Affine and non-linear image warping based on landmarks

Stanley Durrleman

October 19, 2010

1 Introduction

The purpose of this note is to show how one can align two images knowing pairs of corresponding landmarks in each image, which are typically located at the contours of the images or at points of high curvature like corners of objects, for instance. This note goes along with a MATLAB demo. The demo folder contains:

- a `MATLAB` script called `imageWarp.m`
- two images from the Caltech 101 database
- two `.mat` files (`PsrcAnchors.mat` and `PtarAnchors.mat`) containing the coordinates of the landmarks in the source and the target image respectively.

You can run the first part of the demo which corresponds to the introduction part (i.e. until the first two figures are displayed). The first figure shows the two original images as well as the superimposition of the two. To drive the registration (also called the warping) of the source image to the target image, we select pairs of corresponding landmarks in each image, which are shown in the second figure of the demo.

2 Affine registration using triplets of points

The first idea to align images is to use affine transforms, as they are the most general form of linear transform. A generic affine transform is determined by a 2-by-2 matrix A and a 2-by-1 vector t. It maps the point $X = (X^x, X^y)^t$ (when X denotes a vector, its coordinates are written as X^x and X^y with subscript x and y) to the point $AX + t$.

The matrix A has 4 components and the vector t has 2 components. Therefore, an affine transform has 6 degrees of freedom and hence it is entirely determined by 3 pairs of landmarks. The purpose of this section is to show how to estimate this affine transform (A, t) from the positions of three source points

X_1, X_2, X_3 (in the source image domain) and their corresponding target points Y_1, Y_2, Y_3 (in the target image domain).

2.1 Determination of the affine transform

The affine transform needs to satisfy the set of 3 constraints:

$$\begin{cases} AX_1 + t = Y_1 \\ AX_2 + t = Y_2 \\ AX_3 + t = Y_3 \end{cases} \tag{2.1}$$

If one knows the linear part A, then the translation vector t can be determined easily by any of these equations as $t = Y_1 - AX_1 = Y_2 - AX_2 = Y_3 - AX_3$.

The idea is now to focus on the linear part A and to get rid of the translation by subtracting the second and third equation to the first one. This gives:

$$\begin{array}{l} A(X_1 - X_2) = Y_1 - Y_2 \\ A(X_1 - X_3) = Y_1 - Y_3 \end{array} \tag{2.2}$$

A is a 2-by-2 matrix and these last two equations show how the two vectors $X_1 - X_2$ and $X_1 - X_3$ are mapped to the vectors $Y_1 - Y_2$ and $Y_1 - Y_3$. If the vectors $X_1 - X_2$ and $X_1 - X_3$ are not parallel, they define a new basis of the 2D space. These equations show precisely that the matrix A, when the input is written in this new basis, is written as:

$$\tilde{A} = \left[\begin{array}{cc} Y_1 - Y_2, & Y_1 - Y_3 \end{array} \right] = \left[\begin{array}{cc} Y_1^x - Y_2^x & Y_1^x - Y_3^x \\ Y_1^y - Y_2^y & Y_1^y - Y_3^y \end{array} \right] \tag{2.3}$$

To write this matrix in the usual canonical basis ($e_1 = (1,0)^t, e_2 = (0,1)^t$), one needs to use the change of basis which maps the canonical basis (e_1, e_2) to the new basis $(X_1 - X_2, X_1 - X_3)$. This change of basis matrix is given by definition as:

$$P = \left[\begin{array}{cc} X_1 - X_2, & X_1 - X_3 \end{array} \right] = \left[\begin{array}{cc} X_1^x - X_2^x & X_1^x - X_3^x \\ X_1^y - X_2^y & X_1^y - X_3^y \end{array} \right] \tag{2.4}$$

Now, we notice that the equations (2.2) can be written as:

$$\begin{cases} APe_1 = Y_1 - Y_2 \\ APe_2 = Y_1 - Y_3 \end{cases} \tag{2.5}$$

since $X_1 - X_2 = Pe_1$ (i.e. the first column of P) and $X1 - X_3 = Pe_2$ (i.e. the second column of P). This shows that the matrix AP is exactly $\tilde{A}$: $AP = \tilde{A}$, so that the linear part of the affine transform is given as:

$$A = \tilde{A}P^{-1} = \left[\begin{array}{cc} Y_1^x - Y_2^x & Y_1^x - Y_3^x \\ Y_1^y - Y_2^y & Y_1^y - Y_3^y \end{array} \right] \left[\begin{array}{cc} X_1^x - X_2^x & X_1^x - X_3^x \\ X_1^y - X_2^y & X_1^y - X_3^y \end{array} \right]^{-1} \tag{2.6}$$

which is expressed only in terms of the coordinates of the data points.

The solution of our problem is therefore given by:

$$\boxed{\begin{cases} A = \tilde{A}P^{-1} \\ t = Y_1 - AX_1 \end{cases}} \tag{2.7}$$

Question: is P always invertible? P is of rank 2 if and only if the two vectors $X_1 - X_2$ and $X_1 - X_3$ are not collinear, which means that the 3 points X_1, X_2 and X_3 are not aligned. If they are, the problem is under-determined: one only knows how to transform the line (X_1, X_2, X_3) (a subset of dimension 1). Then, there are two cases: if the image points (Y_1, Y_2, Y_3) are not aligned, then there is no solution. If they are aligned, then there is an infinite number of solution. However, these situations almost never happen in practice, and we can assume that P is invertible. Numeric-wise, this also shows that the inversion of P is ill-posed in the triangle build on the vertices X_1, X_2, X_3 is too flat.

2.2 Image warping based on affine transform

Once the affine transform has been estimated based on the position of 3 pairs of landmarks, it can be used to deform the underlying image. Although the transform has been estimated from only 3 pairs of points, it defines a transformation of the whole $2D$ domain: any point X in the source image domain is mapped into the point $AX + t$ in the target image domain.

Let I be the source image and J the target image. Let $\tilde{I}$ be the transformed source image, which is defined in the target image domain. Let Y be the position of a generic pixel in the target image domain. The grey level of the transformed image $\tilde{I}$ at pixel Y is given by the grey level of the source image I at the pixel X such that $AX + t = Y$. We have therefore:

$$\tilde{I}(Y) = I(A^{-1}(Y - t)) \tag{2.8}$$

In the most general case, the position $A^{-1}(Y - t)$ does not correspond to the coordinate of a pixel in the source image domain. In this case, $I(A^{-1}(Y - t))$ stands for the grey-value of the pixel which is the closest to the arbitrary position $A^{-1}(Y-t)$. A smoother transformed image can be obtained by using an interpolation between the gray-levels at the four neighboring pixels surrounding the position $A^{-1}(Y - t)$ instead of using a closest neighbor strategy.

2.3 Demo

Now, you can run the demo corresponding to this section. Figures 3,4 and 5 are displayed. Figure 3 shows the original images with the 3 selected landmarks. Figure 4 shows how the estimated affine transform maps the points of a regular lattice. Figure 5 shows the image warping based on the estimated affine transform.

Questions:

- Change the selected landmarks. How does the choice of the landmarks impact the image warping?

- Adapt the code to implement the interpolation scheme for building the transformed image instead of the closest neighbor strategy. Where does the reconstructed image changes?

- Compute the matching errors between the transported source landmarks and their corresponding target landmarks. This error should be zero for the landmarks used to estimate the affine transform. What happens for the other landmarks? What is the total error? Can you find any triplets which makes this error to vanish? How close to zero can you get?

3 Affine registration in a least square sense using more than 3 landmarks

As you may have noticed from the previous experiments, the choice of landmarks is crucial for the determination of the affine transform and hence for the image alignment. And you may not have found any triplets of landmarks which allows you to satisfactorily align the two images globally.

The idea is then to use the full set of landmarks to find a better affine registration between both images. The problem is that the affine transform has only 6 degrees of freedom, which means that in the most general case, there is no solution (A, t) to the set of equations:

$$\begin{cases} AX_1 + t = Y_1 \\ AX_2 + t = Y_2 \\ \qquad \vdots \\ AX_N + t = Y_N \end{cases} \tag{3.1}$$

for $N > 3$ pairs of landmarks. This means that for any matrix A and vector t, the total error between all the transported source points $AX_i + t$ and the target point Y_i cannot be zero. For want of anything better, we want to find the affine transform (A, t) which minimizes this total error. Denoting the transformed source points as $T(X_i) = AX_i + t$, the criterion to be minimized is:

$$f(A, t) = \sum_{i=1}^{N} \|T(X_i) - Y_i\|^2 = \sum_{i=1}^{N} \|AX_i + t - Y_i\|^2 \tag{3.2}$$

Our purpose is to find the minimum of this function over all possible 2-by-2 matrices A and all possible 2D-vectors t. For that, we will find the values of A and t for which the gradient (i.e. the differential) of f vanishes. Remark that the zero value of the gradient does not necessarily mean that f reaches a minimum at this point: it can be any extremal points like a maximum or a horse saddle point for instance. In our case, I let you prove as an exercise that the criterion is convex, which guarantees that there is only one extremal point and that this extremum is a minimum.

Here the criterion f depends on 6 variables: the 4 coordinates of the matrix A and the 2 coordinates of the vector t. To find the gradient of f, one could compute the 6 partial derivatives with respect to each of these variables. This is particularly long and annoying (you can do it as an exercise, and check that we find the same result...). Here, we will follow a more intrinsic approach, which is based directly on the definition of the gradient. The general sketch is the following. Let $X = (X^1, \dots, X^n)$ a generic set of n variables (i.e. a n-dimensional vector) and $f(x)$ a generic function of these n variables. Then, if we can write the variation of the function $f(X + \delta X)$ for a small variation of the variables δX (which is also a n-dimensional vector) as:

$$f(X + \delta X) = f(X) + B^t \delta X + \varepsilon, \tag{3.3}$$

where B is an n-dimensional vector and $\varepsilon / \|\delta X\|$ tends to zero as $\|\delta X\|$ tends to zero, then, by definition, B is the gradient of f at point X and is denoted: $B = \nabla f(X)$.

Now, let's apply this general sketch to our problem. At first, we will focus on the translation vector t, assuming that the matrix A is fixed. In this case, f is a function of two variables $t = (t^x, t^y)$. The variations of $f(t)$ with respect of the variations δt (a 2D-vector) can be written as:

$$\begin{aligned} f(t + \delta t) &= \sum_{i=1}^{N} \|AX_i + t + \delta t - Y_i\|^2 \\ &= \sum_{i=1}^{N} \|(AX_i + t - Y_i) + (\delta t)\|^2 \\ &= \sum_{i=1}^{N} \left(\|(AX_i + t - Y_i)\|^2 + 2(AX_i + t - Y_i)^t (\delta t) + \|\delta t\|^2 \right) \\ &= \underbrace{\sum_{i=1}^{N} \|(AX_i + t - Y_i)\|^2}_{= f(t)} + 2 \sum_{i=1}^{N} (AX_i + t - Y_i)^t (\delta t) + N \|\delta t\|^2 \end{aligned}, \tag{3.4}$$

where the third equality comes from the fact that $\|X + Y\|^2 = (X + Y)^t (X + Y) = \|X\|^2 + 2X^t Y + \|Y\|^2$.

Since $N \|\delta t\|^2 / \|\delta t\| = N \|\delta t\|$ tends to zero as $\|\delta t\|$ tends to zero, then, by definition the gradient of f with respect to the translation vector t is:

$$\frac{1}{2} \nabla f(t) = \sum_{i=1}^{N} (AX_i + t - Y_i) = \sum_{i=1}^{N} (AX_i - Y_i) + Nt \tag{3.5}$$

Therefore, the gradient vanishes when $\sum_{i=1}^{N} (AX_i - Y_i) + Nt = 0$, which means: $t = \frac{1}{N} \sum_{i=1}^{N} (Y_i - AX_i)$. Denoting $\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i$ the center of mass of the source points and $\bar{Y} = \frac{1}{N} \sum_{i=1}^{N} Y_i$ the center of mass of the target point,

then the translation vector t which enables to achieve the minimum of j is given as:

$$\boxed{t = \bar{Y} - A\bar{X}} \tag{3.6}$$

This gives the optimal translation vector, provided that we know the optimal matrix A. This optimal translation vector has a simple interpretation: it is the difference between the center of mass of the target points and that of the transported source points.

Now, we need to determine the optimal matrix A which achieves the minimum of f. First, we plug the expression of the optimal translation vector into the criterion, so that this criterion depends only on the matrix A. This gives:

$$\begin{aligned} f(A) &= \sum_{i=1}^{N} \left\| AX_i + (A\bar{X} - \bar{Y}) - Y_i \right\|^2 \\ &= \sum_{i=1}^{N} \left\| A(X_i - \bar{X}) - (Y_i - \bar{Y}) \right\|^2 \\ &= \sum_{i=1}^{N} \left\| A\tilde{X}_i - \tilde{Y}_i \right\|^2 \end{aligned} \tag{3.7}$$

where we denote $\tilde{X}_i = X_i - \bar{X}$ the centered source points and $\tilde{Y}_i = Y_i - \bar{Y}$ the centered target points.

To differentiate f with respect to A, we will follow the same sketch as for the translation vector. The only thing to take care of is that the matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is now considered as the 4-dimensional vector $(a, b, c, d)^t$ (the vector of the 4 variables on which f depends). The inner-product between two matrices $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $A' = \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix}$ is then given as $(a, b, c, d)^t(a', b', c', d')$. I let you verify that this is equal to $\mathrm{Tr}(A^t A')$: the trace of the product between the transpose of A and A' ($A^t A'$ is a 2-by-2 matrix). This is called in the literature the "Frobenius norm" on matrices.

Now, let's write a variation of the criterion $f(A)$ with respect to a variation of the matrix $A + \delta A$, where δA is another 2-by-2 matrix. This leads to:

$$\begin{aligned} f(A + \delta A) &= \sum_{i=1}^{N} \left\| A\tilde{X}_i + (\delta A)\tilde{X}_i - \tilde{Y}_i \right\|^2 \\ &= \underbrace{\sum_{i=1}^{N} \left\| A\tilde{X}_i - \tilde{Y}_i \right\|^2}_{=f(A)} + 2\sum_{i=1}^{N} (A\tilde{X}_i - \tilde{Y}_i)^t (\delta A)\tilde{X}_i + \sum_{i=1}^{N} \left\| (\delta A)\tilde{X}_i \right\|^2 \end{aligned} \tag{3.8}$$

The third term tends to zero as $\|\delta A\|$ tends to zero. Therefore, by definition of the gradient, the inner-product between the gradient $\nabla f(A)$ (a 2-by-2 matrix) and δA (another 2-by-2 matrix) is equal to: $\mathrm{Tr}\left(\nabla f(A)^t(\delta A)\right) = 2\sum_{i=1}^{N}(A\tilde{X}_i -$

$\tilde{Y}_i)^t \delta A \tilde{X}_i$. We have now to extract the gradient from this expression. For that, we notice that this last term is a scalar and, as such, it is equal to its trace. This leads to:

$$\begin{aligned}(A\tilde{X}_i - \tilde{Y}_i)^t(\delta A)\tilde{X}_i &= \mathrm{Tr}\left((A\tilde{X}_i - \tilde{Y}_i)^t(\delta A)\tilde{X}_i\right) \\ &= \mathrm{Tr}\left(X_i\left(A\tilde{X}_i - \tilde{Y}_i\right)^t \delta A\right) \\ &= \mathrm{Tr}\left(\left((A\tilde{X}_i - \tilde{Y}_i)\tilde{X}_i^t\right)^t \delta A\right)\end{aligned} \tag{3.9}$$

where the second equality comes from the fact that $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ for any generic matrices A and B. We have therefore:

$$\mathrm{Tr}\left(\nabla f(A)^t \delta A\right) = \mathrm{Tr}\left(\left(2\sum_{i=1}^{N}(A\tilde{X}_i - \tilde{Y}_i)\tilde{X}_i^t\right)^t \delta A\right). \tag{3.10}$$

Since this equality holds for any δA, we get:

$$\nabla f(A) = 2\sum_{i=1}^{N}(A\tilde{X}_i - \tilde{Y}_i)\tilde{X}_i^t \tag{3.11}$$

The optimal matrix A is such that this gradient vanishes. This gives: $A\sum_{i=1}^{N}\tilde{X}_i\tilde{X}_i^t = \sum_{i=1}^{N}\tilde{Y}_i\tilde{X}_i^t$. Therefore, the optimal affine transform is given by the matrix A and translation vector t:

$$\boxed{\begin{cases} A = \left(\sum_{i=1}^{N}\tilde{Y}_i\tilde{X}_i^t\right)\left(\sum_{i=1}^{N}\tilde{X}_i\tilde{X}_i^t\right)^{-1} \\ t = \bar{Y} - A\bar{X} \end{cases}} \tag{3.12}$$

For $\tilde{X}_i$ and $\tilde{Y}_i$ 2D-vectors, $\tilde{Y}_i\tilde{X}_i^t$ and $\tilde{X}_i\tilde{X}_i^t$ are 2-by-2 matrices.

Question: is the matrix $\sum_{i=1}^{N}\tilde{X}_i\tilde{X}_i^t$ always invertible? First, we notice that this matrix is symmetric (i.e. it equals its transpose). This means that it has 2 eigenvalues and 2 eigenvectors. Let v and eigenvector (with unit norm) associated to the eigenvalue λ, then by definition of the eigenvectors, $v^t(\sum_{i=1}^{N}\tilde{X}_i\tilde{X}_i^t)v = \lambda$. The first term can be also written as: $\sum_{i=1}^{N} v^t\tilde{X}_i\tilde{X}_i^t v = \sum_{i=1}^{N}(v^t\tilde{X}_i)^2$. This shows that the eigenvalues are all non-negative. The matrix is invertible if and only if they are all non-zeros. If $\lambda = 0$, then for every i, $v^t\tilde{X}_i = 0$ and the eigenvector is perpendicular to every X_i. In 2D, if the points X_i are not all aligned on a straight line, then there is no such vectors except 0. This shows that λ cannot vanish and that the matrix is invertible. In dimension d, the matrix is invertible if the points X_i "fill up" the space, in the sense that the points cannot be included in a sub-space of smaller dimension than the ambient space of dimension d.

3.1 Demo

You can run the part of the demo, which corresponds to this section. It displays the Figure 6, which shows the position of the source points after registration. Notice that they do not match perfectly the target point. However, the affine transform has been estimated such that it minimizes the total matching error between the transported source points and the target points. Figure 7 shows the image warping based on this estimated affine transform. What do you think about this alignment compared to the previous ones? What happens if you select fewer points? In which regions the mis-alignment is the more pregnant? Is there any way to correct it?

4 Non-linear registration using radial basis function

The affine transform has two intrinsic limitations: it is global and has few numbers of degrees of freedom. It is global: this means that the information some pairs of landmarks in a small part of the image domain will impact the transformation of every point in the image, even if they are far away from the landmarks. It has only 6 degrees of freedom, which means that the possible transformations are very constrained by nature. Such deformation cannot accommodate for "non-linear deformations". For instance, the transversal part of the anchor in the source image of the demo, which has the shape of a chevron, cannot be transformed into a straight line, as it looks like in the target image.

To workaround this issue, the idea is to use more local deformation, to deform local patches of the image. Then, the global image warping is obtained by a superposition of such deformations. Depending on the number of patches, one can adapt the number of degrees of freedom of the deformation to our needs.

4.1 Radial basis functions

In this subsection, we will focus on the case of a single pair of landmarks. The idea of image warping is to estimate a transformation of the image domain from the position of the pairs of landmarks. In the case of a single pair, we have two limit cases. On the one hand, the pair of landmarks can be matched using a translation: every point in the image domain are transported in the same way. This is a linear matching, which is intrinsically a global transformation. On the other hand, one can move the source landmark to the target landmark and left all the other points in the image domain unchanged. This perfectly matches the landmarks points, but pixels of the image in-between do not match at all. Since neither solution is satisfactory, one will use an intermediate way. One wants to diffuse the information containing in the respective position of the landmarks in a neighborhood of the landmarks. This requires the specify the size of the neighborhoods around the landmarks. If this size tends to infinity, we get back to the linear model. If this size tends to zero, only the landmarks

points move and the *discrete* distribution of the landmarks cannot be used to infer a deformation of the *continuous* image domain.

This idea of diffusion can be implemented using radial basis functions. These functions can be seen as an interpolating function. They take two points as input and return a scalar (a weight) which decreases with respect to the distance between the two points. Formally, a radial basis function writes: $K(X,Y) = f(\|X-Y\|)$ (K stands for kernel) where f is a scalar function. Several choices are possible for f. For instance:

$$\begin{aligned} f(\|X-Y\|) &= \exp\left(-\frac{\|X-Y\|^2}{\sigma^2}\right) \\ f(\|X-Y\|) &= \frac{\sigma^2}{\|X-Y\|^2+\sigma^2} \end{aligned} \tag{4.1}$$

These functions are parameterized by a scale σ, which has the dimension of a length. This parameter controls the size of the image patch, which will be deformed by one of the function. We notice that these functions decrease to zero at infinity and that the rate of decay is precisely determined by the parameter σ. You can use `Matlab` to plot the profiles of these functions for varying scales σ.

Now, let (X_0, Y_0) a pair of landmarks in the source and the target image domain respectively. We write a general deformation $T(X)$ as $T(X) = X + v(X)$ where v is the displacement field: for each point X in the image domain, $v(X)$ is a vector (i.e. an arrow) which indicates where to move the point. We define this displacement field as:

$$v(X) = K(X, X_0)\alpha_0 = f(\|X - X_0\|)\alpha_0 \tag{4.2}$$

where α_0 is a 2D-vector to be determined. It is called a momentum. Its direction gives the direction of the deformation. Its magnitude weights the displacements of the source points.

One want to estimate the momentum, so that the deformation matches the landmark points: $T(X_0) = Y_0$. This means $X_0 + v(X_0) = Y_0$, or equivalently $K(X_0, X_0)\alpha_0 = Y_0 - X_0$. Since, in our example we have $K(X_0, X_0) = f(0) = 1$, this gives:

$$\alpha_0 = Y_0 - X_0 \tag{4.3}$$

and the deformation for any point X in the image domain is given as:

$$\boxed{T(X) = X + K(X, X_0)(Y_0 - X_0)} \tag{4.4}$$

This means that at point X_0, the displacement is given by the vector $Y_0 - X_0$. For the other points, the displacement is parallel to the direction $Y_0 - X_0$ but with a magnitude which decreases as the position X get further from that of the landmark X_0. This can be seen as a "local translation", which only impacts the points in a neighborhood of X_0.

To use this deformation to warp an image, we need to compute the inverse deformation. The problem is that a deformation written as $T(X) = X + v(X)$ may not be invertible. And if even it is invertible, the computation of the inverse is not tractable. Indeed, try to find the inverse transform of: $T(X) = X + K(X, X_0)\alpha_0$. However, one can approximate this inverse by $T'(X) = X - v(X)$, i.e. one move the point backward. This is only an approximation: $T(T'(X)) = X - v(X) + v(X - v(X))$ which differs from X. This approximation is valid if $v(X - v(X)) - v(X)$ is small, which means essentially that the norm of the displacement is small. As a consequence, in areas where the displacement field is large, the image warping may not reflect the true deformation of the underlying points. Artifacts may appear like holes or tearing, which essentially reflect the non-invertibility of the deformation or the bad approximation of this inverse when it exists. Therefore, keep in mind that this non-linear approach is only valid for "small deformations".

Demo

You can run the part of the demo entitled 'Non linear registration: close-up', which corresponds to this subsection. This loads a new set of landmarks located in this region. We focus here on a close-up of the image. Figure 8 shows the initial configuration with the single source and target landmarks. Figure 9 shows the displacement field located in the region of the landmark and the corresponding image warping.

Notice that the image warping is only local. Vary the parameter `sigma` and look at the result. Are the images satisfactorily aligned in every neighborhood around the landmarks? What happens in areas where the displacement is large?

Note that the implementation of the demo is not based on Eq. (4.4). It uses the more general formulation derived in the next section for several landmarks instead, so that you will be able to use these close-up images to test the non-linear image warping algorithm presented in the following.

4.2 Superposition of local non-linear deformations for image warping

To obtain a non-linear deformation of the entire image domain, the idea is to build a weighted sum of individual local warps, each one located at one landmark's position. Given $(X_1, \ldots, X_N)$ N landmarks on the source image and $(Y_1, \ldots, Y_N)$ their N corresponding landmarks in the target image. The deformation is written as $T(X) = X + v(X)$, where now the displacement field v is written as:

$$v(X) = \sum_{i=1}^{N} K(X, X_i)\alpha_i. \tag{4.5}$$

The momenta α_i (i.e. 2D-vectors) give the direction and the magnitude of each individual radial basis function. These momenta have to be estimated given the landmark constraints. We notice that we have exactly $2N$ variables to estimate,

for exactly $2N$ constraints. This formulation enables to adjust the number of degrees of freedom of the deformation to the number of constraints.

The set of N 2D-constraints are given as:

$$\begin{cases} T(X_1) = X_1 + \sum_{i=1}^{N} K(X_1, X_i)\alpha_i = Y_1 \\ T(X_2) = X_2 + \sum_{i=1}^{N} K(X_2, X_i)\alpha_i = Y_2 \\ \qquad \vdots \\ T(X_N) = X_N + \sum_{i=1}^{N} K(X_N, X_i)\alpha_i = Y_1 \end{cases} \tag{4.6}$$

This set of constraints can be split in two: one set for each coordinate. For the x-coordinate, the constraints are given as: $X_j^x + \sum_{i=1}^{N} K(X_j, X_i)\alpha_i^x = Y_2^x$ and for the y-coordinate as $X_j^y + \sum_{i=1}^{N} K(X_j, X_i)\alpha_i^y = Y_2^y$. This gives two sets of N scalar constraints, which can be written now in a matrix form.

Let $\mathbf{X}^x$ (resp. $\mathbf{X}^y$, $\mathbf{Y}^x$, $\mathbf{Y}^y$, $\boldsymbol{\alpha}^x$ and $\boldsymbol{\alpha}^y$) be the N dimensional vector: $\mathbf{X}^x = (X_1^x, \ldots, X_N^x)^t$ (resp. $\mathbf{X}^y = (X_1^y, \ldots, X_N^y)^t$, $\mathbf{Y}^x = (Y_1^x, \ldots, Y_N^x)^t$, $\mathbf{Y}^y = (Y_1^y, \ldots, Y_N^y)^t$, $\boldsymbol{\alpha}^x = (\alpha_1^x, \ldots, \alpha_N^x)^t$, $\boldsymbol{\alpha}^y = (\alpha_1^y, \ldots, \alpha_N^y)^t$).

Let $\mathbf{K}$ be the N-by$N$ symmetric matrix whose (i, j)-th element is $K(X_i, X_j) = f(\|X_i - X_j\|)$:

$$\mathbf{K} = \begin{bmatrix} 1 & K(X_1, X_2) & K(X_1, X_3) & \cdots & K(X_1, X_{N-1}) & K(X_1, X_N) \\ K(X_2, X_1) & 1 & K(X_2, X_3) & \cdots & K(X_2, X_{N_1}) & K(X_2, X_N) \\ \vdots & \cdots & \cdots & \ddots & \cdots & \vdots \\ K(X_N, X_1) & K(X_N, X_2) & K(X_N, X_3) & \cdots & K(X_N, X_{N-1}) & 1 \end{bmatrix}$$

Then the two sets of constraints can be written now as:

$$\begin{cases} \mathbf{K}\boldsymbol{\alpha}^x = \mathbf{Y}^x - \mathbf{X}^x \\ \mathbf{K}\boldsymbol{\alpha}^y = \mathbf{Y}^y - \mathbf{X}^y \end{cases}, \tag{4.7}$$

so that the x and y coordinate of the momenta $\boldsymbol{\alpha}$ are given by the two vectors:

$$\boxed{\begin{cases} \boldsymbol{\alpha}^x = \mathbf{K}^{-1}(\mathbf{Y}^x - \mathbf{X}^x) \\ \boldsymbol{\alpha}^y = \mathbf{K}^{-1}(\mathbf{Y}^y - \mathbf{X}^y) \end{cases}} \tag{4.8}$$

The ith coordinate of these vectors allows us to build the ith momenta $\alpha_i = (\boldsymbol{\alpha}_i^x, \boldsymbol{\alpha}_i^y)^t$.

Question: is the matrix K always invertible? This question is more difficult to answer in this case than in the previous cases, since it depends on the radial basis function used. For the two functions given here, the answer

is yes. In this case, K is said a positive definite kernel, which guarantees the invertibility of the matrix $\mathbf{K}$ if the landmarks points are distinct. In short, this property depends on the Fourier Transform of the function f.

However, from a numerical point of view, the matrix $\mathbf{K}$ may have a bad conditioning, meaning that the computation of its inverse, although being possible in theory, is impossible in practice due to numerical instability. This happens if the distance between landmarks (i.e. $\|X_i - X_j\|$) is much smaller that the scale σ. To workaround this numerical issue, we build the matrix $\mathbf{K} + \gamma \mathrm{I}$ instead of $\mathbf{K}$, where γ is a scalar parameter supposed to be small and I stands for the N-by-N identity matrix. As a consequence, the computed solution does not guarantee anymore a perfect matching between source and target landmarks. The total matching errors actually increases with γ. See the exercise below for more insight into this parameter.

Demo

You can now run the part of the demo, which corresponds to this section. Figure 10 shows the initial configuration. Figure 11 shows the displacement field over the whole image domain estimated from the location of the landmarks and the corresponding warped image. How does this registration compare to the affine one? Look at the transversal part of the anchor for instance.

You can also experiment this image warping method on the close-up images of the previous section.

Questions:

- Change the scale of the kernel σ. How does this impact the results?
- Set the scale σ to a large value and decrease the value of γ until `Matlab` catches an error due to numerical instability (the inversion of an bad conditioned matrix leads to complex values in `Matlab`, which later raises an error.) What is the trade-off between numerical stability and precision of the matching?
- What do think about the distribution of the landmarks? How many landmarks should one need to have a good warping? Should the distribution of the landmarks be uniform in the image domain? What happens if you down-sample the set of landmarks (pick one other landmark for instance)?
- Make an artificial experiment: take two landmarks closer to each other than σ and choose two momenta which are parallel but with opposite direction. Compute the corresponding displacement field. How does this displacement field vary with the scale σ? What does this experiment tell you about the spatial integration of "incompatible" constraints?
- What happens to the image warping if the magnitude of the deformation (i.e. the displacement of the points) is large? Is the idea of scaling the magnitude of the displacement field a satisfactory answer to this problem?

- A deformation is called diffeomorphic if it is smooth, one to one with a smooth inverse. In particular, a diffeomorphic deformation cannot create holes or tearing for instance. To which extent the computed deformation is diffeomorphic?

- In these experiments, the source image has been set to the image after the affine alignment. What happens if you start from the original image instead? Do non-linear deformations account well for linear transformations? Can one cascade one linear transformation and a non-linear deformation? In which order? Do they commute? Can we estimate both deformations at the same time?

4.3 Exercise

A registration which perfectly matches the source to the target points is not always desirable, especially since the position of the landmarks may be known only up to a given precision. We would rather use the landmarks as an indication of the direction of deformation instead of a hard constraint.

As we did for the affine registration, we can introduce a least-square criterion, which compute the total error matching between source and target points. This least square criterion is: $\|(\mathbf{X} + \mathbf{K}\boldsymbol{\alpha}) - \mathbf{Y}\|^2$, where here the norm stands for the norm on $2N$ dimensions and where we concatenate the x and y coordinate of the vectors. This norm is equal to $\sum_{i=1}^{N} \|X_i + v(X_i) - Y_i\|^2$.

The minimization of the least-square criterion:

$$\min_{\boldsymbol{\alpha}} \|\mathbf{X} + \mathbf{K}\boldsymbol{\alpha} - \mathbf{Y}\|^2 \tag{4.9}$$

will lead to the solution $\boldsymbol{\alpha} = \mathbf{K}^{-1}(\mathbf{Y} - \mathbf{X})$, as we precisely showed that this value satisfies all the matching constraints, thus making the criterion to vanish.

The idea is to introduce a *regularized* least-square criterion which balances the error matching against the "regularity" of the deformation. The regularity may be quantified by the scalar $\boldsymbol{\alpha}^t \mathbf{K}\boldsymbol{\alpha}$. If this value is small (the norm of $\boldsymbol{\alpha}$ tends to zero), then there is almost no displacement. The larger the momenta, the larger the amount of deformation. This quantity can be seen as a measure of the variance of the momenta distributed at the landmark points. Therefore, the regularized least square criterion can be written as:

$$\min_{\boldsymbol{\alpha}} \left\{ \|\mathbf{X} + \mathbf{K}\boldsymbol{\alpha} - \mathbf{Y}\|^2 + \gamma \boldsymbol{\alpha}^t \mathbf{K}\boldsymbol{\alpha} \right\}, \tag{4.10}$$

where γ is a scalar parameter which quantifies the trade-off between the matching errors and the regularity of the deformation.

Question: Differentiate this criterion with respect to the vector $\boldsymbol{\alpha}$ (you may use the intrinsic method of differentiation illustrated in the affine case) and show that the criterion is minimized for $\boldsymbol{\alpha} = (\mathbf{K} + \gamma \mathrm{I})^{-1} (\mathbf{Y} - \mathbf{X})$. Interpret the role of γ from both a geometrical and a numerical point of view (i.e. with respect to the matching errors and with respect to the numerical stability of the computation of the solution). Discuss the two limit cases: $\gamma \to 0$ and $\gamma \to \infty$.

5 Conclusion

As you must have noticed from the experiments, this non-linear registration scheme enables to correct several misalignments which remained after the affine registration. However, some undesirable artifacts like creation of holes appears when the "amount of deformation" is too large, for instance when two parts of the image, which are close to each other, deform in different ways. Decreasing the magnitude of the displacement field corrects those artifacts, but at the cost of a poor matching. One idea is to move the points in the direction given by the the displacement field, but by only a small fraction in magnitude. Once points have been move slightly, one re-estimates the displacement field at the new locations of the source points (which now account for the matching error between the updated source points and the target points) and again slightly move the points in this new direction. This builds a deformation iteratively. In this case, the source points do not move along a straight line, but along curves instead. The displacement field gives the tangents of these curves and can therefore be interpreted as the velocity field of a deforming material. In this case, the deformations are (almost) always invertible and one can compute their inverse explicitly. Such registrations fall into the category of 'diffeomorphic registrations'. But that's another story...