Introducción al procesamiento de imágenes médicas
UBA
Trabajo Práctico I

Facundo Manuel Quiroga

January 5, 2014

Contents

1 Introduction **2**

2 Materials and methods **6**
2.1 Definitions 6
2.2 Gaussian Filtering 8
2.3 Bilateral Filtering 9
2.4 Thresholding with Otsu's method 11
2.5 Dataset and testing methods 17

3 Results **23**
3.1 Sample images 23
3.1.1 Synthetic images 26
3.1.2 MRI images 30
3.2 Similarity and Regularity Plots 32
3.2.1 Synthetic images plots 34
3.2.2 MRI image plot 38

4 Discussion **38**
4.1 Sample images 38
4.2 Similarity and Regularity Plots 40
4.3 Conclusion 41

1 Introduction

The purpose of this lab is to learn about **segmentation** and **filtering** methods, in order to apply them in medical image processing, particularly in **functional magnetic resonance imaging** (**fMRI**) images [1].

After a short introduction to set the context for the lab, we describe three common image processing algorithms: Gaussian Filtering, Bilateral Filtering and Otsu's method for thresholding. All three algorithms where implemented in Python 2.7 using the Numpy and Matplotlib packages bundled in the PyLab environment, and the PIL image library. The algorithms were tested on a dataset to both (a) have a measure of their correctness (b) evaluate their performance, applicability and, in the case of the two filtering algorithms, their differences.

Functional Magnetic Resonance Imaging

Functional MRI is a medical imaging technique that measures brain activity by detecting changes in blood flow in the brain. It employs a scanner, similar to those used in traditional MRI, that also measures the change in magnetization between oxygen-rich and oxygen-poor blood. The technique depends on the fact that neurons that are stimulated have an increased blood supply, and so the blood around the neuron becomes more oxygen rich. This phenomena is called the **Haemodynamic response**.

Therefore, by measuring the change in magnetization, the change in blood supply can be estimated, which in turn is an indicator of brain activity. This approach is known as the **Blood-oxygen-level dependent (BOLD)** method.

Traditional 4D MRI generates a time-labeled sequence of 3-dimensional grayscale images. Each image is composed of **voxels** (a 3D generalization of a pixels). [2] Different tissue types generate different voxel intensities, and so the images can expose body structures.

Functional MRI generates (in addition to the MRI anatomical images), a time-labeled sequence of 3-dimensional grayscale images. A voxel in such images measures the BOLD signal in a certain volumetric region of the brain, indicative of brain activity.

[1] Digital images.

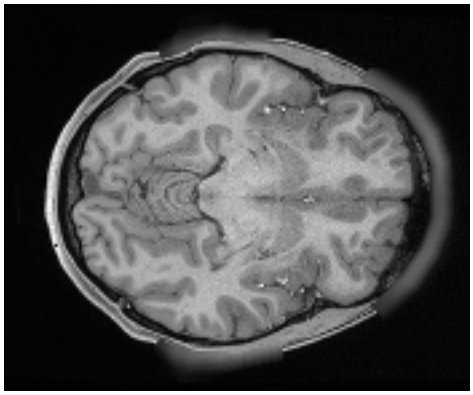
[2] Pixels are 2-dimensional and voxels 3-dimensional; the d-dimensional generalization of both are called **tiles**, but since image processing jargon usually uses the word pixels to refer to 2D tiles which are by far the most common type, we will refer to all three as pixels in the following sections.

Both types of images (functional and anatomical) can come with various types of defects, and so image processing procedures must be applied before performing statistical tests with the information contained in the images.

Segmentation

Segmentation methods partition an image into sets of pixels, assigning some kind of label to each set. Segmentation methods are typically used to identify and locate objects and boundaries in images .

The simplest type of segmentation is called **thresholding**, which splits image pixels into two sets, those above and those below an intensity threshold k.



(a) MRI image

(b) Thresholded MRI image. Pixels with intensity above a certain level k have been converted into white pixels, and pixels with intensity below k to black.

Thresholding can be applied to grayscale images to create binary (ie, black and white) images. Adaptive thresholding techniques find an appropriate intensity level k such that all pixels with intensity greater that k are transformed into white pixels, and those with intensity less than k into black pixels. Thresholding methods thus divide the pixels in an image in two sets C_0 and C_1; those less that k form set C_0, and those greater than k form C_1. What consists of an appropriate intensity level k depends on the problem domain.

In MRI processing, thresholding is used to detect brain areas composed of white matter, gray matter and cerebrospinal fluid, by identifying the sets of pixels that represent each tissue type.

There are of course myriad techniques to perform thresholding; in this lab Otsu's method will be explored [1], a histogram-based technique. Otsu's

method roughly consists of searching, through brute-force, for the intensity level k that minimizes the variance of the intensity values of the two resulting sets of pixels.

Filtering

A filter is a process that removes or enhances certain features or properties of a signal.

In the context of image processing, filtering techniques are commonly used to remove many types of noise, since image acquisition is rarely a straightforward and error-less procedure. Filtering techniques to remove noise from a signal are called **denoising** methods.

The goal of image denoising methods is to recover the original image from a noisy measurement,

$$v(i) = u(i) + n(i) \tag{1}$$

where $v(i)$ is the observed value, $u(i)$ is the "true" value and $n(i)$ is the noise perturbation at a pixel i. The best simple way to model the effect of noise on an image is to consider $n(i)$ as a source of gaussian white noise. In that case, $n(i)$ are i.i.d. gaussian values with zero mean and variance σ^2 .

Several methods have been proposed to remove the noise and recover the true image, u. Even though there is a wide variety of techniques, most share the same basic approach: denoising is achieved by replacing the intensity of each pixel by some kind of average of the intensity of all the pixels [2].

From a frequency perspective, **low-pass filters**, which selectively remove the high and medium frequency contents of an image, leaving mostly the low frequency components, can be employed as denoising filters. The general hypothesis behind such methods is that noise makes an image less smooth, and so higher frequency contents of an image are probably noise.

In the spatial domain, denoising filters usually employ averaging of pixel intensities to **smooth** an image. Every pixel in the resulting filtered image then contains a weighted average of the pixels of the original image. By changing the intensity of a pixel to a weighted average of the pixels of an image, the difference in intensities between pixels is reduced, and so the image become more smooth. The key in this averaging procedure is to calculate appropriate weight for each pixel. Averaging filters typically differ in the way that the weights for pixels are calculated.

We can consider two broad filtering approaches: **domain filtering** and **range filtering**.

Domain filtering only consider pixels that are spatially close to average, according to some distance measure. To calculate the smoothed value for a pixel j, it weights each pixel i in the image in terms of a distance measure $d(i, j)$ between the position of pixel i and that of pixel j.

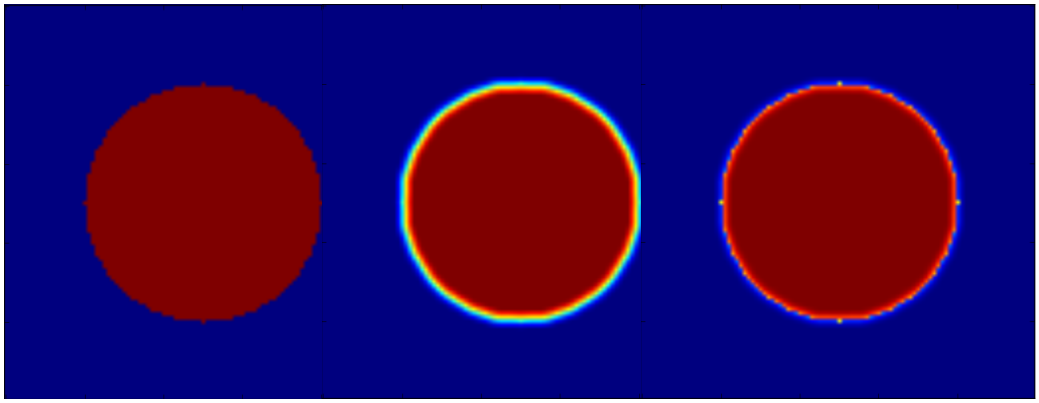
Range filtering weights each pixel i in terms of some similarity measure $s(i, j)$ between the intensity of pixel i , $v(i)$, and the intensity of the pixel (j) whose value is to be replaced, $v(j)$ [3].

This lab explores and compares two simple and well known algorithms: **gaussian filtering** and **bilateral filtering** [3].

Gaussian filtering is a domain filter that computes weights in terms of some variant of the gaussian function $d(i, j) = ae^{(i-j)^T \boldsymbol{A}(i-j)}$, where $a \in \mathbb{R}$ is a constant and $\boldsymbol{A}$ is a constant positive-definite symmetric matrix, such that $d :: \mathbb{R}^d \mapsto \mathbb{R}$.

Bilateral filtering is both a domain *and* range filter. The simplest case of bilateral filtering uses a gaussian function for the range similarity and also for the domain closeness function [4].

Since the noise $n(i)$ is not known, denoising filters must make assumptions that end up distorting the image, and so the transformed image may lack desirable features of the original. Bilateral filtering results in behavior similar to domain filtering, but has the added advantage that it can, in some occasions, preserve edges and other fine detail.



(a) Original image. (b) After applying a Gaussian Filter (c) After applying bilateral filtering.

[3] Although, as we explain in the next section, the concept of range filtering alone makes little sense.

[4] Bilateral filtering uses the concept of closeness instead of distance because it is more compatible with the concept of similarity and yields a simpler formulation.

2 Materials and methods

2.1 Definitions

Images

A grayscale image is a function that maps pixel positions or indexes $\boldsymbol{p} \in \boldsymbol{D}$ to image intensity values $\boldsymbol{S}$:

$$I :: \boldsymbol{D} \mapsto \boldsymbol{S}$$

The domain $\boldsymbol{D}$ is usually defined as $\boldsymbol{D} = \mathbb{R}^d$ for a d dimensional image.

The range $\boldsymbol{S}$ can be defined as $\boldsymbol{S} = \mathbb{R}$ in its most general formulation, o more commonly as $\boldsymbol{S} = [0 \dots 1]$ in the case of **continuous intensities**.

We can also define a **discrete range** such that $\boldsymbol{S} = \{0, 1, \dots, K\}$; a typical value for K is $2^8 - 1 = 255$ to model **8-bit grayscale images**.

The **discrete** version of a grayscale image with intensities $\boldsymbol{S}$ and spatial resolution R_s can be defined by restricting the domain to an integer subset of $\mathbb{R}^d$. Therefore, a discrete image can be described as a matrix $\boldsymbol{I} \in \boldsymbol{S}^{R_s}$.

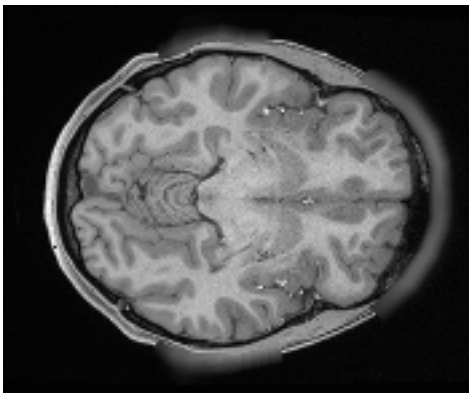
For our purposes, since we are interested in $3D$ grayscale fMRI images, we will mostly assume $d = 3$, so that in the discrete case $R_s = n \times m \times p$, ie, a $\boldsymbol{I}$ is a 3 dimensional array with size $n \times m \times p$ and index set $\boldsymbol{D}_m = \{1, \dots, n\} \times \{1, \dots, m\} \times \{1, \dots, p\}$.

We also define $N = |\boldsymbol{D}_m|$, amount of pixels in the image.

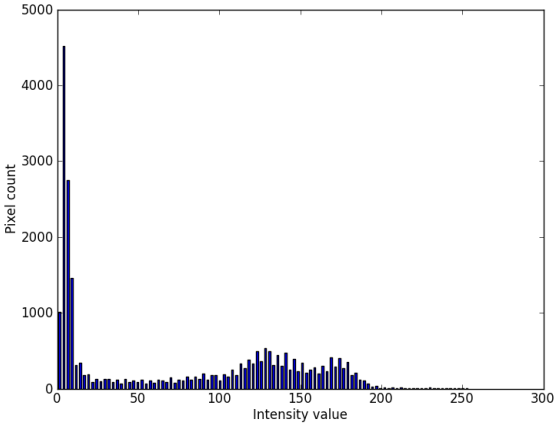
Histograms

The histogram H of a discrete grayscale image $\boldsymbol{I}$ with discrete intensities $\boldsymbol{S} = \{0, 1, 2, \dots, 255\}$ is a function maps intensity values k to the number of pixels with that intensity, that is, the number of pixels $\boldsymbol{p}$ such that $\boldsymbol{I}(\boldsymbol{p}) = k$ [5].

[5] We will employ function notation $A(i)$ to denote the value of index i of matrix A.



(a) 2D slice of a 3D brain MRI image.



(b) Histogram of the 2D slice.

Therefore, $H :: \boldsymbol{S} \mapsto \mathbb{N}$, and it is defined as:

$$H_{\boldsymbol{I}}(k) = \sum_{\boldsymbol{p} \in \boldsymbol{D}_m} \delta(\boldsymbol{I}(\boldsymbol{p}), k)$$

with:

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } otherwise \end{cases}$$

In the case of continuous intensity values $\boldsymbol{S} = [0 \dots 1]$, H can be defined by discretizing $\boldsymbol{S}$ into a finite number h of equally spaced intervals N_i called **bins**, and calculating the number of pixels $\boldsymbol{p}$ with intensities in each bin N_k [6]:

$$H_{\boldsymbol{I}}(k) = \sum_{\boldsymbol{p} \in \boldsymbol{D}_m} \boldsymbol{I}(\boldsymbol{p}) \in N_k$$

where:

$$N_k = [\frac{k}{h}, \dots, \frac{k+1}{h}]$$

And so for the continuous case we have $H :: [0, \dots, h-1] \mapsto \mathbb{N}$.

For simplicity, when presenting Otsu's method we will use the first definition of the histogram. Also, for efficiency reasons the histogram is usually computed as a vector $\boldsymbol{H}$ with a single pass through the image pixels.

[6] Technically, this definition of N_k is inappropriate because there is overlap between intervals, but we leave it as it is for the sake of clarity.

2.2 Gaussian Filtering

The traditional Gaussian filter [7] transforms an image I into an image I' so that:

$$I'(\boldsymbol{p}) = k(\boldsymbol{p}) \int_{\boldsymbol{q} \in \boldsymbol{D}} c(\boldsymbol{p}, \boldsymbol{q}) I(\boldsymbol{q}) d\boldsymbol{q}$$

Where $c(\boldsymbol{x}, \boldsymbol{y}) = e^{\frac{-\|\boldsymbol{p}-\boldsymbol{q}\|}{2\sigma^2}}$ is the closeness function, and $k(\boldsymbol{p}) = k = \frac{1}{2\pi\sigma^2}$ is the scaling factor. The constant σ determines the width of the filter, in terms of the width of the Gaussian bell described.

This formula basically says that the value of the transformed image I' at pixel $\boldsymbol{p}$ is the weighted average of all the pixels of the image I, with c as the weighting function. The scaling factor $k(\boldsymbol{p})$ is chosen so that c integrates to 1, and therefore the *dc* component or average intensity of the image is preserved.

Since we want to implement this transformation in a digital image, the discrete version of the filter is similarly given by:

$$\boldsymbol{I}'(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q} \in R_s} c(\boldsymbol{p}, \boldsymbol{q}) I(\boldsymbol{q})}{S}$$

with:

$$S = \sum_{\boldsymbol{q} \in R_s} c(\boldsymbol{p}, \boldsymbol{q})$$

where $|\cdot|$ denotes set cardinality, and the denominator S preserves the *dc* component by making the weights sum to 1.

The Gaussian function is very close to zero for values away from the center ($\boldsymbol{p}$, in this case) by more than 2σ, so we can save computing time by averaging only within a neighborhood or window around pixel $\boldsymbol{p}$. This yields the formulation:

$$\boldsymbol{I}'(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q} \in \boldsymbol{W}} c(\boldsymbol{p}, \boldsymbol{q}) I(\boldsymbol{q})}{S}$$

with:

$$\boldsymbol{W} = W(\boldsymbol{p}, \sigma, R_s)$$

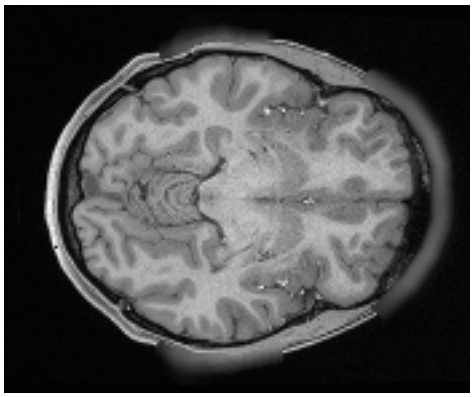
$$S = \sum_{\boldsymbol{q} \in \boldsymbol{W}} c(\boldsymbol{p}, \boldsymbol{q})$$

[7] A gaussian isotropic filter, ie, one that has covariance matrix σI, where I is the identity matrix

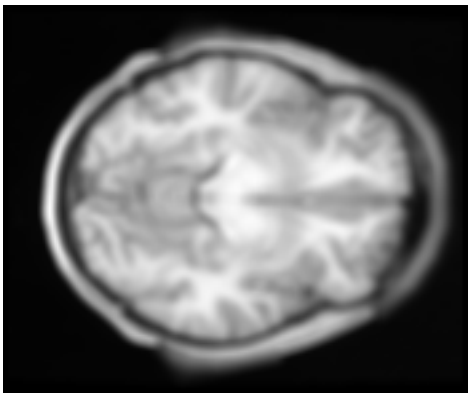
where W is a function that returns a square window of pixels around $\boldsymbol{p}$ such that:

$$W(\boldsymbol{p}, \sigma, R_s) = \{\boldsymbol{q} \in R_s | \, \|\boldsymbol{p} - \boldsymbol{q}\|_\infty \leq ceil(2\sigma)\}$$
where:
$$\|\boldsymbol{x}\|_\infty = \max abs(x_i), \; i = 1, \ldots, d$$

Therefore W selects the pixels $\boldsymbol{q}$ with distance to $\boldsymbol{p}$ (induced by the infinity norm $\|\boldsymbol{x}\|_\infty$) less than 2σ.



(a) Original image.



(b) After applying a gaussian filter

Given that relative to the pixel $\boldsymbol{p}$ whose new value is to be computed, the distances $c(\boldsymbol{p}, \boldsymbol{q})$ are always the same, a matrix G of weights called a **kernel** or **convolution matrix** can be precomputed so that:

$$\boldsymbol{I}'(\boldsymbol{p}) = \sum_{\boldsymbol{q} \in \boldsymbol{W}} G(\boldsymbol{p} - \boldsymbol{q}) I(\boldsymbol{q})$$

In the case of a square 3D image of dimensions $N = M^3$, with a square window of size K, the naive implementation of this formula as an algorithm yields a time complexity of $O(KM^3)$, but there are many schemes that can efficiently calculate or approximate the filtered image [4].

2.3 Bilateral Filtering

Following an analogous argument to domain filtering, a range filtering technique with a Gaussian function s as a similarity function applied without a notion of neighborhood would produce the following:

$$I'(\boldsymbol{p}) = k(\boldsymbol{p}) \int_{\boldsymbol{q} \in \boldsymbol{D}} s(I(\boldsymbol{p}), I(\boldsymbol{q}))I(\boldsymbol{q})d\boldsymbol{q}$$

with $k(\boldsymbol{p}) = \frac{1}{\int_{\boldsymbol{q} \in \boldsymbol{D}} s(I(\boldsymbol{p}), I(\boldsymbol{q}))d\boldsymbol{q}}$ to preserve the *dc* component of the image. The problem with this approach is that range filtering alone makes little sense for denoising and smoothing, since it would just make pixels with similar intensities more similar, regardless of where they are in the image. The concept of derivative and smoothness is an intrinsically local one, and therefore range filtering alone is not useful for smoothing [8].

Bilateral filtering therefore includes both a domain weighting function and a range weighting function, yielding a similar formulation to Gaussian filtering, but with the range-based similarity weighting function s. Given an image I, the filter computes I' according to:

$$I'(\boldsymbol{p}) = k(\boldsymbol{p}) \int_{\boldsymbol{q} \in \boldsymbol{D}} c(\boldsymbol{p}, \boldsymbol{q}) s(I(\boldsymbol{p}), I(\boldsymbol{q}))I(\boldsymbol{q})d\boldsymbol{q}$$

with $k(\boldsymbol{p}) = \frac{1}{\int_{\boldsymbol{q} \in \boldsymbol{D}} c(\boldsymbol{p}, \boldsymbol{q}) s(I(\boldsymbol{p}), I(\boldsymbol{q}))d\boldsymbol{q}}$ to preserve the dc component as before.

In the case of c and s taking the form of a Gaussian similarity function, we have $c = e^{\frac{-abs(s1-s2)}{2\sigma_d^2}}$ as before, and $s(s1, s2) = e^{\frac{-abs(s1-s2)}{2\sigma_r^2}}$, where σ_d and σ_r are the widths of the filter for the domain and the range respectively.

The discrete version of the filter is given by:

$$\begin{aligned} \boldsymbol{I}'(\boldsymbol{p}) &= \frac{\sum_{\boldsymbol{q} \in \boldsymbol{W}} c(\boldsymbol{p}, \boldsymbol{q}) s(I(\boldsymbol{p}), I(\boldsymbol{q}))I(\boldsymbol{q})}{S} \\ \boldsymbol{W} &= W(\boldsymbol{p}, \sigma_d, R_s) \\ S &= \sum_{\boldsymbol{q} \in \boldsymbol{W}} c(\boldsymbol{p}, \boldsymbol{q}) s(I(\boldsymbol{p}), I(\boldsymbol{q})) \end{aligned}$$

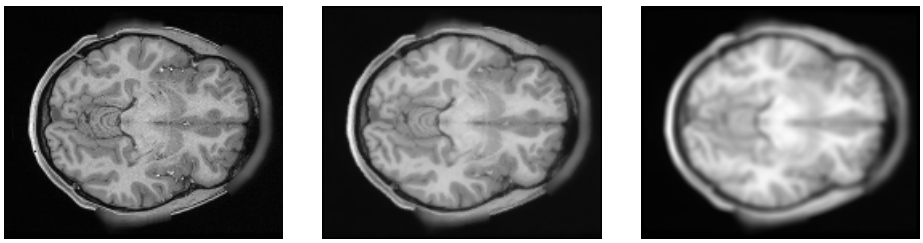
This is the same as Gaussian filtering, except that:

- The range similarity weight, $s(I(\boldsymbol{p}), I(\boldsymbol{q}))$, is included in the sum to compute $\boldsymbol{I}'$
- The size of the window is chosen according to σ_d, since using a window is an approximation technique that can be applied only to the domain filter, because it uses the fact that the filter rapidly decays in strength at distances greater than $2\sigma_d$.

[8] It could, perhaps, be though of as an iteration scheme in a color reduction procedure, since it makes colors more similar.

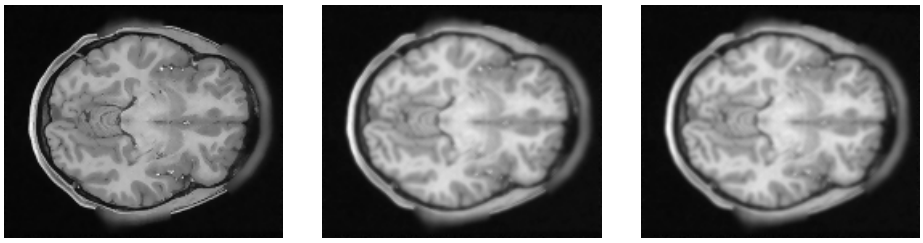
- The normalizing term S now also sums $s(I(\boldsymbol{p}), I(\boldsymbol{q}))$.
- A convolution matrix G cannot be pre-computed because $s(I(\boldsymbol{p}), I(\boldsymbol{q}))$ depends on the actual intensity values of $\boldsymbol{p}$ and $\boldsymbol{q}$, not their distance.

The output of gaussian filtering and bilateral filtering is similar, because the are both performing a smoothing operation, but the bilateral image will typically be less smooth since it preserves some higher frequency information.



(a) Original image. (b) After applying a bilateral filter (c) After applying a gaussian filter

The naive algorithm is of the same order as gaussian filtering, although it is generally slower since a convolution matrix G cannot be precomputed. Also, when $\sigma_r \to \infty$, all distances between pixel intensities become 1, and so in that case bilateral filtering is equivalent to gaussian filtering.



(a) Bilateral filtering with $\sigma_r = 20$, $\sigma_d = 1$. (b) Bilateral filtering with $\sigma_r = 200$, $\sigma_d = 1$. (c) Gaussian filtering, $\sigma_d = 1$.

2.4 Thresholding with Otsu's method

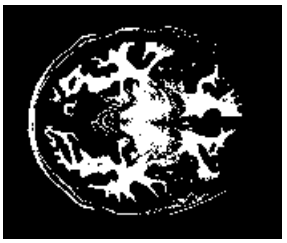
Assuming a discrete set of intensities $\boldsymbol{S} = \{0, 2, \ldots, 255\}$, the simplest thresholding scheme calculates, given a threshold level k and an image I, a thresholded image I' such that:

$$I'(\boldsymbol{p}) = \begin{cases} 255 & \text{if } I(\boldsymbol{p}) > k \\ 0 & \text{otherwise} \end{cases}$$

This formula simply changes every pixel of I to a 0 or 255 depending on whether the pixel is below or above the threshold level. The discrete case is the same, with I replaced by $\boldsymbol{I}$. There are $|\boldsymbol{S}| = 256$ threshold levels to choose from.

(a) MRI image after applying thresholding with $k = 20$. With this value, the cranium is distinguished from the rest of the brain.

(b) MRI image after applying thresholding with $k = 100$. With this value, CSF could be separated from white and gray matter.



(c) MRI image after applying thresholding with $k = 150$. With this value, white matter turns into white pixels, white CSF and gray matter mostly turn into black ones.

The difficult part of an adaptive thresholding scheme is not the application of the threshold, but its determination. Otsu's algorithm is a histogram-based method that approaches the problem of guessing the optimal threshold by defining a measure $M(k)$ of the appropriateness of a threshold k, and performing a brute-force search on the range of the image $\boldsymbol{S}$ to establish the value k^* that maximizes M.

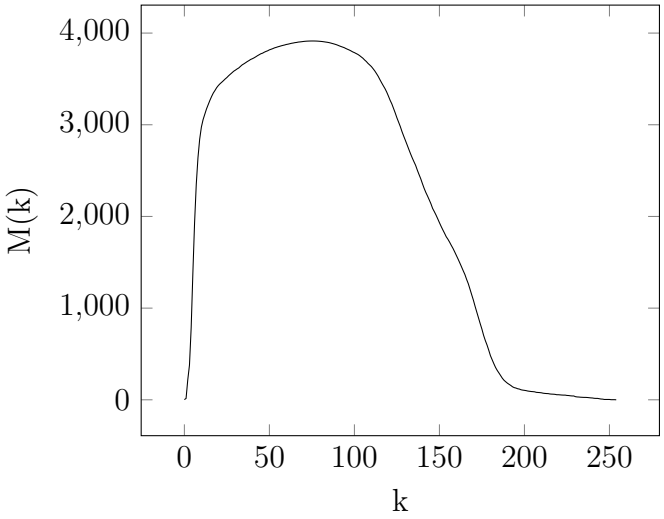


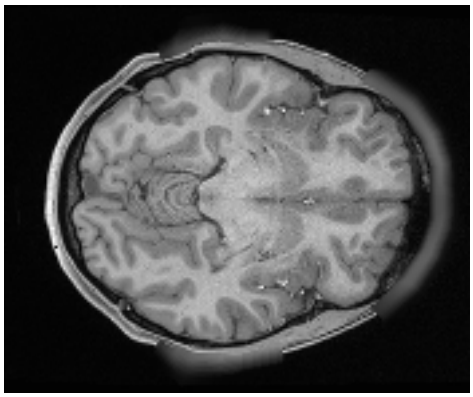
Figure 8: Values of k vs $M(k)$ for the histogram shown before.

We will now formally define the measure M.

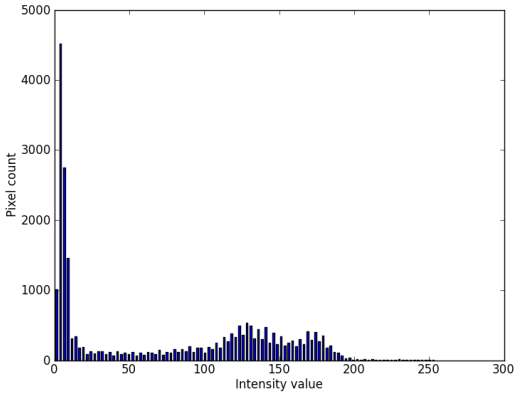
The method computes the histogram H of the image once, and after that only uses the histogram data to determine k^*. For simplicity, we will divide the histogram H by the number of pixels of the image N to obtain a normalized histogram p, such that:

$$p_i = \frac{H(i)}{N}$$

In this way, p_k can be interpreted as the relative frequency of intensity level i, and therefore the histogram can be regarded as a probability distribution.



(a) 2D slice of a 3D brain MRI image.



(b) Normalized histogram of the 2D slice of the image. Values in the y-axis are now frequencies instead of pixel intensity counts.

The measure $M(k)$ is based on considering that the intensities i of each pixel of the image are generated by a probability distribution $P(k)$, of which the normalized histogram given by p_i is our estimator.

Moreover, we consider that there are two sources of grayscale intensity values, one for each class or set C_0 and C_1 (the ones obtained by thresholding of the images based on the value of k). One class can generate the intensity value of more pixels, that is, the probability of a pixel being generated by that class can be greater than for the other.

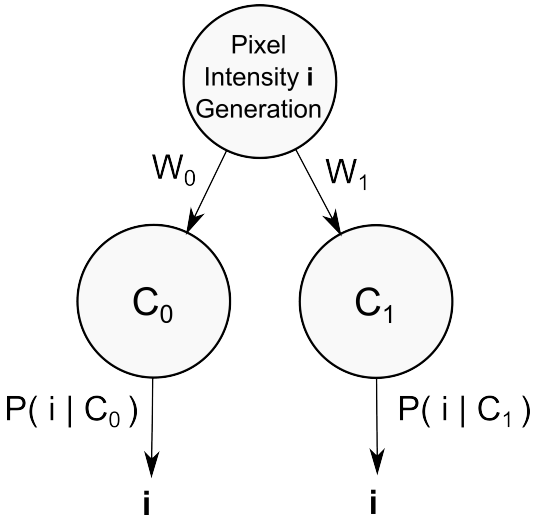


Figure 10: Two probabilistic processes that generate intensity levels of class C_0 and class C_1.

The estimators for the probability of class C_0 and C_1 being chosen to generate the intensity value of a pixel can be calculated in terms of the histogram p_i as:

$$\omega_0 = P(C_0) = \sum_{i=0} p_i$$

$$\omega_1 = P(C_1) = \sum_{i=k+1}^{K} p_i$$

We can now therefore consider the histogram as two different histograms, one for each class:

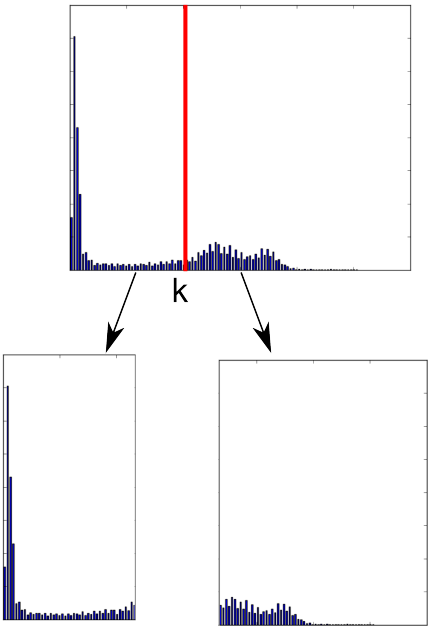


Figure 11: Two histograms, for the pixels of class C_0 and class C_1.

Therefore, for each class we can estimate the conditional probabilities $P(k|C_0)$ and $P(k|C_1)$ as:

$$P(i|C_0) = p_i^0 = \begin{cases} \frac{p_i}{\omega_0} & \text{if } 0 \leq i \leq k \\ 0 & \text{otherwise} \end{cases}$$

$$P(i|C_1) = p_i^1 = \begin{cases} \frac{p_i}{\omega_1} & \text{if } k+1 \leq i \leq K \\ 0 & \text{otherwise} \end{cases}$$

Each process is probabilistic, and so has mean intensity values μ_0 and μ_1, and variances σ_0^2 and σ_1^2. The means and variances can be calculated as:

$$\mu_0 = \sum_{i=0}^{k} p_i^0 i \qquad \sigma_0^2 = \sum_{i=0}^{k} p_i^0 (i-\mu_0)^2$$

$$\mu_1 = \sum_{i=k+1}^{K} p_i^1 i \qquad \sigma_1^2 = \sum_{i=k+1}^{K} p_i^1 (i-\mu_1)^2$$

Given a value of k, we can define M to be:

$$M = \sigma_W^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2$$

The measure σ_W^2 is the weighted sum of the variances of each class, and is called the **within-class variance**. The weights give more importance to classes with more pixels, as the estimation of its variance has better statistical significance. We can easily find the value of k such that maximizes σ_W^2 by brute-force search with $k = 0, \ldots, K$.

As stated before, the hypothesis behind this measure is that the smaller the

Otsu proves that the so called **between-class variance** σ_B^2 is a measure M that is equivalent to σ_W^2 in that minimizing σ_B^2 also maximizes σ_W^2.

We can derive σ_B^2 by thinking of sampling not pixels intensities, but class means μ_0 and μ_1, with ocurrence probabilities ω_0 and ω_1. So μ_0 and μ_1 would be two samples, and therefore we can calculate their mean, which in turn is equivalent to the mean of the image or dc component μ_T given by:

$$\mu_T = \omega_0\mu_0 + \omega_1\mu_1 = \sum_{i=0}^{K} p_i i$$

So, the variance σ_B^2 is naturally given by:

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T) + \omega_1(\mu_1 - \mu_T)$$

Therefore, another scheme of finding the same optimal value of k is to iterate through the values of k to find the value k^* that minimizes σ_B^2.

This approach is also a brute-force search, but can be more efficient since it saves us from calculating σ_0^2 and σ_1^2 in each iteration. Also, since $\mu_1 = \mu_T - \mu_0$, and μ_T does not change with k, we only need to re-calculate μ_0 in each iteration, which does not require a full evaluation of the sum since it is just a matter of summing one floating point number. The implementation details of this scheme can be gleaned from the code attached.

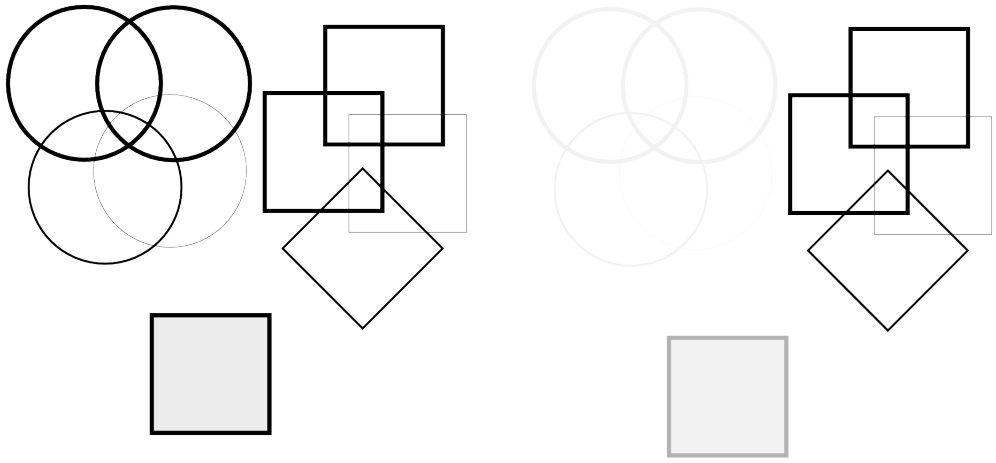
2.5 Dataset and testing methods

We will employ the same dataset to test Otsu's method and both types of filtering.

The dataset consists of both a synthetic image set and an fMRI image set. The first's purpose is to assess the correctness of the implemented algorithms, and glean some basic insights about their advantages and disadvantages. The second is to test them in a real-world setting, particularly in MRI denoising and gray/white matter segmentation.

First dataset:

The first dataset consists of the following synthetic 2D images, created manually using Inkscape, a vector-based image editor, for the *borders* images, and the numpy package for the *gradient* images.

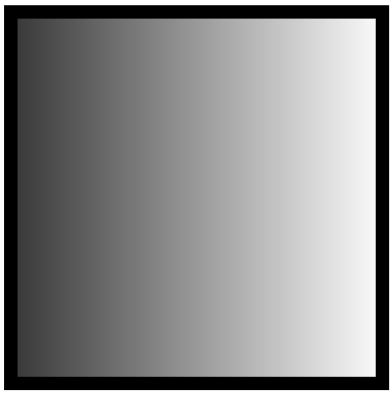


(a) A synthetic image with various types of borders.

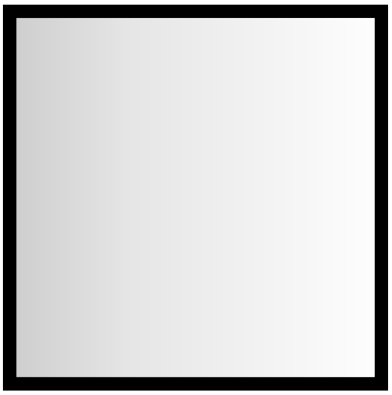
(b) A synthetic image with various types of borders and various types of contrast

For Otsu's methd, the first image serves as a basic test to verify that the algorithm correctly determines a value for k in a trivial case. Since bilateral reportedly preserves object boundaries and lines, we can test this assumption on the various types of borders that appear in the image, and measure the difference with gaussian filtering.

Image b is the same image a, but some objects in b have very little contrast, ie, the intensity of the foreground has very little absolute difference with that of the foreground. In this way, we can explore the effects of varying the intensity levels on both algorithms.



(a) A synthetic image with a central area filled with a dark horizontal gradient



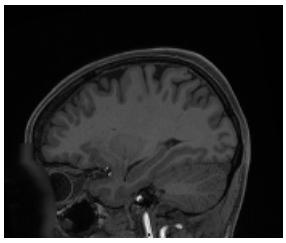
(b) A synthetic image with a central area filled with a light horizontal gradient

This two images include gradients of different strength, to evaluate (a) the threshold determined by Otsu's algorithm when varying the darkness of the gradient, (b) how colors around the borders of an object affect bilateral filtering, and whether it reduces to gaussian filtering in some cases, if it generates color patches, etc.

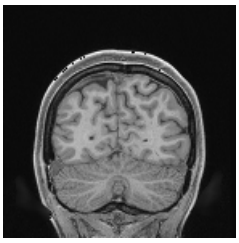
All images have a resolution of 300x300.

Second dataset:

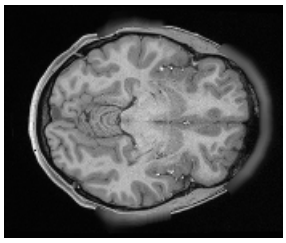
The second dataset consists of a single real-life MRI 3D image of a brain, of resolution $145 \times 174 \times 145$.



(a) A sagittal 2D slice of the 3D MRI image.



(b) A coronal 2D slice of the 3D MRI image.



(c) An axial 2D slice of the 3D MRI image.

The images show three orthogonal slices of the 3D MRI; the algorithms will be run on the 3D image and 2D slices will be generated for visual inspection.

Testing method

We will first run the algorithms on this images with several noise intensities to see how Otsu's method reacts to it and whether the filtering techniques are effective at denoising. Some samples images will be produced to be able to visually inspect and analyze the output of the algorithms.

Remembering our basic noise equation from the introduction, $v(i) = u(i) + n(i)$, where v was the observed image, u the true image and n the noise, given our dataset consists of original images $u(i)$, we will add a known noise $n(i)$ to each image to produce a supposedly observed v. Then, we will apply three different testing schemes:

1. To test the filtering algorithms, we will apply the filter to v, to produce a restored image r. Because we know the original image u, we will then compare r and u to see how well the filter removed the noise from the image.

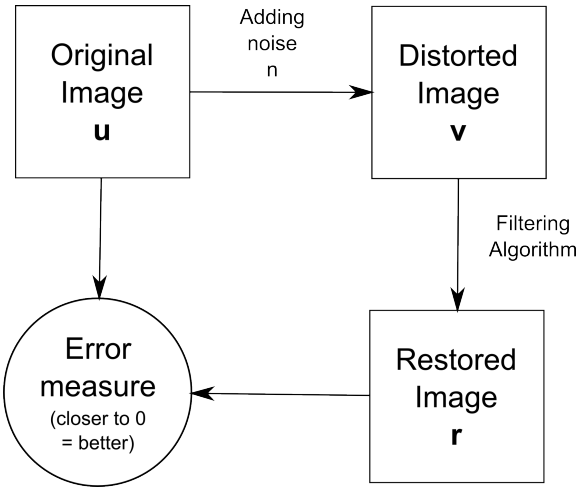


Figure 15: Testing sequence for filtering algorithms.

2. To test Otsu's thresholding, we will threshold both v and u and compare their differences, and so test the resilience of the method in the presence of noise.

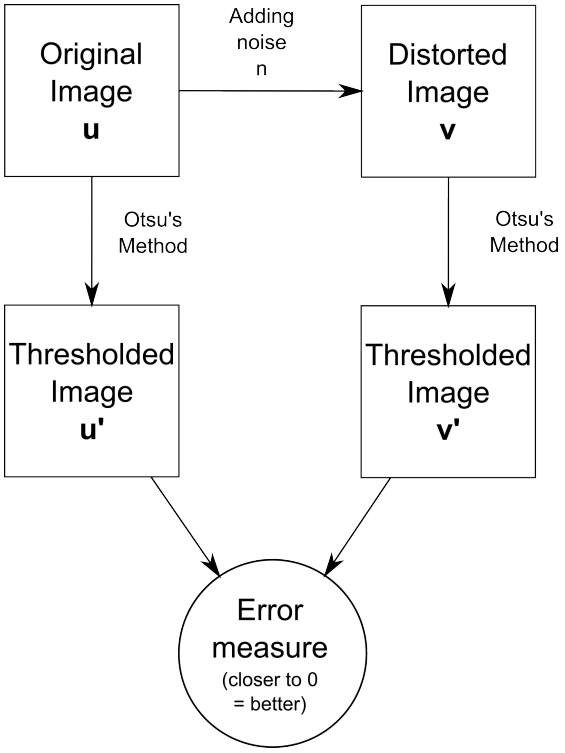


Figure 16: Testing sequence for Otsu's method.

3. To test Otsu's thresholding, we will also apply a denoising filter to v to produce a restored image r, and then threshold r and u, and compare their differences as before.

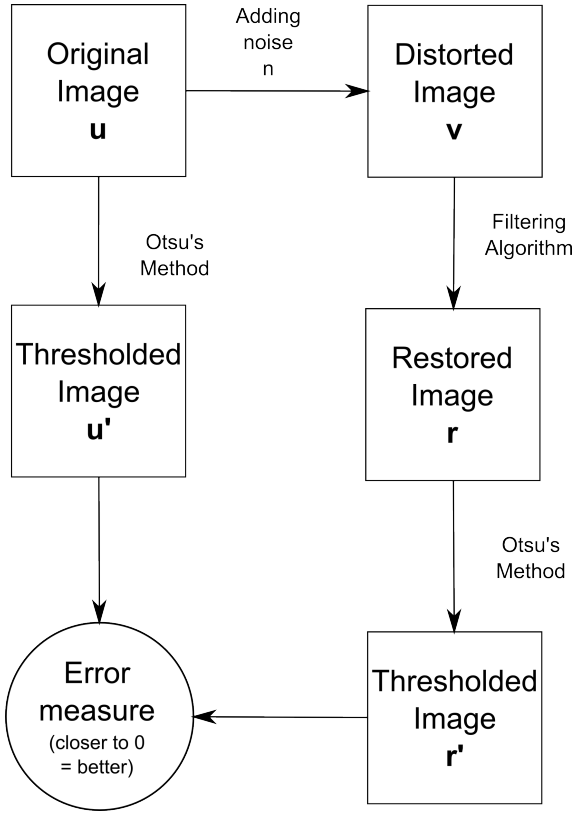


Figure 17: Testing sequence for Otsu's method with a filtering algorithm.

To gauge the difference between images $\boldsymbol{I}$ and $\boldsymbol{J}$, we will employ an objective measure of image similarity. The **similarity measure** is:

$$Sim(\boldsymbol{I}, \boldsymbol{J}) = \frac{1}{N} \sum_{i=1}^{N} (I(\boldsymbol{p}_i) - J(\boldsymbol{p}_i))^2$$

which is related to the **squared frobenius norm** $\|\boldsymbol{I} - \boldsymbol{J}\|_2$.

In the case of the filtering algorithms, it is also useful to evaluate the smoothness of the denoised image. In general, we want smoother results, since a common assumption is that 'normal' images have few drastic changes in intensity. For that, we will use the **regularity** measure of an image:

$$Reg(\boldsymbol{J}) = \frac{1}{N} \sum_{i=1}^{N} |\nabla J(\boldsymbol{p}_i)|$$

Combining this two measures, we get the energy functional $E(\boldsymbol{I}, \boldsymbol{J})$ of the transformation from $\boldsymbol{I}$ to $\boldsymbol{J}$:

$$E(\boldsymbol{I}, \boldsymbol{J}) = Sim(\boldsymbol{I}, \boldsymbol{J}) + Reg(\boldsymbol{J})$$

The similarity term measures the difference between the two images, and the regularization term penalizes images with a high degree of non-smoothness. This is the error measure we refer to in the testing diagrams above [9].

This formulas will be used as objective measures of the goodness of a filtering operation and thresholding operation.

All tests with bilateral and gaussian filtering are repeated varying the value of the corresponding σ that specifies the width of the applied kernels.

3 Results

In this section, we present plots of the similarity and regularity measures using the testing procedures presented in the previous section. This plots show the change of similarity and regularity as a function of the parameters of each method and the artificial noise levels.

For each of the methods or combination thereof, we also present sample images obtained in the tests for illustration and analysis.

3.1 Sample images

In the following, for each sample image, the first three images of the figure consist of the original image, and the results of applying bilateral and gaussian filters. The second row consists of the original image with added noise, and the results of applying both filters, again, but to the noisy image. The last two rows consist of the result of Otsu's method on the original, on the noisy version of the original, and on the versions restored by bilateral and gaussian filtering respectively.

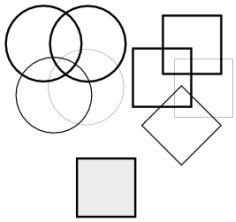
The gaussian and bilateral filter applied to the **original** image both had $\sigma_d = 1$, and the bilateral filter's σ_r was equal to 7. Otsu's method was applied with a histogram of 256 bins.

[9] Technically, because of the regularization term Reg, the value $E(\boldsymbol{I}, \boldsymbol{J})$ is only 0 when both $\boldsymbol{I}$ and $\boldsymbol{J}$ are images with the *same constant intensity* value in every pixel, ie, both images are the same and have exactly the same color in all their pixels. Therefore, the energy function is not truly an error measure, and that's why the fuzzier term *energy function* is used.

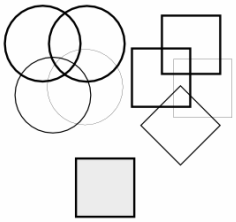
The noisy version of the synthetic images was generated with white gaussian noise, $\sigma = 1.5$.

The gaussian and bilateral filter applied to the **noisy** image both had $\sigma_d = 1.5$, and the bilateral filter's σ_r was equal to 7. Otsu's method was applied with a histogram of 256 bins.

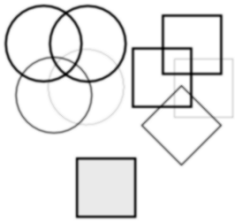
3.1.1 Synthetic images



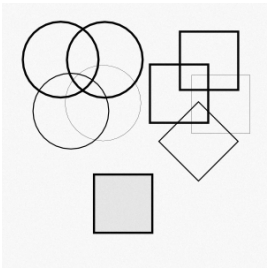
(a) Original



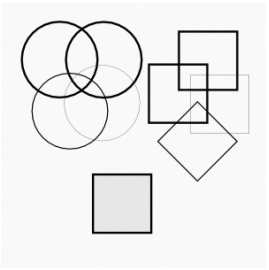
(b) Bilateral



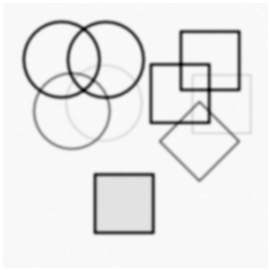
(c) Gaussian



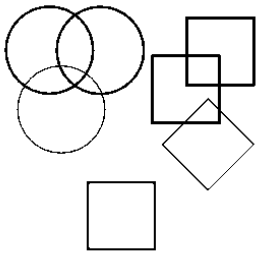
(d) Original with noise



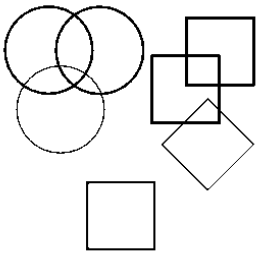
(e) Bilateral with noise



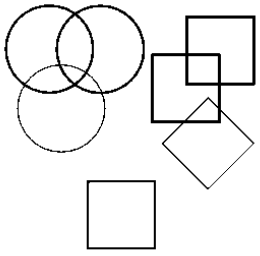
(f) Gaussian with noise



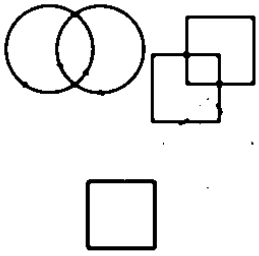
(g) Otsu's algorithm on original image



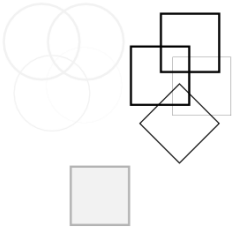
(h) Otsu's algorithm on noisy image



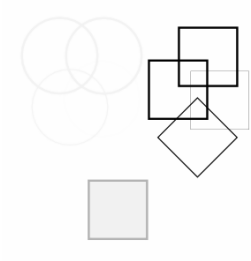
(i) Otsu's algorithm after applying a bilateral filter on the noisy image



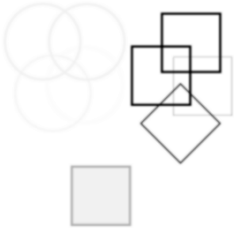
(j) Otsu's algorithm after applying a gaussian filter on the noisy image



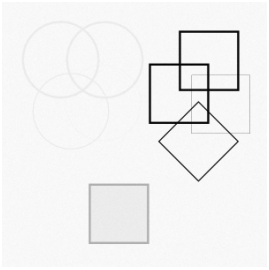
(a) Original



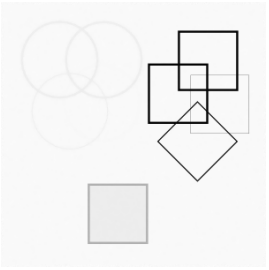
(b) Bilateral



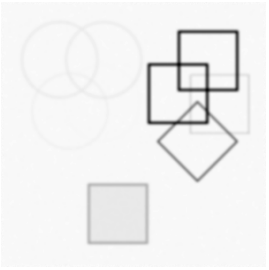
(c) Gaussian



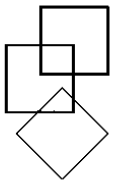
(d) Original with noise



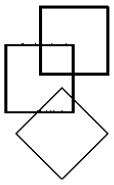
(e) Bilateral with noise



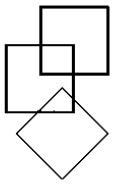
(f) Gaussian with noise



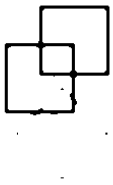
(g) Otsu's algorithm on original image



(h) Otsu's algorithm on noisy image



(i) Otsu's algorithm after applying a bilateral filter on the noisy image

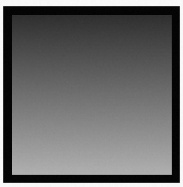


(j) Otsu's algorithm after applying a gaussian filter on the noisy image

(a) Original

(b) Bilateral

(c) Gaussian

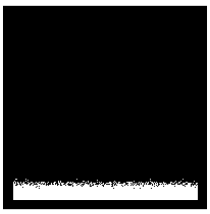


(d) Original with noise

(e) Bilateral with noise

(f) Gaussian with noise

(g) Otsu's algorithm on original image



(h) Otsu's algorithm on noisy image

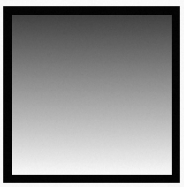
(i) Otsu's algorithm after applying a bilateral filter on the noisy image

(j) Otsu's algorithm after applying a gaussian filter on the noisy image

(a) Original

(b) Bilateral

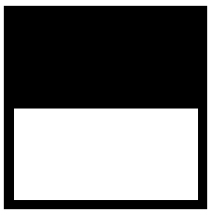
(c) Gaussian



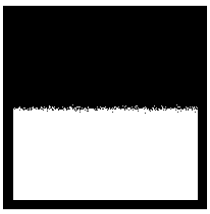
(d) Original with noise

(e) Bilateral with noise

(f) Gaussian with noise



(g) Otsu's algorithm on original image

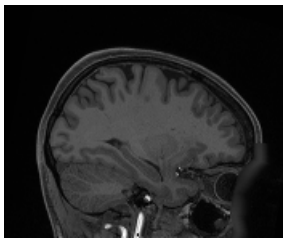


(h) Otsu's algorithm on noisy image

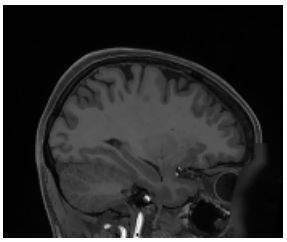
(i) Otsu's algorithm after applying a bilateral filter on the noisy image

(j) Otsu's algorithm after applying a gaussian filter on the noisy image

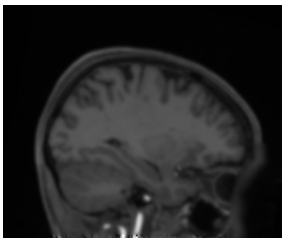
3.1.2 MRI images



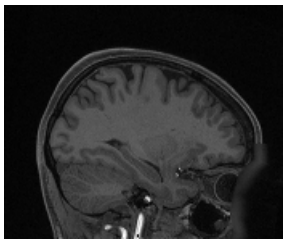
(a) Original



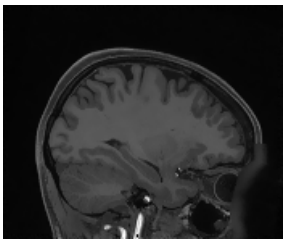
(b) Bilateral



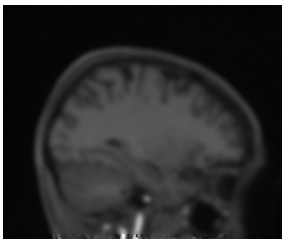
(c) Gaussian



(d) Original with noise



(e) Bilateral with noise



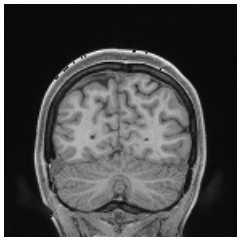
(f) Gaussian with noise

(g) Otsu's algorithm on original image

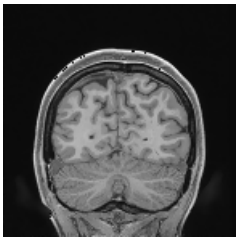
(h) Otsu's algorithm on noisy image

(i) Otsu's algorithm after applying a bilateral filter on the noisy image

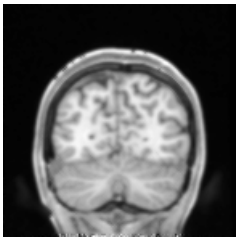
(j) Otsu's algorithm after applying a gaussian filter on the noisy image



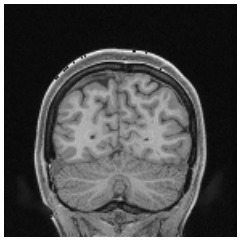
(a) Original



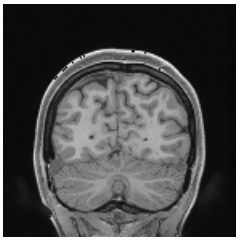
(b) Bilateral



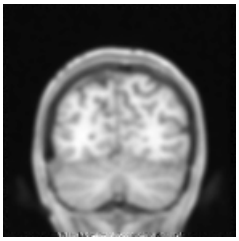
(c) Gaussian



(d) Original with noise



(e) Bilateral with noise



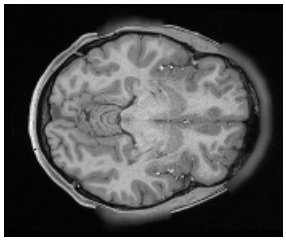
(f) Gaussian with noise

(g) Otsu's algorithm on original image

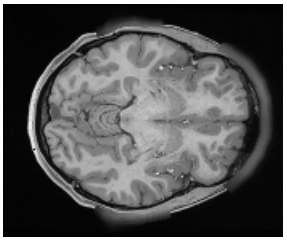
(h) Otsu's algorithm on noisy image

(i) Otsu's algorithm after applying a bilateral filter on the noisy image

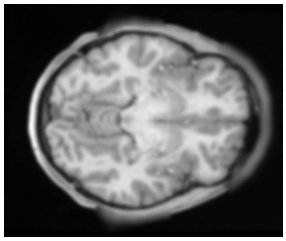
(j) Otsu's algorithm after applying a gaussian filter on the noisy image



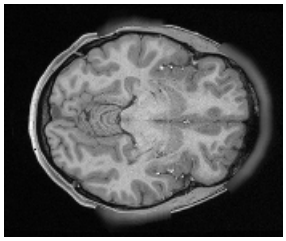
(a) Original



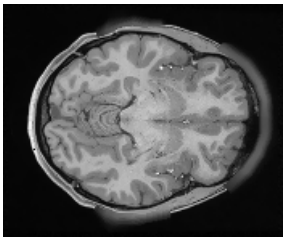
(b) Bilateral



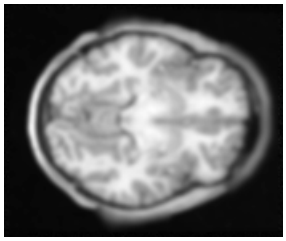
(c) Gaussian



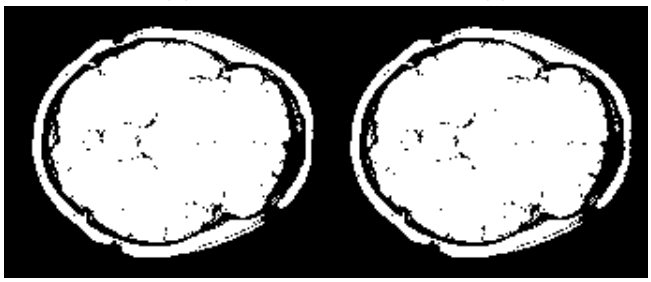
(d) Original with noise



(e) Bilateral with noise

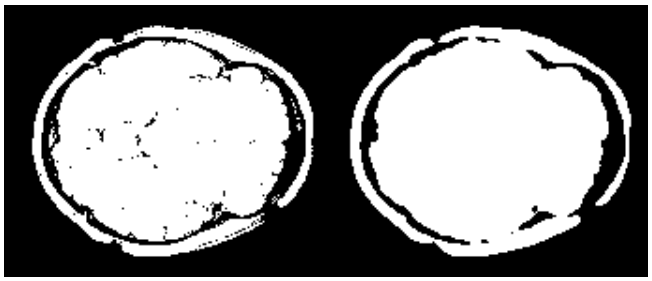


(f) Gaussian with noise



(g) Otsu's algorithm on original image

(h) Otsu's algorithm on noisy image



(i) Otsu's algorithm after applying a bilateral filter on the noisy image

(j) Otsu's algorithm after applying a gaussian filter on the noisy image

3.2 Similarity and Regularity Plots

Here we show plots of the similarity and regularity of the transformations. For each image, we show plots of both Sim and Reg for both types of filtering,

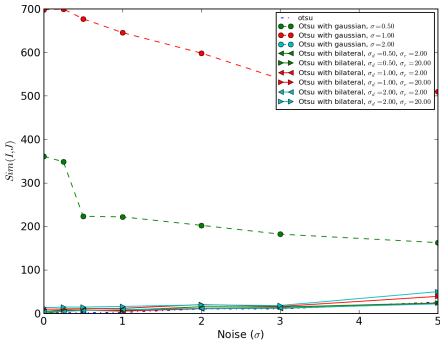
and for Otsu's algorithm, with and without noise and denoising with both types of filters.

Each plot is obtained by varying the noise of the image through values $\sigma \in \{0, 0.25, 0.5, 1, 2, 3, 5\}$. For gaussian filtering, we show results for $\sigma \in \{0.5, 1, 2\}$. For bilateral filtering, we tried the same σ_d as those σ employed for gaussian filtering, and we also varied $\sigma_r \in \{2, 20\}$.

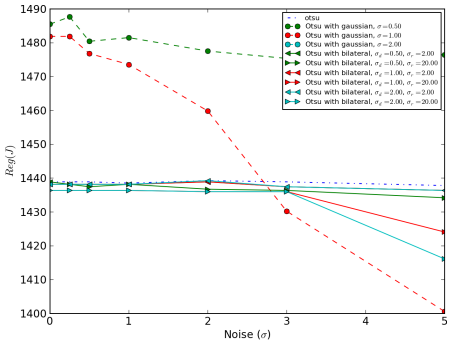
In the plots for Otsu's method, the dashes-and-points blue line shows the Sim and Reg between the application of Otsu's to the original image, and the application of Otsu's to the noisy image without any denoising filter.

In the plots for Otsu's method and synthetic images, we have excluded the noise value $\sigma = 2$ for gaussian filtering because it gave very extreme values of Sig and Reg and hence distorted the plots.

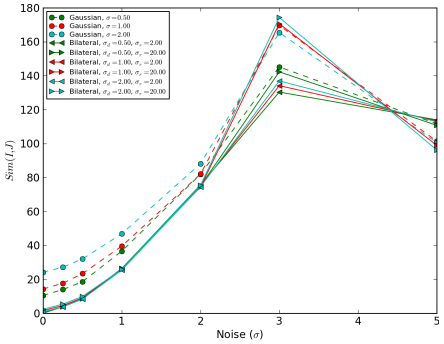
3.2.1 Synthetic images plots



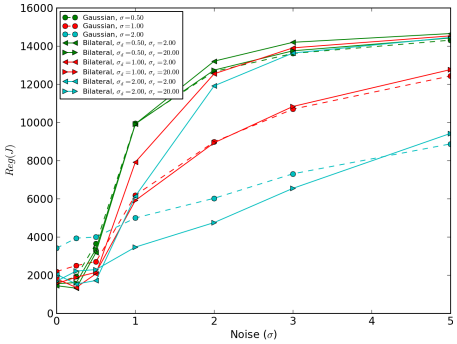
(a) Similarity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



(b) Regularity for thresholding as a function of noise, with bilateral, gaussian and no filtering.

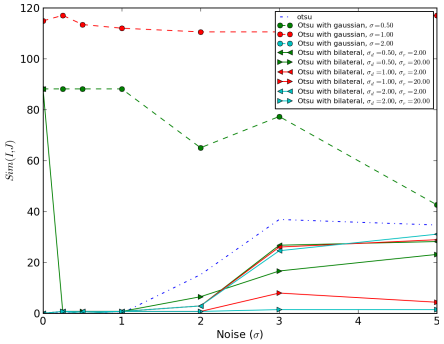


(c) Similarity for denoising with bilateral or gaussian filters, as a function of noise.

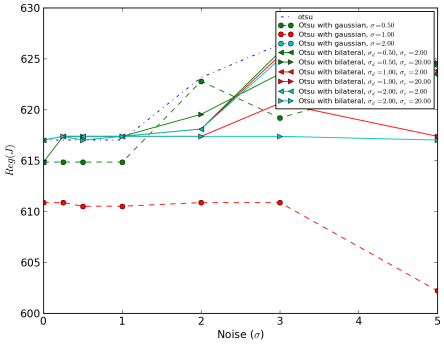


(d) Regularity for denoising with bilateral or gaussian filters, as a function of noise.

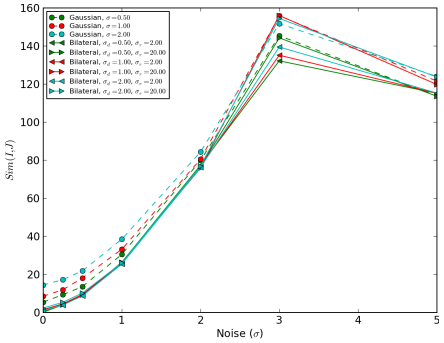
Figure 25: Results for file `borders`.



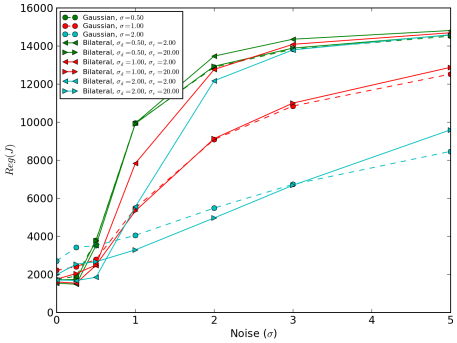
(a) Similarity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



(b) Regularity for thresholding as a function of noise, with bilateral, gaussian and no filtering.

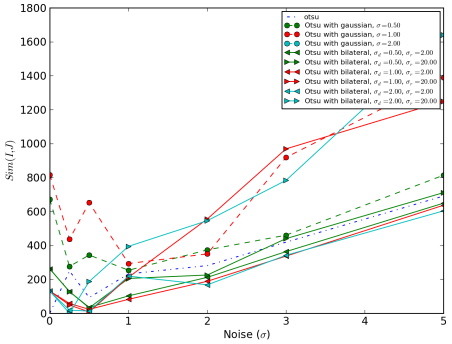


(c) Similarity for denoising with bilateral or gaussian filters, as a function of noise.

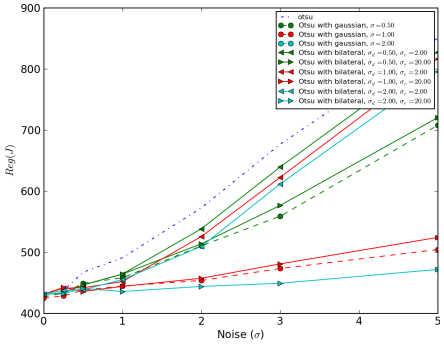


(d) Regularity for denoising with bilateral or gaussian filters, as a function of noise.

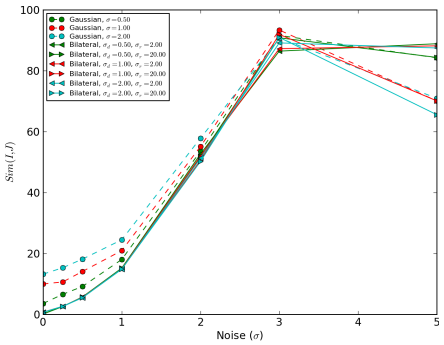
Figure 26: Results for file `borders_contrast`.



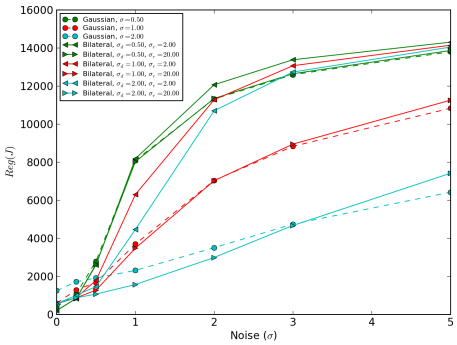
(a) Similarity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



(b) Regularity for thresholding as a function of noise, with bilateral, gaussian and no filtering.

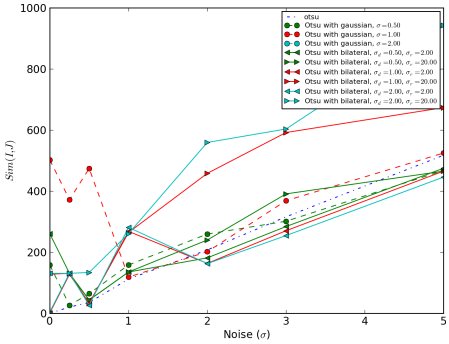


(c) Similarity for denoising with bilateral or gaussian filters, as a function of noise.

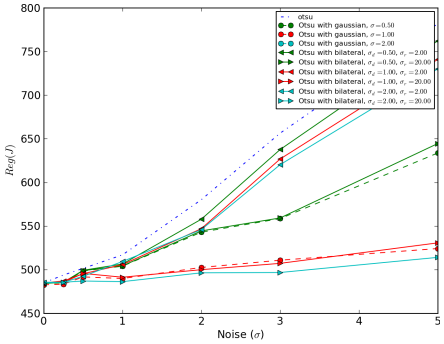


(d) Regularity for denoising with bilateral or gaussian filters, as a function of noise.

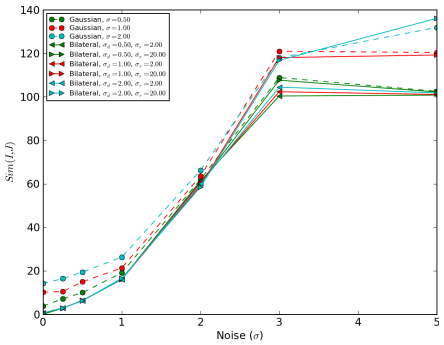
Figure 27: Results for file `gradient1`.



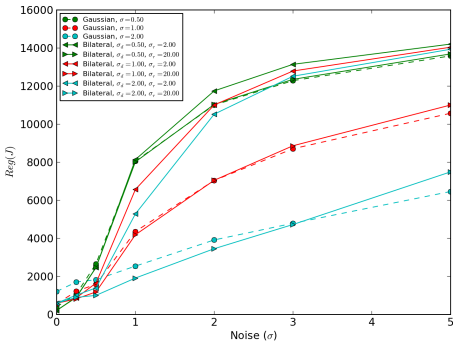
(a) Similarity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



(b) Regularity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



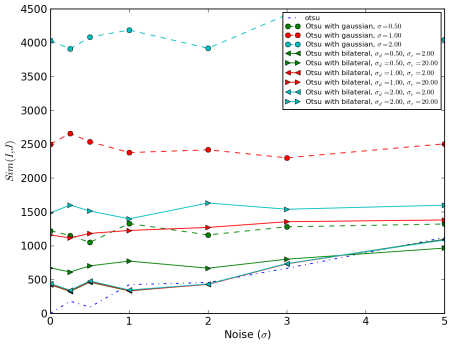
(c) Similarity for denoising with bilateral or gaussian filters, as a function of noise.



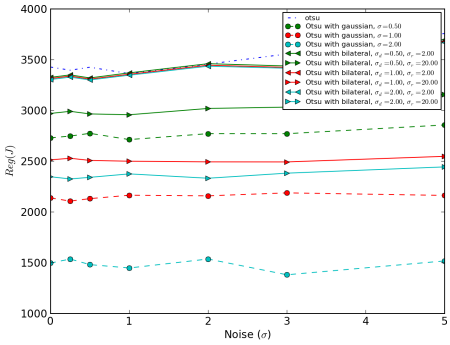
(d) Regularity for denoising with bilateral or gaussian filters, as a function of noise.

Figure 28: Results for file `gradient2`.

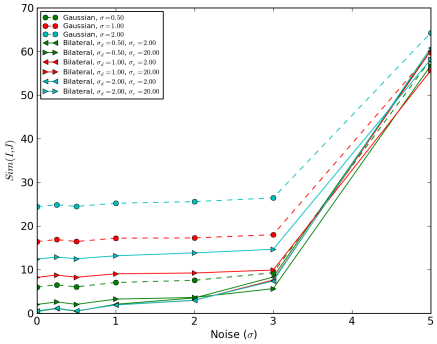
3.2.2 MRI image plot



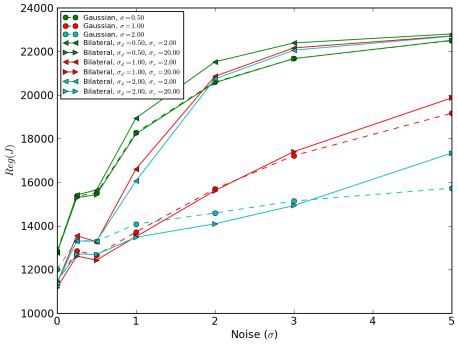
(a) Similarity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



(b) Regularity for thresholding as a function of noise, with bilateral, gaussian and no filtering.



(c) Similarity for denoising with bilateral or gaussian filters, as a function of noise.



(d) Regularity for denoising with bilateral or gaussian filters, as a function of noise.

Figure 29: Results for file `mri`.

4 Discussion

4.1 Sample images

In all sample images, the same story can be seen: bilateral filtering produces little or no image deformation, while the gaussian filter produces noticeable blurring, at the same σ level [10]. It preserves edges better, even and specially

[10] Lower σ levels for gaussian filtering also produced blurring in our experiments, but still it is questionable if the algorithms should be compared at the same σ level, because bilateral filtering will always smooth the image less (except for a constant intensity image, for which bilateral filtering is the same as gaussian filtering)

in the presence of noise. Bilateral filtering can preserve edges even in areas of little contrast, as can be seen in the *gradient* images. Nonetheless, in the gradient areas, both filtering algorithms perform in more or less the same way.

As can be seen in the *borders* and *borders_contrast* synthetic images, bilateral filtering also avoids generating a slight circular splotch where lines cross. This shows the lack of selectiveness in gaussian filtering, which produces effects perceived as color smears.

For the MRI images, the difference in performance between gaussian and bilateral is greater than for the synthetic images, since the images have much more fine detail. Gaussian filtering is more sensitive to noise in this case, making the filtered image almost unusable.

In turn, bilateral filtering can be readily seen to smooth the gray matter, white matter and CSF areas well, without crossing intensity boundaries, so that the resulting images could be more easily processed using other methods after the filter.

In the *borders* synthetic image, Otsu's algorithm produced reasonable results, leaving out the circles and squares with less contrast out of the resulting image, as well as the background of the bottom square. In the *borders_contrast* image, only the blackest figures remained, probably because the intensity peaks of the gray figures was probably close to the peak of white color from the background, and hence the threshold was set lower than the gray intensity.

When noise was added, Otsu carried over some artifacts which where negated by bilateral filtering. Gaussian filtering instead added some of its own, as when some of the splotches previously mentioned turned into black pixels that should not be there because the smoothing increased the mean intensity level of the area.

In the *gradient* images, adding noise caused the limit between the black and white regions inside the square to become fuzzy. This effect was mostly negated by both types of filtering, although the greater strength of gaussian smoothing throws the peaks of the histogram closer to the white color, and hence there is a greater black area than in the other cases. Gaussian filtering also introduces splotches in spots alongside the left and right borders of the inside of the square, probably because a sufficient mass of intensity randomly accumulated.

In the case of MRI, Otsu's method determined a threshold that separated CSF from gray and white matter, but did not distinguish this last two. It's likely that further thresholding with Otsu's on the gray and white matter areas could yield a successful separation. Given that as shown before in section 2.4, an MRI thresholded image with a threshold of 150 did separate

white matter from the rest of the image, this is indeed possible.

Bilateral filtering was again able to negate the effects of noise on the images, making the restored and thresholded image almost the same as the thresholded original. The smoothing strength of Gaussian filtering made it lose almost all fine detail after thresholding, which may be useful when trying to separate the cranium from the rest of the brain, but loses all distinction between CSF and gray/white matter inside the brain.

4.2 Similarity and Regularity Plots

In general, all plots show similar results:

- The mean magnitude of the *Reg* function is much bigger in general than the mean magnitude of the *Sim* function, and so is generally the predominant term in the E function; this is the reason why we did not directly plot the latter function, but preferred to show *Reg* and *Sim* scores separately.
- For the filtering algorithm's tests, *Sim* and *Reg* scores correlate positively with noise, for the thresholding tests the correlation is very low.
- For most tests, *Reg* scores correlate positively with σ_d and σ_r, as should be expected.
- Images processed with gaussian filtering become smoother (lower *Reg* scores) than those treated with the bilateral method, although the latter applied with $\sigma_r = 20$ shows behavior similar to gaussian filtering.
- **Images smoothed with bilateral filtering generally have lower** *Sim* **scores than those treated with gaussian filtering**, and so should tend to perform better at denoising.
- When very little noise has been added to the image, ($\sigma \in \{0.25, 0.5\}$) applying Otsu's algorithm directly on the noisy image can yield better *Sim* scores than those obtained by previously filtering, probably because the filter distorts the image more than the amount of noise removed.
- The *Reg* score for Otsu's without filtering is always the same or more than that of the method with worst score.

The *Reg* and *Sim* plots for the thresholding of the images *gradient1* and *gradient2* should probably be disregarded since the determination of the threshold for this images becomes somewhat arbitrary and the values do not reflect any clear trend.

4.3 Conclusion

We can conclude that bilateral filtering is indeed a better denoising method in most cases that gaussian filtering, as demonstrated by the result images and the similarity plots. In all cases, visual inspection of the resulting image readily shows that the bilateral filter is more appropriate than the gaussian for denoising, as it distorts the image less across object boundaries and fine detail.

Otsu's method seems to provide appropriate threshold values for the synthetic images. For MRI segmentation in general it only discriminated between CSF (black pixels), and both the white/gray matter and the cranium (white pixels), so to separate white matter from gray further experiments should be conducted.

In the presence of non-trivial noise, applying a denoising filter can help Otsu's thresholding scheme, but it can also introduce unwanted additional defects if the filter is too strong.

Finally, it is hard to make sense of what should be the values of *Reg* without any reference to the original image's *Reg* score. Lower *Reg* values are not sufficient evidence of a better transformation, as can be confirmed by visual inspection of the gaussian and bilateral filtered images.

Therefore, the E measure does not correlate well with what we see in the result images. The *Reg* term overpowers the *Sim* term and so would yield a lower E value for gaussian filtering that for bilateral, when, as said before, it can be readily seen that bilateral filtering performs better in terms of denoising.

References

[1] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[2] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[3] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[4] P. Kovesi. Fast almost-gaussian filtering. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 121–125, 2010.