Digital analysis of fingerprints

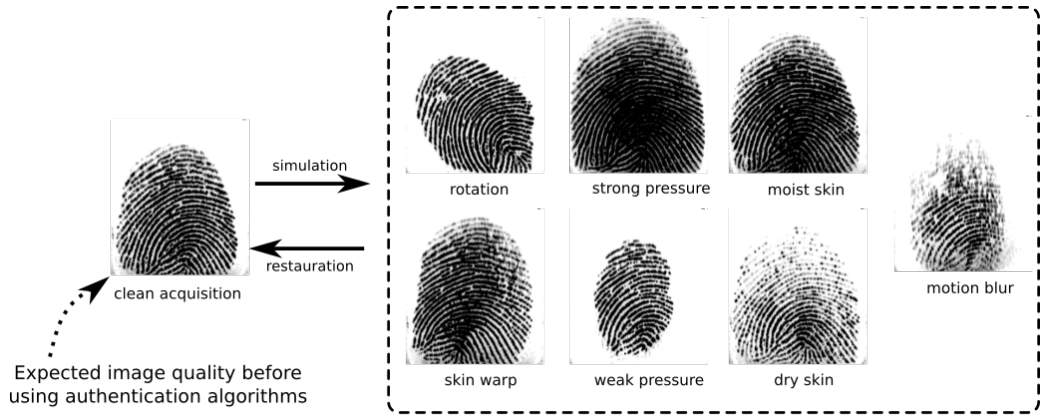
1 – Presentation

1.1 Goal

The goal of the project is to create a fingerprint image processing toolbox. You will have to find mathematical filters and models which explain how a fingerprint is captured with all the artefacts that could occured during its acquisition. An example is to try to answer the following question : given two real images of the same finger, how one goes from the image on the left to the right one ?

In the approach proposed in this project, several algorithms need to be built to cover most common artefacts (motion blur, dry/moist finger, strong/weak finger pressure, rotations).

The following figure explains the two ways the algorithms could be designed :



How fingerprints are usually captured by sensors (most common artefacts)

- The first way will be called **simulation** ; the algorithm will have to take the clean acquisition as an input, and output a distorted acquisition (rotated, dry, blurred, etc). Such algorithm could be useful for restauration algorithms testing or data augmentation for deep neural network training.
- The second way will be named **restauration** ; in this case, the input to the algorithm will be a distorted acquisition, and the output shall be as close as possible to the clean acquisition. This restauration process is usually a required step before any authentication could be performed. Its goal is to enhance the fingerprint image if the acquisition process was of poor quality (low quality image sensor, moist skin of the finger, etc).

In some cases, a single algorithm could solve both ways (simulation and restauration), or one of the way could be easier to model and implement than another. You will be able to choose restauration or simulation for any artefacts. The goal of this project is to write a scientific application that will make use of the different mathematical filters and models suggested to solve fingerprint matching.

1.2 Library development

You will be working in a team. The library needs to be written in C++ and will contains a set of mathematical functions. You may use any software tools you have learn or heard of.

1.3 Work Plan

This document will guide you to the successful completion of your project. It contains multiple exercises with questions, some of which you will be asked to answer in a final report. The assignment covers 5 topics, each having a "starter" and a "main course". You will be asked to take at least three starters and two main courses, starter 1 being mandatory ; note that the main courses are not of the same difficulty.

1.4 Grading

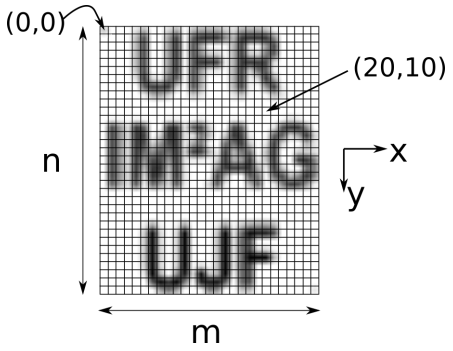
At the end of the project, you will turn a report and make a presentation of your work. The final grade will be based on - the technical writing of your report. - the presentation of your work. - the quality of the library you have develop. - your involvement in the project. - an evaluation by your pairs.

Each grading element will be detailed during the course of this project.

Read all the document at the beginning to make proper choice for libraries and class definitions.

2 – Image Loading, Saving and Pixels Manipulation

An image may be defined as a two-dimensional function $f(x, y)$ where x and y are spatial (plane) coordinates and the amplitude f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point (**Digital Image Processing**, *RC Gonzales, RE Woods*, 3rd Edition). Nowadays most of the images we encounter are digital such that x, y and the intensity values of f are all finite, discrete quantities. A digital image is composed of a finite number of elements named "pixels" (picture elements), each has one coordinate (x, y) and an intensity value in the integer range of $[0, 255]$. The size of an image (n, m) is usually defined as the number of pixels along its height n versus its width m.



The choice of a basis for pixel coordinates is arbitrary ; the convention is to choose the top left pixel as the origin $(0, 0)$ but one may define it in another way. Yet it needs to be defined properly from the beginning because any geometrical operation on the image will use pixel coordinates on this basis.

A common practice in image processing is to map intensity value originally in $[0, 255]$ to $[0, 1]$ when converted to a float ; again it is a convention which use is left to the project team. Note that it is important to map back the values to integers in $[0, 255]$ before saving any image.

2.1 Starter 1

The first round of questions is related to image loading, pixel manipulation and image saving.

- Describe your image frame for the project and the intensity range you will consider for the images.
- Look for an image processing library in C/C++ and load one image of the project. Some good options are openCV or Cimg libraries, but any other relevant solutions are welcomed. Support your choice.

At this point, you should build a class with appropriate members and methods which will help you thereafter in the project. You can assume that all images we will process are grayscale (and not RGB).

- Search for the minimum and the maximum in pixel intensity values, identify their values and data type, understand which one corresponds the black and the white. Create a method to cast all pixels values into floating numbers (most of the operations we will perform needs floating point arithmetic).
- By changing some pixel values, create white and black squares in the image as below :
- Save the result in a new image as a png file and check the integrity of the result.
- Let $s(x, y)$ be the image after performing a symmetry transform of the image $f(x, y)$ along the y axis. Express the mathematical relationship between those two images.

- Implement a method which does this operation, and save the resulting image in a new file. Do the same for a symmetry transform along the x and y diagonal axis.
- According to you, is this pixel swapping operation a rotation ? Justify your answer ; you can think of expressing the matrix which transforms the pixels coordinates during the symmetry operation and compute its determinant.

2.2 Main course 1 (simulation)

As a first order approximation and because of the spherical shape of the finger, one can simulate the pressure variation of the finger onto the sensor surface as a local spot on the image where pixels intensity is high, and decreasing as the distance to the center of the spot increases. The two images below give an example of what is observed in practice, the red square on the right image with a weak pressure of the finger is the location of the spot center, according to our definition :

The pixel operation to perform on the original image f is :

$$g(x, y) = c(x, y)f(x, y)$$

where $c(x, y) \in [0, 1]$ is a scalar coefficient - also known as a weight - and g the resulting image. The coefficient function will tend to decrease the pixel intensity of the original image as it goes towards zero, or keep it the same when close to one. It is assumed here that 0 corresponds to a white pixel, you have to choose the appropriate range of values for c according to your convention.

- Assuming first that the coefficient function is isotropic and defined as $c(r)$ with r being the euclidean distance between the center of the spot in the image and the pixel where the coefficient needs to be computed (with have $c : \mathcal{R} \to [0, 1]$), suggest few mathematical functions which monotonically decrease as $r \to \infty$, with $c(0) = 1$ and $\lim\_r \to \infty c(r) = 0$.
- Implement a function which takes as input the pixel coordinates of the spot, a set of pixels coordinates to which the coefficients should be computed, maybe some extra parameters which tune the function c, and outputs the coefficients values. You may want to test several functions.
- Test your function(s) on the image `clean_finger.png` and comment your results given what the image `weak_finger.png` give in practice.
- Looking at the images in the assignment, you may have noticed that the pixel intensity transform seems anisotropic. Recall the definition of anisotropy and explain why you can make this observation.
- Choose the function $c(r)$ which satisfied you the most in the previous question and adapt it to deal with anisotropic pixel operations (now you should consider $c(x, y)$ in its original definition).

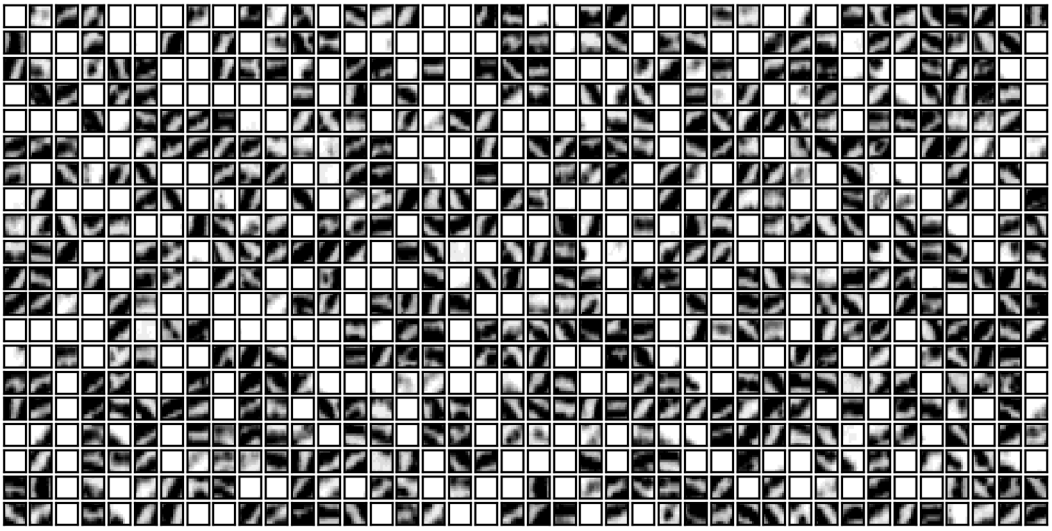
- Implement a function which computes the coefficients according to your new function c, it would be nice to be able to choose the direction of the pixel intensity transform.
- Test it on the image `clean_finger.png` and comment your results compared to the first you got.

2.3 Main course 1 (restauration)

The goal here is to recreate lost information in the surroundings of the fingerprint. You will use a simple dictionary based inpainting algorithm. The idea is to collect small patches from the fingerprint image and smartly copy paste those where the information is missing.

- Collect random patches from the weak pressure image to build a dictionary ; you could use other images, but it should ideally be the weak pressure one. You shoud select an odd number for the patch size (9 is a good guess), and crop several thousands of patches. The next figure is an illustration of what you may expect as an output for the dictionary.

dictionary patch samples - (9x9) pixels



- Create a mask of a region in the fingerprint image where you want to recreate lost information. It should be a matrix of the same size as the image with True or False values, True being pixels location to inpaint and False being valid pixels which shall not be processed.
  For testing purposes, you may consider first as a mask a small square of 30×30 pixels in the middle of the fingerprint image. Once testing is completed, you could try to create a ring shape mask around the fingerprint ends to extend its limits.
- The algorithm is as follow :
  For any True pixel in the mask with coordinates (x, y) :
  1. Crop the surrounding patch $\mathbf{p}$ in the fingerprint image at coordinates (x, y) ($\mathbf{p}$ needs to be the same size as the dictionary patches),
  2. Compute the Euclidean distance between $\mathbf{p}$ and all the patches in the dictionary (only the valid pixels of $\mathbf{p}$ - those with a False value from the mask - should be used for distance computation),
  3. Find the closest patch $\mathbf{p}_s$ from the dictionary (minimum distance to $\mathbf{p}$),
  4. Copy paste the middle pixel value of $\mathbf{p}_s$ into the fingerprint image at coordinates (x, y).

  Be aware that in this processing loop, the chosen path in the pixels coordinates to inpaint may be crucial, especially if the inpainting region has a ring shape.
- Comment on the results (artefacts, possible improvements) ; you may play with the patch size or the number of patches in the dictionnary, or get ideas from the litterature (for ex. Region Filling and Object Removal by Exemplar-Based Image Inpainting).

3 – Geometrical Warps

Image warping is the process through which the pixels coordinates (x, y) of an image are transformed according to a motion model. There exists many of those motion models : translation, rotation, similarity, affine or projective

are amongst the most well known. It is of great use in many computer vision application, for example to create a seamless panorama from several images of a scene. The interested reader can take a look at section 2 of the article **Image Alignment and Stitching : A Tutorial** (*R. Szeliski*, 2006, foundations and Trends in Computer Graphics and Vision) to get a better understanding of the image warping operation.

3.1 Starter 2

This set of questions will explain you how to perform geometrical operations on images.

- Describe the motion model which relates pixels on the left image to the one on the right :

- Find a mathematical function $w : \mathcal{R}^2 \times \mathcal{P} \to \mathcal{R}^2$ which changes a pixel coordinates (x, y) given some parameters p in $\mathcal{P}$ accordingly to this motion model :

$$(x', y') = w(x, y; p)$$

- What are the parameters of this motion model ?
- Implement a method which, given a set of pixels coordinates and model parameters, outputs the new pixels coordinates.
- We now would like to visualize the transformed image ; in other words compute the new pixels intensity values given the pixels of the original image. What is the mathematical operation dedicated to this task ?
- Provide at least 2 examples of such methods with some explanation of how it could be applied in a two dimensional case. Give their algorithm complexity for an (n, m) image.
- Implement one of those methods, it should take as input : the original pixels coordinates, their values, and the transformed pixels coordinates. The output should be the new pixels values. The difficulty of the chosen method will be accounted for the final mark.
- How can you double check your algorithm using mathematical properties of the motion model you previously identified ? Think about w^{-1}. Give some test examples on your report which validate your algorithm, and comment on any disparity you would observe.
- Test your algorithm on the image `clean_finger.png` and try to parametrize at best your function $w(x, y; p)$ so that your output image is as close as possible to `warp1_finger.png`

3.2 Main Course 2

The finger skin elasticity, though useful for precise dexterous manipulation, is the consequence of many fails of fingerprint authentication systems. On the two following acquisitions of the same finger, the right acquisition is the result of a small rotation while the skin was under pressure during the acquisition :

Looking at the red square, one should notice that the black ridges of the fingerprint look squeezed in some part of the region and spaced out in the other part. The rest of the fingerprint image "almost" looks the same, such that this warp can be considered as local. Therefore, the warp function may look something like this :

$$w(x, y) = \big(x + \delta_x(x, y), y + \delta_y(x, y)\big)$$

with $\delta_x : \mathcal{R}^2 \to \mathcal{R}$ and $\delta_y : \mathcal{R}^2 \to \mathcal{R}$ being two functions which influence decreases - they should go towards zero - as the pixel coordinate (x, y) is away form the defined center of the warp. Note that piecewise warp function could be a solution but may create a discontinuity on the boundaries.

- Define a warp function which transform pixels coordinates according to this skin elasticity property. It should mimic this "squeezed and spaced out" observation made on real images. At least two parameters need to be considered for this warp : its location on the image (as a local warp, not all the image should be affected) and its strength. Other parameters could be of interest, like its orientation ; think that once the finger is under pressure, you can still translate or rotate it a bit.
- Implement this function and test it with different values of parameters on the image `clean_finger.png` to see how close to the reality (the image `warp2_finger.png`) your warp model can be.

4 – Linear Filtering

In image processing, linear filters are of great use for image denoising, resampling or features extraction. The convolution operation usually assumes spatially invariant filters, and is equivalent to linear filtering the image. The interested reader could look at **The design and use of steerable filters** (*WT Freeman, EH Adelson*, 1991, transactions on Pattern Analysis and Machine Intelligence) for an enlightening use of linear filters in the context of computer vision.

4.1 Starter 3

- Give a mathematical definition of the discrete convolution operation in 2 dimension. Explain with an illustration how it should work on a 2D grid.
- Implement a method which, given two matrices of arbitrary size, outputs the result of the convolution process between those two matrices ; the resulting matrix should be the same size as the first one in the function call. Find a suitable strategy to deal with image boundaries and explain it.
- What is the algorithm complexity of the convolution operation with respect to the number of pixels in both matrices ? Consider a naive implementation of the convolution.
- Explain the relationship between the Fourier transform and the convolution.
- Implement the convolution operation using the Fast Fourier Transform (use a library to compute 2D Fourier transforms like fftw if the image library you chose does not implement it), and double check your results comparing both convolution algorithms on an image of the project. Display the comparison in your report and try to explain any disparity.
- What is the algorithm complexity using the FFT to do the convolution operation ? Again consider a naive implementation of the FFT.
- Say we have a kernel of fixed size $(15, 15)$ pixels with which we want to convolve to an (n, m) image, when is it more efficient to use one method instead of the other with respect to the number of pixels in the image ?
- As a first order approximation, a motion blurred image f_b can be represented as the result of the convolution of the original (sharp) image f with a blurring kernel k :

  $$f_b(x, y) = k * f(x, y)$$

  Can you suggest a blurring kernel $k(x, y)$ of maximum size $(15, 15)$ to which the image `clean_finger.png` is convolved to obtain the blurred right image `blurred_finger.png` ?

The kernel elements should sum up to 1 so that image energy is preserved. Don't expect to get good results against the real blurred image, but you should be able to simulate blur caused by a translation of the finger. You can find some hints about blur kernels in : **Removing camera shake from a single photograph** (*R. Fergus, B. Singh, A. Hertzmann, ST Roweis, WT Freeman*, 2006, SIGGRAPH).

4.2 Main Course 3 (simulation)

You probably end up with poor results while trying to simulate the motion blur for the last question. If you take a closer look at the image `blurred_finger.png`, you may notice that only the fingerprint surroundings look blurry and faded, while its center is still sharp. This seems consistent with the spherical shape of the finger and the properties of the skin which make the finger stick to the surface. Thus the kernel blur should not be spatially invariant nor preserve the energy in the image :

$$f_b(x, y) = k_{x,y} * f(x, y)$$

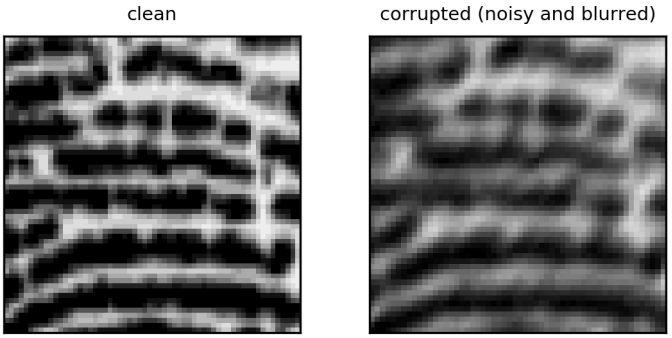
where the subindex (x, y) for $k_{x,y}$ expresses the dependency of the kernel with respect to its location on the image.

- Do you think you can use the FFT to implement this operation ?
- Given the center (x_c, y_c) of the fingerprint (we will assume that it is the point where the pressure is the highest), define a kernel function $k_{x,y}$ whose energy decreases with its distance from the center point (x_c, y_c). The energy could decrease linearly or quadratically, but the sum of the kernel's elements has to remain in $[0, 1]$.
- Implement this function and test it on the image `clean_finger.png` ; you can consider first the identity kernel.
- Given the same center (x_c, y_c), define a kernel function which increases the blur on the image as its distance to the center point increases. You could first define a kernel for the fingerprint boundaries, and think of a way to make it evolve to the identity kernel as it distance to the center decreases.
- Implement this function and test it on the image `clean_finger.png` ;
  try to get closer to the image `blurred_finger.png` in tuning your algorithm.
- Related to restauration, given the kernel and the blurred image, do you think that the convolution operation could be reversed to deblur the image ? Give some explanation to support your answer, feel free to google the problem using appropriate keywords and give your sources (if any).

Note that in real life, the kernel is usually unknown and need to be estimated ; a really nice solution is suggested in **Removing camera shake from a single photograph** (*R. Fergus, B. Singh, A. Hertzmann, ST Roweis, WT Freeman*, 2006, SIGGRAPH).

4.3 Main Course 3 (restauration)

The goal of this restauration task will be to deblur the corrupted fingerprint acquisition f_b (in the right of the figure below) to get as close as possible to the original acquisition f (the clean one in the left) :



The blurring kernel is :

$$k = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{7} \\ 0 & 0 & 0 & \frac{1}{7} & 0 \\ 0 & 0 & \frac{1}{7} & 0 & 0 \\ 0 & \frac{1}{7} & \frac{1}{7} & 0 & 0 \\ \frac{1}{7} & 0 & \frac{1}{7} & 0 & 0 \end{bmatrix}$$

In this exercice, the clean image is the file `clean_blur_ex.png` of pixel size (60×60) (this should help you to evalute your debluring operation) and the corrupted one is `corrupted_blur_ex.png` with size (56×56) (this shall be the input to your deblurring function, as well as the kernel k).

- Convolution as expressed in starter 3 is a linear operation, which means that it could be rewritten as a matrix operation of the form :

$$\mathbf{f}_b = K\mathbf{f} \tag{1}$$

  where $\mathbf{f}$ is the unknown vectorized clean acquisition of length 3600, $\mathbf{f}_b$ is the vectorized corrupted acquisition of length 3136, and K is the blurring matrix operator of size $(3600, 3136)$. Write down the least square estimate for $\mathbf{f}$, and explain why solving this system of linear equations is ill posed.
- Build the K matrix using the coefficients in k. You need to understand precisely the 2d convolution operation to do so, you should use the clean acquisition for debuging (as long as you do not get an image really close to the corrupted one when you multiply K with $\mathbf{f}$, your K is likely to be wrong). You may have to use a sparse matrix representation for K, and you will need a linear systems solver (the library ALGLIB is an example of sparse matrix with linear algebra operation support).
- Solve the system of equation as a least square problem with a linear systems solver. You may :
  - use a linear solver which is able to solve directely 1 as a least square problem,
  - reformulate the problem so that it becomes well-posed (explain why) and could be solved with any linear algebra solver :

$$\begin{bmatrix} \mathbf{f}_b \\ \mathbf{0}_{3600} \end{bmatrix} = \begin{bmatrix} K \\ \lambda I_{3600} \end{bmatrix} \mathbf{f} \tag{2}$$

    where I_{3600} is the identity matrix with 3600 ones on its diagonal, λ is a positive scalar value and $\mathbf{0}_{3600}$ is the vector with 3600 zeros in it. You could set $\lambda = 1$ as a start, and try to play with its value to see the output of the deconvolution operation.
- Comment on your results. If you use a least square linear solver, change the value of the damping coefficient to see its influence. If you choose to reformulate the system of equations, see how different values for λ change the result. Try to understand how it could be improved (for ex. Image and depth from a conventional camera with a coded aperture).

5 – Morphological Filtering

Morphological filtering is a theory developed in the 1960s for the analysis and processing of discrete images (**An overview of morphological filtering**, *J. Serra, L. Vincent*,1992, Circuits, Systems and Signal Processing). It is a collection of non-linear operations related to the morphology of features in an image, such as boundaries, skeletons, ... Similarly to the convolution operation, the image is scanned with a mask (called kernel previously) of predefined shape, this shape defines the region of interest or neighbourhood around a pixel ; it can be a square, circle, or diamond with the origin at the center is used. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images, but those techniques can be extended to grayscale images as well. Their main goal is to remove imperfections on the images which could not be easily treated with linear or median filters, but other applications exists as we will see.

5.1 Starter 4

- Perform a non exhaustive bibliography on morphological filters to understand their action on images. Explain with an illustration how the dilatation and erosion operations work a 2D grid given a mask (aka structuring element). Cite your sources (articles, web links, ...).
- According to you, what morphological operation would you perform to get the image `moist_finger.png` given the image `clean_finger.png` ? Justify.
- Same question with the image on the right `dry_finger.png`.
- Because morphological filters are more easily applied on binary images, provide a mathematical explanation of the binarization operation of an image and implement such a method.

- explain how to properly choose the threshold parameter for the binarization operation. If you cannot find a clever way to do this, we recommend you to read **A threshold selection method from gray-level histograms**, *N. Otsu*, 1975, Automatica.
- Implement erosion and dilatation operations for binary images. You should test your algorithm on the image `clean_finger.png` with several structural elements of different sizes till you approximate `moist_finger.png` and `dry_finger.png`. Detail your results on the report.
- Does the operation of successive erosion and dilation act the same as dilation and erosion ? Explain your answer with examples.

5.2 Main Course 4

The goal here is to extend the morphological operations to grayscale images, and adapt it to the fingerprint case. The input/output of your algorithm will now be grayscale images, however the structuring element will still be binary.

- Search documentation to understand how morphological filters can be adapt to grayscale images. Explain on your report and provide your sources.
- Implement the method to process grayscale images ; check on the image `clean_finger.png` if your result is correct.
- As noticed before, the spherical shape of the finger makes the processing relevant only locally. Suggest a way to adapt your algorithm to this observation either for the `moist_finger.png` and `dry_finger.png`.$; ideally, the transition should be seamless so that the processing does not create new non natural artefacts on the image. Implement your solution and show the improvement on the images.

6 – Optimization for Image Registration

6.1 Starter 5

In several fields of computer science, optimization is defined as a mathematical process through which a function is minimized, or maximized, with respect to a set of predefined parameters. This function is often called objective, cost or loss function. In computer vision, a popular lost function is the sum of squared errors between the pixels of two images :

$$l(p) = \sum_{x,y} (f(x,y) - g(w(x,y;p)))^2,$$

where f and g are two images, the latter g being the one we want to align to f, w is a warp function as defined in section 3 of the document and p its parameters. The sum is taken over all pixels of images. The value of this loss function depends on p, and the goal here is to attain an optimal value for p such that $l(p)$ is minimized. This is commonly written as :

$$\hat{p} = \underset{p}{\text{argmin}}\ l(p),$$

and $\hat{p}$ is the sought values of the parameters set.

To complete this part of the project, you will need an image interpolation function ; you may use the one you developed in section 3 or any function available in OpenCV.

- Consider first that f is `tx_finger.png`, and the warp function is a translation along the x axis such that there is only a single translation parameter p_x to estimate. Assuming that p_x is a discrete value in the pixel coordinate grid, propose a greedy strategy to find the optimal value $\hat{p}_x$. Give the pseudo code for this strategy and draw the loss function with respect to p_x.
- Let f be the image `txy_finger.png`, and the warp function w be a translation along the x and the y axis. Extend your previous greedy strategy to estimate p_x and p_y, still assuming that both parameters have discrete values in the pixel coordinate grid. Draw the loss function surface with respect to p_x and p_y, and comment on the ripples that you may observe.
- Let the loss function now being defined as :

$$l(p) = \frac{\sum_{x,y}(f(x,y) - \bar{f})(g(w(x,y;p)) - \bar{g_w})}{\sqrt{\sum_{x,y}(f(x,y) - \bar{f})^2}\sqrt{\sum_{x,y}(g(x,y) - \bar{g_w})^2}}.$$

Using the same optimization algorithm as before, estimate both p_x and p_y. Draw the new loss function surface and compare it to the previous one, comment on any difference. Display the absolute error image, which is the per pixel absolute difference between f and $g(w(x,y;p^*))$ where p^* are the optimal parameters according to the loss function minimization. Comment on the resulting error image.

- You may notice that both images are not perfectly registered, the main reason being that translation parameters are not integer values. Suggest a strategy to look for subpixel translations, and display the absolute error image after having optimized the translation parameters.
- Now choose f to be the image `rtxy_finger.png` where the warp function is a combination of a rotation and a translation along x and y. Adapt your greedy search strategy to estimate the rotation and the translation and display the absolute error image after image registration. How may you speed up the whole optimization process (from pixel to subpixel search) ? Evaluate the computational time gain between the greedy search strategy and your optimized solution.
- You will now implement a non differentiable coordinate descent (or ascent with respect to your loss function) algorithm to minimize (or maximize) the loss function. The strategy is as follow : starting from an initial p_0, for each entry p in the parameters set, evaluate the loss function when decreasing and increasing the entry by α_p percent. If the change result in a better loss, the change is accepted and the parameter α_p is increased by ten percent, otherwise α_p is decreased by fifty percent. You can choose $\alpha_p = 10\%$ as a start, but you may test several values. Plot the convergence rate of your algorithm, investigate cases for which it does not converge. Is this algorithm suited for global (discrete pixel) or local (subpixel) search of warps ?

6.2 Main Course 5

- We will now focus on gradient descent strategies. Let's consider the sum of squared errors loss function and the translation case where $w(x,y;p)$ is given by :

$$w(x,y;p) = \begin{cases} w_x = x + p_x \\ w_y = y + p_y \end{cases}$$

Compute the partial derivatives of $w(x,y;p)$ with respect to p_x and p_y. Assume that :

$$\frac{\partial g(w(x,y;p))}{\partial p} = \begin{cases} \frac{\partial g_w}{\partial w_x}\frac{\partial w_x}{\partial x} \\ \frac{\partial g_w}{\partial w_y}\frac{\partial w_y}{\partial y} \end{cases}$$

where $\partial g_w/\partial w_x$ and $\partial g_w/\partial w_y$ are given using finite differences estimates, compute the partial derivatives of $l(p)$ with respect to p_x and p_y which will be noted respectively ∇_{p_x} and ∇_{p_y}.

- implement the following gradient descent scheme to estimate translation of `txy_small_finger.png` :

1. initialize α, ϵ, p_x and p_y,
2. warp the image g according to p_x and p_y,
3. compute ∇_{p_x} and ∇_{p_y},
4. update the warp parameters as $p_x \leftarrow p_x - \alpha\nabla_{p_x}$ and $p_y \leftarrow p_y - \alpha\nabla_{p_y}$
5. exit the loop if ∇_{p_x} and ∇_{p_y} are both smaller than ϵ, else continue to step 2.

advice : α is known as the convergence rate, and should be set properly. Start with a small value and make sure that the loss function is decreasing at each loop iteration ; display it at each iteration for better monitoring. If the loss function value is overshooting, then α is likely to be too high.

- how could you use this algorithm for a broader warp like `txy_finger.png` ?
- Consider now the rotation and translation case where the warp function is given by :

$$w(x, y; p) = \begin{cases} w_x = \cos(\theta)x + \sin(\theta)y + p_x \\ w_y = -\sin(\theta)x + \cos(\theta)y + p_y \end{cases}$$

the parameters are (θ, p_x, p_y). Adapt the gradient descent with new formulas to estimate those warp parameters.