the three different intensity estimation algorithms I explained in Section 3.2.1. The current results were made on a 30 pixel image. The foreground region starts at a gray value of 20. Its gradient is 0.5. The intensities were estimated with the local mean of $I_{est} = I_{est}^{-1} \cdot median(I/J)$. The further details are described in Section 3.2.

The first row shows how well the intensities are reconstructed if the piecewise smooth approximation is calculated prior to the intensity estimation. The second row shows the reverse case, that is, if the intensities are first estimated and upon the result the piecewise smoothness is added. The third option is the piecewise constant patch. It assumes that piecewise smoothness is created by a large number of smaller constant patches. In order to reconstruct a piecewise constant intensity on the patch, I first calculate the mean of each region within the patch and then use the region's mean to estimate the true intensity of each region.

Appendix B

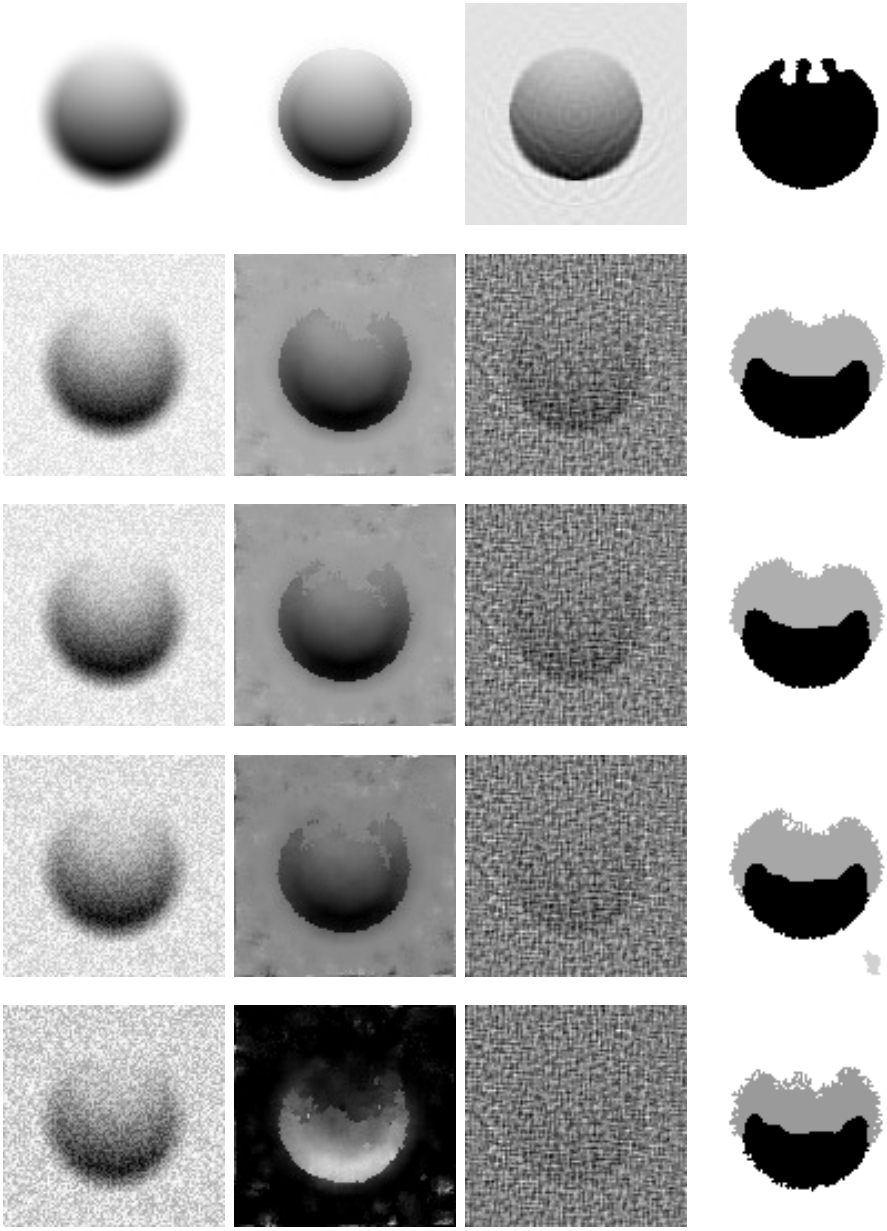
Evaluation Results

This chapter contains the segmentation and image restoration results from Chapter 4. They are sorted by image.



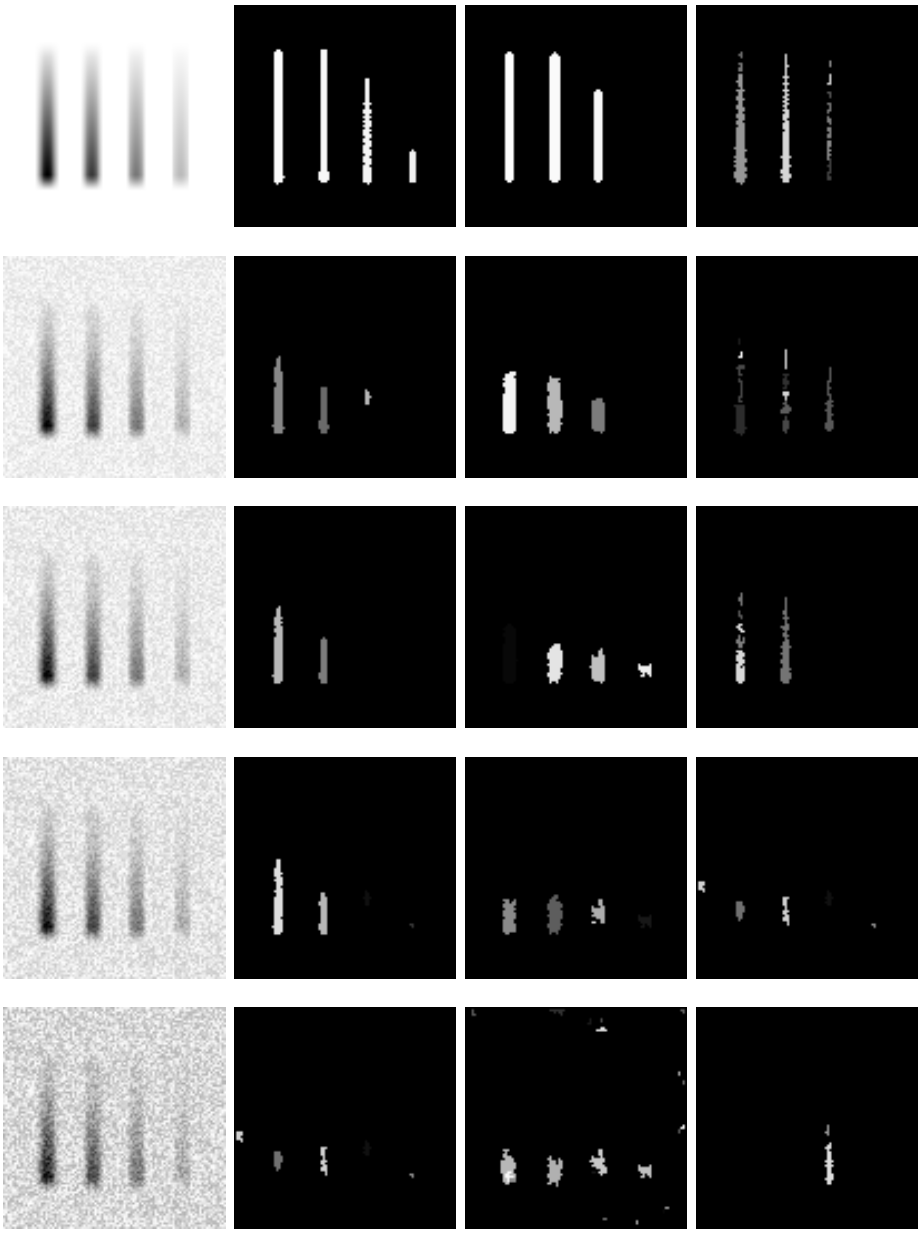
(q) input *(r)* `psDec` *(s)* `ps` *(t)* `pcDec`

Figure B.1: Image segmentation results of the circle example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



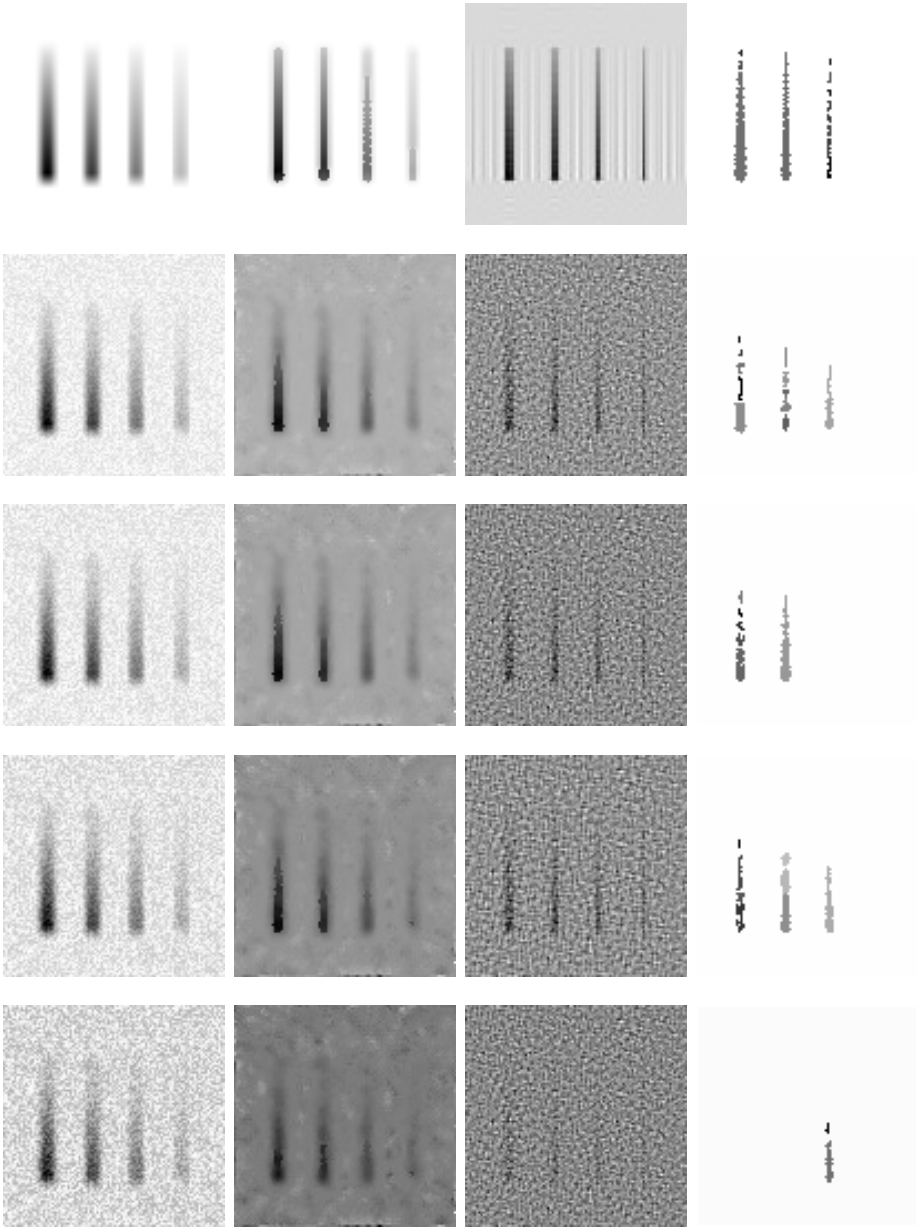
(q) input *(r)* `psDec` *(s)* `Wiener` *(t)* `pcDec`

Figure B.2: Image restoration results of the circle example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



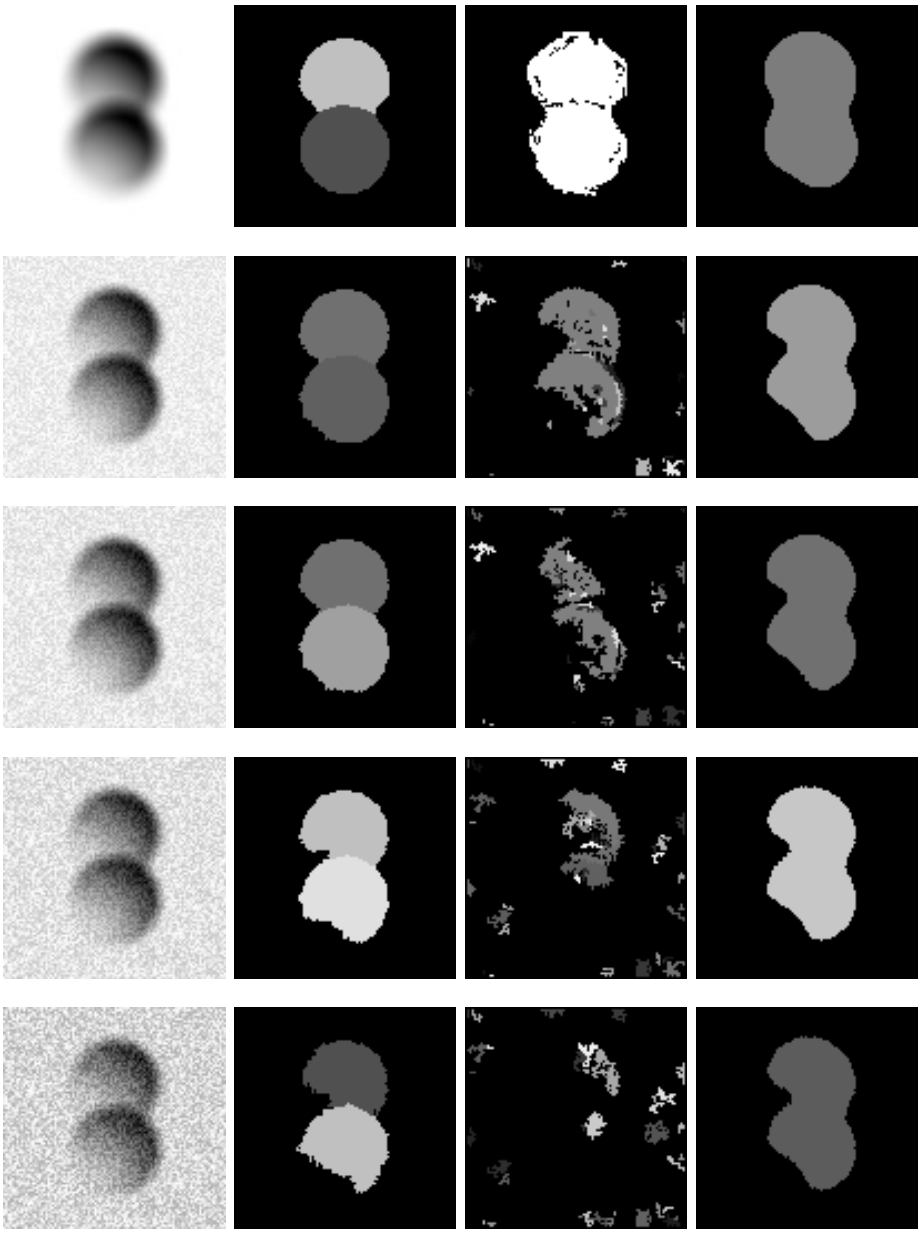
(q) input *(r)* `psDec` *(s)* `ps` *(t)* `pcDec`

Figure B.3: Image segmentation results of the lines example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



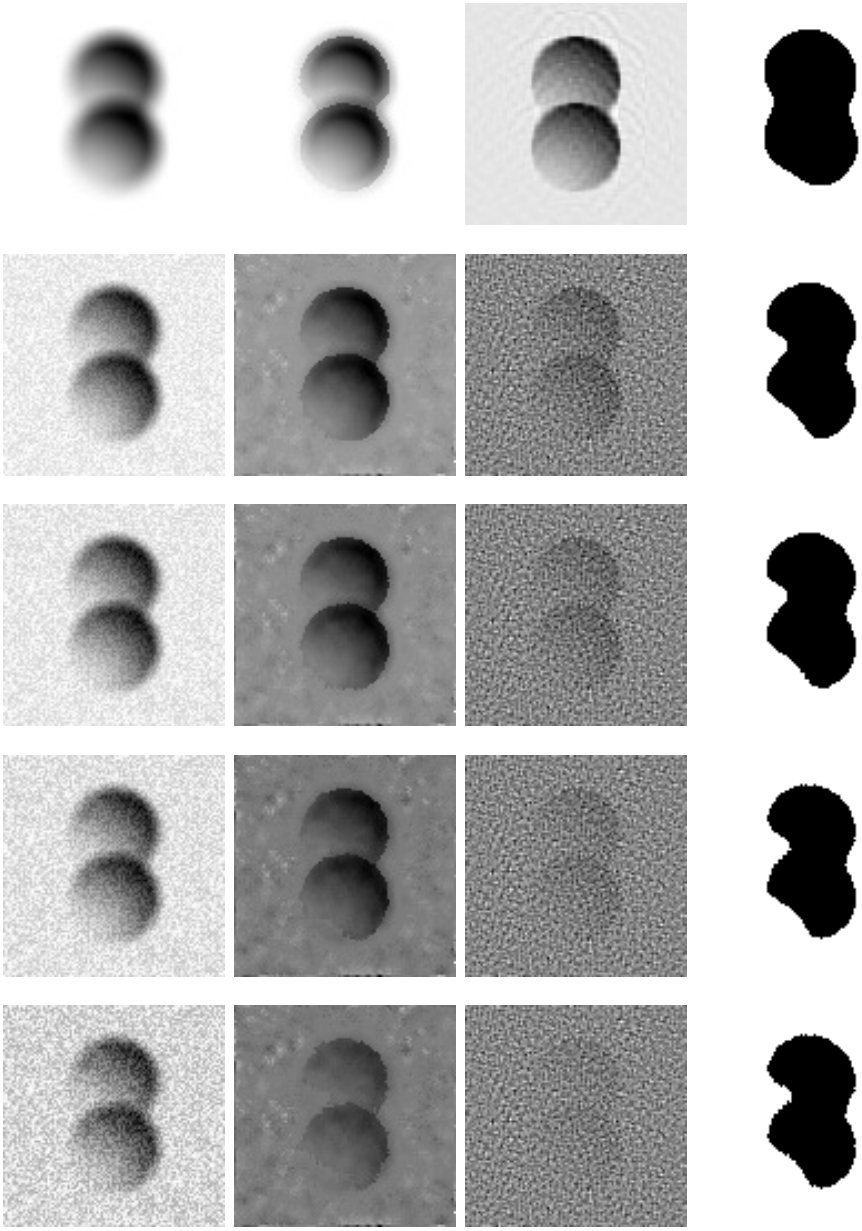
(q) input *(r)* `psDec` *(s)* `Wiener` *(t)* `pcDec`

Figure B.4: Image restoration results of the lines example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



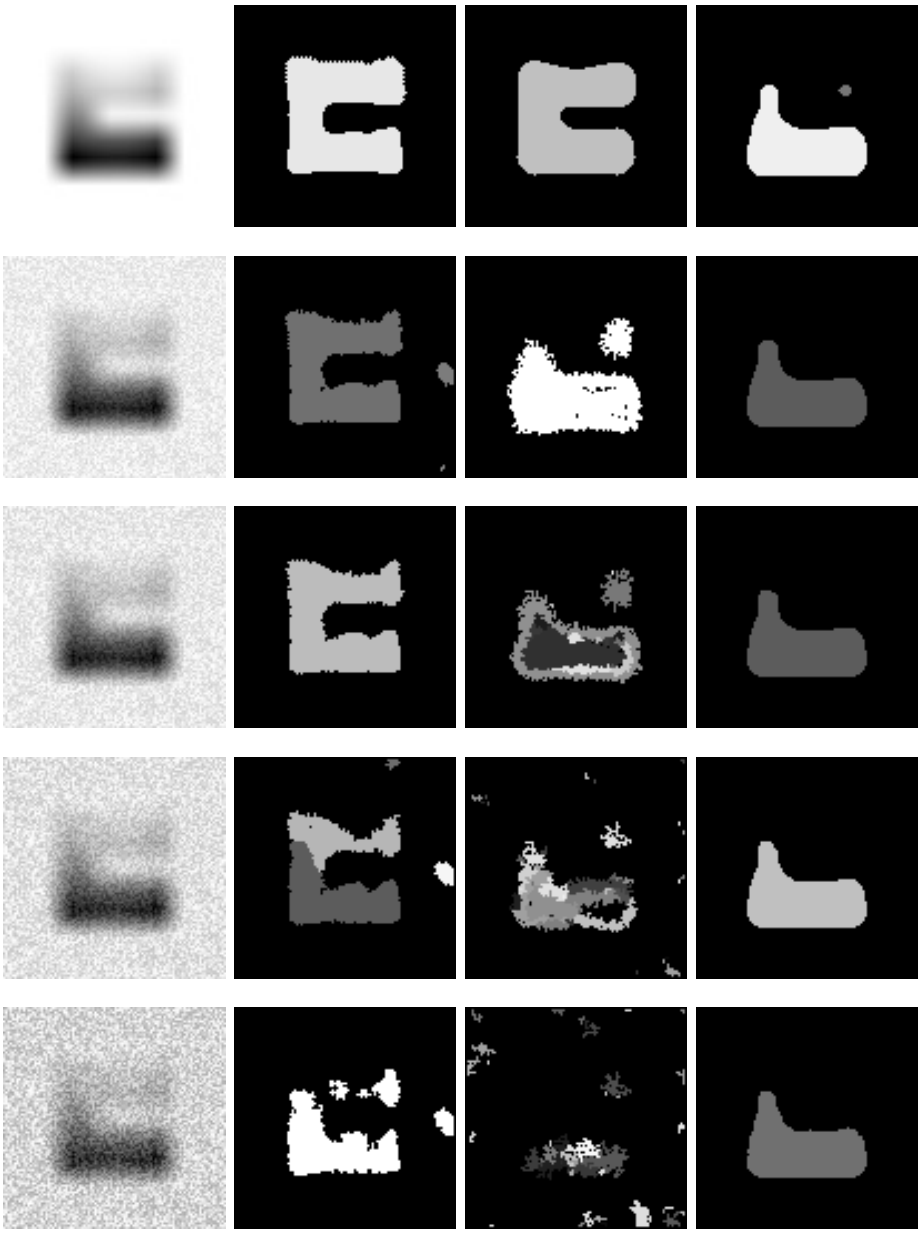
(q) input *(r)* `psDec` *(s)* `ps` *(t)* `pcDec`

Figure B.5: Image segmentation results of the overlapping circles example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



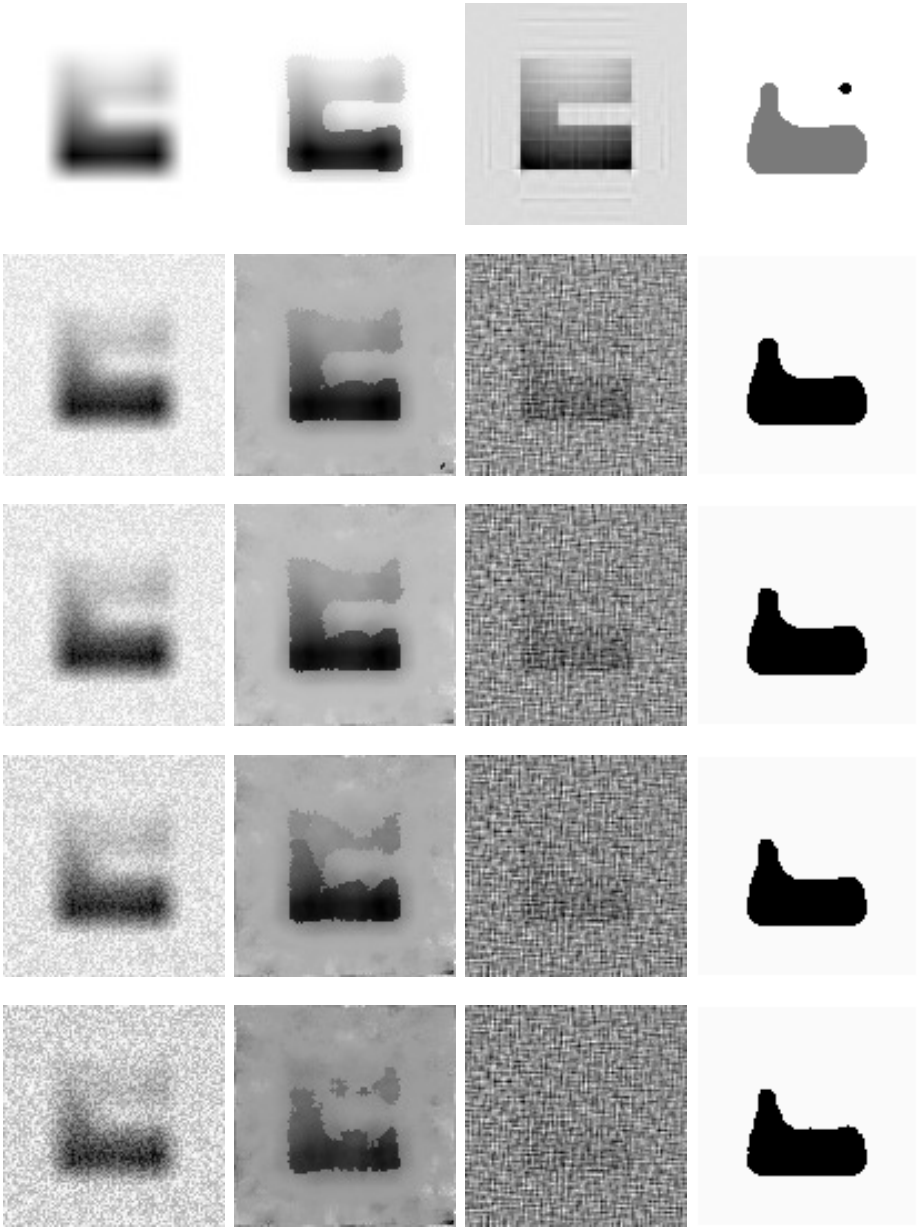
(q) input *(r)* `psDec` *(s)* `Wiener` *(t)* `pcDec`

Figure B.6: Image restoration results of the overlapping circles example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



(q) input *(r)* `psDec` *(s)* `ps` *(t)* `pcDec`

Figure B.7: Image segmentation results of the overturned U example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.



(q) input *(r)* `psDec` *(s)* `Wiener` *(t)* `pcDec`

Figure B.8: Image restoration results of the overturned U example image. The Gaussian noise is amplified from top to bottom. The input image shown in the first row contains no noise. The one in the the second row has a SNR of 2, and so on. I used the following noise levels: 0, 15, 10, 5, and 2.
Some of the figures has been inverted for illustration purposes.

Appendix C

CD Content

The enclosed CD contains the following:

- *Region Competition framework* containing the herein described piecewise smooth deconvolving energy model
- an implementation of the other piecewise smooth deconvolving energy model using $mean(I - (J - I))$ that was shortly described in Section 3.2
- diploma thesis
- Octave scripts used for the simulation described in Section 3.2
- example images
- results made with the *Region Competition framework* using the piecewise smooth non–deconvolving energy model (`ps`), the piecewise constant deconvolving energy model (`pcDec`), and the herein described piecewise smooth deconvolving energy model (`psDec`)
- image restoration results made with the Wiener Deconvolution Filter

The implemented energy model can also be found in the MPI-CBG GIT repository of the *Region Competition framework* in the branch `PSDeconvolution`. The alternative version using $mean(I - (J - I))$ is in the branch `PSDeconvolution_UsingMeanOf-SubtractChange`.

I will hand in another CD at my defense, which will, in addition to above content, also contain the slides used during the defense.

Bibliography

[AT90] Luigi Ambrosio and Vincenzo Maria Tortorelli. Approximation of functional depending on jumps by elliptic functional via Γ–convergence. *Comm. on Pure and Applied Math.*, 43(8):999–1036, 1990.

[BC07] Thomas Brox and Daniel Cremers. On the statistical interpretation of the piecewise smooth Mumford-Shah functional. In *Proceedings of the 1st international conference on Scale space and variational methods in computer vision*, SSVM'07, pages 203–213, Berlin, Heidelberg, 2007. Springer-Verlag.

[BSK04] Leah Bar, Nir Sochen, and Nahum Kiryati. Variational Pairing of Image Segmentation and Blind Restoration. In *Proc. ECCV' 2004, Prague, Czech Republic, Part II: LNCS #3022*, pages 166–177. Springer, 2004.

[Buh03] Martin D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.

[BW04] Thomas Brox and Joachim Weickert. Level Set Based Image Segmentation with Multiple Regions. In *Proceedings of 26th DAGM*, pages 415–423, 2004.

[Car13] Janick Cardinale. *Unsupervised Segmentation and Shape Posterior Estimation under Bayesian Image Models*. PhD thesis, ETH Zürich, February 2013.

[CKS02] Daniel Cremers, Timo Kohlberger, and Christoph Schnörr. Nonlinear Shape Statistics in Mumford-Shah Based Segmentation. In *In European Conference on Computer Vision*, pages 93–108. Springer, 2002.

[CPS12] J. Cardinale, G. Paul, and I.F. Sbalzarini. Discrete Region Competition for Unknown Numbers of Connected Regions. *Image Processing, IEEE Transactions on*, 21(8):3531 –3545, August 2012.

[CPS13] J. Cardinale, G. Paul, and I.F. Sbalzarini. Coupling image restoration and segmentation – A generalized linear model/Bregman perspective. *International Journal of Computer Vision*, 2013. accepted.

[CS05] Tony F. Chan and Jianhong Shen. *Image Processing And Analysis: Variational, Pde, Wavelet, And Stochastic Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[CV01] Tony F. Chan and Luminita A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[dB90] Carl de Boor. *Splinefunktionen*. Lectures in Mathematics, ETH Zürich. Birkhäuser, Basel, 1990.

[Fij] Fiji Is Just ImageJ. `http://fiji.sc`. [Online; accessed 13-March-2013].

[FKL07] L. Fahrmeir, T. Kneib, and S.M. Lang. *Regression. Modelle, Methoden und Anwendungen*. Statistik und ihre Anwendungen. Springer London, Limited, 2007.

[GG84] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741, 1984.

[Had02] Jacques Hadamard. Sur les problèmes aux dérivés partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902.

[HS09] Jo A. Helmuth and Ivo F. Sbalzarini. Deconvolving Active Contours for Fluorescence Microscopy Images. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*, ISVC '09, pages 544–553, Berlin, Heidelberg, 2009. Springer-Verlag.

[ITK02] ITK. The Insight Segmentation and Registration Toolkit. `www.itk.org/`, 2002. [Online; accessed 13-March-2013].

[ITK05] ITK/Coding Style Guide. `www.vtk.org/Wiki/ITK/Coding_Style_Guide`, 2005. [Online; accessed 17-March-2013].

[Jan05] Jiri Jan. *Medical Image Processing, Reconstruction and Restoration: Concepts and Methods (Signal Processing and Communications)*. CRC, 1 edition, November 2005.

[JCS+09] Miyoun Jung, Ginmo Chung, Ganesh Sundaramoorthi, Luminita A. Vese, and Alan L. Yuille. Sobolev gradients and joint variational image segmentation, denoising, and deblurring. pages 72460I–72460I–13, 2009.

[KK09] Jan Kybic and Jakub Krátký. Discrete curvature calculation for fast level set segmentation. In *Proceedings of the 16th IEEE international conference on Image processing*, ICIP'09, pages 2981–2984, Piscataway, NJ, USA, 2009. IEEE Press.

[KTCW02] Junmo Kim, Andy Tsai, Mujdat Cetin, and Alan S. Willsky. A curve evolution-based variational approach to simultaneous image restoration and segmentation. In *Proc. IEEE ICIP*, pages 109–112, 2002.

[MS88] David Mumford and Jayant Shah. Optimal Approximations by Piecewise Smooth Functions and Variational Problems. *Comm. on Pure and Applied Math.*, XLII(5):577–685, 1988.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, November 1992.

[SCV02] Berta Sandberg, Tony F. Chan, and Luminita A. Vese. A Level-Set and Gabor-based Active Contour Algorithm for Segmenting Textured Images. Technical report, UCLA Department of Mathematics CAM report, 2002.

[TA77] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. V. H. Winston & Sons, Washington, D.C.: John Wiley & Sons, New York„ 1977.

[TYW01] Andy Tsai, Anthony Yezzi, Jr., and Alan S. Willsky. Curve Evolution Implementation of the Mumford–Shah Functional for Image Segmentation, Denoising, Interpolation, and Magnification. *Trans. Img. Proc.*, 10(8):1169–1186, August 2001.

[VC02] Luminita A. Vese and Tony F. Chan. A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model. *Int. J. Comput. Vision*, 50(3):271–293, December 2002.

[WSSB13] Wes Wallace, Lutz H. Schaefer, Jason R. Swedlow, and David Biggs. Deconvolution in Optical Microscopy. `http://micro.magnet.fsu.edu/primer/digitalimaging/deconvolution/deconalgorithms.html`, 2013. [Online; accessed 27-February-2013].

[ZH06] Hongwei Zheng and Olaf Hellwich. Extended mumford-shah regularization in bayesian estimation for blind image deconvolution and segmentation. In *Proceedings of the 11th international conference on Combinatorial Image Analysis*, IWCIA'06, pages 144–158, Berlin, Heidelberg, 2006. Springer-Verlag.