university of groningen

faculty of mathematics and natural sciences

Shape Adaptive Kernel Density Estimation as smoothing method for MTObjects.

Master's thesis

June 2021

Student: F. N. Mol

Primary supervisor: Dr. M. H. F. Wilkinson

ABSTRACT

MTObjects is a segmentation method which segments astronomical images. The main difference between the state-of-the-art SExtractor and other techniques, is that MTObjects performs segmentation by means of creating a MaxTree of the image. MTObjects uses a smoothing technique, Gaussian Blur, in a significance test, to see whether a node in the MaxTree is significant. The σ parameter is set to 3. There lies an improvement in distinguishing noise from faint light-emitting objects and nested objects for MTObjects. Hence, adaptive smoothing methods could improve the workings of MTObjects. This thesis looks into different smoothing techniques to denoise astronomical images. Next to the Gaussian blur, we test the Perona Malik Diffusion method and develop a smoothing method based on Kernel Density Estimation. The smoother based on the Kernel Density Estimation can be made adaptive to the local intensity level and the local curvature. We investigate the performance of these techniques by comparing the smoothed noised image to a noiseless image, by means of denoising metrics. The denoising metrics used are the Peak-Signal-to-Noise-Ratio (PSNR), the Structural Similarity Index (SSIM) and the Normalized Root Means Squared Error (NRMSE). We perform a grid-search to investigate the optimal parameter settings. Both the Perona Malik Diffusion and the smoothing based on the Kernel Density Estimation outperform the Gaussian Blur, where it is also interesting that the optimal Gaussian Blur has a $\sigma = 1.14$. The Perona Malik Diffusion which uses a conduction gradient function of the gradient of the blurred image is generating the best results in terms of the denoising metrics, in a standard image and an astronomical image. Also, a major drawback of the KDEsmoother is the running time, which is 1166 seconds for a 500 by 500 pixel image. The large running time makes it infeasible to use this method on large astronomical images. Hence, this research suggests that MTObjects could improve its workings by adjusting the significance test by using the Perona Malik Diffusion instead of a Gaussian Blur.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Astronomical images contain enormous amount of information about the universe, ranging from easily identifiable planets and stars to vague light-emitting sources and noise. The latter two give a challenge to distinguish and when done by humans is, apart from very time consuming, vulnerable to errors. Techniques have been developed to recognize what is a light-emitting object and what is noise, and the challenge for such techniques is to be as good as the human classification, and eventually outperform the labeling done by humans.

For the last two decades, SourceExtractor, or SExtractor in short, has been a state-of-the-art technique for the above posed problem [3]. The technique defines the sources by using a thresholding approach and comparing the resulting peak values with the local determined background value. In the near past, several techniques have been developed which aim to outperform SExtractor. The most promising techniques are Profound [12], NoiseChisel [1] and MTObjects [17]. Where the latter technique is unique in the workings compared to the former three techniques since MTObjects considers multiple threshold levels per object.

MTObjects approaches the same former stated problem by dividing the objects in the image hierarchically by using the Max-Tree technique for thresholding. The root node in this tree consists of the image thresholded at the background value, hence the complete image, where the tree is divided further based on increasing threshold values. Then, MTObjects uses a significance test to filter out noise, followed by a step which finds objects out of these significant nodes. The main advantages of MTObjects compared to SExtractor are that MTObjects performs better at extracting faint parts of objects and at extracting and subdividing nested objects [17].

To reduce the noise in the image, MTObjects smooths the image by using a Gaussian Blur while doing the significance test. This smoothing technique is adopted from SExtractor. The Gaussian Blur technique is the widely used technique to reduce noise in an image, where the image is being convolved by a Gaussian Kernel. The Gaussian Blur is equivalent to the solution of a two-dimensional isotropic diffusion [8].

Since the technique is used uniformly for the entire image, we could question its applicability for the above stated problem. The reason for this is since there is a large difference in the intensity and size of light-emitting objects, as in the density of such objects in different parts of the image. We could pose that to reduce the noise in such an image, that a different type of smoothing could be used for large objects with a high intensity compared to smaller light emitting objects. Hence, it would be worthwhile to look in to adaptive smoothing methods.

For this thesis, we will consider two adaptive smoothing methods and evaluate their workings while used in the MTObjects technique. The first adaptive smoothing technique we will consider is a technique which will be based on the Kernel Density Estimate (KDE) [14]. We will smooth the image by using the KDE to multiply each intensity value of each pixel by it's locally determined kernel. The second adaptive smoothing technique Anisotropic Diffusion (AD) [10]. We will test whether these smoothing techniques will obtain resulting images which are more similar to a given image when we feed in those images with Poisson noise added. In terms of the goal to adding one of these smoothing methods to MTObjects, is to preserve the vague light-emitting objects better and the structure of nested objects. For the latter, we hypothesize that the anisotropic part for both smoothing methods will improve the outcome .

One of the main questions that arise is whether the KDE has to be used to decide the intensity value of the smoothed image, or whether this method can be used to determine the weights for the weighted sum of the resulting pixel in the smoothed image. We want to scale the kernel based on the square root of the intensity level, since the noise follows a Poisson distribution on the photon count. Hence, the larger the intensity, the smaller the kernel, where the upper limit corresponds to the delta function, i.e. a 1 by 1 pixel kernel with a value of 1. How to determine what the minimum and maximum kernel size must be needs to be simulated. Next to making the kernel size adaptive, we aim to make the kernels shape adaptive also by adjusting the shape of the kernel based on the local curvature, as is done in the recognition of blood vessels [15]. Adapting the shape of the kernel in Kernel Density Estimation has been done earlier by [2]. To obviate from the infinite support kernel when using the Gaussian kernel, we will use an Epanechnikov Kernel [6], as is done for the Kernel Density Estimation in [20]. We will determine the eigenvalues of the local second order derivatives for a given pixel, and make the kernel smaller in the direction of the highest curvature, and expand the kernel width at the direction of the lowest curvature. The rate in which the kernel

needs to be squished will be determined on the local eigenvalues of the curvature. We will investigate whether we need to keep the size of the surface kernel to be dependent only on the pixel intensity, or also on the eigenvalues.

The aim of using Kernel Density Estimation is to be able to determine what parameters are optimal for using Anisotropic Diffusion. Kernel Density Estimation will have a significant higher computational cost than the Anisotropic Diffusion, and hence the aim of developing a smoother based on that technique will be to gain information about AD parameters instead of actually using the technique as a smoothing technique for MTObjects. Since the astronomical images for MTObjects are 80,000 by 80,000 pixel images containing double precision floats, we will small images and add Poisson Noise to those images to simulate the effect of those astronomical images.

A brief explanation of MTObjects, the considered smoothing methods and the feedback metrics will be presented in the Methods section, Chapter 2. The results of the outcomes of testing MTObjects with the different smoothing methods will be presented in the Results section, Chapter 3. Thereafter, we will conclude this thesis with presenting our conclusions on the research we have done and will propose topics for future research in the Conclusion section, Chapter 4.

2 METHODS

This section is divided into three parts. At First, we discuss how we present and handle images in this paper. After that, we are showing the workings of MTObjects. And the last part of this section will be the smoothing method which is based on the Kernel Density Estimation.

2.1 IMAGE

2.1.1 *Input Image*

The input image is represented as $\mathbf{I}[i, j]$, with width w and height h such that $i \in [0, w]$ and $h \in [0, h]$. We use gray-scale images such that for every pair (i, j) we have a single intensity value, $I[i, j] \in [0, 255]$ for standard 8-bit images. The astronomical images consist of 64-bit float values, however the range of possible values does not cover the whole float range. A pixel represents the photon count for that given area, where the range of possible value is between $1e - 14$ and $1e - 12$, and hence we have $I[i, j] \in [1e - 14, 1e - 12]$.

2.1.2 *Poisson Noise*

The goal of creating a smoother based on the Kernel Density Estimation is to estimate the underlying photon count distribution for the given pixels of the input image of astronomical images. To model noise as is present in such images, we add Poisson Noise to the input image. To generate Poisson Noise, we can simply create an image for every pixel $i \in [0, w]$ and $j \in [0, h]$ we have

$$\mathbf{I_{PN}}[i, j] = Pois(\mathbf{I}[i, j]) \tag{2.1}$$

2.1.3 *FITS Images*

To be able to compare the impact that the different smoothing techniques have on the workings of MTObjects, we have chosen to run (a first set of) experiments on 10 simulated images. The advantage of using such a type of images instead of using real images, is that the ground-truth is exact, as was discussed in Chapter 1. All 10 simulated images have a size of 10,000 by 10,000 pixels. As is common for astronomical images, the images are stored in FITS files. To handle such images we use the `astropy.io` package [11], which is a package which specially developed for working with astronomical images and data.

2.1.4 *Denoising Metrics*

In the former section, we noted that the astronomical images have a photon distribution where the noise is close to Poisson distributed on the photon count. The aim of the smoothing method we will develop is to denoise the image as best as possible. There exists several denoising metrics for images, which give a metric to the difference of two given images. One of the most-used metric is the Peak Signal-to-noise[16]. However, this denoising metric also receives a lot of criticism, since there exist cases where the visual outcome of the denoising is not close to the metric, i.e. PSNR can give a resulting metric which states that the image is close to the original, while visually this is not true [7]. Hence, we will generate two alternative metrics, Structural Similarity [18] and the Normalized Root Mean Squared Error. These three metrics will be used to set the parameters of our smoothing method, and will compare the resulting smoother with (1) no smoothing or the noise image, (2) a simple Gaussian Blur and (3) Anisotropic Diffusion.

The PSNR metric is given by

$$PSNR(I_1, I_2) = 10 \log_{10} \frac{max(I_1^2)}{MSE(I_1, I_2)} \tag{2.2}$$

where

$$MSE(I_1, I_2) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (I_1(i,j) - I_2(i,j))^2 \tag{2.3}$$

The SSIM metric [18] is defined by

$$SSIM_{i,j} = l(x_{i,j}, y_{i,j}) \times c(x_{i,j}, y_{i,j}) \times s(x_{i,j}, y_{i,j}) \tag{2.4}$$

where

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.5}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{2.6}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{2.7}$$

Where μ_x and σ_x represent the mean and standard deviation of a 7 by 7 pixel window where the center pixel of this window is the observed pixel in image in the noiseless image, and μ_y and σ_y represent the mean and standard deviation of a 7 by 7 pixel window where the center pixel of this window is the observed pixel in image in the (smoothed) noisy image. C_i represents a small constant determined by the data range of the image. This data range is set by the minimum and maximum intesnity value of the noiseless image.

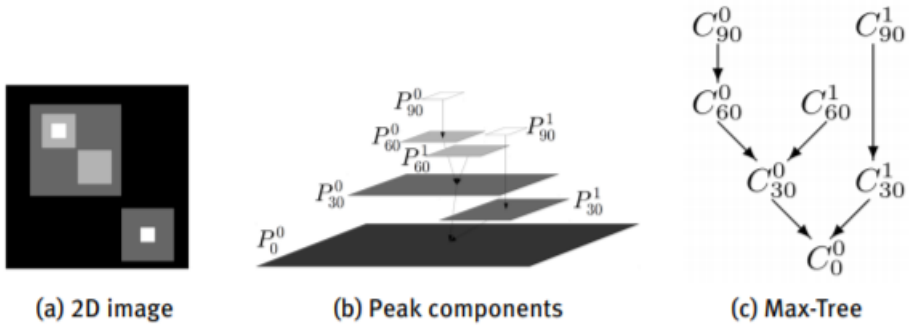


Figure 2.1: Max-Tree approach for an image where (a) is the 2D image (b) is the division of the image in it's Peak Components and (c) the corresponding Max-Tree build from the given peak components. Image used from Teeninga et al. [17]

Then, the final SSIM metric is given by mean of all the $SSIM_{i,j}$.
And finally the NRMSE is given by

$$NRMSE = \frac{\sqrt{MSE}}{\max(I_1) - \min(I_1)} \tag{2.8}$$

2.2 MTOBJECTS

MTObjects is a method which segments astronomical images into background, nebulas and light-emitting objects. MTObjects first determines a background value for the whole image. This value is calculated by taking the mean of the pixel values of flat tiles, where these flat tiles are parts of the image which do not have any objects in them. This background is set as parent node in the Max-Tree, where the tree is expanded by using an increasing threshold value, until the leaves consist of the peak components of the input image. The building of such a Max-Tree is presented in Figure 2.1. In the original paper [17], the authors present 4 statistical tests to determine whether each node of the Max-Tree is a significant node, i.e. if the node is an object. The significance tests are based on a power attribute of the parent node or the ancestor node of the node which is observed. The most used significance test of these 4 tests is the test which is based on the power attribute on the ancestor node, and the area of the node while a Gaussian Blur is used on the image. For this statistical test, we will research whether we can increase the workings of MTObjects by changing the smoothing method from Gaussian Blur to a more advanced and adaptive smoothing method. Finally, MTObjects uses deblending to identify nested objects.

2.3 KERNEL DENSITY ESTIMATION

2.3.1 *Kernel Density Estimation*

Kernel Density Estimation (KDE) is a technique which attempts to find an underlying distribution from a given set of observed data points. We call this a non-parametric methodology to calculate the distribution [14].

To give an introduction to this concept, a similar more crude way to determine a distribution of observed data points in a one dimensional data set is a histogram. To create the histogram, we simply determine an interval size t in which we count all the observations which lie in between n and n+t for any given n. A two dimensional graph is created this way which is an approximation of the actual distribution on which the observed data points lie. The interval size n and the placing of the intervals (determining of the start interval which is between $0 =< n =< t$) are prone to errors which could lead to complete different outcomes of the estimated distribution.

Hence, a more sophisticated method is developed which is less sensitive to the way the initial model is set, as mentioned above with the case of a histogram, the Kernel Density Estimation has been developed. This technique does not set an interval size where it sums the number of observations in between, but sets a distribution with mean $\mu = x_i$ where x_i is the value of the observation i and the variance σ^2 is chosen by the user of the model. To make such a Kernel Density Estimation, a distribution needs to be chosen which will be placed on all the observations, where the uniform distribution and the normal distribution are the most commonly used. Each of these distributions are added to create the approximated underlying distribution of the observed data points. Another advantage of Kernel Density Estimation, is that whenever we use a distribution such as the Normal Distribution, we end up with a continuous distribution as result, compared to the discrete histogram.

Since Kernel Density Estimation is a way to smooth data points, we could also use it as a smoothing method for images. Note, however, that an image is a 2d histogram, where every pixel is a 2d bin where such a bin represents the counted photons for this given area. Hence, using Kernel Density Estimation is not an alternative density estimator to a histogram when used for images. Next to that, using isotropic kernels in Kernel Density Estimation, we have an equivalent technique to a Gaussian Blur. The difference between these two techniques, is that the Gaussian Blur uses an weighted average of neighbourhood pixels to determine a smoothed pixel value, and the Kernel Density Estimation adds a kernel multiplied by the considered pixel at the pixel location in the smoothed image. This is again equivalent for

isotropic kernels. The advantage, however, of using the Kernel Density Estimation comes when we change the kernel size and/or shape based on the local information. The local information of the pixel influences the size and/or the shape of the kernel, and hence influences neighbourhood pixels. If we would make a Gaussian Blur adaptive, the resulting pixel would be influenced on the local information.

2.3.2 *Epanechnikov Kernel*

Since the Gaussian Kernel has infinite support and since we want to fold the kernel inwards at the borders of the image, we do need to cutoff the Kernel. However, the cutoff will be hard to determine with a shape adapted kernel. As an alternative, we use the Epanechnikov Kernel[6] which has a finite support. The d-dimensional Epanechnikov Kernel with bandwidth b is given by:

$$K_\epsilon(\mathbf{x}) = \begin{cases} \frac{d+2}{2 \cdot c_d}(1 - \mathbf{x} \cdot \mathbf{x}) & \text{if } \mathbf{x} \cdot \mathbf{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

In Figure 2.2 we show both the Gaussian Kernel and the Epanechnikov Kernel with a $\sigma = 7$ and bandwidth $b = 7$ To get a variance of 1, we need to multiply the standard deviation by $\sqrt{5}$.
Since we only use the 2-dimensional variant, we can fill in $d = 2$, where c_d is the volume of a d-dimensional unit sphere, which equals to π for $d = 2$. Hence, we get

$$K_\epsilon(\mathbf{x}) = \begin{cases} \frac{1}{b\sqrt{5}} \frac{4}{2\pi} \left(1 - \frac{\mathbf{x}}{b\sqrt{5}} \cdot \frac{\mathbf{x}}{b\sqrt{5}}\right) & \text{if } \mathbf{x} \cdot \mathbf{x} < b^2 \cdot 5 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

To use an anisotropic kernel, we must input a $2x2$ bandwidth matrix instead of a bandwidth scalar. This 2×2 matrix is the same as the

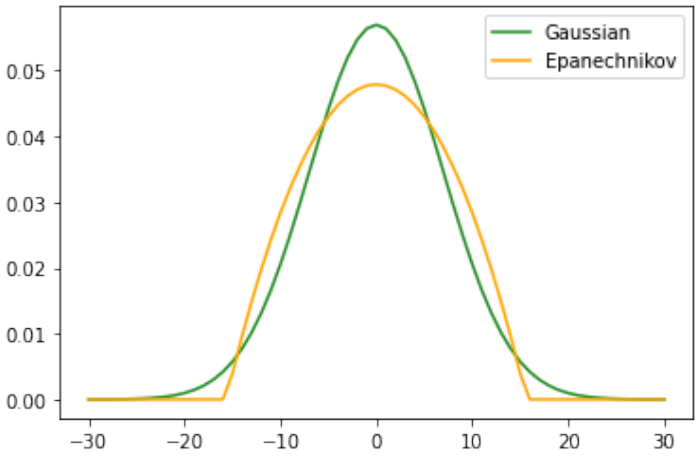


Figure 2.2: kernels

co-variance matrix for a simple 2d Gaussian distribution. We end up with the following formula for the kernel

$$K_\epsilon(\mathbf{x}) = \begin{cases} \frac{1}{\det \mathbf{B}\sqrt{5}} \frac{4}{2\pi} \left(1 - \mathbf{B}^{-1} \left(\frac{\mathbf{x}}{\sqrt{5}} \cdot \frac{\mathbf{x}}{\sqrt{5}}\right)\right) & \text{if } \mathbf{x} \cdot \mathbf{x} < \det \mathbf{B}^2 \cdot 5 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

2.3.3 *Boundaries*

When using convolution techniques, such as Gaussian Blur, to smooth an image, it is common to pad the image such that the smoothing can cope with pixels where the kernel otherwise would consider pixels outside of the boundaries of the given input image. In the case of convolutions, one can simple pad the image by $(k-1)/2$ pixels, where one of the most used types of padding is mirror padding. We could apply padding for the KDE smoothing, where we could use mirror padding determined by the size of the maximum kernel we use. However, since we want to adjust the shape, next to the size, of the kernel, and the shape is determined on the local curvature, we would need to determine the upper bound of the kernel size. This could lead to a large padding which leads to a large increase of the computation time, especially if we consider the 80,000 by 80,000 FITS images. Whenever we would use mirror padding, we would obtain the same results if we mirror (or fold) the kernel inwards instead of mirroring the image at the borders, as is shown in Figure 2.3. For this folding technique, we do not need to determine any padding size and we also do not need to increase the image size.

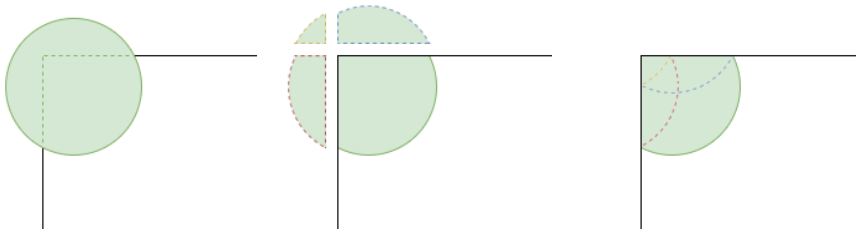


Figure 2.3: Whenever the kernel is exceeding the image border (a), we fold the exceeding parts of the kernel (b) inwards and add the kernel values to the corresponding pixels which lie in the image (c).

2.3.4 *Pilot Estimate*

Whenever an adaptive kernel is used for Kernel Density Estimation, where the adaptivity depends on the local information, a pilot estimate could be used. The pilot estimate is a technique which uses a simple,

or simpler, approximation of the underlying distribution which creates a continuous support of the observed data[20]. Consider for example that we use the Gaussian Blur or a KDE with the normal distribution. The continuous support is needed to calculate the first and second order derivatives of the observed data which can be used to shape the kernel. Another advantage of the using a pilot estimate is whenever there exists noise which is an outlier, the local information, or local differences, can be such that it influences the shape of the kernel too extremely. Whenever the pilot estimate is used, the extreme outliers are flattened or smoothed out, and hence the adaptivity of the kernel will be less influenced. We refer to the pilot estimate as: $\mathbf{I_{PE}}$.

2.3.5 *Kernel Size*

We let the size of the kernel be dependent on the intensity of the pixel, the pilot estimated photon count, of the blurred image. The higher intensity pixels shouldn't have a widespread influence on neighbouring pixels, where lower intensity pixels (background pixels) should have a high ranging influence. Hence, we want the kernel to be smaller for higher intensities. We use the bandwidth matrix $\mathbf{B}$ to adjust the size. Note that if we use a isotropic kernel, we can use the bandwidth scalar b, since the matrix will be $B = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix}$. To calculate b, we try the following options, for the resulting smoothed images see Chapter 3.

2.3.6 *Kernel Shape*

We adapt the kernels of the KDEsmoother based on the local information[2]. To adjust the shape of the kernel based on local curvature, we need to determine the eigenvalues with corresponding eigenvectors of the local Hessian [15]. Hence, we use Sobel operators, denoted as S_x and S_y for the Sobel operators in the x and the y directions, to generate the first- and second-order derivatives of the image. For input image I we have $S_x(I) = I_{dx}$ and $S_y(I) = I_{dy}$ for the first-order derivative images in the x and the y direction, respectively. Consequently, we have $S_x(I_{dx}) = I_{dxx}$, $S_x(I_{dy}) = S_y(I_{dx}) = I_{dxy}$ and $S_y(I_{dy}) = I_{yy}$ for the second-order derivative images. For pixel pair i, j we have that the Hessian matrix is given by

$$H_{i,j} = \begin{bmatrix} I_{dxx}(i,j) & I_{dxy}(i,j) \\ I_{dxy}(i,j) & I_{dyy}(i,j) \end{bmatrix} \tag{2.12}$$

An eigenvalue decomposition gives us the eigenvalues λ_1 and λ_2 with corresponding eigenvectors e_1 and e_2. Let λ_1 be the eigenvalue with the largest absolute value, i.e. $|\lambda_1| \geq |\lambda 2|$. The eigenvalues represent

the magnitude of the curvature in the direction of the eigenvectors. Hence, we adjust the shape of a given kernel by these eigenvalues, and then rotate such that the kernel aligns with the eigenvectors.
In the former section, we determined how the size of the kernel will be adjusted. To keep the size of the surface of the kernel equal, we adjust the size of the kernel in e_1 and e_2 direction such that the determinant of a given bandwidth matrix equals the determinant of an isotropic kernel for that given size. This determinant is equal to the squared value of the single bandwidth value b, i.e. $\det(B) = b^2 = b$ We use the following ratio

$$r = \frac{|\lambda_1|}{|\lambda_1| + |\lambda_2|} \tag{2.13}$$

Where $0.5 \leq r \leq 1$. Hence, we specify

$$b_\alpha = b \times (1 + \alpha \times (r - 0.5)) \tag{2.14}$$

$$\mathbf{B} = \begin{bmatrix} b_\alpha & 0 \\ 0 & \frac{b^2}{b_\alpha} \end{bmatrix} \tag{2.15}$$

For any value of $0.5 \leq r \leq 1$ we have that $\det B = b^2$, and hence the kernel surface remains equal for all kernels at a given intensity value.

After creating the anisotropic bandwidth matrix based on the eigenvalues, we rotate this matrix such that is is perpendicular to the eigenvector corresponding to the highest eigenvalue. The idea is that we want the kernel to be smaller in the direction of the highest eigenvalue, and wider in the other direction.

2.3.7 *Smoothed Image*

To generate the smoothed image, we start with an $w \times h$ sized image $\mathbf{I_S}$ where $\mathbf{I_S}[i, j] = 0$ for all $i \in [0, w]$ and $j \in [0, j]$. Then, for every (i, j), we generate a kernel $K_{i,j}$, depending on the intensity in $\mathbf{I_{PE}}[i, j]$. The size of this kernel $K_{i,j}$ is (w_K, h_K). We multiply this kernel by the pixel intensity of the original input image, i.e. $K_{i,j} * \mathbf{I}[i, j]$ and add the resulting kernel to the smoothed image $\mathbf{I_S}$ where the center of the kernel is placed at $\mathbf{I_S}[i, j]$. If there exist parts of the kernel which exceeds the pixel boundaries of the smoothed image, we fold the kernel inwards.
The number of pixels the kernel has to it's boundary are: $x_K = \frac{w_K - 1}{2}$ in the x-direction and $y_K = \frac{h_K - 1}{2}$ in the y-direction. Hence, we have the following equation:

$$\mathbf{I_S}[i - x_K : i + x_K, j - y_K : j + y_K] = K_{i,j} * \mathbf{I}[i, j] \tag{2.16}$$

To see what the resulting distribution of kernels is, we also generate an image $w \times h$ sized zero image $\mathbf{I_{BK}}$, where for every pixel we add the kernel without multiplying the given kernel by the pixel intensity. Which results in

$$\mathbf{I_{BK}}[i - x_K : i + x_K, j - y_K : j + y_K] = K_{i,j} \tag{2.17}$$

The pseudo-code of the KDEsmoother can be seen in Appendix A.

2.4 ANISOTROPIC DIFFUSION

Gaussian smoothing is a form of isotropic diffusion, which basically means that the smoothing is done equally in all directions. Anisotropic Diffusion is the counterpart to this standard form of smoothing, and one of the first adaptive smoothing methods introduced which does not smooth in all directions. The first form of this method was introduced by Perona and Malik in [10]. We will use the traditionally introduced smoothing technique, however, we must note that there exist many variations on this technique [19].

To show how Anisotropic Diffusion works, we first introduce the following notation for the Gaussian Blur

$$\frac{\partial I(x,y,t)}{\partial t} = \Delta I \tag{2.18}$$

where I is the image function, x, y are the x- and y-coordinates, t is the variance used for the Gaussian Kernel. Δ is the notation for the Laplacian operator. Hence, we note that

$$\Delta I = (I_{xx} + I_{yy})$$

Now, we introduce the Perona Malik formula for Anisotropic Diffusion

$$\frac{\partial I(x,y,t)}{\partial t} = \nabla(c(x,y,t)\nabla I) \tag{2.19}$$

Where div is the divergence operator and ∇ is the gradient operator. The diffusion process is steered by the $c(x,y,t)$, where this function is given by

$$c(x,y,t) = g(|\nabla I(x,y,t)|) \tag{2.20}$$

Finally, we have to choose the conduction function $g(\dot{)}$. Perona and Malik propose two different conduction tensor functions, where the first is given by:

$$g(\nabla I) = e^{(-(|\nabla I(x,y,t)|/\kappa)^2} \tag{2.21}$$

And the second conduction tensor function is given by:

$$g(\nabla I) = \left(1 - \left(\frac{|\nabla I(x,y,t)|}{\kappa}\right)^2\right)^{-1} \tag{2.22}$$

In their paper, Perona and Malik [10] state that the second increases the stability of the diffusion operation. Hence, we choose Equation 2.22 as conduction tensor function.

2.4.1 *Perona-Malik diffusion on intensity*

The Perona-Malik diffusion, as presented above, has the aim to smooth an image with preserving its edges. In the astronomical images which MTObjects attempts to segment, there exist no hard edges except for peaks. Hence, it seems interesting to adjust the Perona-Malik diffusion such that the diffusion is steered by the intensity level instead of the local gradients. Adjusting the diffusion method this way, gives the following formula:

$$g(\nabla I) = \left(1 - \left(\frac{I(x,y,t)|}{\kappa}\right)^2\right)^{-1}$$

2.4.2 *Gaussian Blur for Perona-Malik Diffusion*

In Catté et al.[4], it is suggested to increase the stability of the Perona Malik diffusion, the input of the conduction tensor function could be lightly blurred by a Gaussian Blur. The reason for this is to reduce the influence of noise, e.g. if there is a peak signal which is noise, the local information of the structure has a high impact on the diffusion equation. Hence, when you blur the input of the conduction tensor function, you reduce the impact of noise in this equation. The resulting equation is

$$g(\nabla G_\sigma * I) = \left(1 - \left(\frac{|\nabla G_\sigma * I(x,y,t)|}{\kappa}\right)^2\right)^{-1}$$

Interestingly, the blurring the input of the conduction tensor function is equivalent to the approach of the pilot estimate for the Kernel Density Estimation, both techniques use a Gaussian Blur to reduce the impact of noise on local information which is used for both smoothing methods.

2.4.3 *Algorithm*

In each iteration of the Anisotropic Diffusion, the resulting image is the sum of the image at time step $t-1$ and the result of Equation 2.19.

The second part of the equation is multiplied by a weight γ, where Perona and Malik[10] state that the performance is best when setting $\gamma = [0.10, 0.25]$. We adopt this by setting the γ parameter equal to $\gamma = 0.10$. The reason for setting the lower bound is that γ and the number of iterations together influence the degree of how smoothed the resulting image is, and since we test the number of iterations, we want the influence per iteration by the γ parameter be low. We will test the influence on what is the optimal set of parameters for the number of iterations and the κ parameter on the three different performance metrics, the MSE, SSIM and PSNR, as mentioned in Chapter 2. Furthermore, we will test both conduction tensor functions as given in Equation 2.21 and Equation 2.22 where we will generate results for using the gradient on the image as input, and using the raw intensity as input in both cases. Finally, we will test the influence of blurring the input of both conduction tensor functions by the Gaussian Blur. We want the image to be slightly blurred, and hence we use $\sigma = 1$. The pseudo-code of this algorithm can be seen in Appendix A.

We will generate results of using both the gradient and the intensity on the conduction tensor function, using a Gaussian Blur on the input of the conduction tensor function, and testing the parameter κ and the number of iterations.

2.5 TECHNICALITIES

The source-code of the methods and experiments is done in Python version 3.7x. Experiments are done on the Zeus server and the Peregrine HPC. Zeus is a server managed by Dr. M.H.F. Wilkinson[1] at the Rijksuniversiteit Groningen. The server has Dell R815 Rack Server with four 16-core AMD Opteron processors and 512 GB of RAM memory [9]. Peregrine is the the High Performance Cluster (HPC) from the Rijksuniversiteit Groningen. We run the training on a CPU node consisting of 24 cores @ 2.5 GHz (two Intel Xeon E5 2680v3 CPUs) [2]. The running times of the different methods will be compared on the Peregrine HPC.

1 m.h.f.wilkinson@rug.nl
2 https://wiki.hpc.rug.nl/peregrine/

3 RESULTS

In this section we present the results of the different smoothing methods, where we test the parameters for the Gaussian Blur, the Perona-Malik Diffusion and the KDEsmoother method. We perform a grid-search on the different parameters, and determine based on the denoising metrics how to tune the parameters. We split the results section up into two parts, where we test the method on a standard photographic image, and on an astronomical telescope image which is relevant for MTObjects. We will use a simulated image in the astronomical case, to be able to compare the resulting smoothed images to a noiseless image and be able to know the underlying ground truth segmentation. Finally, we will run the optimal smoothing method with corresponding parameters on the complete set of simulated images, to see the increase on MTObjects when compared to the original smoothing method, which is a Gaussian blur with $\sigma = 3$. Also, we will generate the results of the performance of MTObjects on the noiseless images, to see where the opportunities of improvement lie, in the preprocessing or in the MTObjects method itself.

3.1 STANDARD PHOTOGRAPHIC IMAGE

In this section, we use a photographic image called 'Lena', which is a common used image in the field of Image Processing due to the high amount of detail. The image is presented in Figure 3.1. The reason we perform the smoothing methods on a standard image, next to an astronomical image, is that we can compare whether a given technique's performance is due to the technique, or is only better suited for one of these images. Also, since we have developed a new smoothing method, it is interesting to see its performance not only on MTObjects related images, but also to a more general image.

Figure 3.1: Colored image of Lena, which we use as a standard image to compare the performance of the different proposed smoothing techniques on.

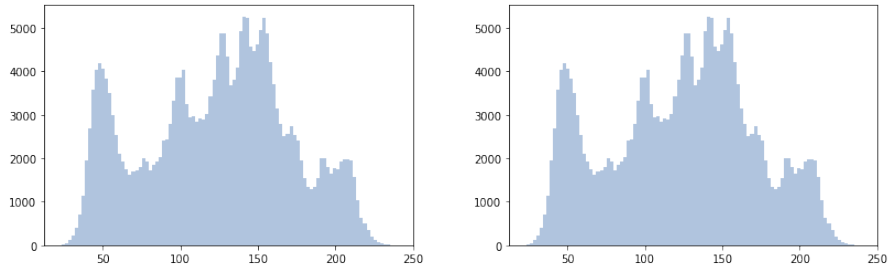
We convert the image of Lena to a one channel gray-scale image, which can be seen in Figure 3.2a. As described in Chapter 1, we use Poisson noise on the image to simulate the noise distribution on the astronomical images. The gray-scale image with Poisson noise can be seen in Figure 3.2b. The histograms of the pixel intensities of the gray-

(a) The gray-scale image of Lena.

(b) The gray-scale image of Lena, with added Poisson noise.

scale image of Lena and the Lena image with Poisson noise added are presented below in Figure 3.3. We are interested in the distribution of the pixel intensities, since we use a modified Perona-Malik diffusion based on the local pixel intensity, and use a size adaption for the kernel in the KDEsmoother based on the intensity level. For the latter, we use a scaling based on the minimum and maximum of the pixel intensity of the whole image, hence, we want to verify whether there are for example no outliers which changes the behaviour of this scaling. When we look at Figure 3.3, we see that this is not the case. Finally, we can also compare the two histograms to see whether the noise model

is correct, and the distribution is still similar to the noiseless image, which is also the case.



(a) (b)

Figure 3.3: Histogram of pixel intensities for (a) the gray-scale image of Lena and (b)the gray-scale image of Lena with Poisson noise.

3.2 TUNING OF PARAMETERS.

In this section, we show the tuning of the parameters for the Gaussian Blur, the Perona-Malik diffusion and the KDEsmoother. The tuning is done by performing a grid-search on the different denoising metrics. We use the PSNR as the metric to optimize on. We present the results of 3 different denoising metrics, PSNR, SSIM and NRMSE, as was proposed in Chapter 2. As a benchmark, we use the above mentioned metrics given by the gray-scale image and the Poisson noise image shown in Table 3.1. Also, we present the results of the metrics using the original noiseless image as input, which gives the upper-bounds (or lower-bound for the NRMSE) of these metrics. The denoising metrics on a noiseless image and the simulated image are given below, in Table 3.1.

	PSNR	SSIM	NRMSE
Noised image	25.814	0.572	0.084
Original Image	∞	1.000	0.000

Table 3.1: The denoising metrics PSNR, SSIM and NRMSE given for Lena. The metrics are given where the input image is compared to the original gray-scale image of Lena. As input, the noised and the noiseless image are used, to give the upper-bounds and lower-bounds.

3.2.1 *Gaussian Blur*

In the original MTObjects paper [17], a Gaussian Blur with $\sigma = 3$ is used. To get a full comparison how well both the adaptive smoothing methods, the Perona-Malik diffusion and the KDEsmoother are

working, we test different values for σ of the Gaussian Blur on the three denoising metrics. Hence, we can see how a non-adaptive, or isotropic, smoother is working when tuned to the denoising metrics. The resulting plot can be seen in Figure 3.4.

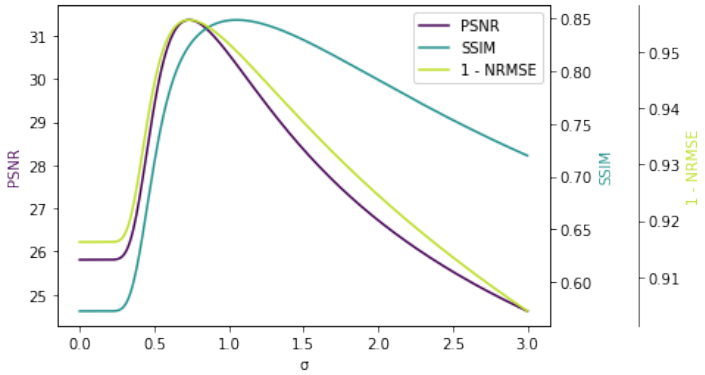


Figure 3.4: Denoising metrics PSNR, SSIM and NRMSE plotted on σ as parameter for Gaussian Blur, using the gray-scale Lena image as ground truth and the smoothing on the Poisson noise image. Note that we have inverted NRMSE. The optimal σ for each metric is at 0.73, 1.05 and 0.73 for PSNR, SSIM and NRMSE, respectively.

As can be seen in Figure 3.4, the slopes of PSNR, SSIM and the inverse of the NRMSE metrics are similar, where the optimal values for σ are also close, i.e.the optimal σ is equal to 0.73, 1.05 and 0.73 for PSNR, SSIM and NRMSE, respectively. In in Table 3.2 below, the optimal denoising metrics for the Gaussian Blur on the Lena image are given next to the denoising metrics when using a Gaussian Blur with $\sigma = 3$.

	PSNR	SSIM	NRMSE
Optimal value	31.344	0.848	0.044
$\sigma = 3$	24.601	0.719	0.096

Table 3.2: Tuning of the σ-parameter for the Gaussian Blur smoothing method, used on the gray-scale Lena image with Poisson noise. The denoising metrics for $\sigma = 3$ are also displayed, the value for the parameter used in the Gaussian Blur in MTObjects.The optimal values of σ are 0.73, 1.05 and 0.73 for the PSNR, the SSIM and the NRMSE, respectively.

3.2.2 *Perona-Malik diffusion*

The Perona-Malik diffusion has 2 parameters which we want to optimize by means of a grid-search on the denoising metrics. We want to optimize κ and the number of iterations. The third parameter of the Perona-Malik diffusion, γ, is a stability parameter which is set to 0.1 for all experiments.

As proposed in Chapter 2 the Perona-Malik diffusion in 4 different variants:

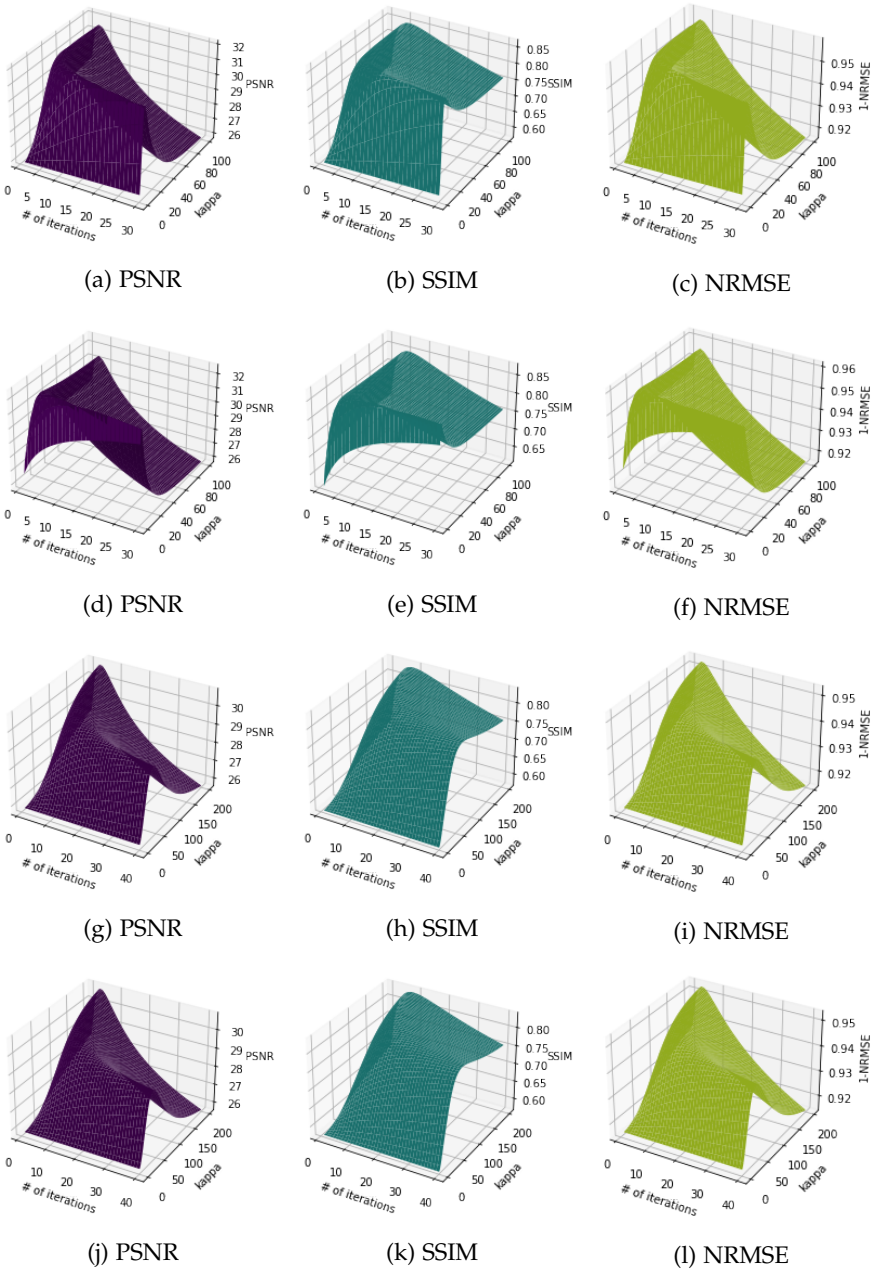
- Conduction Function uses the local gradients
- Conduction Function uses the local gradients of a slightly blurred image
- Conduction Function uses the local intensity value
- Conduction Function uses the local intensity value of a slightly blurred image

We optimize the these two parameters on the three different denoising metrics, where the resulting graphs are depicted in Figure 3.5 on the next page and the resulting values of the optimal metrics can be seen in Table 3.3.

	PSNR	SSIM	NRMSE
Perona-Malik	32.105	0.866	0.041
Perona-Malik with Blur	32.494	0.875	0.039
Perona-Malik intensity	30.813	0.836	0.047
Perona-Malik intensity with Blur	30.783	0.835	0.047

Table 3.3: Tuning of the κ-parameter and the number of iterations for the Perona-Malik diffusion, used on the gray-scale Lena image with Poisson noise. The denoising metrics are displayed for the 4 different variants of the Perona-Malik diffusion.

For standard images, such as the Lena image we use, it seems that the variant of the Perona-Malik Diffusion where the conduction function uses the gradient of the blurred image outperforms the other variants, for all three denoising metrics. The corresponding parameters for this optimal variant are $\kappa = 5$ with 9 iterations for the PSNR and the NRMSE, and $\kappa = 3$ with 23 iterations. For the variants which run on the intensity instead of the gradient, a higher κ parameter increases the resulting image in terms of denoising. When κ converges, the influence of the conduction function is reduced, and hence the influence of the local information is reduced. We can interpret that in



(a) PSNR (b) SSIM (c) NRMSE

(d) PSNR (e) SSIM (f) NRMSE

(g) PSNR (h) SSIM (i) NRMSE

(j) PSNR (k) SSIM (l) NRMSE

Figure 3.5: Denoising metrics used on Perona-Malik Diffusion where we plot on κ and the number of iterations used. We display the results for all variants of the Perona-Malik Diffusion we use, where the Conduction Function has as input: for (a)-(c) gradients, (d)-(f) gradients of blurred image, (g)-(i) intensities, (j)-(l) intensities of blurred image. Where for each variant, we display the PSNR, SSIM and NRMSE from left to right. Note, that we use 1-NRMSE, to compare the distribution of the grid-search to the PSNR and the SSIM metrics.

the case of using the intensity as input for the conduction function, the diffusion prefers to be as isotropic as possible. Comparing these

results, we see that the Perona-Malik Diffusion indeed performs worse than the Gaussian Blur for the intensity based versions. However, the gradient based methods outperform the Gaussian Blur.

3.2.3 *Kernel Density Estimation*

In this section, we tune the parameters of the smoothing method based on the Kernel Density Estimation, the KDEsmoother, where we use three different versions of the algorithm. First, we present the results of the method which uses isotropic kernels with uniform size. The parameter we tune is the bandwidth parameter which determines the size of the kernel used for all pixels in the smoothing method. In Figure 3.6 the results of the grid-search optimization of the bandwidth parameter for the KDEsmoother are presented. Since we use uniform

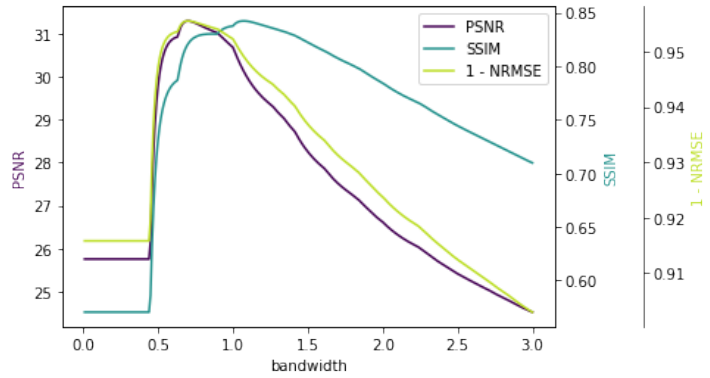


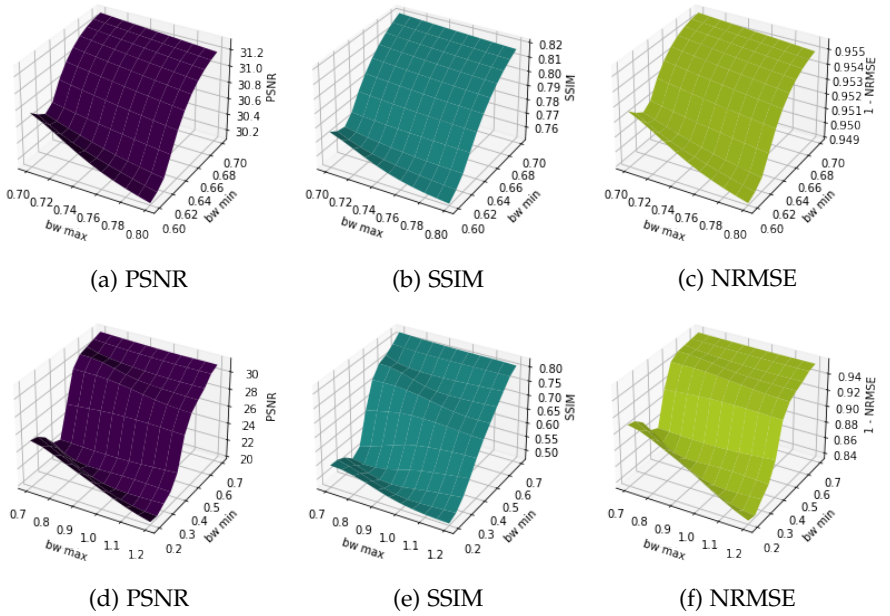
Figure 3.6: Denoising metrics PSNR, SSIM and NRMSE plotted on b as parameter for KDE, using the gray-scale Lena image as ground truth and the smoothing on the Poisson noise image. Note that we have inverted NRMSE. The optimal σ for each metric is at 0.73, 1.05 and 0.73 for PSNR, SSIM and NRMSE, respectively.

kernels, we expect that this variant of the KDEsmoother performs similar to Gaussian Blur, with a slight variation due to the difference between the Gaussian distribution and the Epanechnikov distribution. We can see that indeed the graphs in Figure 3.6 look similar to the graph in Figure 3.4. Note that the denoising metrics for the KDEsmoother seems to be noncontinuous. This non-continuity occurs when increasing the bandwidth results in increasing the used kernel size. In Table 3.4 we show the results of the tuning of the bandwidth on all three denoising metrics. We show the resulting optimal value, maximum in case of PSNR and SSIM and minimum in case of NRMSE, with the corresponding bandwidth value. Comparing the results of the KDE smoother in Table 3.4 with the results of the Gaussian Blur in Table 3.2, we see that the Gaussian Blur has similar results, but the results are slightly better.

	PSNR	SSIM	NRMSE
Value	31.301	0.842	0.044
Bandwidth	0.70	1.07	0.70

Table 3.4: Uniform kernel size, tuned on the parameter bandwidth. We display the pairs of optimal denoising metric with corresponding parameter.

For the size adaptive version of the KDEsmoother, we perform a grid-search for the parameters b_{min} and b_{max} together. The resulting graphs of the grid-search for each of the denoising metrics is presented in Figure 3.7. We use the optimal bandwidth parameter of the uniform KDEsmoother for the PSNR and NRMSE as upper-bound and lower-bound, for b_{min} and b_{max} respectively. Note that this can result in a sub-optimal value for the SSIM, since the bandwidth value of 0.7 is not optimal for this denoising metric in the uniform case.



(a) PSNR (b) SSIM (c) NRMSE

(d) PSNR (e) SSIM (f) NRMSE

Figure 3.7: Denoising metrics used on size adaptive variant of the KDEsmoother. We display the results for all variants of the Perona-Malik Diffusion we use, where the Conduction Function has as input: for (a)-(c) 11 steps of 0.01, (d)-(f) 11 steps of 0.05, Where for each variant, we display the PSNR, SSIM and NRMSE from left to right. Note, that we use 1-NRMSE, to compare the distribution of the grid-search to the PSNR and the SSIM metrics.

As we can see in Figure 3.7, the optimal value for all denoising metrics is the highest value for b_{min}, and the lowest value for b_{max}. These are the values which represent the non-size adaptive version of the KDEsmoother, and hence, the resulting optimal denoising values are the same, as can be seen in Table 3.5. For the shape-adaptive

	PSNR	SSIM	NRMSE
Value	31.301	0.821	0.044
(b_{min}, b_{max})	(0.70, 0.70)	(1.07,1.07)	(0.70,0.70)

Table 3.5: KDEsmoother with adaptive kernel size, tuned on the b_{min} and b_{max} parameters. We display the pairs of optimal denoising metric with corresponding parameter.

version of the KDEsmoother, we use a bandwidth of $b_{min} = 0.7$ and $b_{max} = 0.7$, i.e. the version which is not adaptive in kernel size. We use these values for the bandwidth since these are optimal for the PSNR and NRMSE metrics in both testes version of the KDEsmoother. We have performed a grid-search on the α parameter, which determines how much a kernel is adjusted in its shape. The resulting graph of this grid-search performed for all three denoising metrics is displayed in Figure 3.8 As we can see in Figure 3.8, increasing the adaptivity in

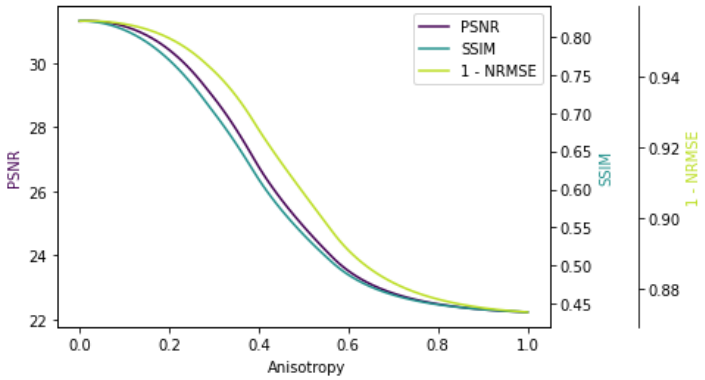


Figure 3.8: Denoising metrics PSNR, SSIM and NRMSE plotted on α as parameter for the shape-adaptive KDEsmoother, using the gray-scale Lena image as ground truth and the smoothing on the Poisson noise image. Note that we have inverted NRMSE. The optimal α for each metric is at 0.02, 0.00 and 0.02 for PSNR, SSIM and NRMSE, respectively.

shape for the KDEsmoother, decreases its performance for all three denoising metrics. However, a small value for α has a slightly better PSNR and NRMSE than the uniform KDEsmoother. These results are presented in Table 3.6.

	PSNR	SSIM	NRMSE
Value	31.307	0.821	0.044
α	0.02	0.00	0.02

Table 3.6: Uniform kernel size, tuned on the parameter bandwidth. We display the pairs of optimal denoising metric with corresponding parameter.

3.3 COMPARISON OF SMOOTHING TECHNIQUES.

From the former sections, we summarize all the denoising metrics for each smoother. Each smoother is optimized on the given denoising metric. The results are displayed in Table 3.7.

	PSNR	SSIM	NRMSE
Without smoothing	25.814	0.572	0.084
Same image	∞	1.000	0.000
Gaussian Blur $\sigma = 3$	24.601	0.719	0.096
Gaussian Blur	31.344	0.848	0.044
Perona-Malik	32.105	0.866	0.041
Perona-Malik with Blur	**32.494**	**0.875**	**0.039**
Perona-Malik on Intensity	30.813	0.836	0.047
Perona-Malik on Blurred Intensity	30.783	0.835	0.047
KDEsmoother uniform	31.301	0.842	0.044
KDEsmoother size-adaptive	31.301	0.842	0.044
KDEsmoother shape-adaptive	31.307	0.821	0.044

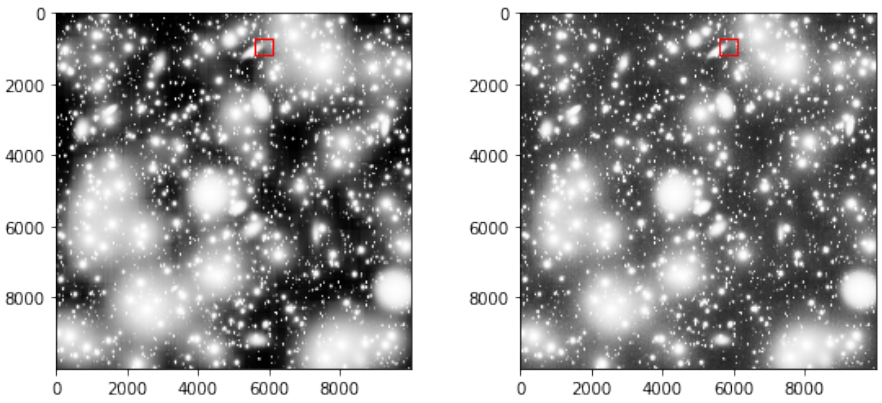
Table 3.7: Overview of the denoising metrics for all smoothing techniques with optimal parameter setting. These are all results of smoothing the Lena image with noise and comparing the result to the noiseless image of Lena.

As we can see in the overview in Table 3.7, the optimal smoother in terms of the three denoising metrics is the Perona-Malik Diffusion where the conduction function used the gradient combined with a Gaussian Blur. The KDEsmoother we have developed has similar performance to the Gaussian Blur, and the size-adaptivity and shape-adaptivity of the KDEsmoother does not add significant increases in performance when using the standard photographic image of Lena. Finally, note that all smoothing methods outperform the Gaussian Blur with a $\sigma = 3$, even when we do not smooth the image at all. We display the smoothed image for the optimal parameters for the best performing variant of each smoothing method in Figure 3.9 on the next page.

Figure 3.9: Overview of the original and the smoothed images of Lena, where the following images are presented: (a) Gray-scale Lena, (b) Lena with Noise, (c) Gaussian Blur with $\sigma = 3$, (d) Optimal Gaussian Blur, (e) Optimal Perona-Malik Diffusion, (f) Optimal KDEsmoother.

3.4 ASTRONOMICAL IMAGES

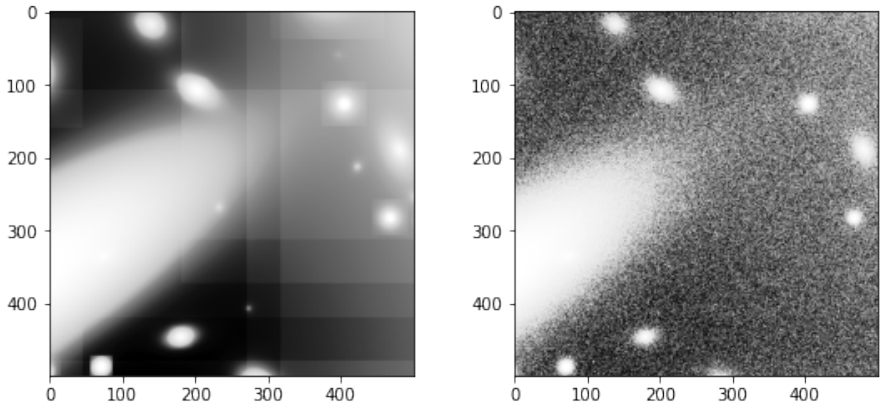
The KDEsmoother was developed with the objective to reduce the noise in astronomical images, and hence we also perform the same experiments, as on the standard photographic image presented in the former section, on an astronomical image. We use an image from a set of simulated images, which are similar to the real astronomical images which are used for MTObjects. The advantages of using simulated images are that we have the exact underlying ground truth segmentation of these images and we can generate images with and without noise. These noiseless images are crucial for our experiments to generate the denoising metrics. We use a set of 10 simulated images, where each of these images consist of 10,000 by 10,000 pixels. We display the first of these images, `img0.fits` below in Figure 3.10.



(a) noiseless (b) with noise

Figure 3.10: The first of the ten simulated Fits images, where we have the noiseless image in (a), and the same image with a Poisson noise model added in (b). To visualize, we have used an Histogram Equalization stretch.

Since the running time of the KDEsmoother is high, we were not able to perform a grid-search to search for the optimal parameters for the KDEsmoother on these full images. Also, performing a grid-search on the Perona-Malik Diffusion is slow on these large images. Hence, we use a crop of the `img0.fits` simulated image. This crop is presented in Figure 3.11.



(a) noiseless (b) with noise

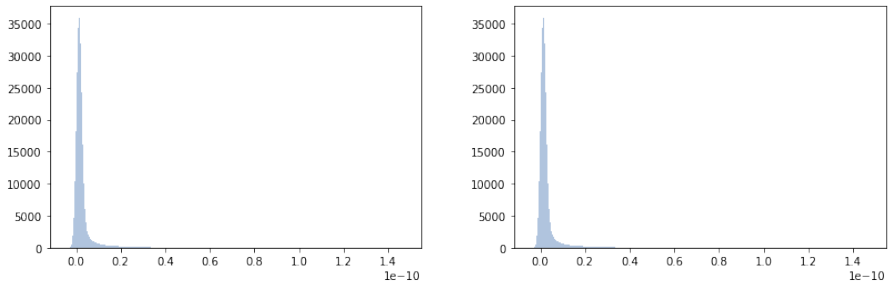
Figure 3.11: Crops of the simulated image (Fig. 9), where (a) is the original image without noise and (b) is the same crop with a Poisson noise model added. To display the images, we used a Histogram Equalization Stretch.

Note, that the above presented images are generated by using a Histogram Equalization. Combined with the simulation technique to generate these images, there seem to be artifacts in the noiseless image which look like squares. It should be further investigated if the simulation of these noiseless images can be improved to exclude these artifacts, but the Histogram Equalization makes signifies these artifacts, while in the actual data they are less significant. We generate the denoising metrics where we use the noiseless and the image without noise as input, to see what the upper-bound and lower-bound are. These metrics are displayed below in Table 3.8.

	PSNR	SSIM	NRMSE
Noised image	42.279	0.936	0.226
Original image	∞	1.0	0.0

Table 3.8: The denoising metrics PSNR, SSIM and NRMSE given for the crop of the astronomical image. The metrics are given where the input image is compared to the noiseless image. As input, the noisy and the noiseless image are used, to give the upper-bounds and lower-bounds.

For these astronomical images, we also display the histogram of the pixel intensities to see the distribution of these intensities. These histograms below in Figure 3.12. Interestingly, there seem to be high outliers such that the most pixel values lie within the smallest 10% range.



(a) (b)

Figure 3.12: Histogram of pixel intensities for (a) the gray-scale astronomical image and (b)the gray-scale astronomical image with noise.

3.4.1 *Gaussian Blur on Astronomical Image*

We tune the parameter σ of the Gaussian Blur, and get the resulting graphs which are depicted in Figure 3.13 below.

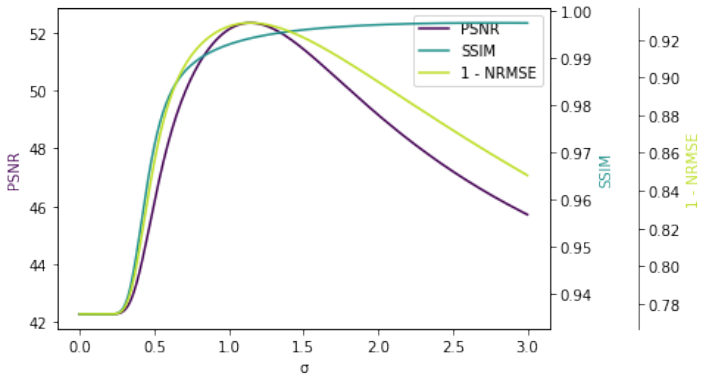


Figure 3.13: Denoising metrics PSNR, SSIM and NRMSE plotted on σ as parameter for Gaussian Blur, using the gray-scale astronomical image as ground truth and the smoothing on the Poisson noise image. Note that we have inverted NRMSE. The optimal σ for each metric is at 1.14, 2.76 and 1.14 for PSNR, SSIM and NRMSE, respectively.

From the graph presented above, we can see that the SSIM value is converges to 1 and has minimal differences when changing σ. Furthermore, increasing σ further did not decrease the value of the SSIM a lot. The reason for this is when comparing the astronomical image to the photographic image of former section, the underlying structure of the astronomical image does stay similar, i.e. there does not exist that much detail within the image as was the case with 'Lena'. Hence, it is harder to deduce an optimal value for the SSIM when reducing the noise in this image, while we aim for reducing noise. Hence, we can conclude that the SSIM is not a suitable denoising metric for the

use of smoothing the astronomical images. For sake of completeness, we report the SSIM with the remainder of the experiments, but we will focus mainly on the PSNR and NRMSE to draw conclusions from about the smoothing techniques. For the PSNR and the NRMSE, we can see that the distribution is similar to the grid-search performed on the photographic image in Figure 3.4, but that the curve is less steep. This also is reflected in the higher optimal bandwidth values presented in Table 3.9.

	PSNR	SSIM	NRMSE
Optimal Value	52.338	0.997	0.071
$\sigma = 3$	45.716	0.997	0.152

Table 3.9: Tuning of the σ-parameter for the Gaussian Blur smoothing method, used on crop of the simulated astronomical image. The denoising metrics for $\sigma = 3$ are also displayed, the value for the parameter used in the Gaussian Blur in MTObjects. The optimal σ for each metric is at 1.14, 2.76 and 1.14 for PSNR, SSIM and NRMSE, respectively.

Interestingly, when looking at the above presented table, we see a large difference in performance when smoothing the image with an optimal σ as with $\sigma = 3$, as was used originally by MTObjects.

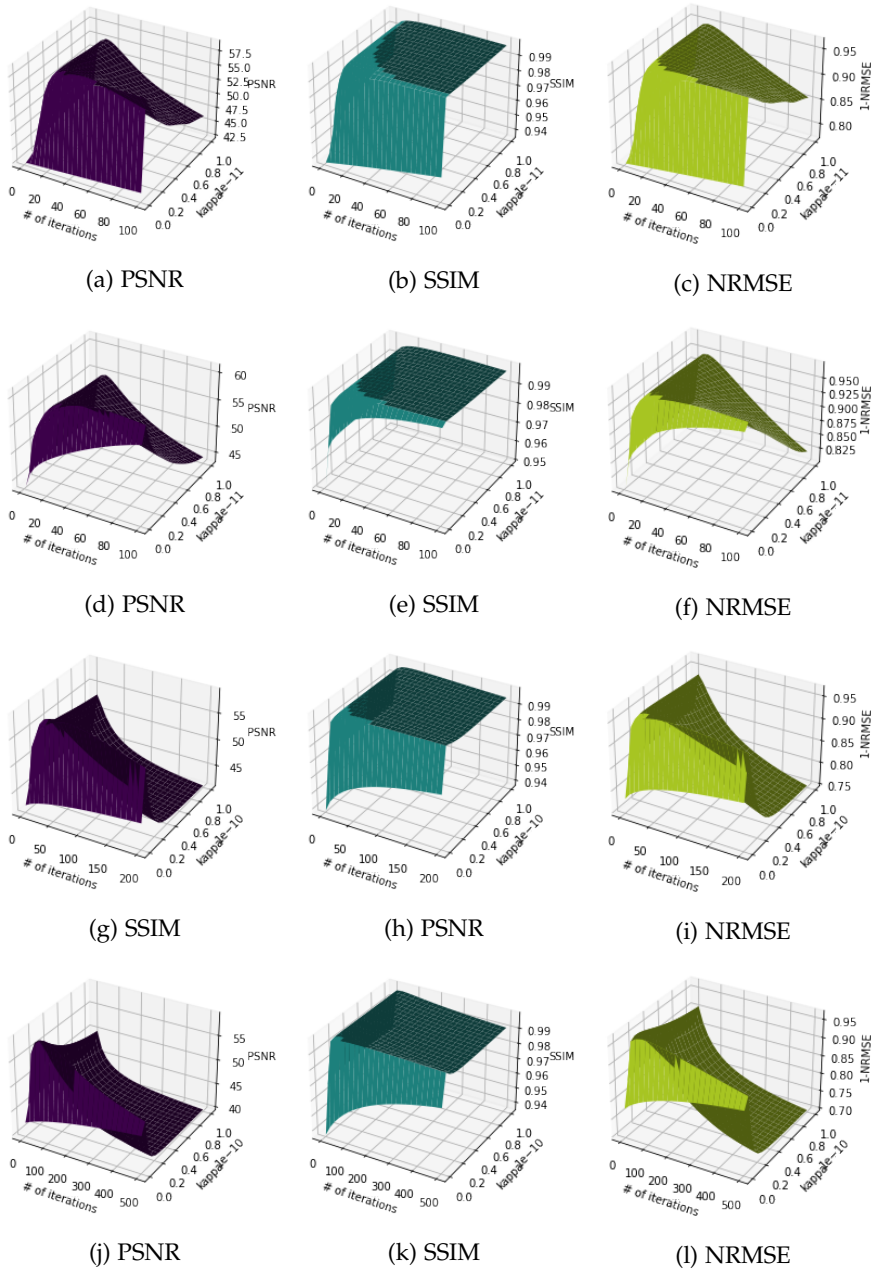
3.4.2 *Perona-Malik diffusion*

In this section we present the results of performing a grid-search for the parameters of the four different variants of the Perona-Malik diffusion tested on the denoising metrics PSNR, SSIM and NRMSE using the crop of the astronomical image. We again, as in Section 3.2.2, use $\gamma = 0.1$ as regularizing parameter. Since κ is a determining the influence of the local gradient, or intensity level, we adjust the range of this parameter to be suitable to the astronomical images. Since, the range of the intensity level is between $1e-14$ and $1e-12$, we set the range of κ to this range. The results of the performed grid-search for each denoising metric is presented in Figure 3.14.

	PSNR	SSIM	NRMSE
Perona-Malik	58.990	0.999	0.033
Perona-Malik with Blur	61.681	0.999	0.024
Perona-Malik intensity	59.221	0.999	0.032
Perona-Malik intensity with Blur	59.350	0.999	0.032

Table 3.10: Tuning of the κ-parameter and the number of iterations for the Perona-Malik diffusion, used on the crop of the astronomical image. The denoising metrics are displayed for the 4 different variants of the Perona-Malik diffusion.

The variant of the Perona-Malik diffusion which has the best performance when looking at the PSNR and the NRMSE, is the variant which uses the blurred gradient as input for the conduction function, as was also the case with the 'Lena' image. The optimal metrics are obtained with $\kappa = 2.7e-13$ for 132 iterations. Note, that the number of iterations is a lot higher compared to the case when using the Perona-Malik Diffusion on a standard image. We can also see this behaviour in the Gaussian Blur, where a higher value for σ means a preference for a more blurred image, which is also how we can interpret the high number of iterations. The two variants which use the Intensity as input for the conduction function have a relative higher performance when comparing with the results for the photographic image, which was what we expect. However, these variants are still outperformed by using the version based on the Gradient.



(a) PSNR (b) SSIM (c) NRMSE

(d) PSNR (e) SSIM (f) NRMSE

(g) SSIM (h) PSNR (i) NRMSE

(j) PSNR (k) SSIM (l) NRMSE

Figure 3.14: Denoising metrics for the smoothed cropped astronomical image by the Perona-Malik Diffusion where we plot on κ and the number of iterations used. We display the results for all variants of the Perona-Malik Diffusion we use, where the Conduction Function has as input: for (a)-(c) gradients, (d)-(f) gradients of blurred image, (g)-(i) intensities, (j)-(l) intensities of blurred image. Where for each variant, we display the PSNR, SSIM and NRMSE from left to right. Note, that we use 1-NRMSE, to compare the distribution of the grid-search to the PSNR and the SSIM metrics.

3.4.3 *KDEsmoother on Astronomical Image*

In this section we present the results of the KDEsmoother used on the cropped simulated astronomical image, which is shown in Figure 3.11(b). We use three variants of the KDEsmoother:

- KDEsmoother with uniform kernels (same size and isotropic for each pixel).
- KDEsmoother with size-adaptive kernels
- KDEsmoother with shape-adaptive kernels

First, we start with using the KDEsmoother with uniform kernels. The results of the grid-search performed on the denoising metrics for different values of the bandwidth parameter are shown in in Figure 3.15.

Again, we can compare this graph to the graph of the grid-search

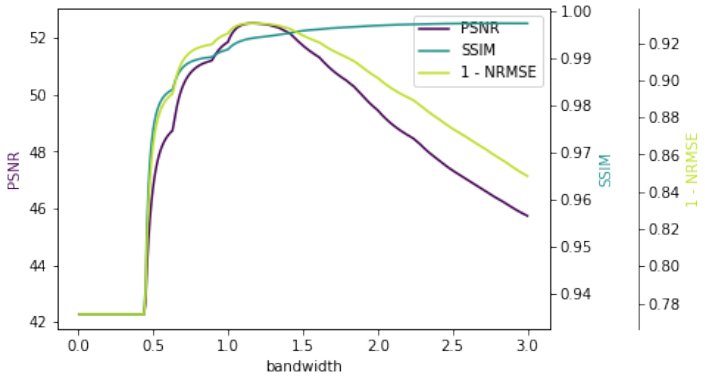


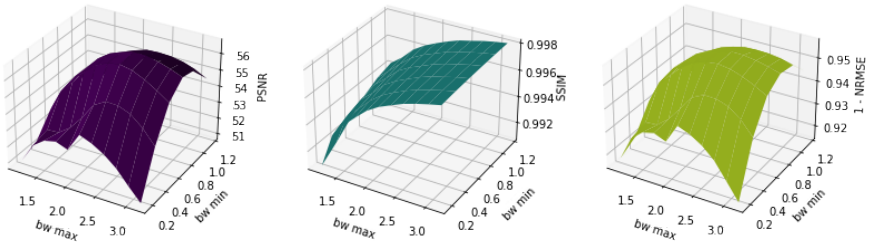
Figure 3.15: Denoising metrics PSNR, SSIM and NRMSE plotted on b as parameter for KDE, using the gray-scale Lena image as ground truth and the smoothing on the Poisson noise image. Note that we have inverted NRMSE. The optimal σ for each metric is at 1.16, 2.79 and 1.16 for PSNR, SSIM and NRMSE, respectively.

optimization of the σ-parameter for the Gaussian Blur, in Figure 3.13. As was the case in former section with the photographic image, we see that the distributions are equivalent, and that the largest difference is the discontinuity in the KDEsmoother. This discontinuity is due to the change in kernel size at the given bandwidth value. The optimal values for the bandwidth parameter for each of the denoising metrics is given in Table 3.11

Performing a grid-search on the size-adaptive version of the KDE-smoother results in the following graphs, displayed in Figure 3.16. We use the optimal bandwidth value for the PSNR and the NRMSE of the KDEsmoother with uniform kernels as upper-bound for the b_{min} parameter and as lower-bound for the b_{max} parameter, which is a value of 1.16. As we can see in Figure 3.16, in contrast with the photographic

	PSNR	SSIM	NRMSE
Value	52.514	0.997	0.069
Bandwidth	1.16	2.79	1.16

Table 3.11: Uniform kernel size, tuned on the parameter bandwidth using the crop of the astronomical image. We display the pairs of optimal denoising metric with corresponding parameter.



(a) PSNR (b) SSIM (c) NRMSE

Figure 3.16: Denoising metrics used on size adaptive variant of the KDEsmoother. We display the results for combinations of b_{min} and b_{max} such that: for (a)-(c) 11 steps of 0.1. Note, that we use 1-NRMSE, to compare the distribution of the grid-search to the PSNR and the SSIM metrics.

image of Lena, we do have optimal values for the denoising metrics where $b_{min} \neq b_{max}$. Hence, the hypothesis that increasing the smoothing at lower intensity values and vice versa holds. The optimal values for these bandwidth parameters are given in Table 3.12. Note, that the

	PSNR	SSIM	NRMSE
Value	56.793	0.998	0.042
(b_{min}, b_{max})	(0.71, 2.41)	(1.16,3.16)	(0.71, 2.41)

Table 3.12: Optimal denoising metrics for the size-adaptive version of the KDEsmoother, where the grid-search is performed on the b_{min} and the b_{max} parameters.

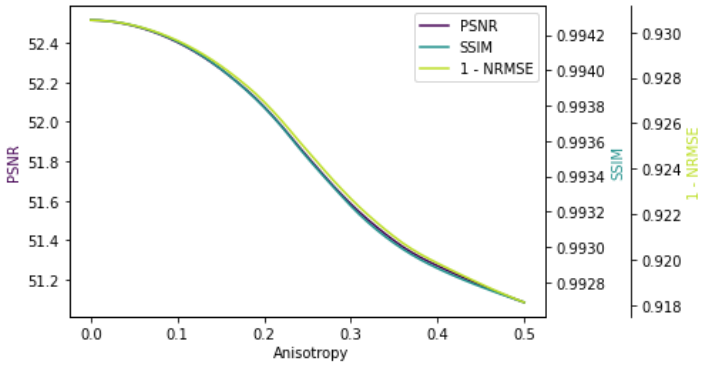
value of the SSIM can be neglected, since it uses the maximum values of the grid for both the b_{min} and the b_{max} parameter. Also, we did not use the optimal bandwidth value for corresponding to the SSIM as upper-bound and lower-bound for the b_{min} and b_{max}.

Finally, we introduce the shape-adaptivity to the KDEsmoother. We test whether the shape-adaptivity increases the performance for:

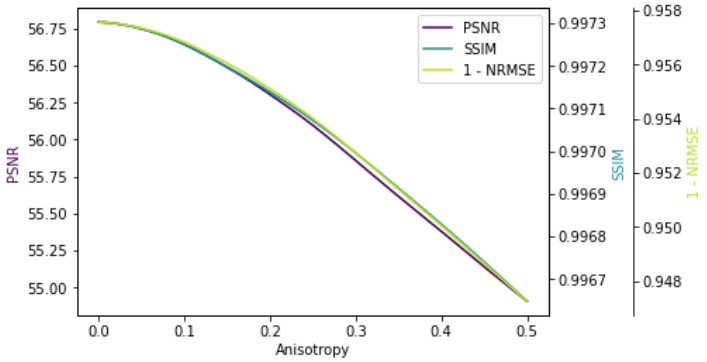
- The KDEsmoother with uniform kernels, where $b = 0.70$, corresponding to the optimal bandwidth for the PSNR and the NRMSE

- The KDEsmoother with size-adaptive kernels, where $(b_{min}, b_{max}) = (0.71, 2.41)$, corresponding the the optimal bandwidth parameters for the PSNR and the NRMSE.

The results of the grid-search performed on α are given in Figure 3.17.



(a)



(b)

Figure 3.17: Denoising metrics PSNR, SSIM and NRMSE plotted the α parameter, which determines how much the kernel adapts its shape. In (a) we have plotted the shape adaptive KDEsmoother with optimal uniform size kernels, and in (b) we have plotted the shape adaptive KDEsmoother with optimal adaptive size kernels. In both graphs, we can see that increasing α decreases the performance, and hence the KDEsmoother works best with isotropic kernels.

As can be seen in both graphs in Figure 3.17, increasing α reduces the performance of the KDEsmoother, and hence the optimal values are in both situation where $\alpha = 0$. We display the result in the overview of all smoothing methods in Table 3.13.

3.4.4 *Overview of the results*

The denoising metrics for all smoothing methods used in the former sections are summarized in the table presented below.

	PSNR	SSIM	NRMSE
Without smoothing	42.279	0.936	0.226
Same image	∞	1.000	0.000
Gaussian Blur $\sigma = 3$	45.716	0.997	0.152
Gaussian Blur	52.338	0.997	0.071
]Perona-Malik	58.990	0.999	0.033
Perona-Malik with Blur	**61.681**	**0.999**	**0.024**
Perona-Malik intensity	59.221	0.999	0.032
Perona-Malik intensity with Blur	59.350	0.999	0.032
KDE isotropic	52.514	0.997	0.069
KDE size adaptive	56.739	0.998	0.042
KDE size & shape adaptive	56.739	0.998	0.042

Table 3.13: Overview of the denoising metrics for all smoothing techniques with optimal parameter setting. These are all results of smoothing the crop of the astronomical image with noise and comparing the result to the noiseless image.

From the overview of the results presented in Table 3.13, we see similar results when comparing with the results of the photographic image in Table 3.7. For the astronomical image, we also have that is the Perona-Malik Diffusion where the conduction function used the gradient combined with a Gaussian Blur is the best performing smoothing method in terms of the three denoising metrics. The differences between the astronomical image and the photographic image are that the variants of the Perona-Malik Diffusion which use the local intensities instead of the local gradients seems to perform relatively better. Also, the size-adaptivity increases the performance of the KDEsmoother when comparing to the KDEsmoother using uniform kernels. Both these results suggest that the hypothesis that the astronomical images are better denoised when increasing the smoothing for lower intensity values, and vice versa, holds.

We display the smoothed image for the optimal parameters for the best performing variant of each smoothing method in Figure 3.9 on the next page.

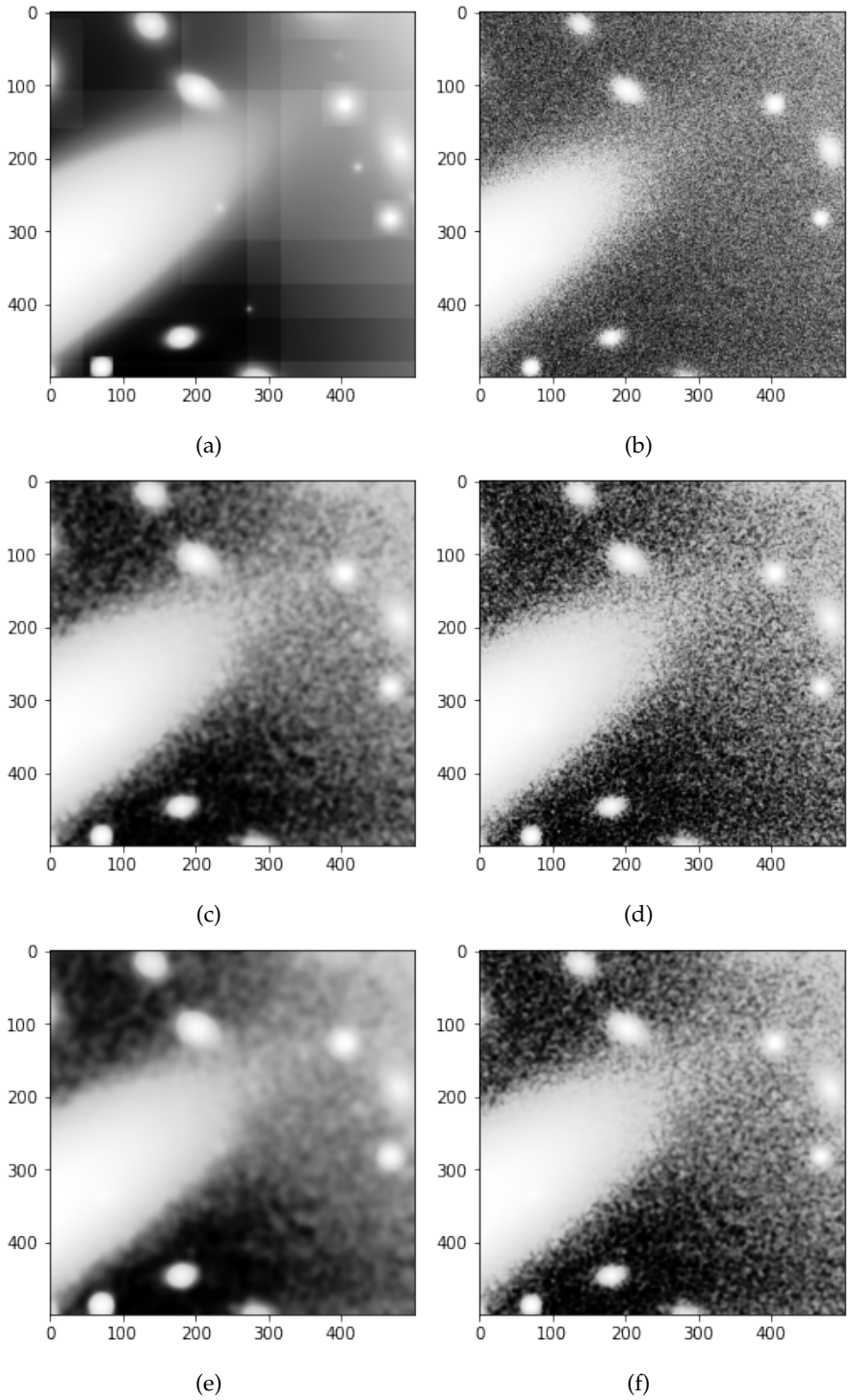


Figure 3.18: Overview of the original and the smoothed images of Lena, where the following images are presented: (a) Astronomical image without noise, (b) Astronomical image with noise, (c) Gaussian Blur with $\sigma = 3$, (d) Optimal Gaussian Blur, (e) Optimal Perona-Malik Diffusion, (f) Optimal KDEsmoother. All presented images are displayed by using a Histogram Equalization.

3.4.5 *Run-time comparison*

The running times of the optimal methods for the Gaussian Blur, the Perona-Malik Diffusion and the KDEsmoother are given in the table below, Table 3.14.

	Run-time (in seconds)
Gaussian Blur	0.0099
Perona-Malik Diffusion	2.1360
KDEsmoother	1166.4242

Table 3.14: Run-times of the different smoothing techniques on the 500 by 500 pixel crop of the simulated astronomical image. The running times are displayed in seconds, where we have used the optimal parameter setting resulted from the former performed grid-search on the denoising metrics.

The high running time of the KDEsmoother comes from the number of operations performed on each individual pixel in the image. For each pixel, the local information is gathered, and a new kernel is generated depending on this local information. It should be further investigated how to reduce the running times of the KDEsmoother, for example by setting parts of the operations in parallel. The running time for the KDEsmoother also depends highly on the bandwidth parameter, and the ratio of low intensity pixels. The latter increases the running time since low intensity pixels use larger kernels. When the running time is optimized for the KDEsmoother, this technique could be considered as a smoothing technique for the astronomical images used by MTObjects, but in this form the running time is to high. When extrapolating the running time to the original image size, which is 10,000 by 10,000 pixels, we would have a running time of 5 to 6 days for smoothing one image. Note, that these are even simulated images, where the actual astronomical images have sizes of 80,000 by 80,000 pixels, which would result in a running time of almost a year on the Peregrine HPC.

3.4.6 *Combining with MTObjects*

In the former section, we have shown that based on the denoising metrics, the optimal smoothing method seems to be the Perona-Malik Diffusion where the conduction function uses the gradient of the blurred image. the corresponding parameters are $\kappa = 2.7e - 13$ where the method is used for 132 iterations. The next step of this research is to see how this smoothing technique improves the performance of MTObjects on the full simulated 10,000 by 10,000 pixel images. We have displayed the resulting segmentation maps in Appendix B. For each of the 10 simulated images, we display the following

- Image without noise
- Image with noise
- MTObjects segmentation map on noiseless image
- MTObjects segmentation map on noised image
- MTObjects segmentation map on noised image smoothed by optimal PM diffusion
- 'Ground Truth' segmentation map

As can bee seen in these images, the 'ground truth' segmentation maps we received corresponding to these image seem incorrect. Hence, comparing the performance with the noiseless and the noised images makes no sense and thus should be done in future work. For this research, we wanted to compare the following metrics

- Precision
- Recall
- F1-score
- Under-merging Error
- Over-merging Error

It should be investigated what is causing these 'ground truth' segmentation maps to be off, and then the above suggested metrics can be compared for different inputs of MTObjects of the same underlying image. Looking at the segmentation maps in Appendix B, we can see that the smoothed image by the Perona-Malik Diffusion seems to be off in some areas. However, in other areas the noise seems to be reduced and the segmentation looks more like the segmentation as with the noiseless image as input. Note that the noiseless image as input does not result in a 'ground truth' segmentation map, and no hard conclusions can be drawn from these images yet.

4

CONCLUSION AND FUTURE WORK

MTObjects is a promising technique, which separates itself from the alternative methods by constructing a Max-Tree. Since MTObjects performs better prepossessing the image by a smoother, where in the original paper a Gaussian Blur with $\sigma = 3$ is used, we considered and constructed shape adaptive smoothing methods. We have developed an image smoothing technique based on the Kernel Density Estimation, which uses the local curvature and intensity to adapt it's shape and it's size. Also, we have used the Perona Malik Anisotropic Diffusion, where we have adjusted this function to run on intensity also. Using the denoising metrics PSNR, SSIM and NRMSE to set the parameters for these methods, we see that these adaptive smoothing methods outperform the Gaussian Blur in terms of denoising. Hence, it looks promising to use these techniques as preprocessing step for MTObjects.

The proposed smoothing technique, the KDEsmoother, is tested on a 'natural' image and on a MTObjects specific image, and compared to the Gaussian Blur and different variants of Perona-Malik diffusion. The denoising metrics PSNR, SSIM and NRMSE are used to evaluate how well the image is denoised. Interestingly, the SSIM does seem not work well for the simulated images. It could be that the window size of 7 by 7 pixels for the SSIM is too small, and hence this should be investigated further. Also, the SSIM could be tested on the whole image instead of using windows. Hence, we concluded that PSNR and the NRMSE are better suited metrics to obtain results with. All grid-search results on these parameters had the same parameters as optimum. As future work it could be tested how well these metrics translate to the performance of the segmentation done by MTObjects. Also, a further investigation in denoising metrics could be done.

The results showed that the KDEsmoother using uniform kernels has similar performance compared to the Gaussian Blur. Size-adaptivity increases the performance for the KDEsmoother when used for the astronomical images, and hence outperforms the Gaussian Blur. The shape-adaptivity did not increase performance for either photographic or astronomical images. We have made some design choices in the shape-adaptive variant, such as preserving the kernel size, which could be investigated further to see if shape-adaptivity can improve this smoothing technique. For example, the local information of the curvature could be prone to small objects and noise, and hence an interesting field of research could be performed on steering the kernels by using the information of a downsampled image. Finally,

both the KDEsmoother and the Gaussian Blur where outperformed by the Perona-Malik Diffusion for all denoising metrics. The version of the Perona-Malik Diffusion which had the best performance, is the variant which used the blurred gradient as input for the conduction tensor function.

Both the KDEsmoother and the Perona-Malik Diffusion had increased performance when increasing the smoothing for lower intensity values for the astronomical images, where this was not the case for the photographic images. Hence, the hypothesis holds that in the astronomical images, the regions with high photon counts should be smoothed less than the regions with low photon counts.

The running time of the KDEsmoother was in orders of magnitude slower compared to the Gaussian Blur and even the Perona-Malik diffusion. Hence, this method should be improved in terms of running time when used on real astronomical images. Future research can be done on using the KDEsmoother on the simulated images as presented in this thesis, but another interesting research opportunity is to see how well the methodology applies to real data.

Kernel Density Estimation can be used as a smoothing method, but developing such a method introduced parameters to tune. Combined with the slow run-time, this technique seems not suitable to set the parameters for the Perona-Malik Diffusion. Also, the techniques are comparable, but a big difference is that the KDEsmoother uses local information of a given pixel to influence the local neighbourhood of that pixel, where the Perona-Malik Diffusion uses local information of a given pixel to influence that given pixel.

In this research we have focused on two adaptive smoothing methods which are flux preserving. It is interesting to test if the assumption that flux preserving adaptive smoothing methods outperform non-flux preserving methods. Since we use the smoothed images for segmentation, and keep the non-smoothed image for further operations, there seems to be not a requirement to have a flex-preserved smoothed image for the segmentation part. Hence, we propose for future work to investigate into more smoothing methods and compare these to the smoothing methods from this research. Also, Deep Learning has proven to be a well suited method to denoise images in a field called Super Resolution [5], and hence this could also be done in future work. Next to using Deep Learning as a smoothing or denoising technique, Deep Learning has proven to be a well suited technique for segmentation [13] . Hence, another interesting research topic would be to see how well a Deep Learning Architecture can perform segmentation on

astronomical images compared to MTObjects.

Furthermore, MTObjects has been developed for segmentation for astronomical images which are made with telescopes. One of the requisites of the images is that the underlying values in the image represent the photon counts. In the field of microscopy, a lot of imaging is done with photon counts, and there exists some similarity to astronomical structures and, for example, protein structures. Hence, an interesting divergence would be to test whether MTObjects is a suitable method for segmentation in microscopy.

A

APPENDIX - PSEUDOCODE

Algorithm 1 KDEsmoother

procedure KDESMOOTHER(I)
  $(w, h) \leftarrow shape(I)$
  $I_S \leftarrow zeros(w, h)$
  $I_{BK} \leftarrow zeros(w, h)$
  $I_{PE} = GaussianBlur(image = I, \sigma = 3)$
  for $i \leftarrow 0, w - 1$ **do**
    for $j \leftarrow 0, h - 1$ **do**
      $B \leftarrow determineBW(I_{PE})$
      $K_s \leftarrow EpanechnikovKernel(B)$
      $K, x_b, x_e, y_b, y_e \leftarrow FoldBoundaries(K_s, i, j, w, h)$
      $I_S[i - x_b : i + x_y, j - y_b : j + y_e] += K \times I[i, j]$
      $I_{BK}[i - x_b : i + x_y, j - y_b : j + y_e] += K$
  return I_S, I_K

Algorithm 2 EpanechnikovKernelGrid

procedure EPANECHNIKOVKERNELGRID(B)
  $r \leftarrow floor(\det(B) * 7)$
  $l \leftarrow r * 2 + 1$
  $x \leftarrow linspace(-r, r, l)$
  $y \leftarrow linspace(-r, r, l)$
  $K \leftarrow zeros(l, l)$
  for $i \leftarrow 0, l$ **do**
    for $j \leftarrow 0, l$ **do**
      $u \leftarrow [x[i], y[j]]$
      $K[i, j] \leftarrow EpanechnikovKernel(u, B)$
  return K

Algorithm 3 EpanechnikovKernel

procedure EPANECHNIKOVKERNEL(u, B)
    $d_B \leftarrow \det(B)$
    $x \leftarrow inverse(B) \cdot (u)$
    if $x \cdot x < 5$ **then**
        $k \leftarrow \frac{1}{5 \times d_B} \frac{4}{2 \times \pi} \left(1 - \frac{x}{\sqrt{5}} \cdot \frac{x}{\sqrt{5}}\right)$
    else
        $k \leftarrow 0$
    return k

Algorithm 4 Perona Malik Diffusion

procedure PMD(I, it, kappa, gamma, op)
    $Out \leftarrow I$
    for $i \leftarrow it$ **do**
        $grad_v, grad_h \leftarrow gradient(Out)$
        if $op == 1$ **then**
            $m1 \leftarrow c(Out) * grad_v$
            $m2 \leftarrow c(Out) * grad_h$
        if $op == 2$ **then**
            $m1 \leftarrow c(grad_v) * grad_v$
            $m2 \leftarrow c(grad_h) * grad_h$
        $m1, \leftarrow grad(m1)$
        $, m2 \leftarrow grad(m2)$
        $Out \leftarrow Out + \gamma \times (m1 + m2)$
    return Out

B

SIMULATED IMAGES AND SEGMENTATION MAPS

In this Appendix we show the 10 simulated astronomical images, where we display each image with and without noise. Also, we display the segmentation maps generated by MTObjects using both the image with and without noise, but also the image with noise smoothed by the Perona-Malik Diffusion with optimal parameter setting which resulted from the performed grid-search analysis in this thesis. Finally, we display the ground truth segmentation maps of each simulated image, but note, these segmentation maps needs to be investigated since they seem wrong.

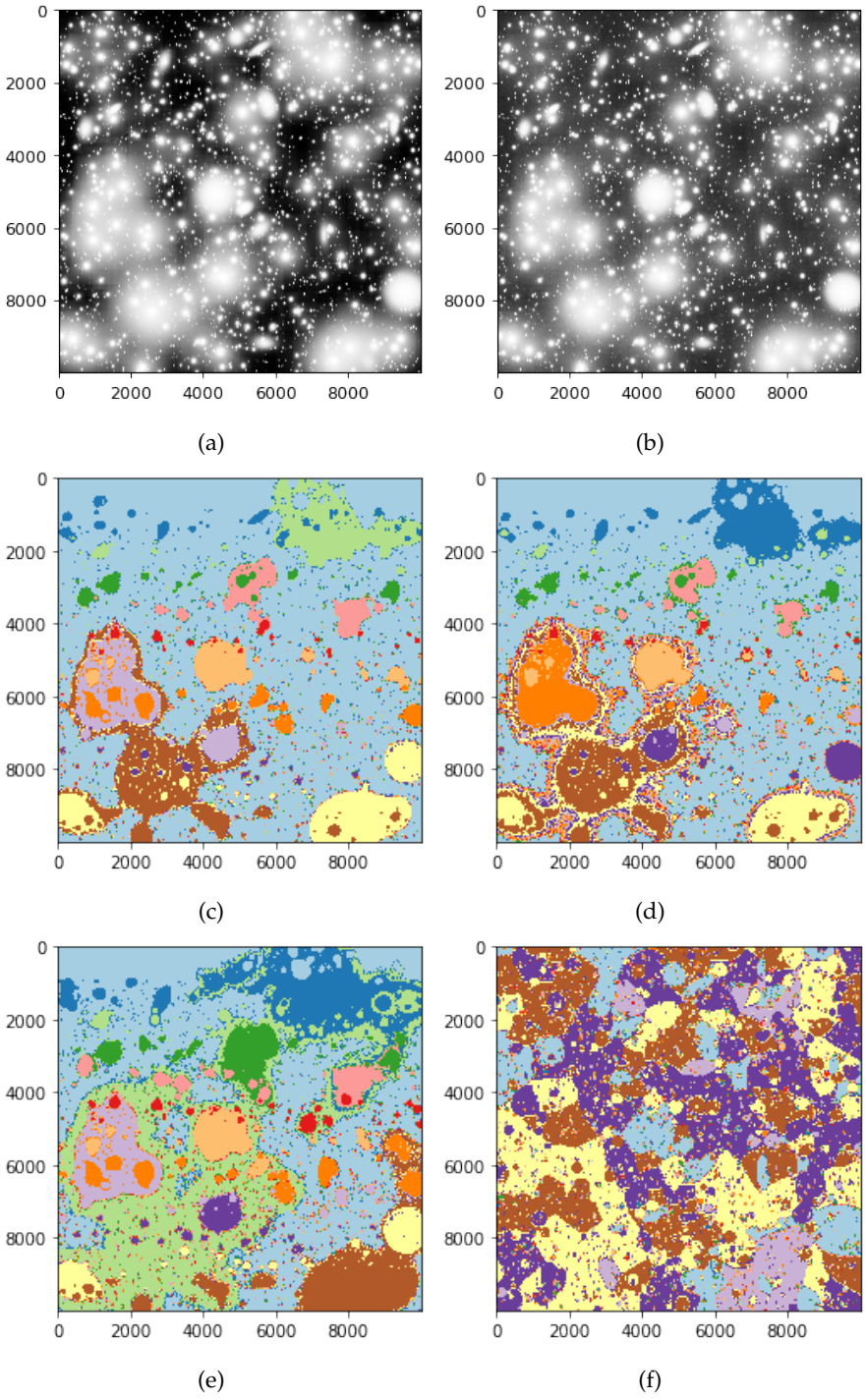


Figure B.1: Simulated Image 0, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

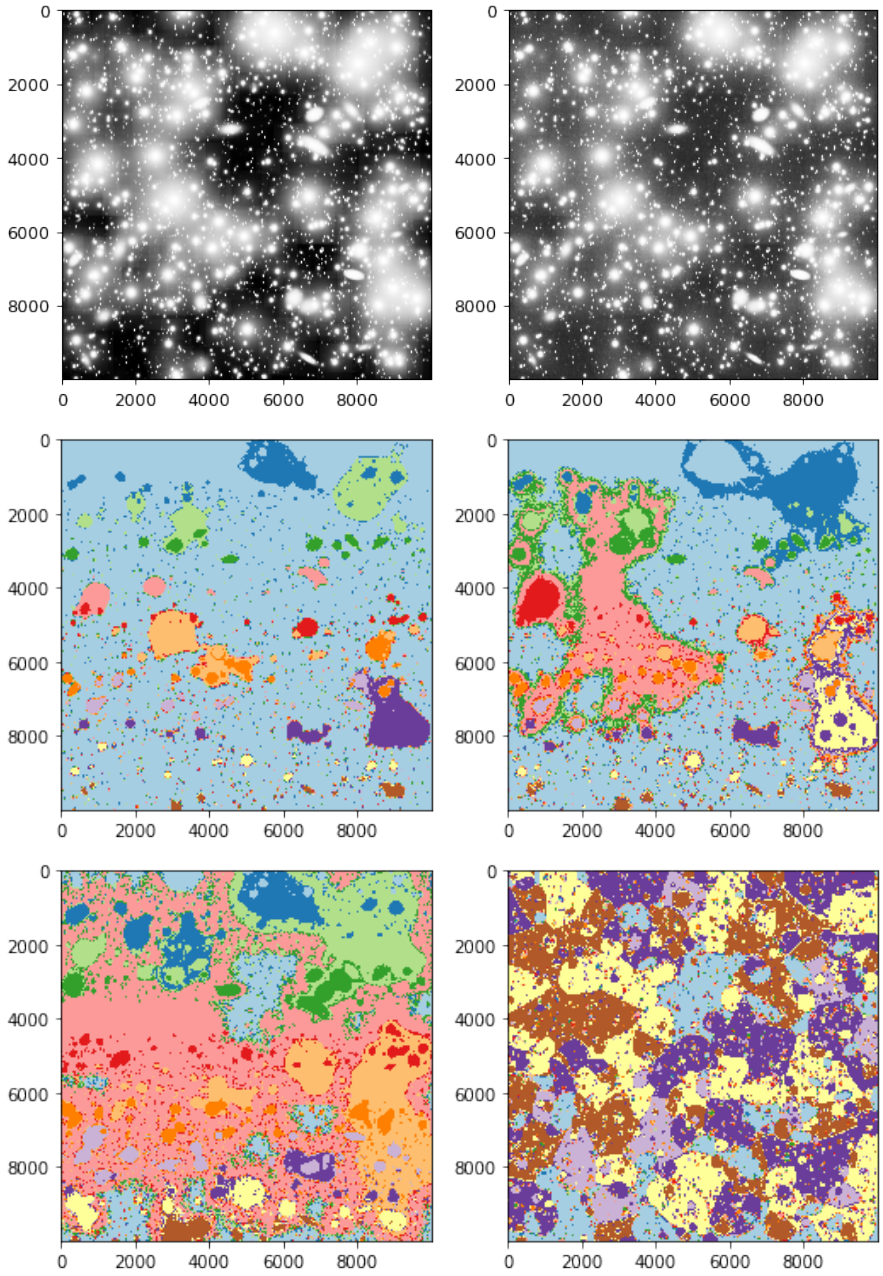


Figure B.2: Simulated Image 1, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

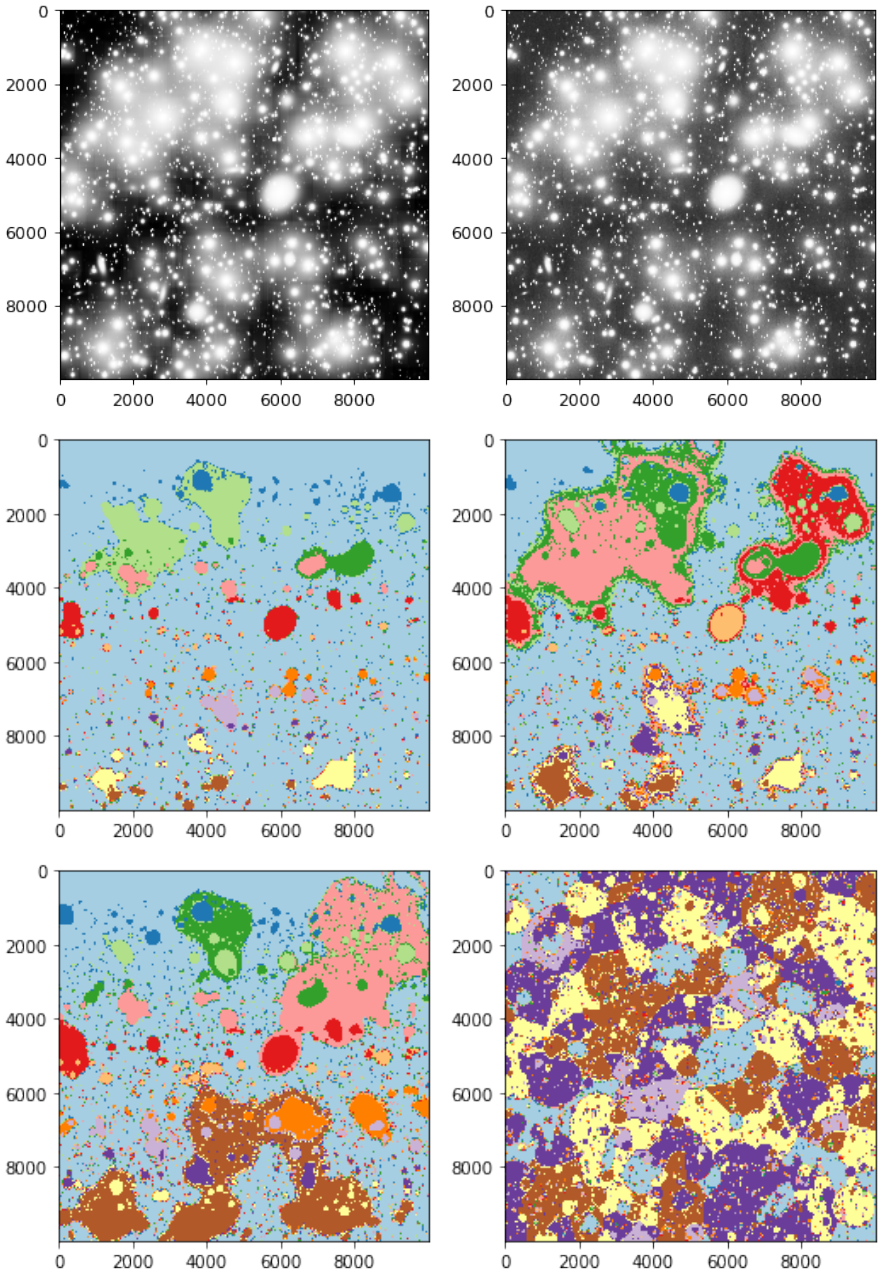


Figure B.3: Simulated Image 2, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

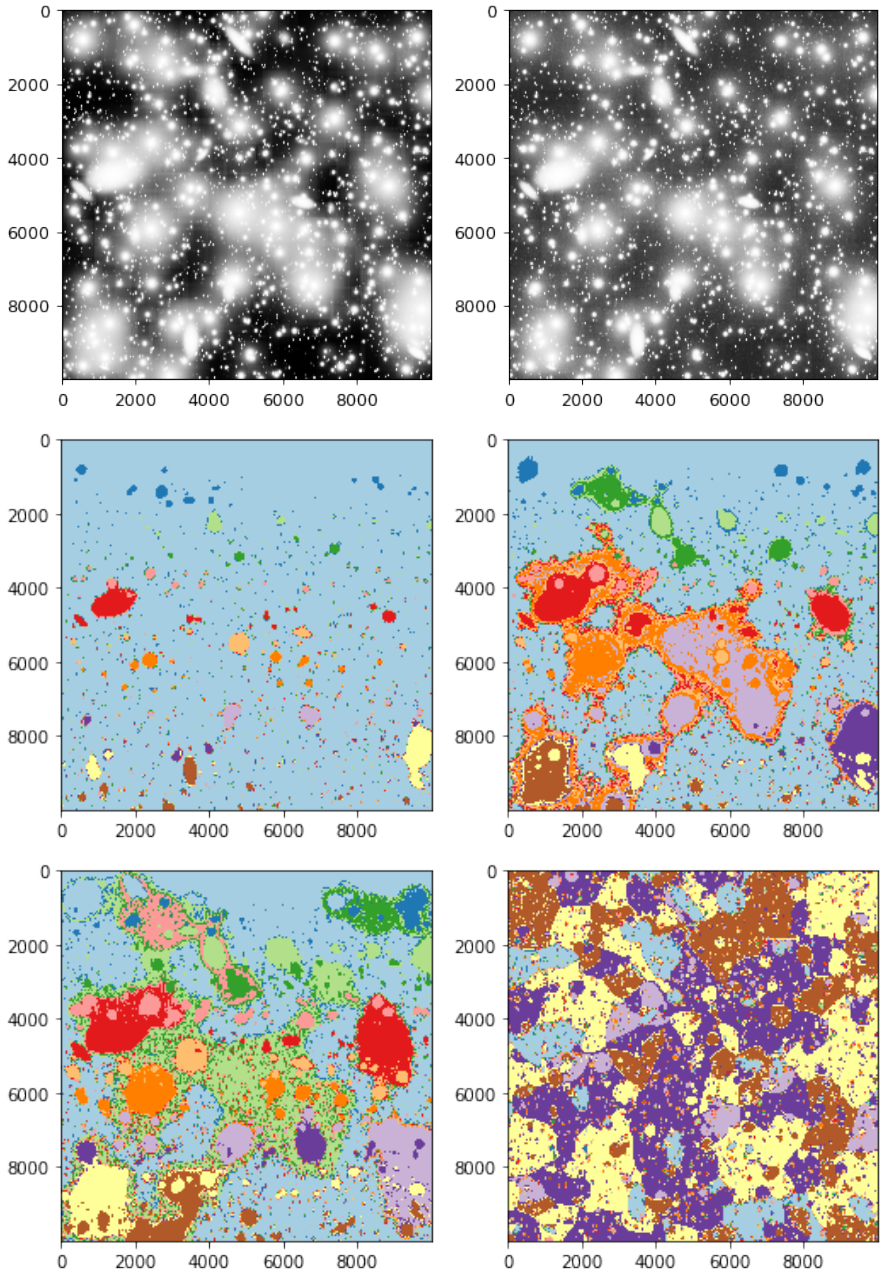


Figure B.4: Simulated Image 3, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

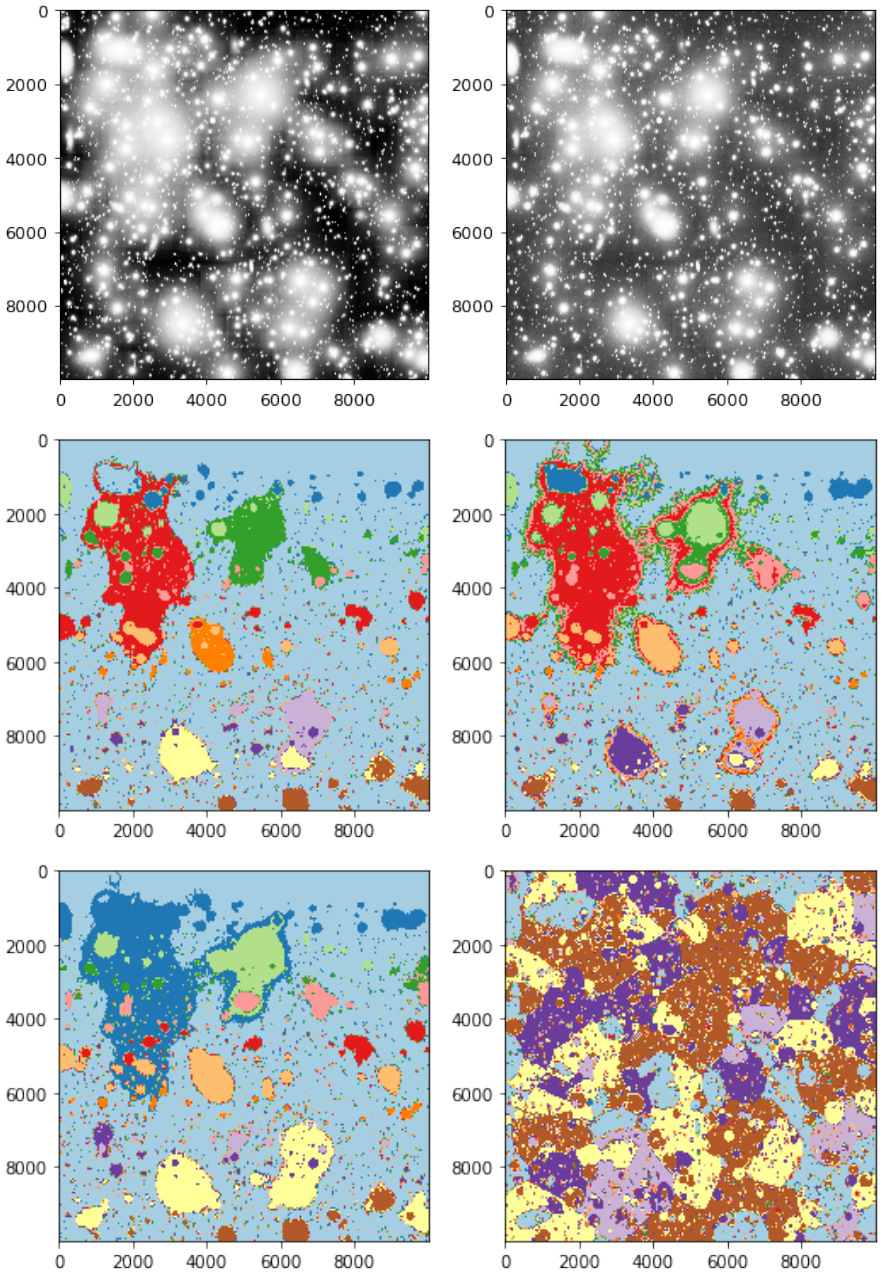


Figure B.5: Simulated Image 4, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

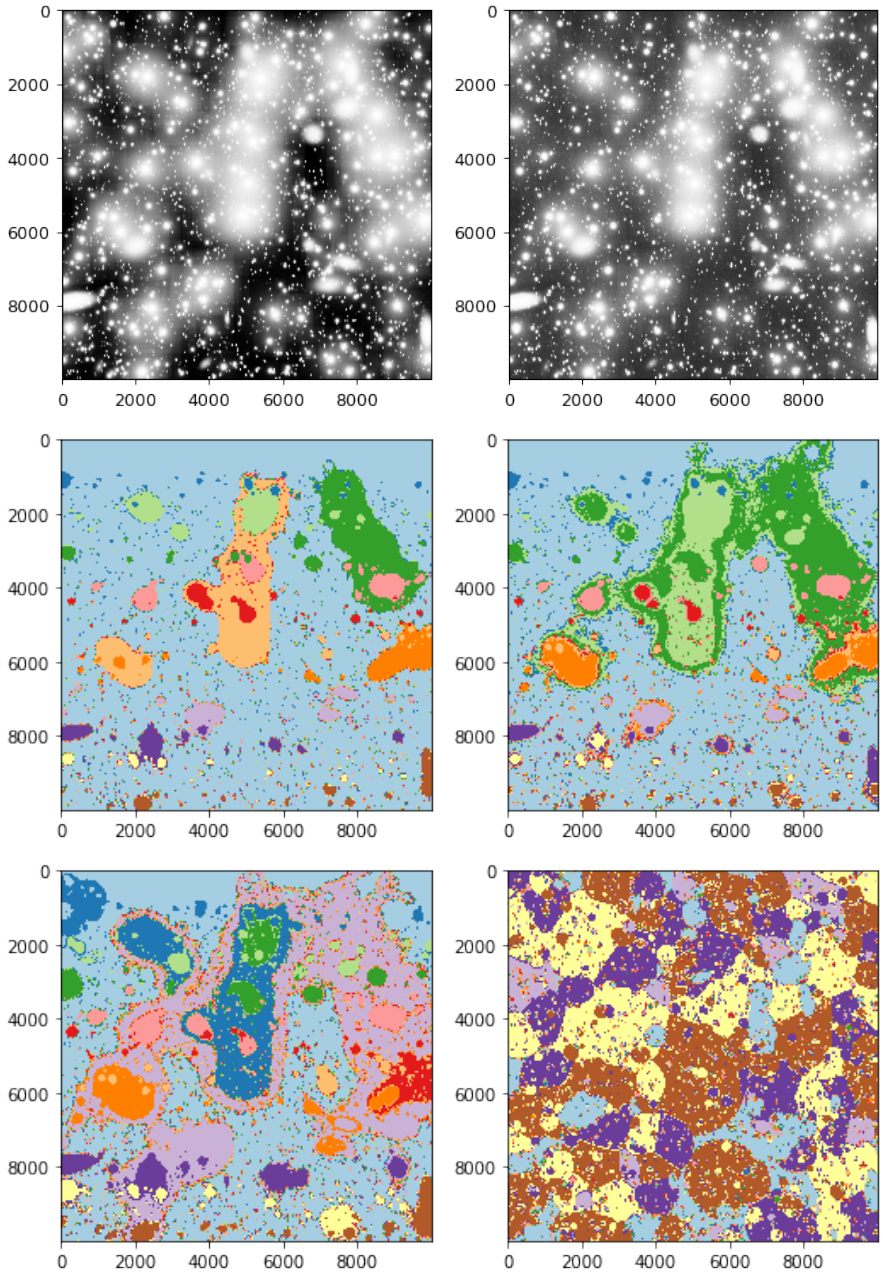


Figure B.6: Simulated Image 5, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

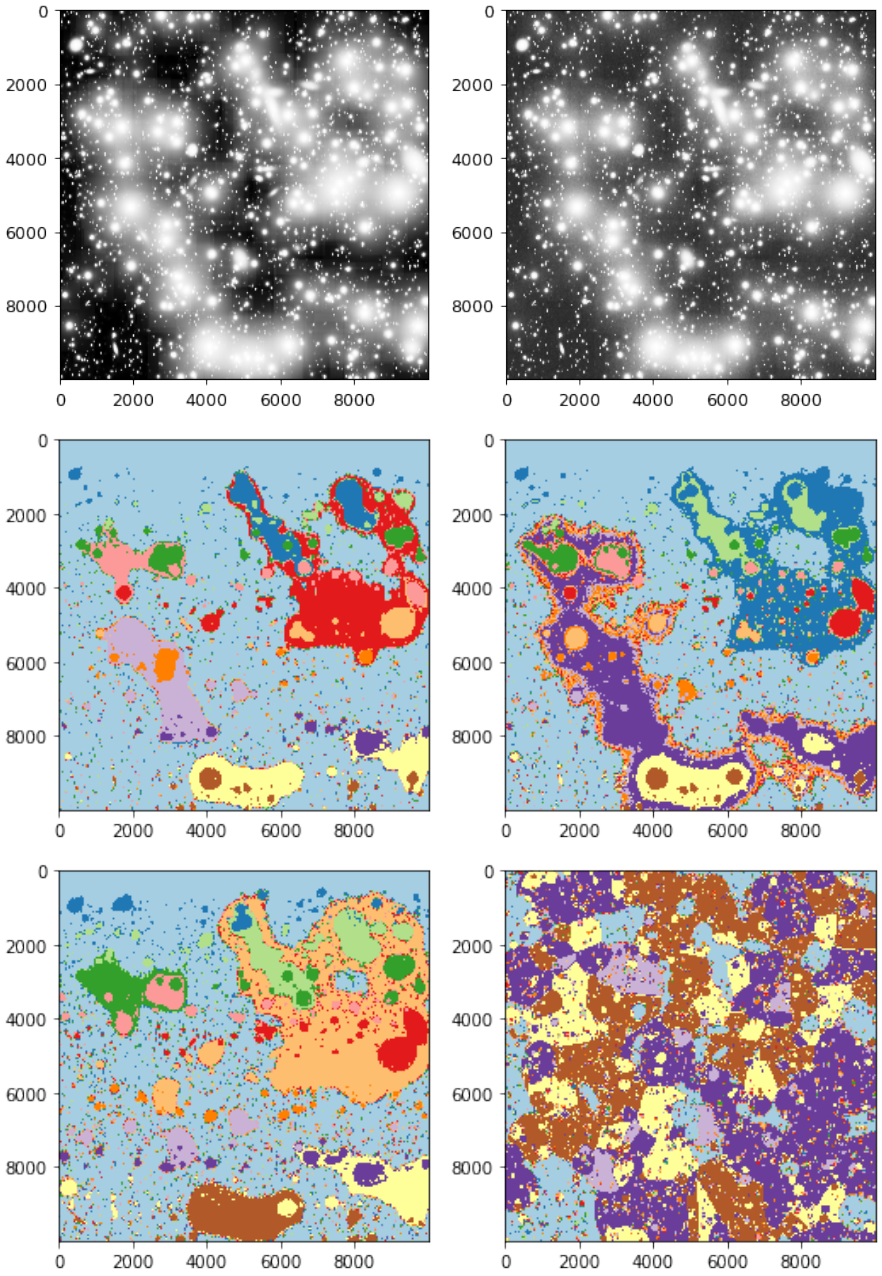


Figure B.7: Simulated Image 6, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

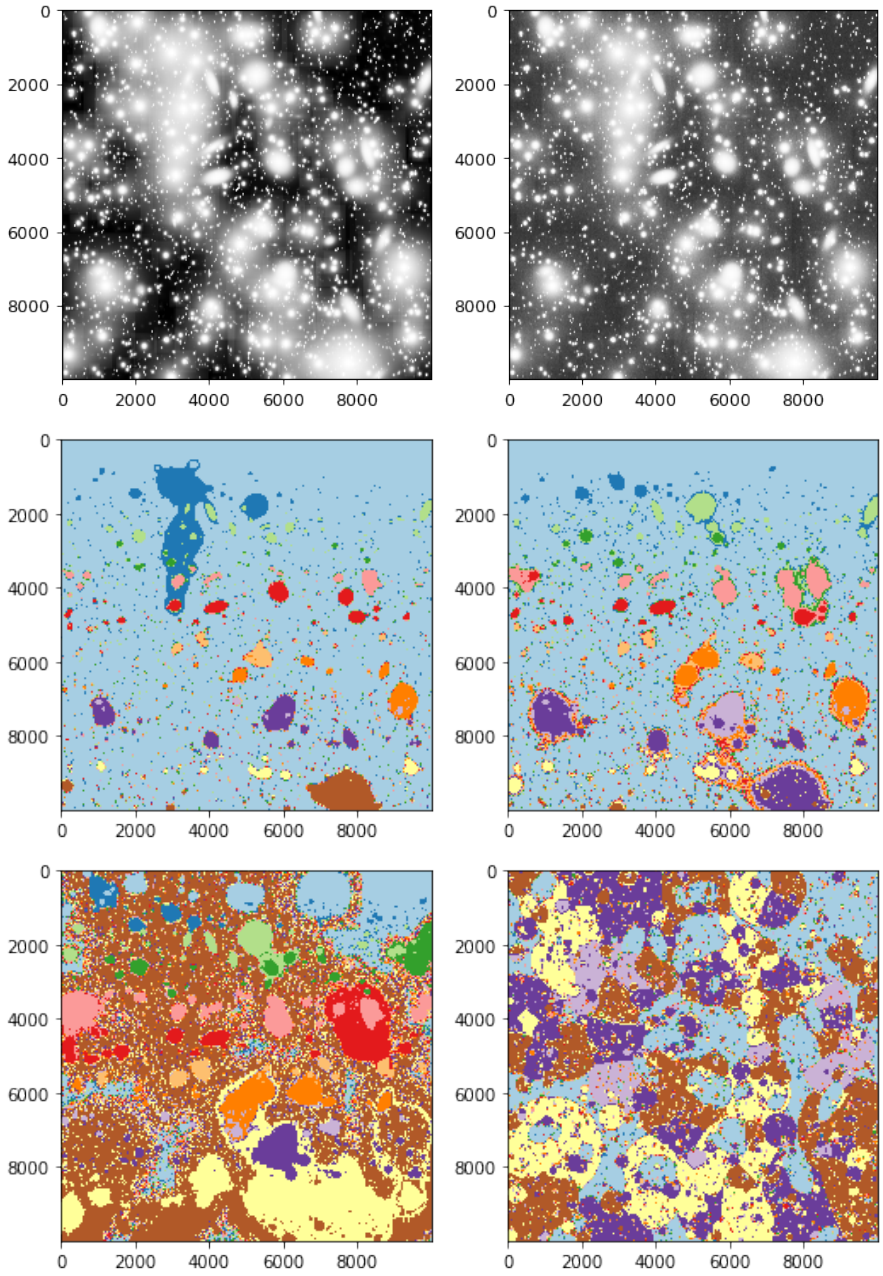


Figure B.8: Simulated Image 7, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

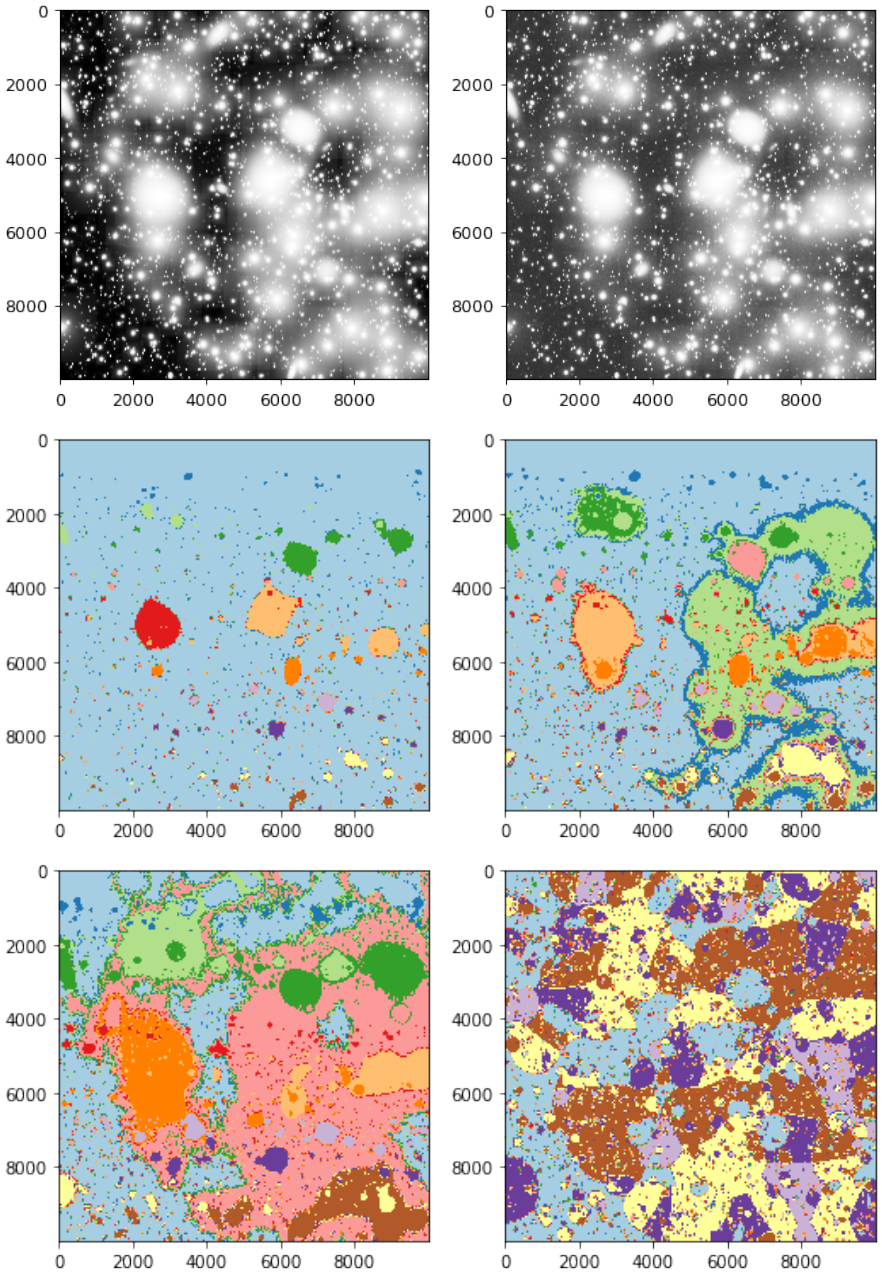


Figure B.9: Simulated Image 8, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

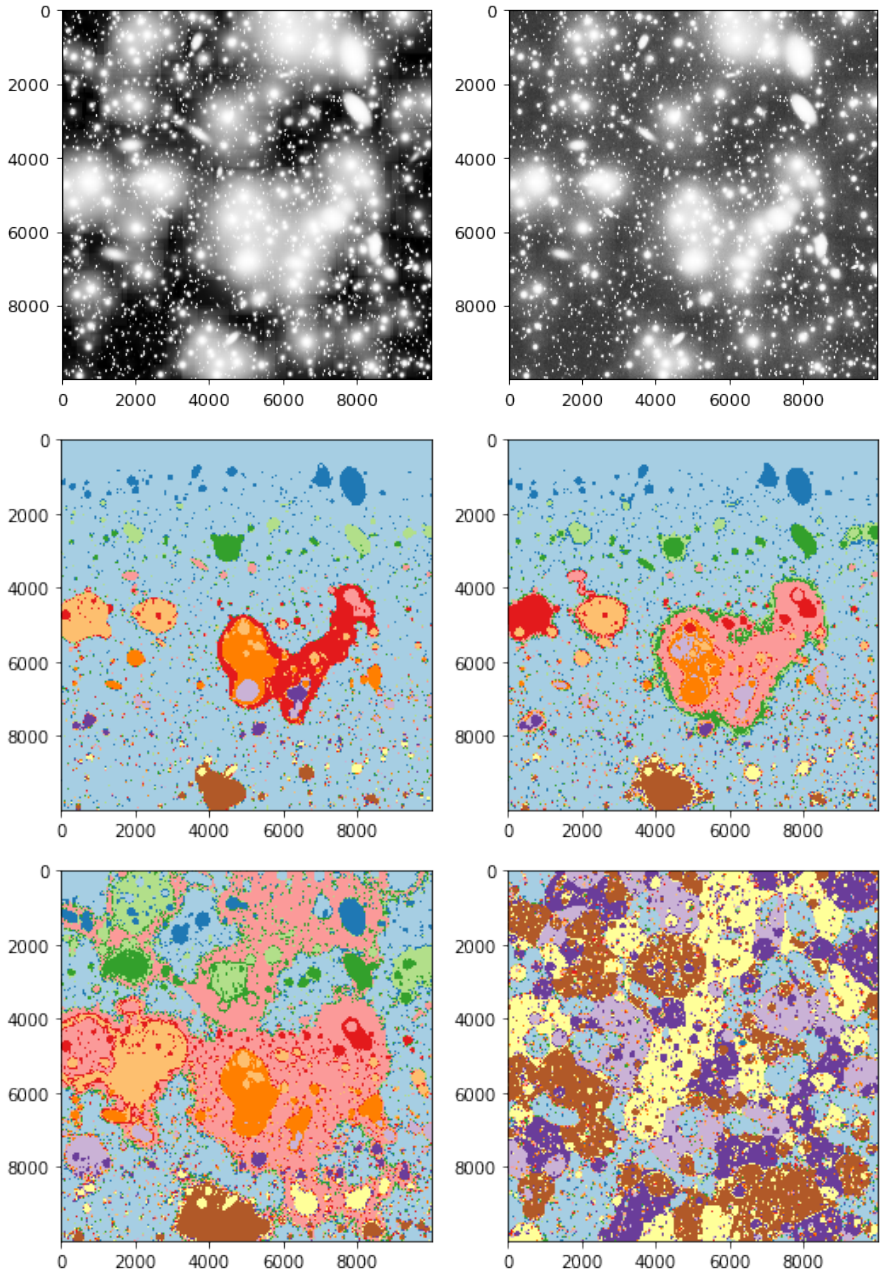


Figure B.10: Simulated Image 9, with (a) Noiseless, (b) With noise, (c) MTObjects segmentation of noiseless, (b) MTObjects segmentation of image with noise, (e) MTObjects segmentation of image with noise smoothed by optimal Perona-Malik Diffusion and (f) Ground truth segmentation map of the simulated image.

BIBLIOGRAPHY

[1] Mohammad Akhlaghi and Takashi Ichikawa. "Noise-based detection and segmentation of nebulous objects." In: *Astrophysical Journal, Supplement Series* (2015). ISSN: 00670049. DOI: 10.1088/0067-0049/220/1/1. arXiv: 1505.01664.

[2] Laura E.N. Baakman. "Shape-Adaptive Kernel Density Estimation." 2017.

[3] E. Bertin and S. Arnouts. "SExtractor: Software for source extraction." In: *Astronomy and Astrophysics Supplement Series* (1996). ISSN: 0365-0138. DOI: 10.1051/aas:1996164. arXiv: 0512139 [astro-ph].

[4] Francine Catte, Pierre Louis Lions, Jean Michel Morel, and Tomeu Coll. "Image selective smoothing and edge detection by nonlinear diffusion." In: *SIAM Journal on Numerical Analysis* (1992). ISSN: 00361429. DOI: 10.1137/0729012.

[5] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. "Image Super-Resolution Using Deep Convolutional Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016). ISSN: 01628828. DOI: 10.1109/TPAMI.2015.2439281. arXiv: 1501.00092.

[6] V. A. Epanechnikov. "Non-Parametric Estimation of a Multivariate Probability Density." In: *Theory of Probability & Its Applications* (1969). ISSN: 0040-585X. DOI: 10.1137/1114019.

[7] Alain Horé and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM." In: *Proceedings - International Conference on Pattern Recognition*. 2010. ISBN: 9780769541099. DOI: 10.1109/ICPR.2010.579.

[8] Jan J. Koenderink. "The structure of images." In: *Biological Cybernetics* (1984). ISSN: 0340-1200. DOI: 10.1007/BF00336961.

[9] Ugo Moschini, Arnold Meijster, and Michael H.F. Wilkinson. "A Hybrid Shared-Memory Parallel Max-Tree Algorithm for Extreme Dynamic-Range Images." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018). ISSN: 01628828. DOI: 10.1109/TPAMI.2017.2689765.

[10] P Perona and J Malik. "Scale-space and edge detection using anisotropic diffusion." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1990). ISSN: 01628828. DOI: 10.1109/34.56205. arXiv: 1102.0183.

[11] Thomas P. Robitaille et al. "Astropy: A community Python package for astronomy." In: *Astronomy and Astrophysics* (2013). ISSN: 00046361. DOI: 10.1051/0004-6361/201322068. arXiv: 1307.6212.

[12] A. S.G. Robotham, L. J.M. Davies, S. P. Driver, S. Koushan, D. S. Taranu, S. Casura, and J. Liske. "Profound: Source extraction and application to modern survey data." In: *Monthly Notices of the Royal Astronomical Society* (2018). ISSN: 13652966. DOI: 10.1093/mnras/sty440. arXiv: 1802.00937.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2015. ISBN: 9783319245737. DOI: 10.1007/978-3-319-24574-4_28.

[14] Murray Rosenblatt. "Remarks on Some Nonparametric Estimates of a Density Function." In: *The Annals of Mathematical Statistics* (1956). ISSN: 0003-4851. DOI: 10.1214/aoms/1177728190.

[15] Yoshinobu Sato, Shin Nakajima, Nobuyuki Shiraga, Hideki Atsumi, Shigeyuki Yoshida, Thomas Koller, Guido Gerig, and Ron Kikinis. "Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images." In: *Medical Image Analysis* (1998). ISSN: 13618415. DOI: 10.1016/S1361-8415(98)80009-1.

[16] Hamid Rahim Sheikh and Alan C. Bovik. "Image information and visual quality." In: *IEEE Transactions on Image Processing* (2006). ISSN: 10577149. DOI: 10.1109/TIP.2005.859378.

[17] Paul Teeninga, Ugo Moschini, Scott C. Trager, and Michael H.F. Wilkinson. "Statistical attribute filtering to detect faint extended astronomical sources." In: *Mathematical Morphology - Theory and Applications* (2016). ISSN: 2353-3390. DOI: 10.1515/mathm-2016-0006.

[18] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. "Image quality assessment: From error visibility to structural similarity." In: *IEEE Transactions on Image Processing* (2004). ISSN: 10577149. DOI: 10.1109/TIP.2003.819861.

[19] Joachim Weickert. "Anisotropic Diffusion in Image Processing." In: *Theoretical foundations of computer vision* (1996). ISSN: 10959572. DOI: 10.1.1.11.751.

[20] M. H.F. Wilkinson and B. C. Meijer. "DATAPLOT: A graphical display package for bacterial morphometry and fluorimetry data." In: *Computer Methods and Programs in Biomedicine* (1995). ISSN: 01692607. DOI: 10.1016/0169-2607(95)01628-7.