Combined Report for all Practicals

M32 Signal and image processing

Benjamin AKERLUND benker-3@student.ltu.se
Louis BARBOUTIE loubar-3@student.ltu.se

M2 TSI - Space Technology and Instrumentation
Faculty of Science and Engineering (*Faculté sciences et ingénierie*)
Université de Toulouse
February 24, 2025

Preface

This report is a submission for the *M32 Signal and Image Processing* course at Université de Toulouse during the academic year 2024/2025. This submission contains all practical assignments given in the four parts of the graded assignments for this course, i.e., *BE1_IntroComputerVision*, *BE2_IntroMorphoMath* and BE3_ShapeDetection and *classif*.

This report is a joint effort of the students mentioned on the title page and is submitted as a group project, as instructed by the lecturer. The submission is made by email and contains only this report. The full list of project files and source code can be found in the following GitHub repository: https://github.com/benjaminakerlund/ut3-Image_Processing.

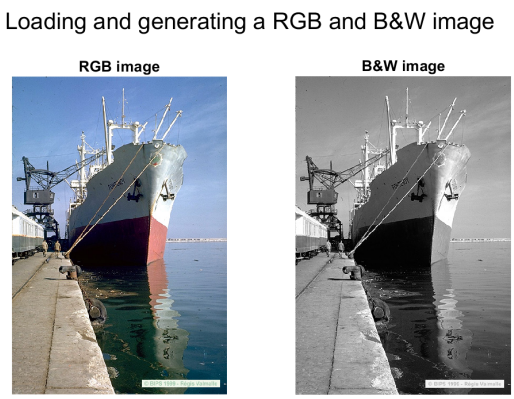
Contents

1 Introduction to Computer vision

Exercises for *BE1_IntroComputerVision* practical.

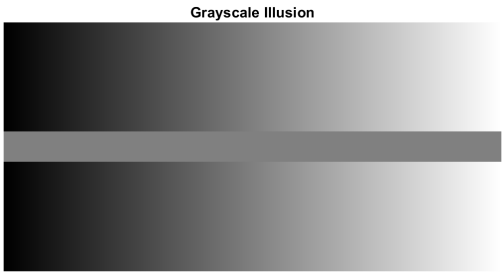
1.1 Basics

Question 1: *Load and Display a grayscale image and a color image. How do you interpret the image coding under MatLab? What is the data type?*

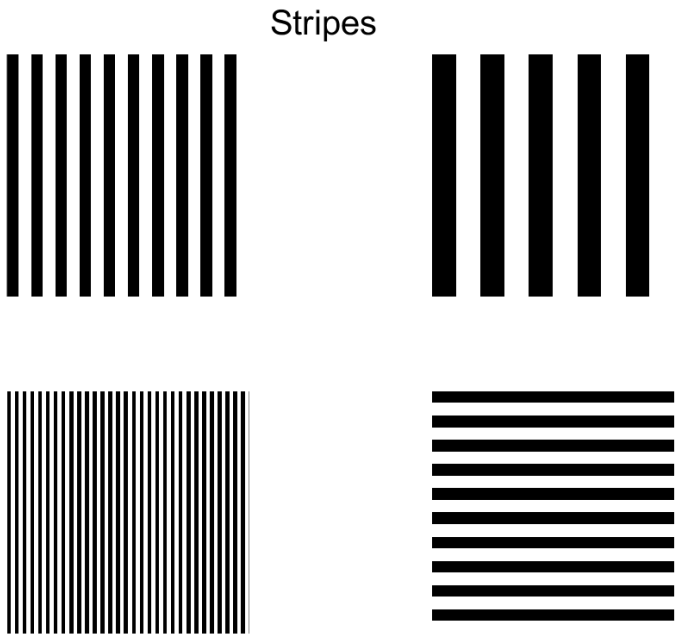


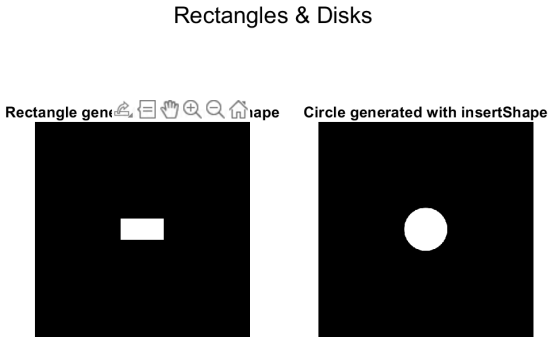
1.1.1 Greyscale Image Coding

Question 2: *Build a matrix with a gradual value of intensity and an horizontal line with a constant value; the representative image is shown Fig. 1.*



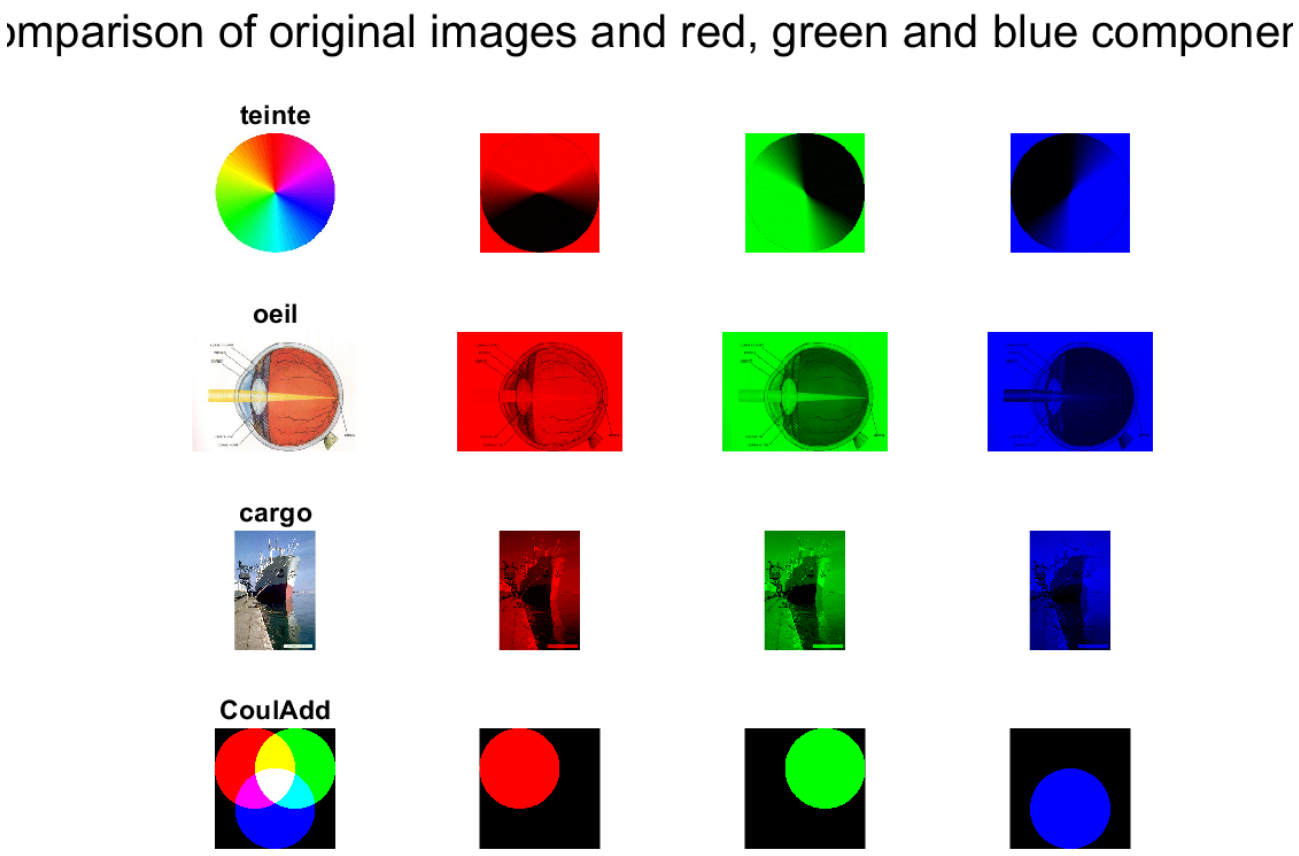
Question 3: *Build a matrix of black & white stripes, and with a rectangle and disk as shown Fig. 2. All parameters (size of stripes, size of rectangle, radius) should be easily modifiable.*



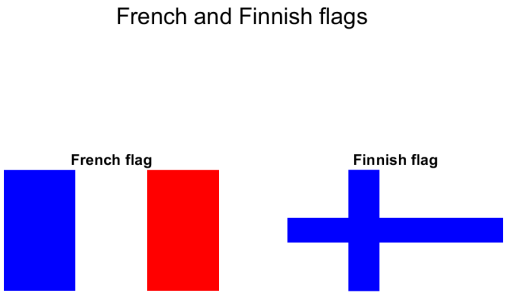


1.1.2 Colour Image Coding

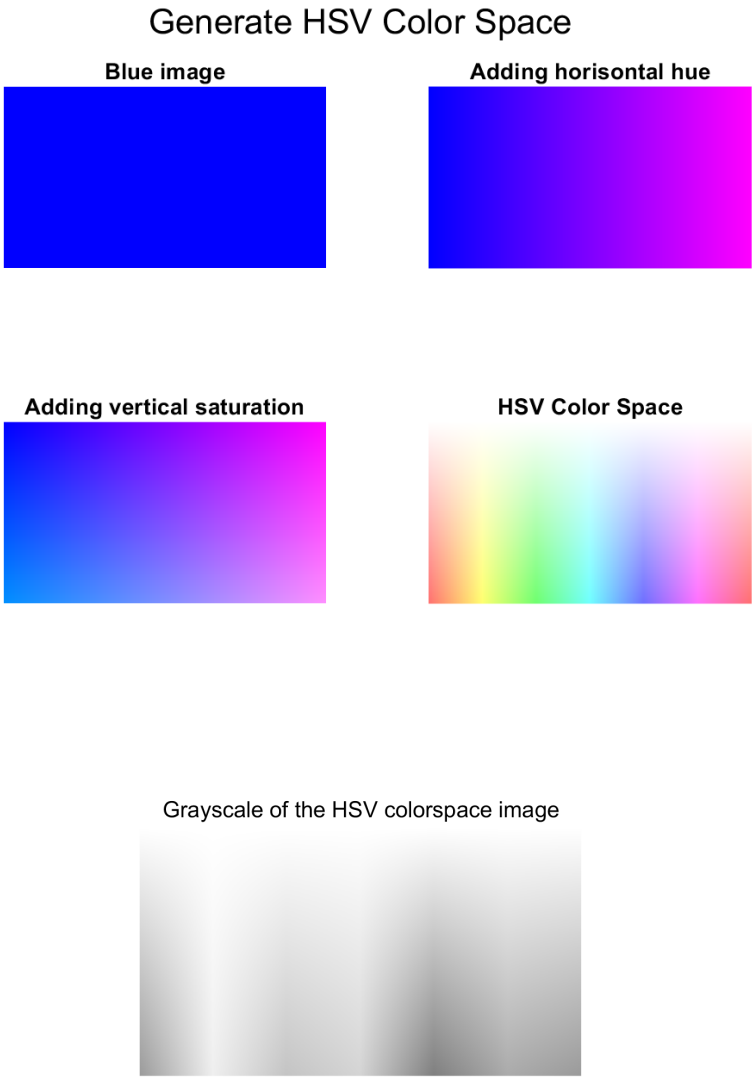
Question 4: *Next, display Teinte.jpg and its red, green and blue components. Interpret and analyse. Same with oeil.jpg, cargo.jpg and CoulAdd.jpg.*



Question 5: *Build and display the french flag. Build and display your flag.*



Question 6: *Use the HSV code (with the command rgb2hsv and interpret images. How is the type of the new matrix? Build and display the image Fig. 4.*



Question 7: *What are the values of α, β and γ?*

When converting an RGB image to grayscale, the channels get summed according to the weights given by α, β and γ:

$$I = \alpha \cdot r + \beta \cdot g + \gamma \cdot b \tag{1}$$

Their values are

$$\alpha = 0.299$$
$$\beta = 0.587$$
$$\gamma = 0.114$$

to best reflect the way the human vision works, which is much more sensitive to green and red than to blue.

Question 8: *Load and display SpainBeach.png and isolate the beach.*

We can do some primitive feature extraction based on color ranges. While it is difficult to so in RGB as the colors are highly correlated, it is easier to do so in HSV space. We select beach-like colors by selecting a range of hue values, which leads to the result in Figure 1.

(a) Original (b) Filtered beach

Figure 1: Hue color filtering

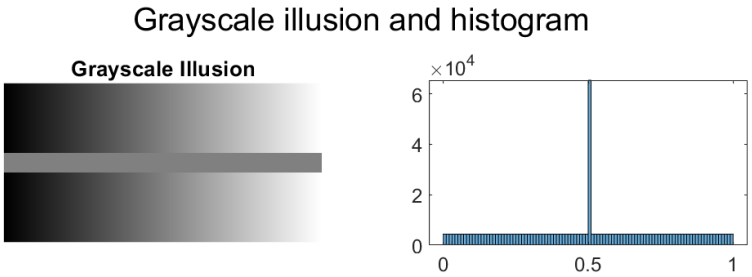
1.1.3 Histograms

Question 9: *What is a histogram? What is the use? Display and interpret histograms of images.*

A histogram of an image represents the intensity level repartition in the image according to the following equation:

$$H_I(n) = \sum_u \sum_v \{I(u,v) = n\}$$

However, spatial information is lost. If we take the histogram of the Grayscale illusion we can see that the color value in the center bar is clearly overrepresented.



On the image `SpainBeach.jpg`, the histogram in Figure 2 is less symmetrical. An equalized option is also available, which adjusts the bin size dynamically.

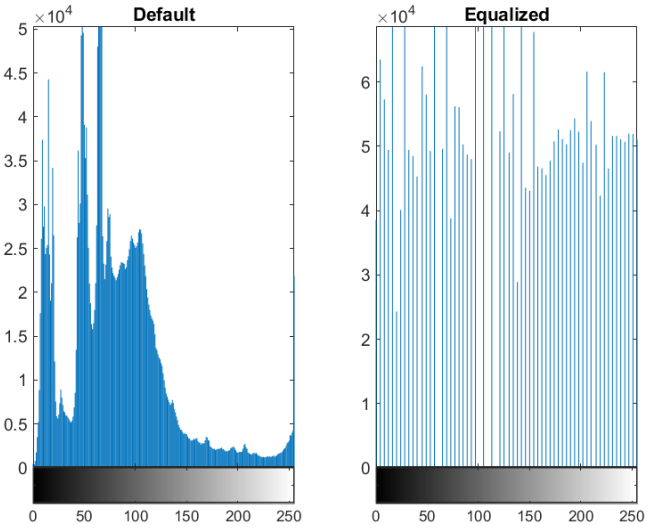


Figure 2: Grayscale histogram

Question 10: *Work the mysterious images called Imagex.bmp and Imagexx.bmp.*

Images with the extension `.bmp` not only include pixel colors, but also a color map. In the case of imagexx this map is empty, leading to a black-looking image by default. However plotting the image as a double data type increases the saturation, and what seems to be the Earth appears, in wrong colors however.

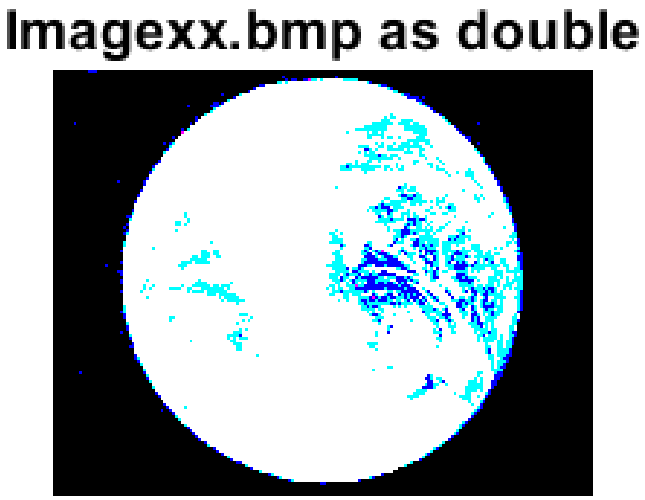


Figure 3: Mystery image

1.1.4 Filtering

Filtering can be associated to blur and to edge detection. Commands ***`imfilter`*** *and* ***`fspecial`*** *can be used to respectively filter images and define kernel (which can be also defined as a matrix).*

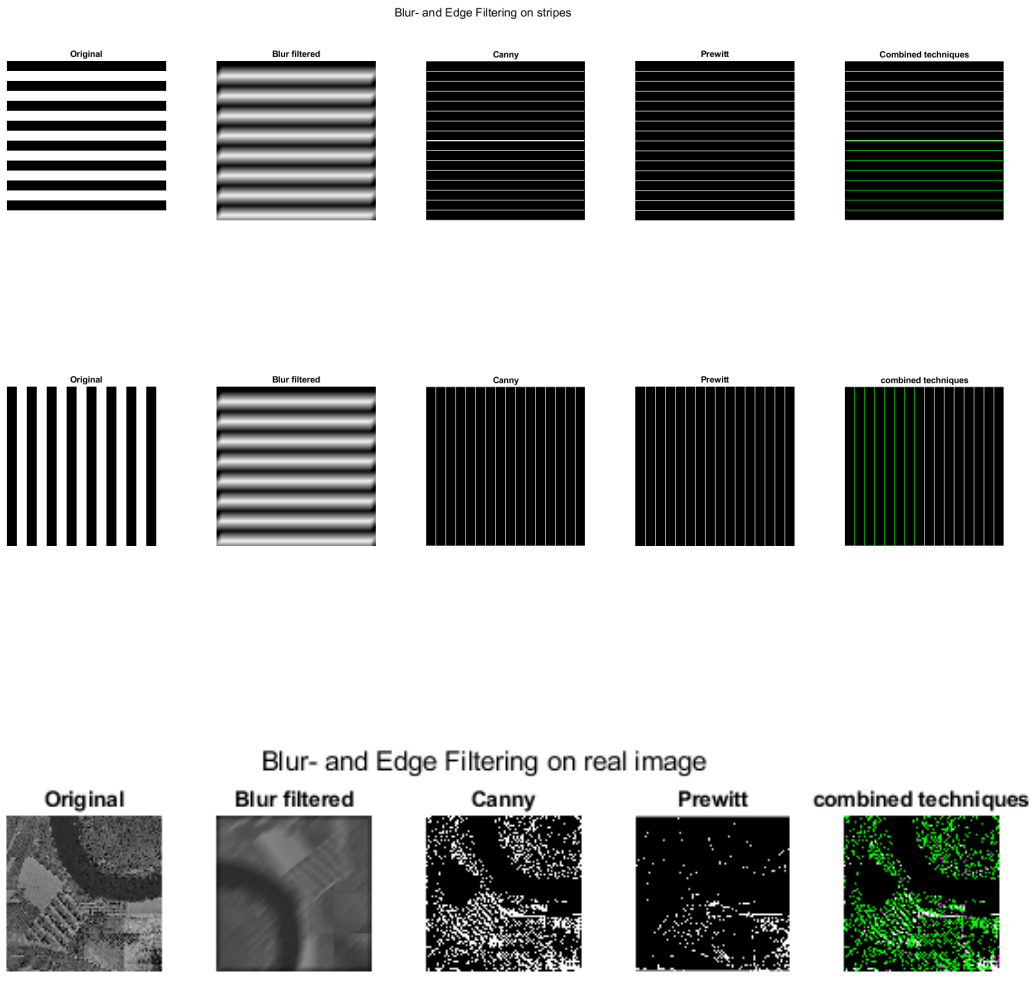
Question 11: *Apply blur Filtering and Edge filtering on the Stripes images and on a 'real' image. What are the main associated Kernels ?*

The kernel options for the `fspecial()` function are: Average, Disk, Gaussian, Laplacian, Log, Motion, Prewitt, and Sobel. For Edge detection, it is easier to use the `edge()` function in MatLab, which has the following methods: Sobel, Prewitt, Roberts, Log, Zerocross, Canny and Approxcanny.

The Disk and Motion kernels seem to work best for blurring the stripes images and the Gaussian did not seem to do much. However, the Motion kernel required specific parameters of length and θ while all tests for other kernels were done with default parameters. The Laplacian and Log kernels seemed to blur everything other than the edges of the stripes, and not even blurring, but not showing anything else just like an edge detection function would. The Prewitt and Sobel kernels performed similarly to no surprise as they are edge detection algorithms, however, they did not particularly work for the horizontal stripes. When applied on a real image, all of these kernels seem to do a decent job at blurring the image.

For edge detection, the Canny method clearly performs best. For some cases, a combination of the Prewitt and Sobel kernels seemed to perform similarly well, while a combination of the Prewitt and Canny methods gave interesting "green" results.

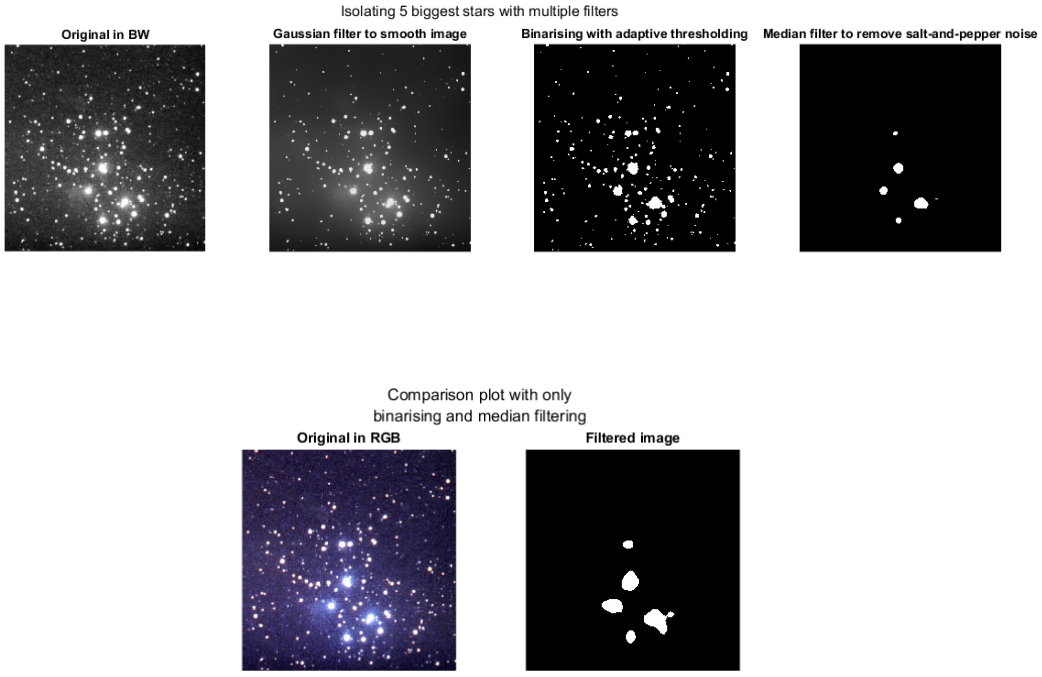
Different blurring and edge detection kernel- and method options applied on both the stripes image and a real image (`Champs.jpg`)) can be seen in the image below. In particular the Motion kernel for blurring and the Canny and Prewitt methods for edge detection as well as a combination of the two latter methods showing the strange green results.



Question 12: *Thanks to successive filtering operators, isolate the main 5 stars of the image Etoiles.png.*

The process to isolate the 5 largest stars can be to first convert the image to grayscale, then smooth the image with a Gaussian filter, then binarise it with an adaptive threshold and finally

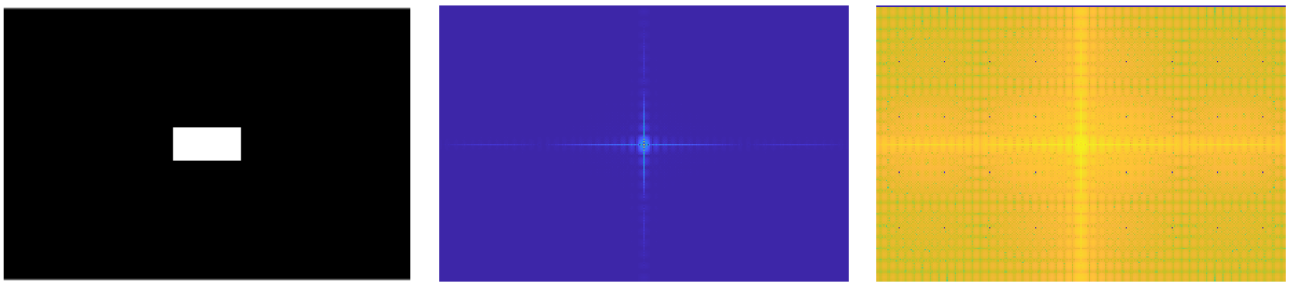
apply a median filter to remove "salt-and-pepper noise". A comparison was made between a more complicated and simpler approach and the results of the better approaches can be seen below:



1.2 Fourier Transform

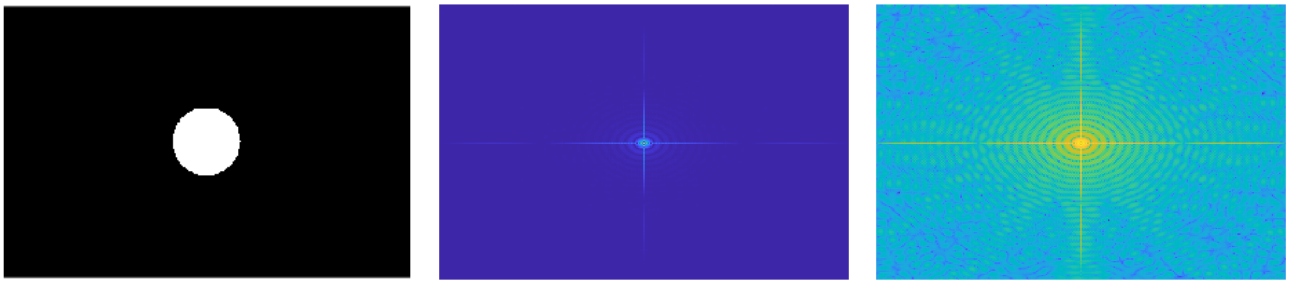
Question 13: *Get the FT and analyze the spectrum if images with stripes (Fig. 2), rectangle and disks (Fig. 3).*

The Fourier transform seems to exhibit a repeating pattern that is close to the original shape, such as the concentric circles and the rectangles. As a rule of thumb the orientation of the periodic features, such as the stripes can also be found in the spectrum again.



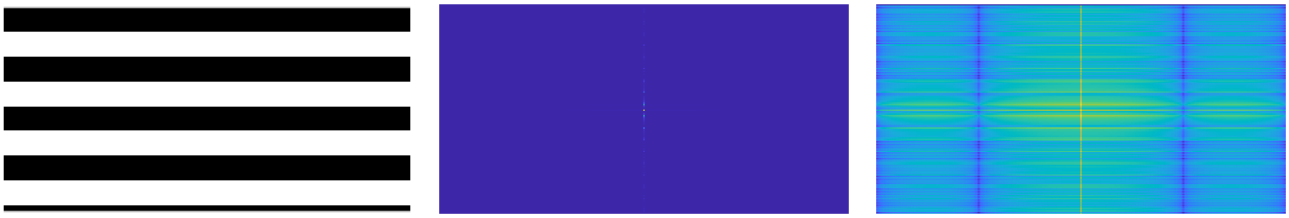
(a) Original (b) Linear Spectrum (c) Log Spectrum

Figure 4: Fourier transform of rectangle



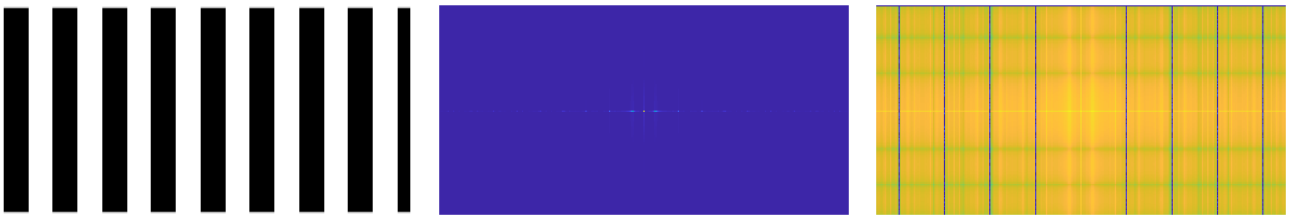
(a) Original (b) Linear Spectrum (c) Log Spectrum

Figure 5: Fourier transform of circle



(a) Original (b) Linear Spectrum (c) Log Spectrum

Figure 6: Fourier transform of horizontal stripes



(a) Original (b) Linear Spectrum (c) Log Spectrum

Figure 7: Fourier transform of vertical stripes

Question 14: *Blur the image with different kernels and interpret the spectrum.*

As we blur the images with larger disks, the blur is more pronounced. In the Fourier transform spectrum this results in high frequencies getting eliminated and the low frequencies around the origin becoming more pronounced. This is to be expected, as blurring eliminates small details which yield high frequencies in the spectrum.

At the same time, the blurred spectra also show that similar features to both the image and the blurring kernel. In this case, we blur the image with a disk, which leads to the emergence of circular patterns in the spectrum. This can be lead back to the mathematical relation that a convolution is equal to a multiplication in the frequency domain.

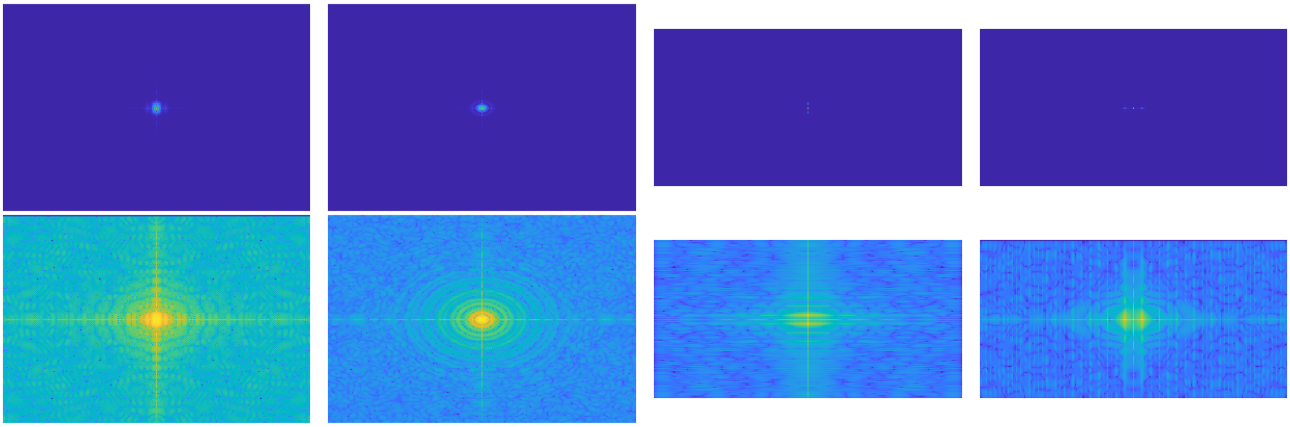


Figure 8: Blurring with disk of radius 10

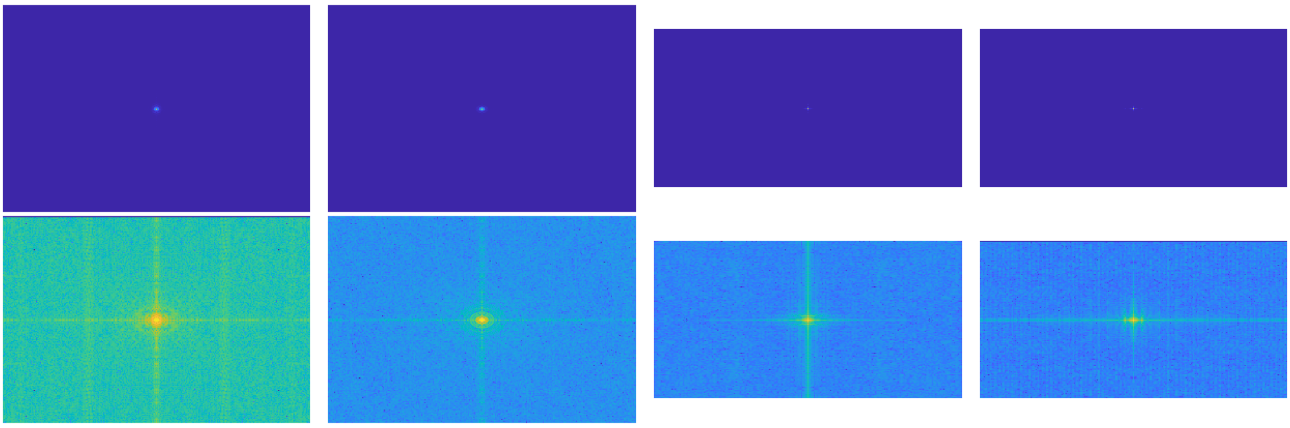
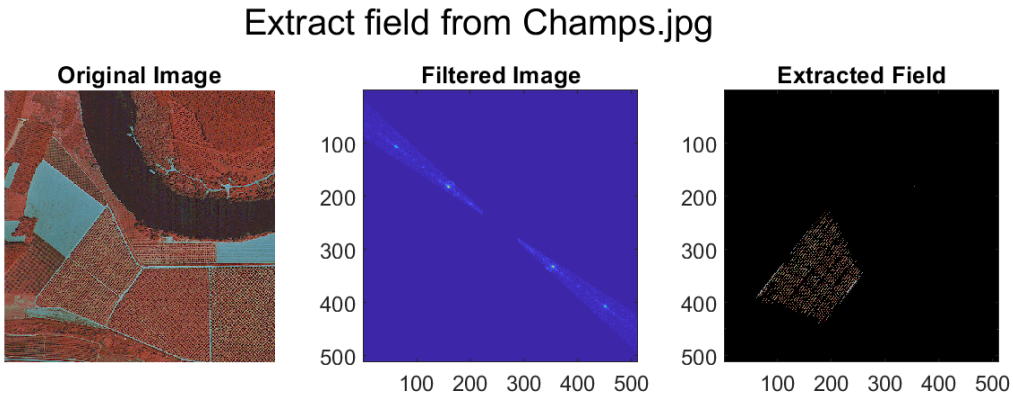


Figure 9: Blurring with disk of radius 50

Question 15: *Write a program that extracts the specific field in the image Champs.jpg (see Fig. 5).*

Since the field has periodic grooves, we can isolate it by narrowing down the range of allowable frequencies from the Fourier spectrum. We want to match the orientation of the field in the spectrum, while also blocking of values close to the origin of the spectrum, which carry noise and are too low in frequency to be relevant.



1.3 Deblurring

In this section, we will see some examples of image deblurring algorithms through linear modelization.

The main sources of blur in an image are:

- Bad focalization,
- Moving blur,
- Atmospheric turbulence,
- etc.

1.3.1 Linear Modelization

As a first approximation, a blurred image can be modelled as follows:

$$Y = H * I + B \tag{2}$$

where:

- Y is the blurred image to be restored,
- H is the convolution kernel,
- I is the original image to be estimated,
- B is Gaussian additive noise.

Question 16: *Why do we have chosen this model? What are the limits of this model with respect to a single lens model?*

This model was most likely chosen because of its simplicity in explaining how noise can be modelled (and added) to an image. However, the limits of this model is that it assumes a "single lense model", i.e., that only one form of noise (the one modeled) is present in a given image. This model is not enough to describe real-life situations, as very seldom does an image only contain such noise as represented in this model.

The model also does not account for spatially varying blur, i.e., the blur for each pixel is based on one single value present throughout the whole convolution kernel. In a real-life situation, images might be blurred quite unevenly throughout the image. objects closer to the camera lense might also be differently blurred than objects further away, however, in this image since it is taken from a great distance, that approximation might be reasonable.

The model also assumes only linear distortion effects and does not take into account nonlinear effects such as sensor saturation (certain intensities might get clipped), optical aberrations (e.g. distortion and chromatic aberration) and motion blur (objects at different speeds blur differently). The latter of which is most likely neglectable for an earth observation application such as this.

Finally, the model assumes the only present noise form would be Gaussian noise, however, many other noise forms exist in real images such as Poisson or salt-and-pepper noise.

*Load the image of Toulouse (**toulouse.bmp**), blur this image with a $(2T+1)$-sized square kernel (Typically $T = 3$). The convolution kernel is thus $h(x, y) = \alpha$ if $|x| \leq T$ and $|y| \leq T$, with $\alpha = \frac{1}{(2T+1)^2}$.*

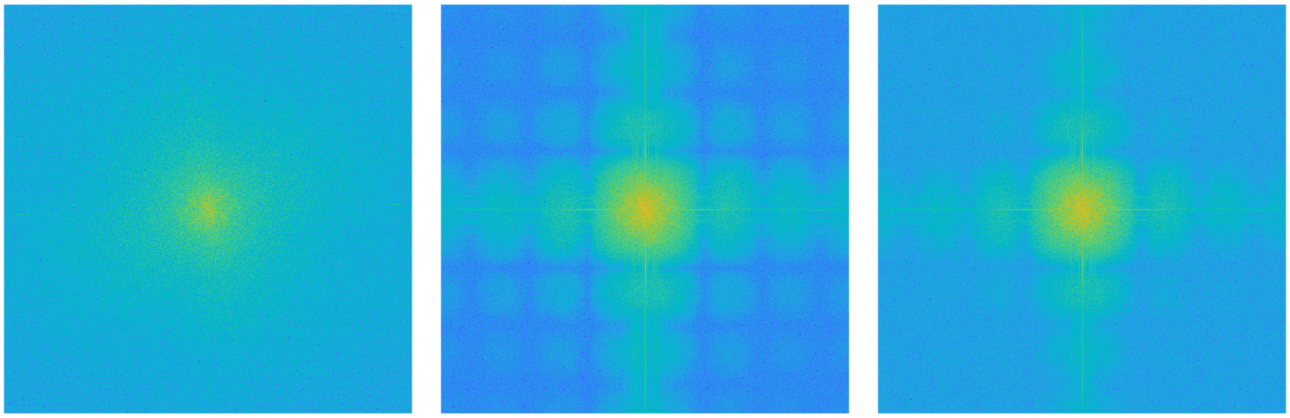
The original image with added blur and noise can be seen in Figure 10. The chosen parameters for the processing are $T = 3$ and a standard deviation of 2 for the gaussian noise.

(a) Original (b) Blurred (c) Blurred and noisy

Figure 10: Processed images

Question 17: *Compare the spectrums of the original image and of the blur image: what do you observe? Justify.*

We can see that the spectrum of the blurred images has a repeating pattern of squares, which decrease in intensity. This is due to the blurring kernel being essentially a box function, whose Fourier spectrum is a cardinal sine function in both axes. Once noise is added, the sharp features are washed out, and the aforementioned repeating patterns are most visible towards the axes, where the Fourier transform of the blurring kernel yielded the largest amplitudes initially.



(a) Original (b) Blurred (c) Blurred and noisy

Figure 11: Fourier Spectrum

1.3.2 Blur Estimation

The objective is here to estimate a posteriori the value of T, *i.e.* from the blurred image.

In dim 1, $h(x) = \sqrt{\alpha}$ if $|x| \leq T$. The DFT is

$$\mathcal{H}(u) = \sum_{x=-T}^{+T} h(x) e^{-j2\pi \frac{ux}{N}} = \frac{1}{2T+1} \sum_{x=-T}^{+T} w^x \tag{3}$$

However,

$$\sum_{x=-T}^{+T} w^x = \frac{w^{-T} - w^{T+1}}{1-w} = \frac{w^{-T-\frac{1}{2}} - w^{T+\frac{1}{2}}}{w^{-\frac{1}{2}} - w^{\frac{1}{2}}}$$

Thus,

$$\mathcal{H}(u) = \frac{1}{2T+1} \frac{\sin\left(2\pi \frac{u}{N}(T+\frac{1}{2})\right)}{\sin\left(\pi \frac{u}{N}\right)} \tag{4}$$

In this case, $N = 512$, $u = -512 \ldots 512$. In dim 2, $h(x,y) = h(x)h(y)$.

Question 18: *Propose a cardinal sinus function that have the same zeros than H(u) (superpose the two functions). What conclusion could you give from these properties? Could you estimate T ?*

If we work the above Equation 4 a bit we can extract from it two separate cardinal sine functions. As a reminder, $sinc(x) = \frac{sin(x)}{x}$:

$$\begin{aligned} \mathcal{H} &= \frac{1}{(2T+1)} \frac{2\pi \frac{u}{N}(T+\frac{1}{2})}{2\pi \frac{u}{N}(T+\frac{1}{2})} \frac{\sin 2\pi \frac{u}{N}(T+\frac{1}{2})}{\sin(\pi \frac{u}{N})} \\ &= \frac{\pi \frac{u}{N} \cancel{(2T+1)}}{\cancel{(2T+1)} 2\pi \frac{u}{N}(T+\frac{1}{2})} \frac{\sin 2\pi \frac{u}{N}(T+\frac{1}{2})}{sin(\pi \frac{u}{N})} \\ &= \frac{sinc(2\pi \frac{u}{N}(T+\frac{1}{2}))}{sinc(\pi \frac{u}{N})} \end{aligned} \tag{5}$$

The zeros of this function, will be zero when the sinc function in the nominator approaches zero. This happens when the nominator of this sinc function in turn approaches zero:

$$\begin{aligned} sinc(2\pi \frac{u}{N}(T+\frac{1}{2})) &= 0 \\ \frac{sin(2\pi \frac{u}{N}(T+\frac{1}{2}))}{2\pi \frac{u}{N}(T+\frac{1}{2})} &= 0 \\ sin(2\pi \frac{u}{N}(T+\frac{1}{2})) &= 0 \end{aligned} \tag{6}$$

This occurs when:

$$2\pi \frac{u}{N}(T+\frac{1}{2}) = k\pi, k \in \mathbb{Z}$$

or when:

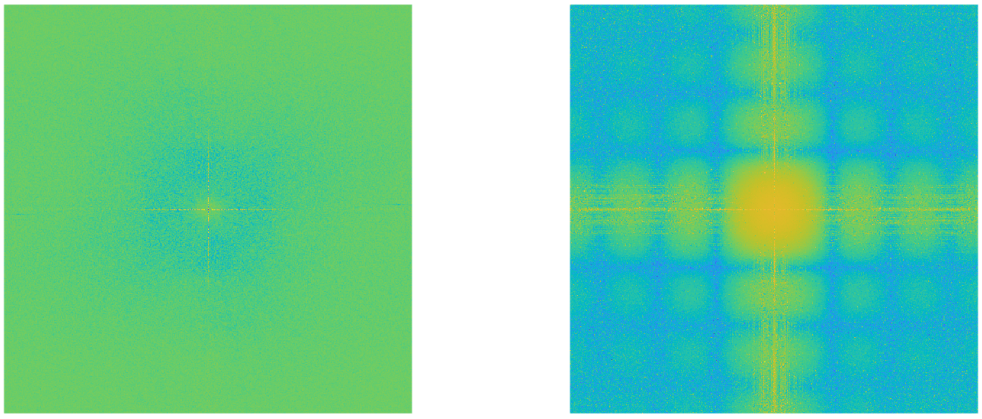
$$u = \frac{kN}{2(T+\frac{1}{2})} \quad \Rightarrow \quad T = \frac{kN}{2u} - \frac{1}{2} \tag{7}$$

One conclusion we can draw from this is that the blur removes certain frequencies, i.e., zeros in the spectrum. This indicates a loss of information at those points. This means that blurred images lose high-frequency details making sharp edges appear smoother.

An algorithmic approach to estimate T-values would then be to first study the image and find out at what frequencies, u, the FT approaches zero values at the different harmonics (probably enough at $k = 1$). Then we could based on this frequency calculate the T-value from Equation 7 for a known N and the already identified frequency u.

Compare the original image spectrum and the blurred image spectrum (using eventually the log function):

The difference between the original image and the blurred image is as expected very pronounced in the square pattern, but also very large in the center of the spectrum.

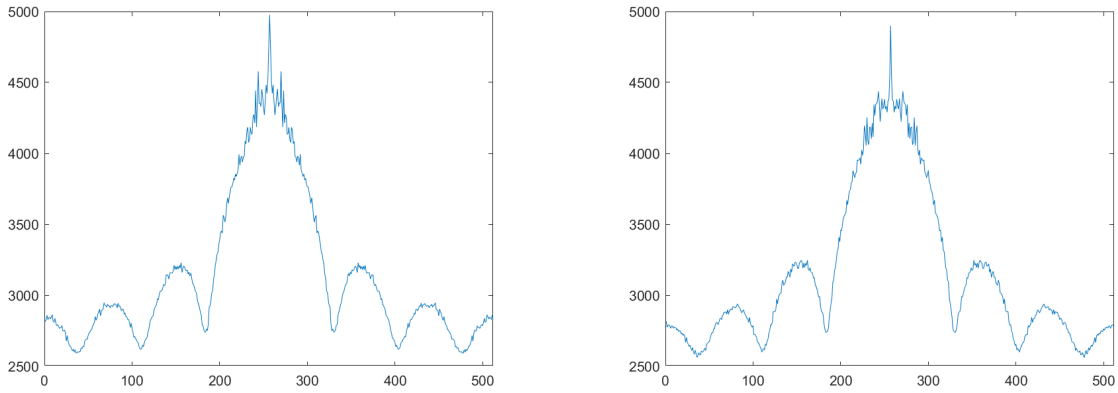


(a) Linear scale (b) Logarithmic scale

Figure 12: Difference of the Fourier spectra

Question 19: *Estimate T.*

Because of the periodicity, we can sum the spectrum in both dimensions to smooth out the noise and contribution from the images spectrum.



(a) x-direction (b) y-direction

Figure 13: Sum of the Fourier Spectrum in one dimension

We can see how the oscillations due to the cardinal sine can be found in the summed up spectrum. In fact, this is due to the images spectrum being multiplied with the cardinal sine, hence the appearance being not just the cardinal sine. We can nonetheless try to estimate T by fitting sinc functions whose zeros correspond to the local minima of the sums. Depending on which minimum we fit, the value we obtain is roughly $T = 2.0$ to $T = 2.7$.

1.3.3 Image Deblurring

Now, image deblurring (or image restoration) can be done using multiple methods. We focus on two of them:

- Inverse Filtering:

- Wiener Filtering:

Let $g(x, y)$ be the inverse filter of $h(x, y)$. In the spectral domain, we have:

$$G = \frac{1}{H} \tag{8}$$

The estimated restored image $\hat{I}$ is:

$$\hat{I} = GY = GHI + GB = I + GB \tag{9}$$

The following program allows to apply this process:

```
SeuilMax = 11 ;
hh = zeros(TailleImage);
centre = [1 1] + floor(TailleImage/2) ;
ext = (TailleFiltre-[1 1])/2;
ligs = centre(1) + [-ext(1):ext(1)];
cols = centre(2) + [-ext(2):ext(2)];

h = ones(TailleFiltre)/prod(TailleFiltre);
hh(ligs,cols) = h;
hh = ifftshift(hh);

H = fft2(hh);

ind = find(abs(H)<(1/SeuilMax));
H(ind) = (1/SeuilMax)*exp(j*angle(H(ind)));

G = ones(size(H))./H;

(...)
```

Question 20: *Complete this program (only 2 lines!) to process inverse filtering method.*

The full code is provided in the same source code directory in the `Restauration_ToBeCompleted.m` file. The two lines to complete this program (with some small other adjustments) can be seen in the *Restauration.m* file in this directory on lines 106 and 107 – and below:

```
TFdIrest = G.*TFdIf;
Irest = real(ifft2(TFdIrest));
```

Question 21: *What's happen with the image marcheur.jpg ?*

We can see that the motion blurred, but non-noisy `marcheur.jpg` picture is reconstructed partially, but not particularly well. The situation is the same also when noise is added. We can see that the algorithm performs significantly better for the `toulouse.bmp` image. This is because the inverse filtering method here only applies the reverse operation to blurring with a square kernel. Since the motion blur kernel is not square, this method doesn't work on the `marcheur.jpg` image.

Image non bruitée restaurée par filtrage inverse

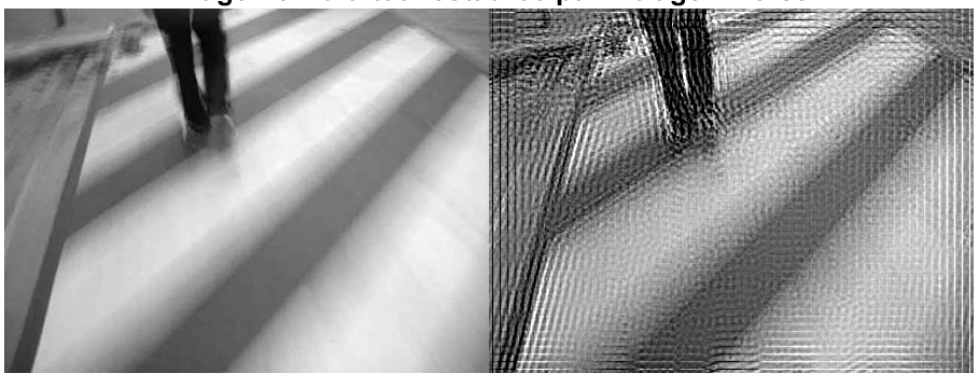
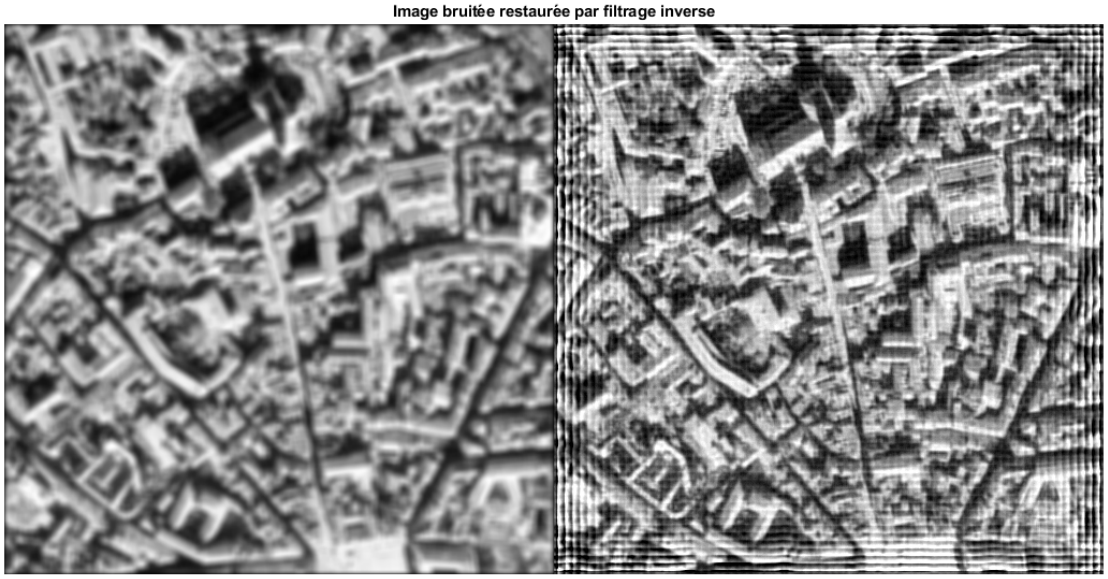


Image bruitée restaurée par filtrage inverse



2 Introduction to Mathematical Morphology

2.2 Erosion and Dilation

Question 1 *Define erosion and dilatation from an ensemblist point of view and an functional point of view. Give some properties related to these operators.*

The ensemblist (or set-theoretic) perspective treats an image as a set of foreground pixels. In this context, the erosion $(A \ominus B)$ of set A (the image) by a structuring element B results in the set of points where B, translated at each point, is entirely contained within A. This operation shrinks objects and removes small structures.

$$A \ominus B = \{x \in \mathbb{Z}^n | B_x \subseteq A\}$$

The dilation $(A \oplus B)$ of A by B results in the set of points where the reflection of B, translated at each point, intersects A. This operation expands objects and fills small gaps.

$$A \oplus B = \{x \in \mathbb{Z}^n | B_x \cap A \neq \emptyset\}$$

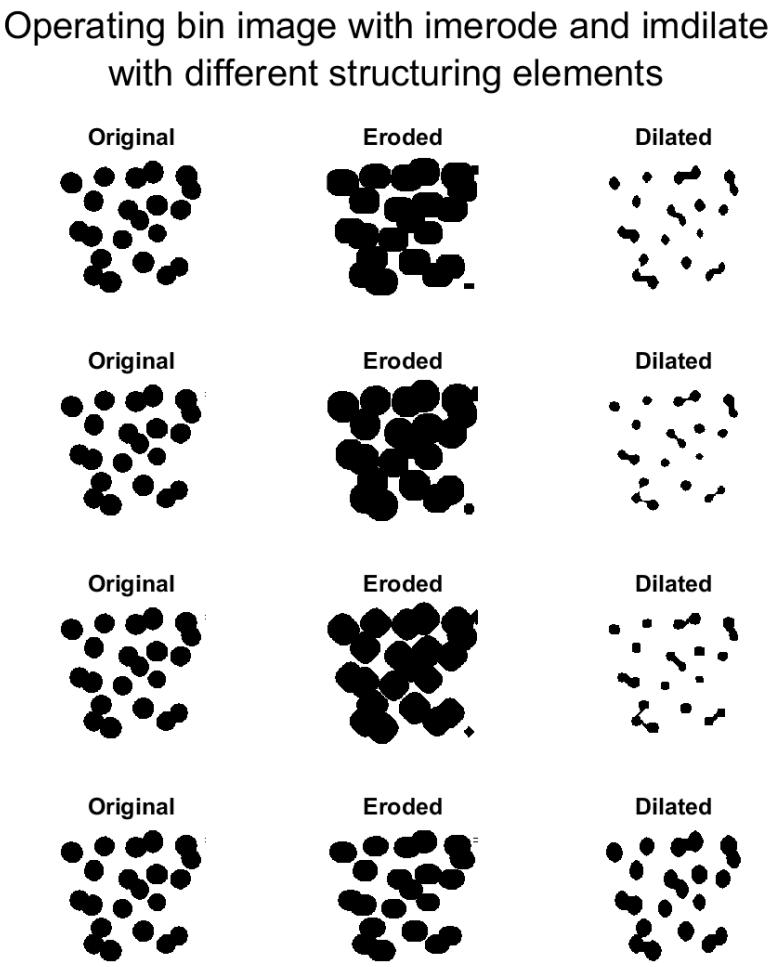
From a functional perspective, images are seen as functions and morphological operations are defined using the maximum and minimum. For erosion: $(f \ominus B)(x) = \inf_{b \in B} f(x + b)$ this replaces each pixel with the minimum value under the structuring element which makes bright regions shrink. And for dilation: $(f \oplus B)(x) = \sup_{b \in B} f(x - b)$ this replaces each pixel with the maximum value under the structuring element which makes bright regions expand.

Some properties of Erosion and Dilation include:

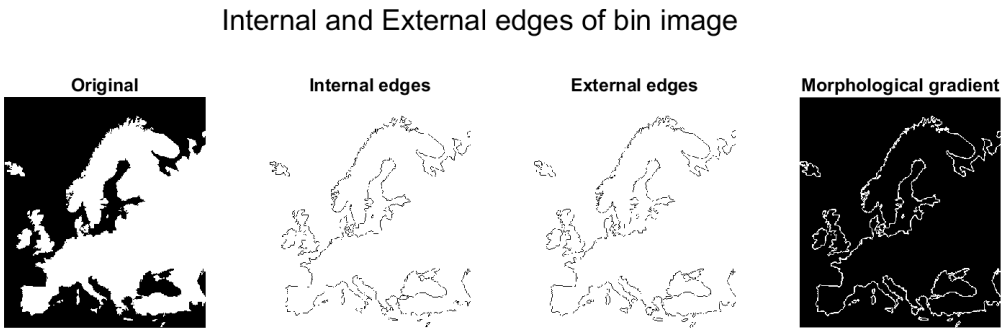
1. **Duality:** $A \ominus B = (A^c \oplus B)^c$ where A^c is the complement of set A.
2. **Increasing Property:** If $A \subseteq C$, then: $A \ominus B \subseteq C \ominus B, \quad A \oplus B \subseteq C \oplus B$
3. **Extensive and Anti-extensive Properties:** indicate that dilation expands the set, while erosion shrinks it. $A \subseteq A \oplus B, \quad A \ominus B \subseteq A$
4. **Associativity:** Erosion and dilation are associative operations: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$
5. **Translation Invariance:** For any translation T_t (shift by t), we have: $(T_t A) \ominus B = T_t(A \ominus B), \quad (T_t A) \oplus B = T_t(A \oplus B)$ where $T_t A = \{x + t \mid x \in A\}$ is the translation of A.

Question 2 *Operate in a binary image and a greyscale image with the MatLab commands imerode and imdilate. What are the effects on binary and grayscale images? Justify. Try with different structuring elements (different shapes, different sizes).*

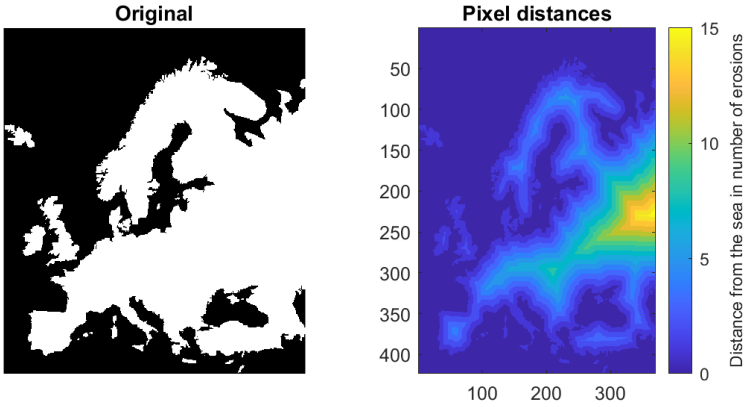
In the figure below, the eroded and dilated images can be seen plotted next to the original image for each structuring element with the following shapes in order: rectangle, disk, diamond and line.



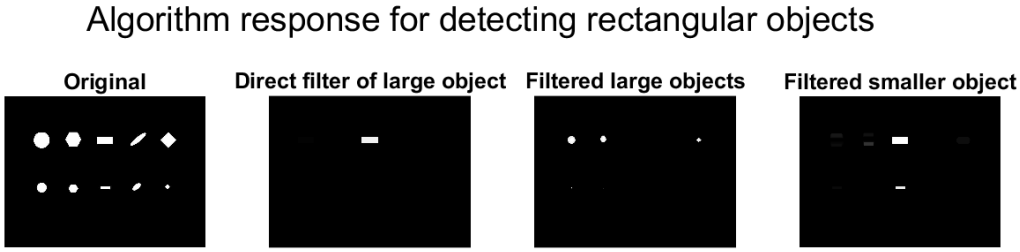
Question 3 *Extract internal and external edges of a binary image, and the morphological gradient.*



Question 4 *As an exercise, write an algorithm that show, in the map of Europe, the distance of each pixel w.r.t. the sea.*



Question 5 *Find an algorithm that detect rectangular objects of 'image2.jpg'.*



2.3 Morphological Filtering

2.3.1 Filters

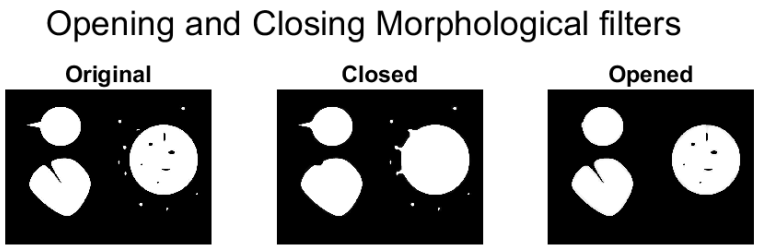
Question 6 *Define the two morphological filters called opening and closing. What are the effects on a binary image sur as 'image1.jpg' (use the commands imopen and imclose)?*

- **Opening filter:** is basically erosion followed by dilation. It removes small objects and noise from the foreground and smooths object boundaries.

- **Closing filter:** is basically dilation followed by erosion. It fills small holes and gaps in the foreground and smooths object boundaries.

The `imopen()` command seems to reduce some amount of "black-colored noise" from the image and slight sharp inserts in the circle in the bottom left of the original image. The `imclose()` command seems to remove "white-colored noise" and the outlet of the circle in the top left.

The effects of the commands on `image1.jpg` can be seen below.



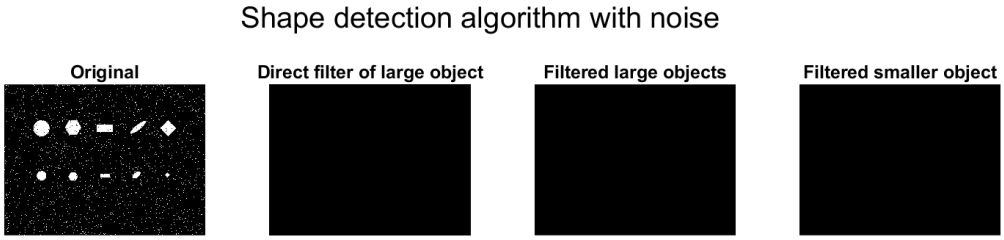
2.3.2 Form Detection

Question 7 *The objective being to recognize some forms on images, find a simple algorithm to operate form detection*

See Question 5, can easily be modified for different shapes, i.e. circle, diamond etc.

Question 8 *Apply a salt-and-pepper noise: what's happen with your previous algorithm?*

The algorithm breaks and does not function at all due to the noise in the picture. See below:

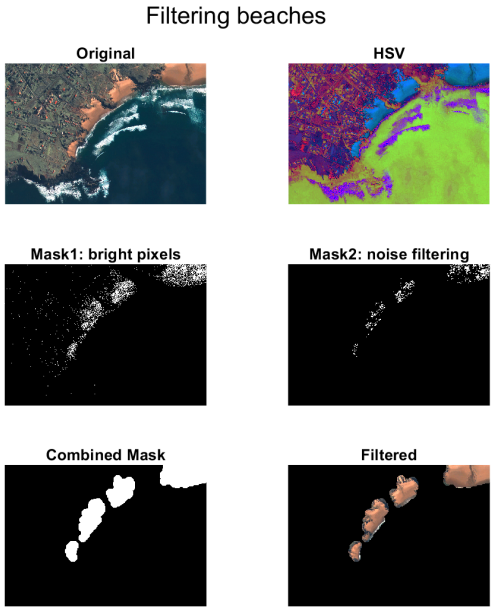


2.3.3 Denoising

Question 9 *Use the image Nebuleuse.jpg and apply a salt-and-pepper noise. De-noise the image by filtering.*

Figure 14: Nebula with added and then removed noise

Question 10 *Apply the same process to the 'Spain Beach' image to isolate the beach itself.*

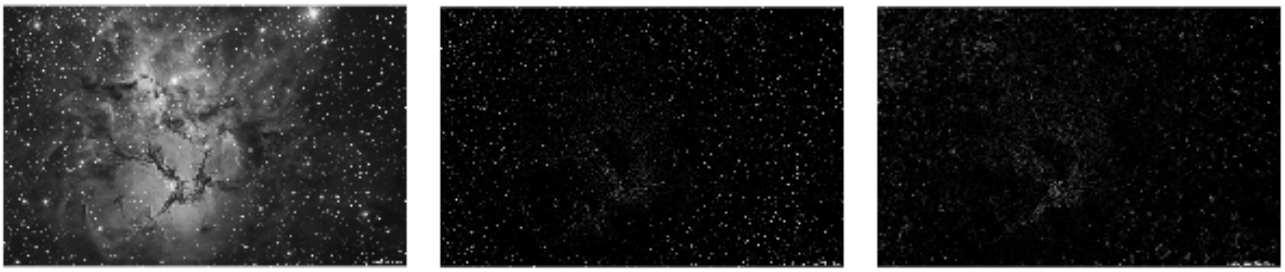


2.3.4 Top-Hat & Black-Hat Filters

Question 11 *Define Top-hat and black-hat process in a 1D function: what is the associated process?*

- **Top-Hat:** This transform is the difference between the original signal and its opening, highlighting small bright features or peaks in the signal.
- **Black-Hat:** This transform is the difference between the closing of the signal and the original signal, highlighting small dark features or valleys.

Question 12 *Define and operate top-hat and black-hat on a greyscale image. What do you observe?*



(a) Original in grayscale (b) Top-hat (c) Black-hat

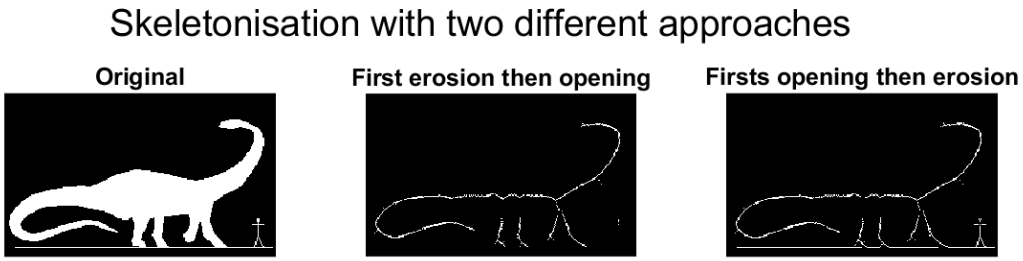
Figure 15: Transform results

We can observe that the top-hat transform preserves the small, bright details, while the black-hat transform keeps the dark details, such as the dark gas clouds in the nebula.

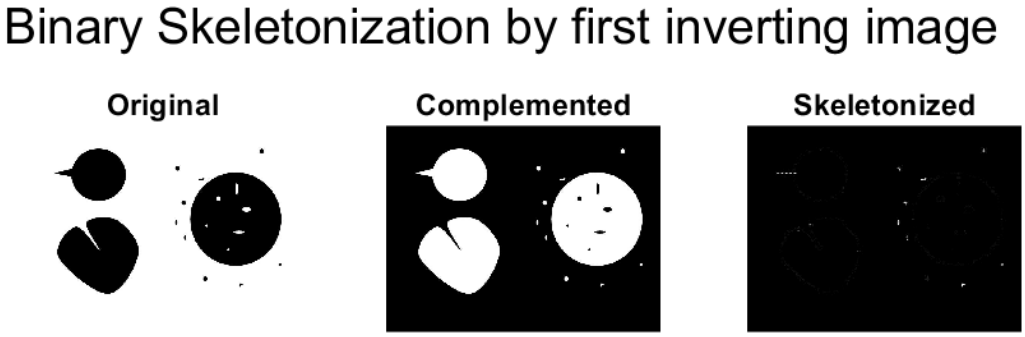
2.4 Morphological Skeletonization & Segmentation

2.4.1 Skeletonization Process

Question 13 *Write and operate a Skeletonization on the diplodocus.*

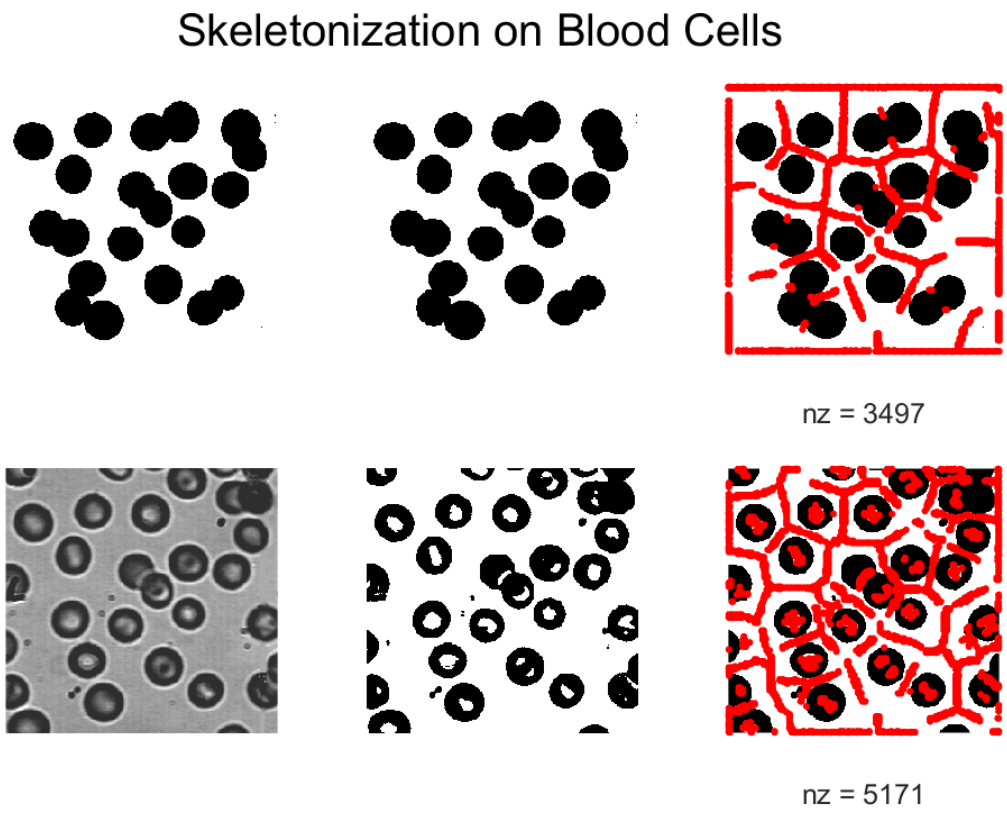


Question 14 *Based on skelittization, find an algorithm that operate a segmentation in a binary image. Apply on 'image1.jpg'.*



2.4.2 Image Segmentation

Question 15 *Find a Skeletonization algorithm and operate on the Blood Cells image.*



3 Shape Detection

3.1 Basics

Question 1: Remind the principle of pseudo-inverse approach.

Now you observe a positions of an asteroid at different times, and the objective is to estimate its trajectory.

Question 2: Complete the program `'AsteroidTry'` to estimate the asteroid trajectory. Detail the theoretical approach.

3.1.1 Principles of the Pseudo-Inverse approach

The pseudo-inverse approach can be applied to find the best fit models to data. This is works well for linear models, such as

$$\vec{y} = a\vec{x} + b \tag{10}$$

where $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_n)$ are our data points and a and b are the sought after parameters. The linear equation can be rewritten in matrix form for each data point as

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}}_{Y} = \underbrace{\begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{P} \tag{11}$$

The parameter vector is then found by inverting the equation to

$$P = X'Y \tag{12}$$

An alternative way to formulate the problem, which is handy when the equations aren't linear, is to write it out a linear relationship as follows:

$$a\vec{x} + b\vec{y} + c = 0 \Leftrightarrow -\frac{a}{c}\vec{x} - \frac{b}{c}\vec{y} = 1 \equiv \alpha\vec{x} + \beta\vec{y} = 1 \tag{13}$$

The matrix form is then expressed as

$$\begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \mathbb{1} \Leftrightarrow XP = \mathbb{1} \tag{14}$$

The pseudo-inverse is then defined by multiplying by the transpose of X on both sides, before inverting the equation:

$$X^tXP = X^t\mathbb{1} \Rightarrow P = (X^tX)^{-1}X^t\mathbb{1} \tag{15}$$

where the $pinv(X) = (X^tX)^{-1}X^t$ is the pseudo-inverse of X. With this formulation, more complex models that involve powers of x and y can be used to fit data. For example, an asteroid's trajectory is an ellipse, whose mathematical model is given by the expression

$$ax^2 + 2bxy + cy^2 + dx + ey = 1 \tag{16}$$

Here we have both linear and quadratic terms, as well as cross-terms, which describe the overall shape and orientation of the ellipse. Each term is also not independent from each other, and

we can reconstruct d and e from a, b and c. Still, the pseudo-inverse can be constructed by arranging the terms in the matrix as follows:

$$\begin{pmatrix} x_1^2 & 2x_1y_1 & y_1^2 & x_1 & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & 2x_ny_n & y_n^2 & x_n & y_n \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \mathbb{1} \tag{17}$$

3.1.2 Estimation of an asteroid's trajectory

The well-known formula for an ellipse is given as

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1 \tag{18}$$

This however does not account for the rotation of an ellipse. The parameters we get out of the pseudo inverse approach are for the rotated ellipse. We can consider the ellipse to be the rotation of a traditional, axis-aligned ellipse, that was rotated through the transformation

$$\begin{cases} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{cases} \tag{19}$$

This ellipse's conic section equation won't have any cross terms which arise from the rotation. That is, we expect the form

$$a'x'^2 + c'y'^2 + d'x' + e'y' = 1 \tag{20}$$

If we plug the transforms Equation 19 into Equation 16, we obtain the relations between the old and new coefficients as follows:

$$\begin{cases} a' = a\cos^2\theta + b\cos\theta\sin\theta + c\sin^2\theta \\ b' = b(\cos^2\theta - \sin^2\theta) + 2\cos\theta\sin\theta(c-a) \\ c' = a\sin^2\theta - b\cos\theta\sin\theta + c\cos^2\theta \\ d' = d\cos\theta + e\sin\theta \\ e' = -d\sin\theta + e\cos\theta \end{cases} \tag{21}$$

We can immediately recover the angle of rotation by imposing

$$b \overset{!}{=} 0 \Leftrightarrow \frac{2b}{a-c} = \frac{2\cos\theta\sin\theta}{\cos^2\theta - \sin^2\theta} \tag{22}$$

One can apply the trigonometric identity $\tan(2\theta) = \frac{2\tan\theta}{1-\tan^2\theta}$ to find that

$$\theta = \frac{1}{2}\arctan\left(\frac{2b}{a-c}\right) \tag{23}$$

Next we can factorize Equation 20 to make the traditional ellipse equation appear. By expanding the square, we obtain

$$a'\left[x' + \frac{d'}{2a'}\right]^2 + c'\left[y' + \frac{e'}{2c'}\right]^2 = 1 + \frac{d'^2}{4a'} + \frac{e'^2}{4c'} \tag{24}$$

From this we can read off the values of the semi-major and semi-minor axes compared to Equation 18, which will be the maximum and minimum respectively of the set

$$\left\{ \sqrt{\frac{1 + \frac{d'^2}{4a'} + \frac{e'^2}{4c'}}{a'}}, \sqrt{\frac{1 + \frac{d'^2}{4a'} + \frac{e'^2}{4c'}}{c'}} \right\} \tag{25}$$

The center of the ellipse is trivially

$$\begin{cases} x'_0 = -\frac{d'}{2a'} \\ y'_0 = -\frac{e'}{2c'} \end{cases} \tag{26}$$

Which must of course be rotated back into place with Equation 19 to have the center point of the rotated ellipse.

With the mathematical framework at hand, the result of the MatLab implementation is shown in Figure 16. The asteroid's trajectory is generated at random every time the script is run, and the fit shows a very good match to the original trajectory.

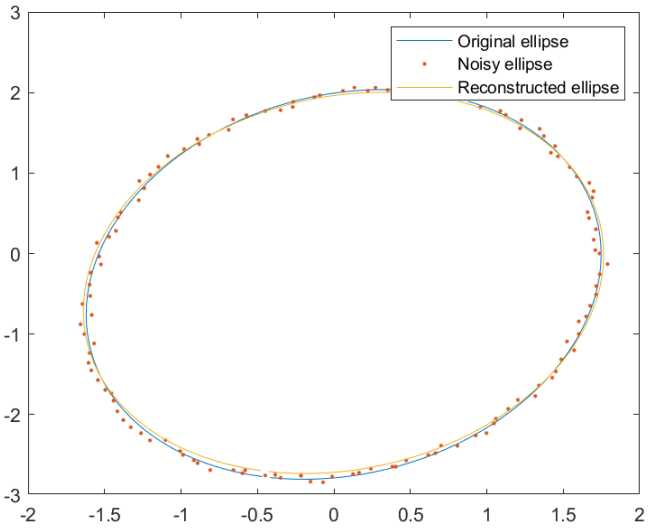


Figure 16: Reconstructed asteroid trajectory

3.2 Shape-Based Approaches

Question 3: Recall basic principle of shape detection in Computer Vision.

Now, let us consider one planet to be detected and characterised (Fig. 1) in 3 different configurations.

Question 4: Detect & Characterise (position, size) the 1st planet in Conf. 1.

Question 5: What's happen with Conf. 2 and Conf. 3?

The underlying idea is to search a pre-defined shape, which is usually a good approximation of the object we are searching for. In our case, we look for disks, which are good approximations for planets and moons. With the defined shape, we must apply an optimization algorithm, which computes the overlap between the shape's dimensions and positions and the objects in the image, and iterates until a maximum overlap is found. We apply the algorithm to the pictures shown in Figure 17, Figure 18 and Figure 19.

While the recognition of a perfect disk, such as for the moon, is very good, we can already see the flaws of this recognition method. As soon as there is more than one object, the method is biased to the larger one, as it will yield a larger overlap value in terms of pixel numbers; additionally, the method is also searching for a single object only, and doesn't try to fit multiple disks. The method also breaks down as soon as the planet is not fully illuminated and has a croissant shape instead of a circle, which leads to a significant mis-estimation of the objects shape.

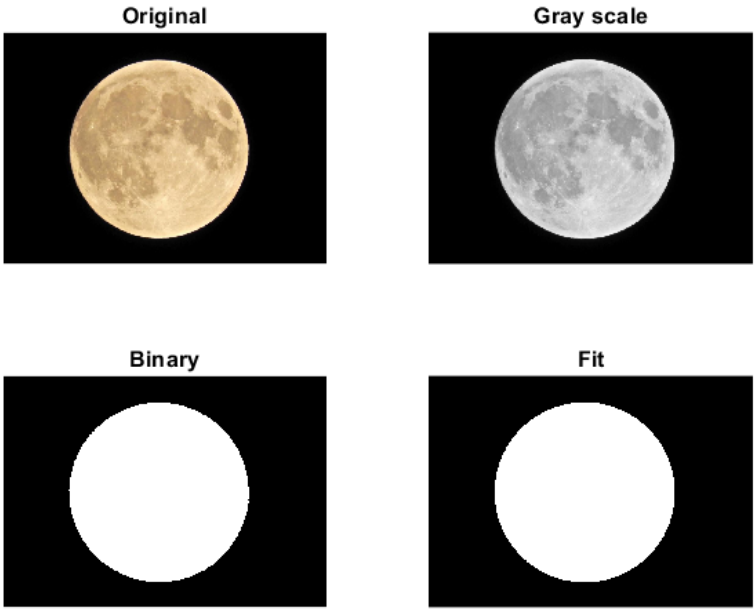


Figure 17: Shape approach on the Moon image

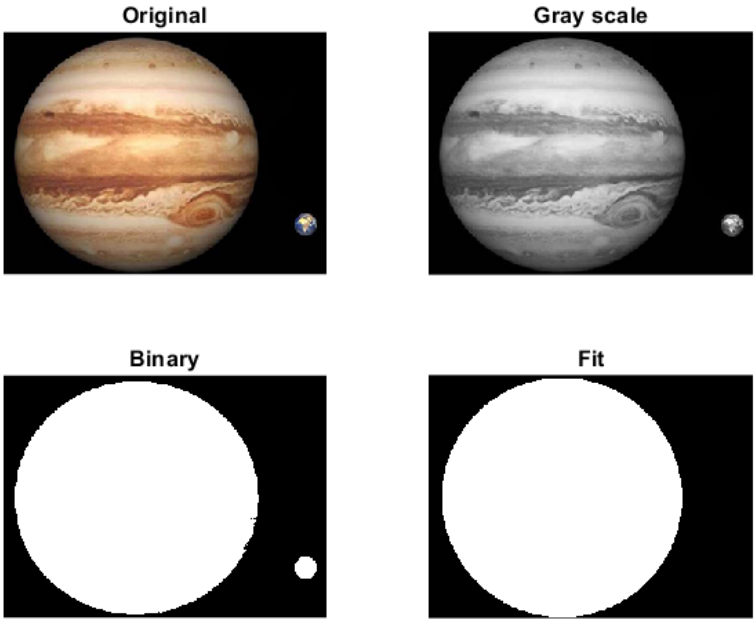


Figure 18: Shape approach on the Jupiter and Earth image

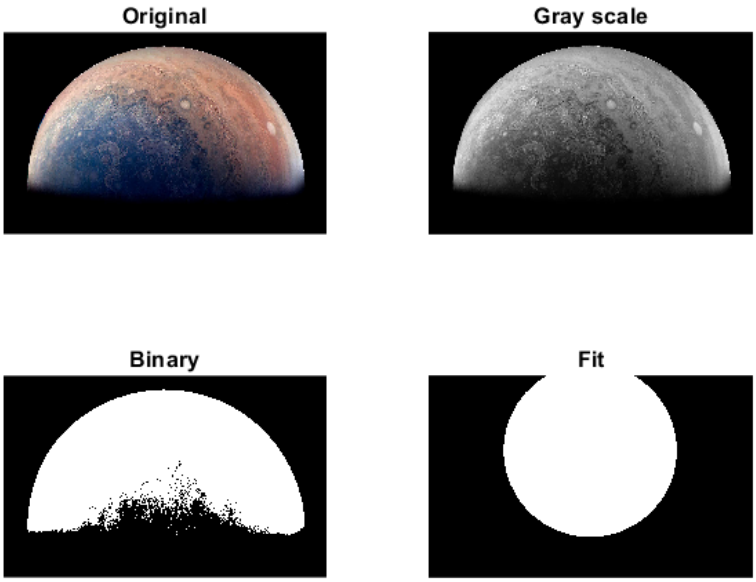


Figure 19: Shape approach on the partial Jupiter image

3.3 Contour-Based Approaches

Question 6: Extract the planet (Conf. 1) using N points and next a Pseudo-Inverse Approach. What's happen in other configurations?

Question 7: Same as 6.

Question 8: Extract the planet (Conf. 1) using an Optimisation algorithm. What's happen in other configurations?

Question 9: Explain the RANSAC Algorithm. Use it in Conf. 2 and Conf. 3.

Question 10: Compare with the circular Hough Transform.

3.3.1 Pseudo-inverse Approach

Circle equation:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{27}$$

As conic section:

$$A(x^2 + y^2) + Bx + Cy = 1 \tag{28}$$

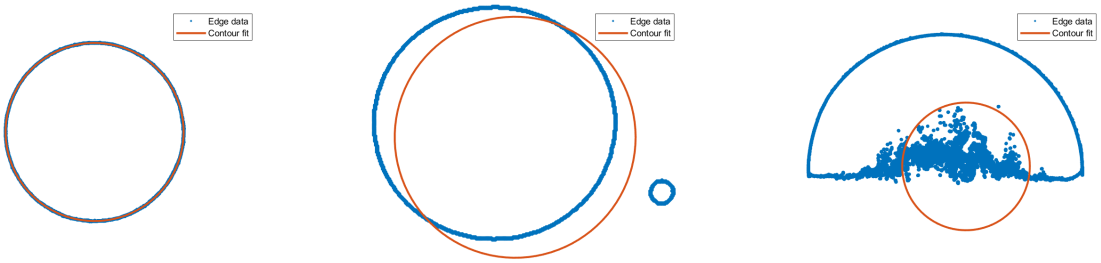
Similarly to the ellipse, we recover the center coordinates by expanding the square, and the radius by comparing with the traditional circle equation:

$$\begin{cases} x_0 = \frac{-B}{2A} \\ y_0 = \frac{-C}{2A} \\ r = \sqrt{\frac{1}{A} + \left(\frac{B}{2A}\right)^2 + \left(\frac{C}{2A}\right)^2} = \sqrt{\frac{1}{A} + x_0^2 + y_0^2} \end{cases} \tag{29}$$

Pseudo-inverse formulation

$$\begin{pmatrix} x_1^2 + y_1^2 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ x_n^2 + y_n^2 & x_n & y_n \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \mathbb{1} \tag{30}$$

To apply the method, we must first extract the edges of the image, which we do by applying a laplacian filter kernel. This method yields the outer of the objects, which is exactly what we want.



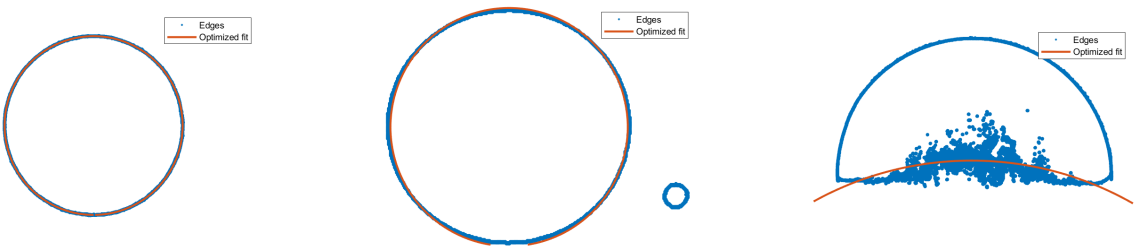
(a) Full Moon (b) Jupiter and Earth (c) Jupiter croissant

Figure 20: Contour approach and pseudo-inverse method results

We can observe again, that the method is excellent for the full moon, which has the exact shape we are trying to fit. When there is more than one object, the fit is biased towards what seems to be the center of mass of the system. In the case of a partial shape the method breaks down again because of all the noise and the fact that it tries to take all data points into account.

3.3.2 Optimization approach

With this approach, we use MatLab's optimization function to search for the minimum in a quadratic cost function, which is simply the sum of the errors of the fit. Interestingly, the optimization works well on the first two configurations. The fact that it fits nicely on Jupiter, but not Earth, is that the optimization method is subject to finding local minima. There is two of such points which correspond to the two planets. Finally on Jupiter's croissant, the method doesn't work at all, as there is to many points which do not correspond to the circumference of the circle, but the algorithm still treats equally important. This results in the fit going through the 'noise' moreso than through the actual points we are interested.



(a) Full Moon (b) Jupiter and Earth (c) Jupiter croissant

Figure 21: Optimization approach results

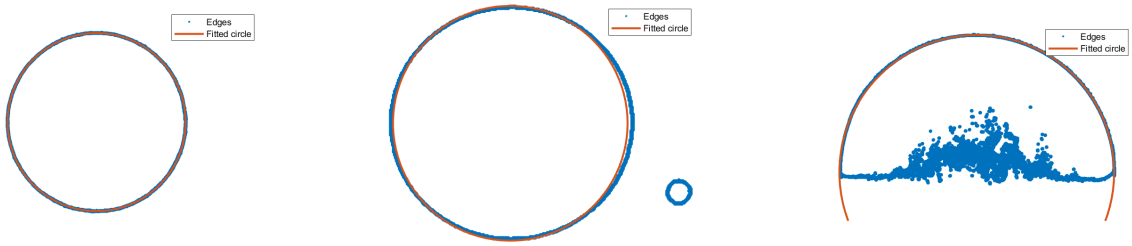
3.3.3 RANSAC approach

The goal of the **RAN**dom **SA**mple **C**onsensus (RANSAC) algorithm is to identify only the points which are actually correspond to the thing we are interested in. Therefore we can define 'inliers' and 'outlier' data points which have get sifted through. When trying to fit a circle, we have to define a tolerance region around the line of best fit, which defines the boundary between the inliers and outliers. A minimum amount of points to be fitted must also be specified, so that we do not under-fit the data, potentially fitting on noise. The algorithm is built in a way that it generates multiple fit solutions based on randomized start conditions, of which the best solution must be selected by the end. This is decided based on either inlier count or least error.

The steps of the algorithm (for fitting circles) are as follows:

1. Select 3 data points randomly (minimum to uniquely identify a circle)
2. Fit the model to these points
3. Identify the inliers based on the threshold
4. If the inlier count is less than the minimum allowed, go back to step 1
5. Fit the model using all the identified inliers
6. Identify the inliers based on the threshold
7. If the inlier count is less than the minimum allowed, go back to step 1
8. Determine the error of the candidate model fit
9. Add the fit, the inlier count and error to the list of candidate model fits.
10. Repeat until the maximum allowed iteration number
11. Examine the list of candidate fits and select the one with highest point number, and/or least error

The results are shown in Figure 22. We can see that the model is excellent at fitting noisy data, as is the case for the Jupiter croissant. In the case of the Jupiter-Earth image, it also successfully excludes the Earth from the fit. The minimum number of inliers to be considered is based on the total amount of data points in the image, and is subject to tuning. The tolerances for identifying inliers also need tweaking through some trial-and error until an optimal fit is found. This leads to certain sets of input parameters to yield no results.

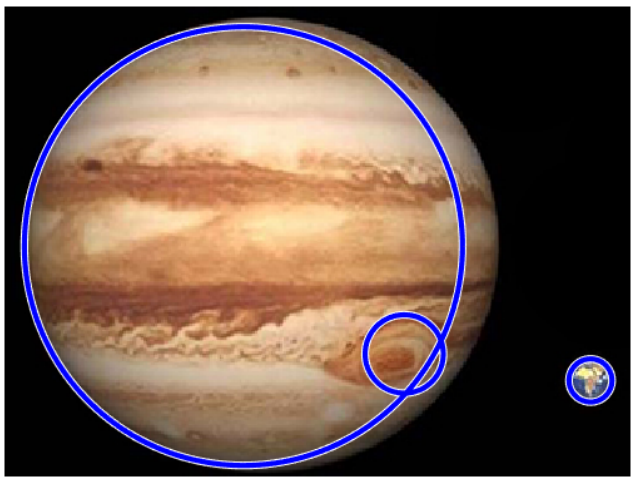


(a) Full Moon (b) Jupiter and Earth (c) Jupiter croissant

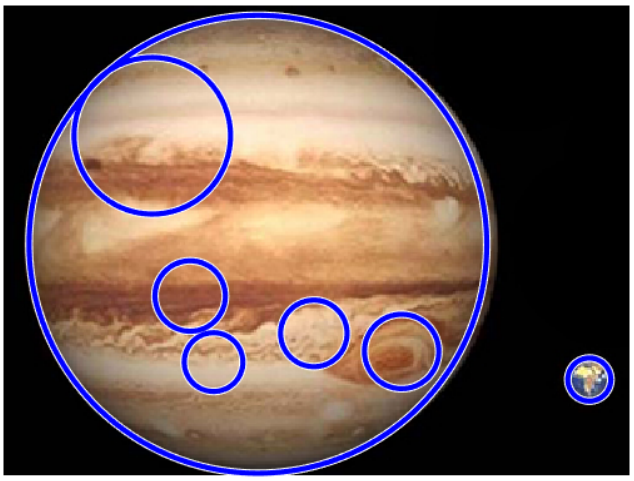
Figure 22: RANSAC approach results

3.3.4 Circular Hough transform approach

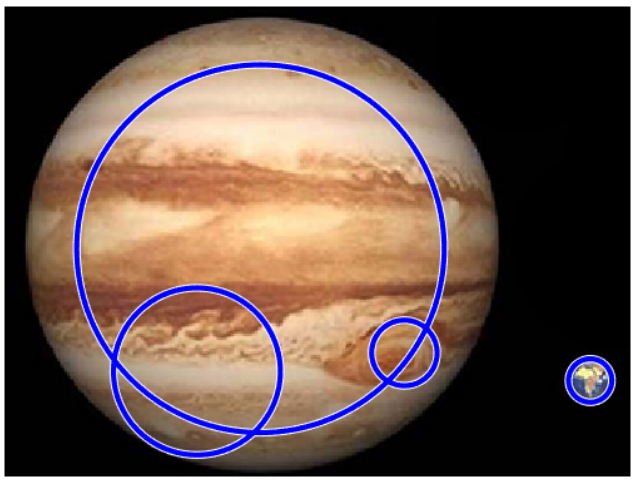
The circle Hough transform works by examining the image in the Hough parameter space to identify possible circular features. It requires either the radius or range of values of radii of the circles to be known. MatLab has a built-in method for this: `imfindcircles()`. While it works on colored and grayscale images, the results seem best when the image is binarized, as the method won't be as sensitive to shapes on the surface, as can be seen in Figure 23. The method seems rather inconsistent overall, and requires a lot of tuning to get right. It seems to detect circles where there aren't any visible features to the human eye, which might be due to slight shadows. It is however able to detect both planets at once in the Jupiter-Earth configuration.



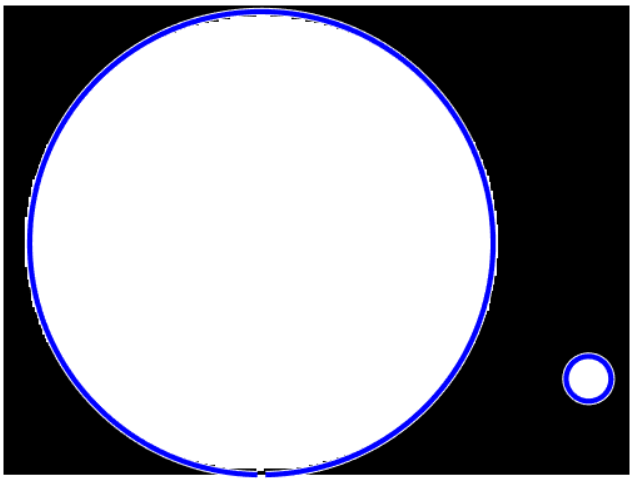
(a) 0.85 Sensitivity, Two Stage Method



(b) 0.9 Sensitivity, Two Stage Method



(c) 0.9 Sensitivity, Phase Code Method



(d) Default parameters on binary image

Figure 23: Hough Method results

4 Introduction to Data Analysis: Data Classification

4.1 Supervised Classification

Question 1: *Enumerate main supervized classification techniques and describe them in few lines.*

Supervised Classification is one of the two broad types of digital classification methods used by the majority of image processing systems to analyse remote sensing images. It requires selection of an appropriate classification scheme, and then identification of training areas in the image that best represents each class. In supervised classification, the analyst identifies in the image homogenous representative areas of different land use and land cover types (information classes) of interest. These are called *training areas* or *training samples*.

The three distinct stages of Supervised Classification are training, classification and output. Training is the identification of a sample of pixels of known classes gathered from reference data, such as field visits, existing maps, reports and aerial photographs. These could be e.g. water, sand, dirt, roads, clouds etc. The key characteristics of selecting training areas are shape, location, number and uniformity.

[1]

Some typical Supervised Classification techniques are listed and described below:

1. **k-Nearest Neighbours (k-NN):** The most simple classification algorithm where the principle is to find the k closest data in the dataset to determine its class. First a distance has to be defined, which is not always easy.

2. **Maximum Likelihood:** A simple classifier based on the probability distribution. It assumes the data follows a known probability distribution (typically normal or Gaussian distr.) for each class. ML estimates the probability that a given data point belongs to each possible class and assigns it to the class with the highest likelihood.

$$C^* = \underset{C_i}{\operatorname{argmax}}\, P(C_i \mid o)$$

3. **Bayesian Classification:** Based on the bayesian rule where given an observation o, the *prior* (a priori probability), is described as:

$$P(C_i \mid o) = \frac{P(o \mid C_i)P(C_i)}{P(o)}$$

   where $P(C_i \mid o)$ is called *posterior* (a posterior probability).

   More text??

4. **Linear Classification:** The most simple classification rule, where the objective is to establish a hyperplane, *HP*, that separates the data into two groups:

$$\mathcal{H} : w_0 + \vec{w} \cdot \vec{x} = 0$$

   In a n-D space:

$$\mathcal{H} : w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$$

   Linear Classification is only possible if the data is *linearly separable*. If not, there are infinite solutions An example of Linear Classification is the Perceptron Algorithm, which bla bla

5. **Neural Networks:**
6. **Decision Trees:**
7. **Concept Lattices:**
8. **Support Vector Machine**
9. **Random Forests:**
10. **etc.**

Question 2: *Apply and compare the following algorithms:*

1. *1-NN (your programme)*
2. *Neural Network*

on 'Spain Beach' image or another of your choice

1. 1-NN

   The first step is to select the data for our classes, which we select at random in predefined areas for 'land', 'beach', 'foam' and 'ocean' classes.

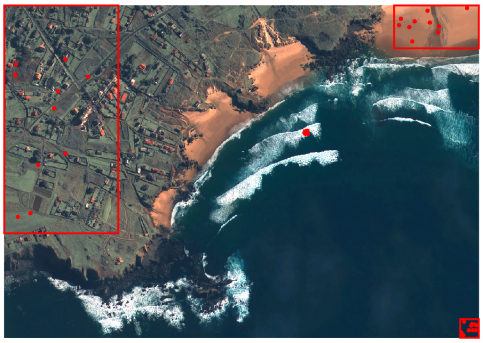


Figure 24: Randomly selected training data

The output of the algorithm gives us the beach we want to extract

(a) Raw output (b) Smoothened output

Figure 25: Extracted beach

2. Neural Network

   We select two sets of training data, which gives us two classes: 'beach' and 'not beach'.

(a) Beach (b) Not beach

Figure 26: Training sets

The target assigned to beach RGB values is 1 and 0 for the rest. Training the network goes smoothly, and its output results in the following extraction of the beach:

(a) Raw output (b) Smoothened output

Figure 27: Extracted beach

3. Comparison

   There are two noticeable differences: speed and noise level. The neural network is much more performant in terms of speed and produces an output with much less noise and better classification (like the river delta in the top right corner).

Question 3: *Search on internet one of the two the following algorithms and apply it to the same image*

1. *SVM*
2. *Ramdom Forest*

4.2 Unsupervised Classification

Question 4: *Enumerate main unsupervized classification techniques and describe them in few lines.*

Unsupervised Classification in essence reverses the process of supervised classification. It requires less input from the analyst to process the image, as the analyst does not predefined the land use and land cover types. However, the analyst can define the number of output classes required and spectral variance (such as the standard deviation, σ) within each class. The image is divided into those defined number of classes based entirely on the spectral data and with no prior knowledge of what land use and land cover types may be present in the image. [1]

- **Clustering**

  K-Means Algorithm

  Kohonen Maps

- **Regression**

- **Dimension Reduction**

  Principal Component Analysis is the most efficient method for Dimension Reduction. It is based on the variance / covariance matrix, where the main eigenvalues are the most important. Associated eigenvectors describe the reduced state space.

Question 5: *Apply the following algorithms with the dedicated objective*

1. *k-means algorithm to define the four classes automatically and speed-up the classification process*
2. *Pseudo-Inverse technique to estimate the position of a planet*
3. *PCA technique to reduce the size of an image*

4.2.1 Principal Component Analysis (PCA)

Since the images data is three, sometimes four, dimensional, it is of interest to reduce the amount of dimensions to optimize memory usage. This is done with principal component analysis. The core of the method is to find a line or plane of best fit of the data, onto which the data is then projected. This results in less dimensions being needed to store very similar information, which means PCA presents itself as a possible compression algorithm.

The steps to find the principal axes are as follows:

1. Center the data by shifting it by its mean and then calculate the covariance
2. Determine the eigenvalues and eigenvectors of the covariance matrix
3. Select the eigenvectors with the largest eigenvalue as the direction vectors of either the plane or the line

Then the data can be compressed by projecting the data onto the plane or line. In order to recover the image after compression, the operation needs to be reversed. For the projection, a simple dot product is enough. In the case of a plane, this looks like

$$D_{\text{proj}} = (D_{\text{original}} - \text{mean}) \cdot \begin{pmatrix} | & | \\ \vec{v}_1 & \vec{v}_2 \\ | & | \end{pmatrix} \tag{31}$$

where $\vec{v}_1$ and $\vec{v}_2$ are the selected eigenvectors with three components, D_{original} is the matrix containing the data of the original image, with dimensions of height x width x 3. The compressed data D_{proj} then has dimensions height x width x 2.

To decompress the image, we just need to invert this relation, which yields:

$$D_{\text{original}} = D_{\text{proj}} \cdot \begin{pmatrix} | & | \\ \vec{v}_1 & \vec{v}_2 \\ | & | \end{pmatrix}^T + \text{mean} \tag{32}$$

We apply the PCA method to the full moon image as a demonstration, and we can see that it is indeed very performant in Figure 28. The only noticable difference is that the reconstructed image is slightly darker. Since PCA does not touch the number of pixels, the resolution is unaffected by this method.

(a) Original

(b) Reconstructed

Figure 28: PCA compression

4.2.2 K-means clustering

The k-means clustering method enables us to select a number of classes we want, and the algorithm will try its best to cluster the data in this number of classes.

The algorithm is as follows:

1. Select the number of classes k
2. Initialize k centroids from the data points at random
3. Calculate the distance of each data point to each cluster
4. Assign each data point to its closest cluster
5. Compute the new centroid from the cluster set
6. If the cluster set is empty, select one new centroid at random
7. Repeat from step 3 until the centroids stop changing.

The advantage of this method is that we can determine an ideal number of classes for our purpose. For example in Figure 29 we tested multiple values of k on the image `SpainBeach.jpg` and reconstruct the image so that each pixel is colored based on the centroid of the cluster. We found that to properly extract the beach, at least five or six classes are necessary. In this

configuration, the distinction between beach and sea foam is made successfully. Once k goes above six, the image starts to become over-classified, where more details get their own classes, but then classes lose their generality.

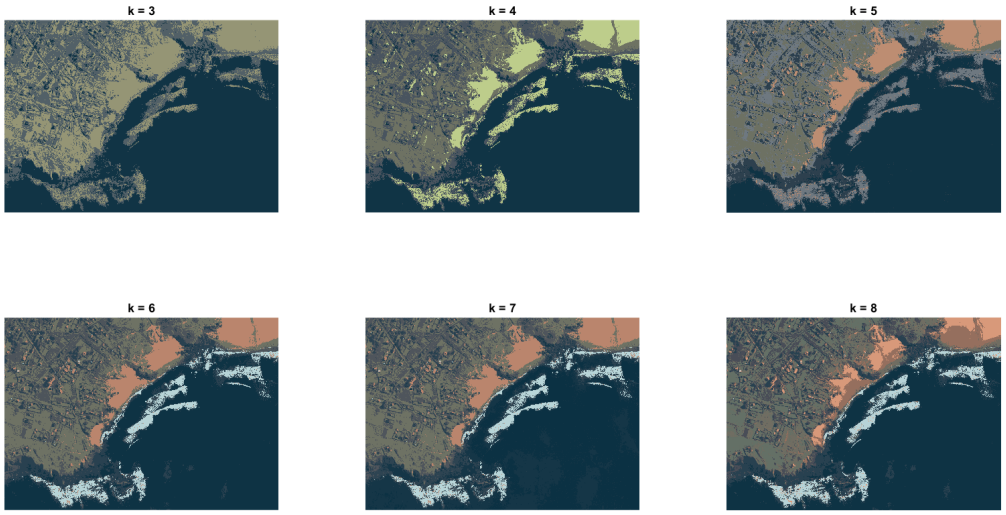


Figure 29: k-means clustering tests

References

[1] P. K. Garg. *Remote Sensing: Theory and Applications*. Mercury Learning and Information, June 17, 2024. Chap. 5 Image Preprocessing Approaches and Chap. 6 Image Classification. ISBN: 978-1-5015-2284-0. DOI: 10.1515/9781501522840. URL: https://www.degruyter.com/document/doi/10.1515/9781501522840/html?lang=en&srsltid=AfmBOoqIUf2vmBDbSUc-Wmk5Fa9PcPRoMIXFWjOu3yFk3mPC1jedww71 (visited on 02/12/2025).