Introduction to image registration

Pietro Gori

Enseignant-chercheur
Equipe IMAGES - Télécom Paris
pietro.gori@telecom-paris.fr

TELECOM Paris

IP PARIS

Plan

Summary

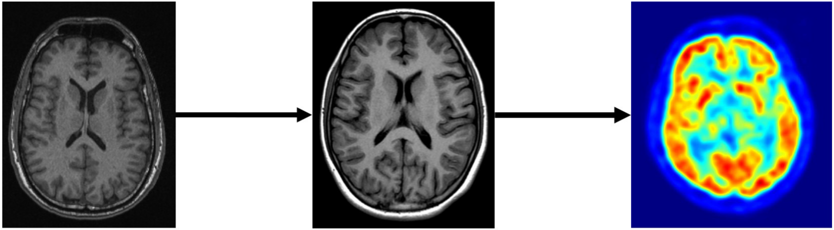
Image registration

Definition

- **Geometric**: find the *optimal* parameters of a *geometric transformation* to spatially align two different images of the same object. It establishes *spatial correspondence* between the pixels of the source (or moving) image with the ones of the target image
- Photometric: Modify the intensity of the pixels and not their position

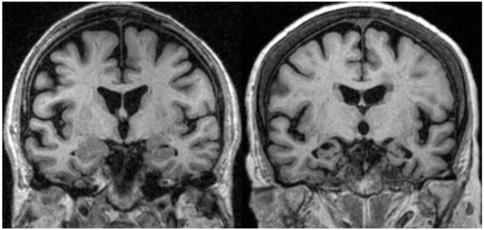
Medical Image Applications

- Compare two (or more) images of the same modality (e.g. T1-w MRI of the brain of two different subjects)
- Combine information from multiple modalities (e.g. PET, DWI, T1-w MRI of the brain of the same subject)
- Longitudinal studies (e.g. monitor anatomical or functional changes of the brain over time)
- Relate preoperative and postoperative images after surgery

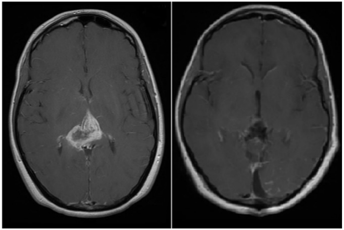


Same modality
Different subjects

Different modalities
Same subject



Longitudinal study



Pre and postoperative

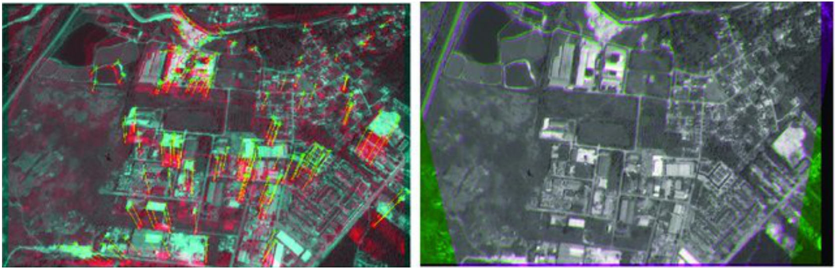
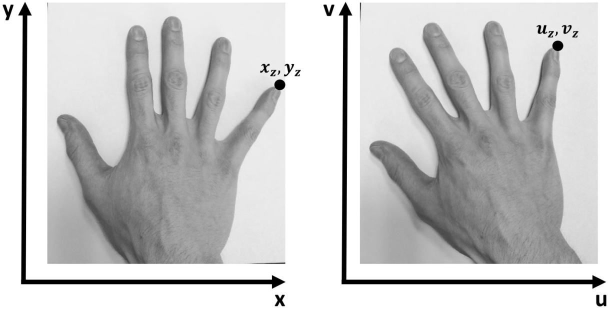


Figure 1: Remote sensing example - From 'Geometric feature descriptor and dissimilarity-based registration of remotely sensed imagery' PLoS One, 2018

Image registration components

- **Dimensionality**: 2D/2D, 3D/3D, 2D/3D
- **Transformation** (linear / non-linear)
- **Similarity metric** (e.g. intensities, landmarks, edges, surfaces)
- **Optimization procedure**
- **Interaction** (automatic / semi-automatic / interactive)
- **Modalities** (mono-modal / multi-modal)
- **Subjects** (intra-subject / inter-subject / atlas construction)

- Let I and J be the source and target images. They show the same anatomical object, most of the time with a different field of view and resolution (sampling)
- $I(x, y)$ and $J(u, v)$ represent the intensity values of the pixels located onto two regular grids: $\{x, y\} \in \Omega_I$ and $\{u, v\} \in \Omega_J$
- For the same subject, the same anatomical point z can be in position x_z, y_z in I and in u_z, v_z in J



Mathematical definition

- Both I and J are functions:

$$\begin{array}{lccc} I(x,y): & \Omega_I \subseteq \mathbb{R}^2 & \rightarrow & \mathbb{R} \\ & (x,y) & \rightarrow & I(x,y) \end{array}$$

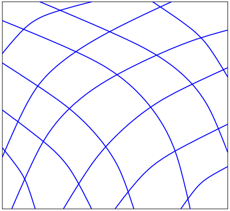
- We look for a geometric transformation $\mathbf{T}$, which is a 2D warping parametric function that belongs to a certain family Γ:

$$\begin{array}{lccc} \mathbf{T}_\phi(x,y): & \mathbb{R}^2 & \rightarrow & \mathbb{R}^2 \\ & (x,y;\phi) & \rightarrow & \mathbf{T}_\phi(x,y) \end{array}$$

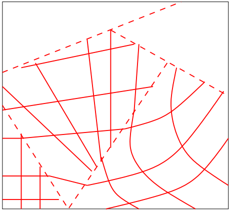
- ϕ is the vector of parameters of $\mathbf{T}$. We look for a transformation that maps (x_z, y_z) to (u_z, v_z)

Mathematical definition

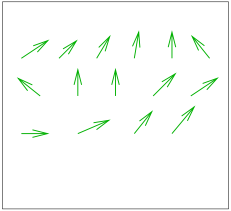
- Most of the time, I and J are simply seen as matrices whose coordinates (x, y) and (u, v) are thus integer-valued (number of line and column)
- The values of the intensities of the pixels can be real numbers $\mathbb{R}$ (better for computations) or integer, usually in the range $[0, 255]$ for a gray-scale image
- There are several kind of transformations:



modèle global



modèle par morceaux (régional)



modèle local

Summary

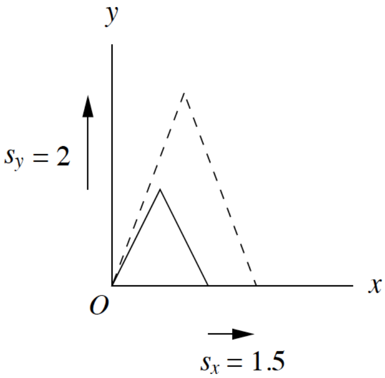
Global transformations

- Global transformations can be defined with matrices
- Application: $\mathbf{T}_\phi(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = A\mathbf{x}$
- Inverse (if invertible): $\mathbf{T}_\phi^{-1}(x,y) = A^{-1}\mathbf{x}$
- Composition: $\mathbf{T}_1(\mathbf{T}_2(x,y)) = (\mathbf{T}_1 \circ \mathbf{T}_2)(x,y) = A_1A_2\mathbf{x}$
- Note: order of transformation is important : A_1A_2 is not equal to A_2A_1 in general

Global transformations

- Global means that the transformation is the same for any points p
- **Scaling** - multiply each coordinate by a scalar

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

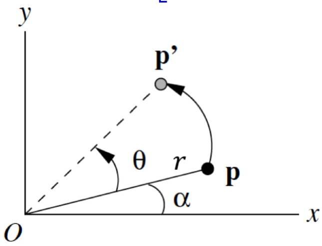


- **Rotation** - WRT origin. Let $p = [x\ y]^T$ and $p' = [u\ v]^T$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\alpha) \\ r\sin(\alpha) \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r\cos(\alpha+\theta) \\ r\sin(\alpha+\theta) \end{bmatrix} = \begin{bmatrix} r(\cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\theta)) \\ r(\sin(\alpha)\cos(\theta) + \cos(\alpha)\sin(\theta)) \end{bmatrix}$$

$$= \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ y\cos(\theta) + x\sin(\theta) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



- $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
- $\det(R) = 1$
- $R^{-1} = R^T$

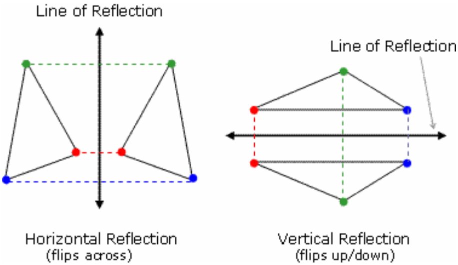
Global transformations

- **Reflection**
- Horizontal (Y-axis)

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r\cos(\pi - \alpha) \\ r\sin(\pi - \alpha) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Vertical (X-axis)

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r\cos(\frac{3}{2}\pi + \alpha) \\ r\sin(\frac{3}{2}\pi + \alpha) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



- $\det(R) = -1$
- $R^{-1} = R^T$

- **Shear** - Transvection in french

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & \lambda_x \\ \lambda_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & \tan(\phi) \\ \tan(\psi) & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

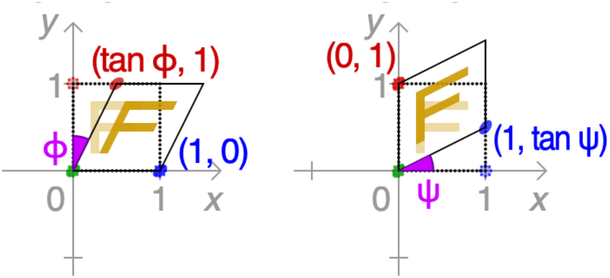


Figure 2: Shear in x and y direction. Image taken from Wikipedia.

2D Linear transformations

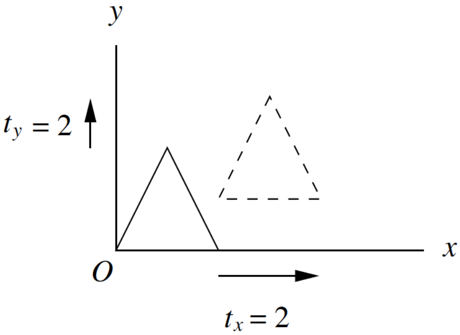
$$\begin{bmatrix} u \\ v \end{bmatrix} = \underbrace{\begin{bmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} 1 & \lambda_x \\ \lambda_y & 1 \end{bmatrix}}_{\begin{bmatrix} a & b \\ c & d \end{bmatrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Linear transformations are combinations of:
  - scaling
  - rotation
  - reflection
  - shear
- Properties of linear transformations:
  - origin is always transformed to origin
  - parallel lines remain parallel
  - ratios are preserved
  - lines remain lines

Translation

- It does not have a fixed point $\rightarrow$ no matrix multiplication

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} \tag{2}$$



Homogeneous coordinates - 2D affine transformation

- Instead than 2D matrices we use 3D matrices.
- **Affine transformation**: combination of linear transformations and translations
- Let $p = [x \ y]^T$ and $p' = [u \ v]^T$, we obtain $p' = \mathbf{T}p$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- $\mathbf{T} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^t & 1 \end{bmatrix}$, where A is the 4 dof linear component and $\mathbf{t}$ the 2 dof translation

2D affine transformation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Properties of affine transformations:
  - origin is *not* always transformed to origin
  - parallel lines remain parallel
  - ratios are preserved
  - lines remain lines

2D Projective transformations (homographies)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a & b & g \\ c & d & h \\ e & f & 1 \end{bmatrix}}_{\mathbf{T}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- $\mathbf{T} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^t & 1 \end{bmatrix}$, where A is the 4 dof affine component, $\mathbf{t}$ the 2 dof translation and $\mathbf{v}$ the 2 dof elation component
- If we consider only $\mathbf{v}$, so $\mathbf{t} = \mathbf{0}$ and $A = \mathcal{I}$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ e & f & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ ex + fy + 1 \end{bmatrix}$$

$$u = \frac{x}{ex + fy + 1} \quad v = \frac{y}{ex + fy + 1}$$

2D Projective transformations (homographies)

$$u = \frac{x}{ex + fy + 1} \quad v = \frac{y}{ex + fy + 1}$$

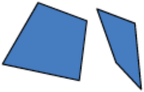
- **Elation**: points are scaled by a scaling factor which is a linear combination of x and y. Points can be mapped to (resp. from) infinite to (resp. from) a finite scalar value. (Ex. if you set f=0 and e=1, then the point $[\infty, \infty]$ is mapped to $[1, 1]$)
- Properties of projective transformations:
  - origin is *not* always transformed to origin
  - parallel lines do *not* necessarily remain parallel
  - ratios, length and angle are *not* preserved
  - lines remain lines

Examples

A square transforms to:

Projective
8dof

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

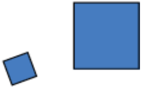


Affine
6dof

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

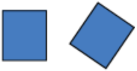
Similarity
4dof

$$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Euclidean
3dof

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

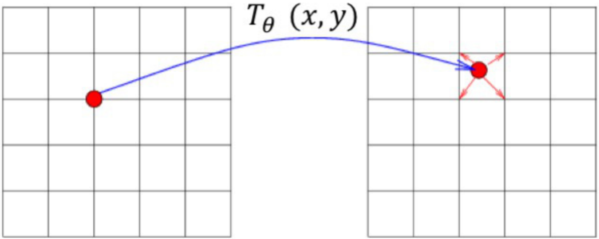


Summary

Forward warping

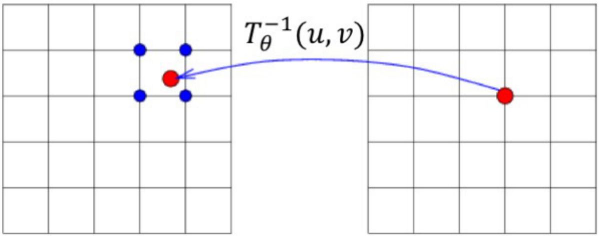
$$\begin{array}{cccc} \mathbf{T}_\phi(x,y): & \mathbb{R}^2 & \rightarrow & \mathbb{R}^2 \\ & (x,y;\phi) & \rightarrow & \mathbf{T}_\phi(x,y) \end{array}$$

- Ideally, if Ω_I and Ω_J were continuous domains, we would simply map (x,y) to $(u,v) = \mathbf{T}_\phi(x,y)$ and then compare $I(x,y)$ with $J(u,v)$
- However, Ω_I and Ω_J are regular grids ! What if $\mathbf{T}_\phi(x,y)$ is not on the grid Ω_J ? $\rightarrow$ Splatting: add weighted contribution to neighbor pixels.

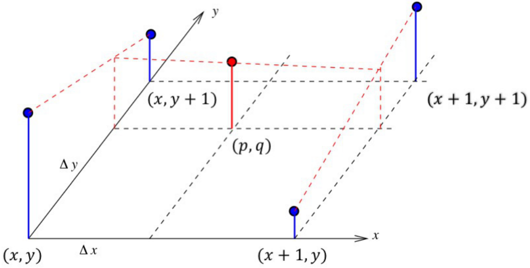


Inverse warping

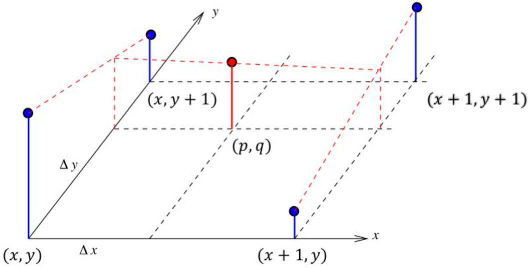
- Find the pixel intensities for the deformed image $I_{\mathbf{T}}$ starting from Ω_J: $(x,y) = \mathbf{T}_\phi^{-1}(u,v)$
- Assign to $I_{\mathbf{T}}(u,v)$ the pixel intensity in $I(x,y)$
- What if $\mathbf{T}_\phi^{-1}(u,v)$ is not on Ω_I ? $\rightarrow$ Interpolation !



- **Goal**: estimate the intensity value on points not located onto the regular grids
  - nearest neighbor
  - bilinear
  - cubic
  - lanczos
  - ...



- **Nearest neighbor**: $J(u,v) = I(\text{round}(x), \text{round}(y))$
- **Bilinear**: $\frac{f(x,q)-f(x,y)}{\Delta y} = \frac{f(x,y+1)-f(x,q)}{1-\Delta y} \rightarrow$
  $f(x,q) = (1-\Delta y)f(x,y) + f(x,y+1)\Delta y$. Similarly,
  $f(x+1,q) = (1-\Delta y)f(x+1,y) + f(x+1,y+1)\Delta y$. Then,
  $f(p,q) = f(x,q)(1-\Delta x) + f(x+1,q)\Delta x$



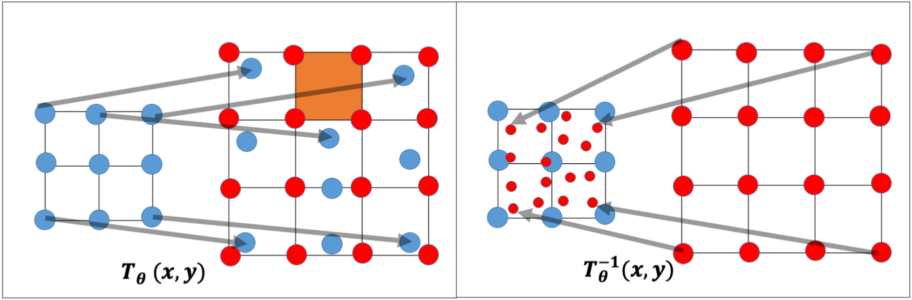
Interpolation examples

Every time we deform an image, we need to interpolate it. For instance, this is the result on Lena after 10 rotations of 36 degrees:

Figure 3: Original image - Nearest Neighbour - Bilinear

Source : http://bigwww.epfl.ch/demo/jaffine/index.html (Michael Unser)

Forward versus Inverse warping



Forward warping **Inverse warping**

In the case of forward warping, holes can occur in the warped image (shaded in orange). Using the inverse warping, we avoid this issue. However, we need to be able to define the inverse transformation !

Deformation algorithm

Recipe:

- Given a source image I and a global transformation $\mathbf{T}$, compute the forward warping of I. The min and max values of the x and y coordinates of the resulting deformed image $I_{\mathbf{T}}$ will give the bounding box
- Given the bounding box of $I_{\mathbf{T}}$, create a new grid within it with, for instance, the same shape of Ω_J to allow direct comparison between $I_{\mathbf{T}}$ and J
- Use the inverse warping and interpolation to compute the intensity values at the grid points of the warped image $I_{\mathbf{T}}$ (we avoid holes)
- Be careful! During the inverse warping, points that are mapped outside Ω_I are rejected.

Recap - Matrix inversion

- The inverse of a square (invertible) matrix $\mathbf{T}$ can be computed as the ratio between the adjoint of $\mathbf{T}$ and its determinant: $\mathbf{T}^{-1} = \text{adj}(T)/\det(T)$
- The adjoint $\text{adj}(\mathbf{T})$ of $\mathbf{T}$ is the transpose of its cofactor matrix
- Given $T = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ its cofactor matrix $\mathbf{C}$ is:

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix},$$

Transformation parameters

- Given a transformation $\mathbf{T}$ defined by a set of parameter θ and two images I and J, how do we estimate θ ?

Transformation parameters

- Given a transformation $\mathbf{T}$ defined by a set of parameter θ and two images I and J, how do we estimate θ ?
- By minimizing a cost function:

$$\theta^* = \arg\min_{\theta} d(I_{\mathbf{T}}, J) \tag{3}$$

- The similarity measure d might be based on the pixel intensities and/or on corresponding geometric objects such as control points (i.e. landmarks), curves or surfaces

Summary

Same modality

- Sum of squared intensity differences (SSD). Best measure when I and J only differ by Gaussian noise. Very sensitive to “outliers” pixels, namely pixels whose intensity difference is very large compared to others.

$$d(I_{\mathbf{T}}, J) = \sum_u \sum_v (J(u,v) - I(\mathbf{T}_\phi^{-1}(u,v)))^2 = (J(u,v) - I_{\mathbf{T}}(u,v))^2 \tag{4}$$

- Normalized Cross-Correlation. Assumption is that there is a linear relationship between the intensity of the images

$$d(I_{\mathbf{T}}, J) = \frac{\sum_u \sum_v (J(u,v) - \bar{J})(I_{\mathbf{T}}(u,v) - \bar{I}_{\mathbf{T}})}{\sqrt{\sum_u \sum_v (J(u,v) - \bar{J})^2 \sum_u \sum_v (I_{\mathbf{T}}(u,v) - \bar{I}_{\mathbf{T}})^2}} \tag{5}$$

Multi modality

- Mutual information. We first need to define the joint histogram between $I_{\mathbf{T}}$ and J. The value in (a,b) is equal to the number of locations (u,v) that have intensity a in $I_{\mathbf{T}}(u,v)$ and intensity b in $J(u,v)$. For example, a joint histogram which has the value of 2 in the position (4,3) means that we have found two locations (u,v) where the intensity of the first image was 4 ($I_{\mathbf{T}}(u,v)=4$) and the intensity of the second was 3 ($J(u,v)=3$). By dividing by the total number of pixels N, we obtain a joint probability density function (pdf) $p_{I_{\mathbf{T}},J}$.
- The sum over the rows or columns gives the marginal pdf of J (p_J) and of $I_{\mathbf{T}}$ ($p_{I_{\mathbf{T}}}$) respectively

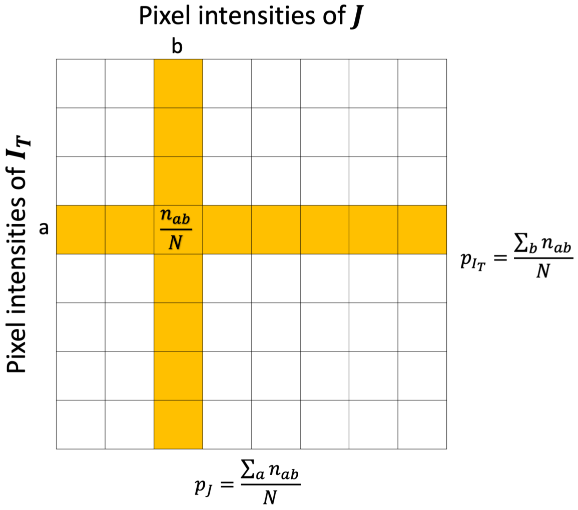


Figure 4: Normalized joint histogram example. n_{ab} indicates the number of locations where the intensity of I_T is equal to a and the intensity of J is equal to b

- We suppose that the pixels of I_T and J take only 8 different intensity values, or that we can group them into 8 bins
- What's the difference between the two normalized histograms ? Which one represents a perfect alignment ?

7	0	0	0	0	0	0	0	0.05
6	0	0	0	0	0	0	0.1	0
5	0	0	0	0	0	0.15	0	0
4	0	0	0	0	0.1	0	0	0
3	0	0	0	0.1	0	0	0	0
2	0	0	0.1	0	0	0	0	0
1	0	0.2	0	0	0	0	0	0
0	0.2	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7

7	0	0	0	0	0	0.01	0.03	0.05
6	0	0	0.04	0.01	0.1	0.05	0.01	0.05
5	0	0	0.02	0.01	0.05	0.01	0.01	0.05
4	0	0	0.02	0	0.01	0.04	0.01	0.01
3	0	0.01	0.2	0.01	0.03	0.01	0.01	0.01
2	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0
1	0.01	0.02	0	0	0	0	0	0
0	0.02	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7

Intensity based registration - similarity measures

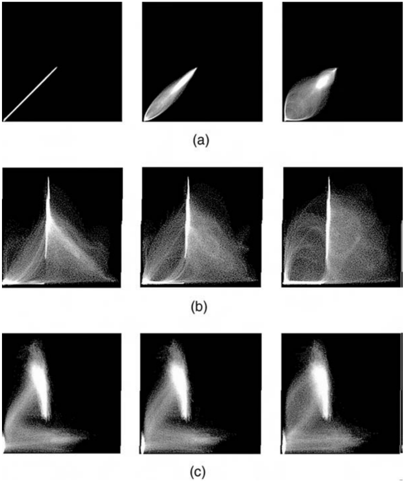


Figure 5: Joint histogram of a) same modality (IRM) b) different modality (MR-CT) c) different modality (MR-PET). First column, images are aligned. 2nd and 3rd columns images are translated. Taken from [2].

Intensity based registration - similarity measures

- The definition of joint entropy is:

$$H(I_{\mathbf{T}}, J) = -\sum_a \sum_b p_{I_{\mathbf{T}},J}(a,b) \log(p_{I_{\mathbf{T}},J}(a,b)) \tag{6}$$

- where a and b are defined within the range of intensities in $I_{\mathbf{T}}$ and J respectively
- The individual entropies of $I_{\mathbf{T}}$ and J are:
  $H(I_{\mathbf{T}}) = -\sum_a p_{I_{\mathbf{T}}}(a) \log(p_{I_{\mathbf{T}}}(a))$ and
  $H(J) = -\sum_b p_J(b) \log(p_J(b))$ respectively.
- We know that $H(I_{\mathbf{T}}, J) \leq H(I_{\mathbf{T}}) + H(J)$ and the more similar the distributions $p_{I_{\mathbf{T}}}$ and p_J, the lower the joint entropy compared to the sum of individual entropies

Intensity based registration - similarity measures

- The definition of Mutual information is:

$$M(I_{\mathbf{T}}, J) = H(I_{\mathbf{T}}) + H(J) - H(I_{\mathbf{T}}, J) \tag{7}$$

- It results:

$$M(I_{\mathbf{T}}, J) = \sum_a \sum_b p_{I_{\mathbf{T}},J}(a,b) \log \frac{p_{I_{\mathbf{T}},J}(a,b)}{p_{I_{\mathbf{T}}}(a) p_J(b)} \tag{8}$$

- It can be seen as the Kullback–Leibler divergence between $p_{I_{\mathbf{T}},J}$ and $p_{I_{\mathbf{T}}} \otimes p_J$. It measures the cost for considering $I_{\mathbf{T}}$ and J as independent random variables, when in reality they are not.

$$M(I_{\mathbf{T}}, J) = H(I_{\mathbf{T}}) + H(J) - H(I_{\mathbf{T}}, J) = H(J) - H(J|I_{\mathbf{T}}) \quad (9)$$

- where the conditional entropy $H(J|I_{\mathbf{T}}) = -\sum_a \sum_b p_{I_{\mathbf{T}},J}(a,b) \log p_{J|I_{\mathbf{T}}}(b|a)$.
- Mutual information measures the amount of uncertainty about J minus the uncertainty about J when $I_{\mathbf{T}}$ is known, that is to say, how much we reduce the uncertainty about J after observing $I_{\mathbf{T}}$. It is maximized when the two images are aligned. [8]
- Maximizing M means finding a transformations $\mathbf{T}$ that makes $I_{\mathbf{T}}$ the best predictor for J. Or, equivalently, knowing the intensity $I_{\mathbf{T}}(u,v)$ allows us to perfectly predict $J(u,v)$.

Intensity based registration - optimization procedure

- Pixel-based similarity measures need an **iterative** approach where an initial estimate of the transformation is gradually refined using the gradient and, depending on the method, also the Hessian of the similarity measure with respect to the parameters θ
- Possible algorithms: gradient descent, Gauss-Newton, Newton-Raphson, Levenberg-Marquardt, etc.
- Problem of the "local minima" $\rightarrow$ stochastic optimization, line search, trust region, multi-resolution (first low resolution and then higher resolution)
- **Validation** $\rightarrow$ visual inspection, alignment of manually segmented objects, value of similarity measure

Intensity based registration

- The goal is to minimize a similarity measure d between a transformed source image $I_{\mathbf{T}}$ and a target image J with respect to the parameters θ of the transformation $\mathbf{T}$
- If we choose the SSD as similarity measure, it results:

$$\begin{aligned} \theta^* = \arg\min_{\theta} \sum_u \sum_v (I(\mathbf{T}_{\phi}^{-1}(u,v)) - J(u,v))^2 = \\ \sum_u \sum_v (I_{\mathbf{T}}(u,v;\theta) - J(u,v))^2 \end{aligned} \tag{10}$$

- This a non-linear optimization procedure even if the transformation $\mathbf{T}$ is linear in θ because the pixel intensities are (in general) not (linearly) related to the pixel coordinates (u,v)

Intensity based registration - Lucas-Kanade Algorithm

- Probably the first image registration algorithm was the Lucas-Kanade one (1981)
- We start with an initial guess for the parameters θ and we look for the best increment to the parameters $\Delta\theta$, by minimizing:

$$\Delta\theta^* = \arg\min_{\Delta\theta} \sum_u \sum_v (I_{\mathbf{T}}(u, v; \theta + \Delta\theta) - J(u, v))^2 \tag{11}$$

- After that, the parameters are updated as: $\theta^* = \theta + \Delta\theta^*$
- These two steps are iterated until convergence (typically $||\Delta\theta||_{<\epsilon}$)

- We first linearize the non-linear expression in Eq.11 by performing a first order Taylor expansion on $I_{\mathbf{T}}(u, v; \theta + \Delta\theta))$, obtaining:

$$\sum_u \sum_v (I_{\mathbf{T}}(u, v; \theta) + \nabla I_{\mathbf{T}}(u, v; \theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u, v; \theta)\Delta\theta - J(u, v))^2 \quad (12)$$

- Reminder: the first order Taylor expansion of a composite scalar function is:

$$f(g(x + h)) \approx f(g(x)) + f'(g(x))g'(x)h \quad (13)$$

$$\sum_u \sum_v (I_{\mathbf{T}}(u,v;\theta) + \nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta)\Delta\theta - J(u,v))^2 \tag{14}$$

- $\nabla I_{\mathbf{T}}(u,v;\theta) = \left(\frac{\partial I_{\mathbf{T}}(u,v;\theta)}{\partial u}, \frac{\partial I_{\mathbf{T}}(u,v;\theta)}{\partial v}\right)^T$ is a $[2 \times 1]$ column vector and is the *gradient* of the image I evaluated at $\mathbf{T}_\phi^{-1}(u,v)$. This means computing the gradient of ∇I in the coordinate frame of I and then warp it back onto the coordinate frame of J using the current estimate of $\mathbf{T}$

Reminder - Image gradient

- The image gradient can be computed as the convolution of the original image I with one filter for the x direction and one for the y direction
- $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})^T$ where $\frac{\partial I}{\partial x} = G_x * I$ and $\frac{\partial I}{\partial y} = G_y * I$
- Common choices for G_x and G_y are:
  - **Sobel**: $G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$ and $G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
  - **Scharr**: $G_x = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$ and $G_y = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$

Intensity based registration - Lucas-Kanade Algorithm

$$\sum_u \sum_v (I_{\mathbf{T}}(u,v;\theta) + \nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta)\Delta\theta - J(u,v))^2 \qquad (15)$$

- $\frac{\partial \mathbf{T}}{\partial \theta}$ is a $[2 \times d]$ matrix where d is the number of parameters θ and is the *Jacobian* of the transformation. Let $\mathbf{T}(u,v;\theta) = (T_u(u,v;\theta), T_v(u,v;\theta))^T$ be a 2D column vector then:

$$\frac{\partial \mathbf{T}}{\partial \theta} = \begin{bmatrix} \frac{\partial T_u}{\partial \theta_1} & \frac{\partial T_u}{\partial \theta_2} & \dots & \frac{\partial T_u}{\partial \theta_d} \\ \frac{\partial T_v}{\partial \theta_1} & \frac{\partial T_v}{\partial \theta_2} & \dots & \frac{\partial T_v}{\partial \theta_d} \end{bmatrix} \qquad (16)$$

- We follow the notational convention that the partial derivatives with respect to a column vector are laid out as a row vector. This convention has the advantage that the chain rule results in a matrix multiplication as in Eq.15

Intensity based registration - Lucas-Kanade Algorithm

- By substituting the linearization in the original cost function Eq.11, we obtain:

$$\Delta\theta^* = \arg\min_{\Delta\theta} \sum_u \sum_v (I_{\mathbf{T}}(u,v;\theta) + \nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta)\Delta\theta - J(u,v))^2 \tag{17}$$

- The partial derivative with respect to $\Delta\theta$ is:

$$2\sum_u \sum_v \left(\nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta) \right)^T (I_{\mathbf{T}}(u,v;\theta) + \nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta)\Delta\theta - J(u,v)) \tag{18}$$

Intensity based registration - Lucas-Kanade Algorithm

- Setting equal to zero the previous Eq., we obtain:

$$\Delta\theta = H^{-1} \sum_u \sum_v \left(\nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta) \right)^T (J(u,v) - I_{\mathbf{T}}(u,v;\theta)) \tag{19}$$

- where H is a $[d \times d]$ matrix and is an approximation of the Hessian matrix. This is a *Gauss-Newton gradient descent algorithm*.

$$H = \sum_u \sum_v \left(\nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta) \right)^T \left(\nabla I_{\mathbf{T}}(u,v;\theta)^T \frac{\partial \mathbf{T}}{\partial \theta}(u,v;\theta) \right) \tag{20}$$

- Please note that if we approximate H with an identity function, we obtain the *steepest descent parameter updates*

Intensity based registration - Lucas-Kanade Algorithm

- Let d the number of parameters for the transformation T and n the number of pixels in J, the total computational cost of each iteration is $O(nd^2 + d^3)$
- The two most expensive steps are: computing the Hessian matrix ($O(nd^2)$) and inverting it ($O(d^3)$).
- The Lucas-Kanade Algorithm is one possible solution but other approaches exist. See [9] for more details.

Intensity based registration - Lucas-Kanade Algorithm

Algorithm 1 Lucas-Kanade Algorithm

Get (r, c) the number of rows and columns of J, initialize θ_0, maximum number of iterations K and iteration index $k = 0$
while $||\Delta\theta||_2 >= \epsilon$ and $||J(u, v) - I_{\mathbf{T}}(u, v; \theta)||_2 >= \tau$ and $k < K$ **do**
  Initialize $\Delta\theta = (0, ..., 0)^T$, $H = ((0, 0), (0, 0))$
  for $u = 1$ to r **do**
    for $v = 1$ to c **do**
      Compute $I_{\mathbf{T}}(u, v; \theta_k)$ and $\nabla I_{\mathbf{T}}(u, v; \theta_k)$
      Evaluate the Jacobian $\frac{\partial \mathbf{T}}{\partial \theta}(u, v; \theta)$
      $\Delta\theta \mathrel{+}= \left(\nabla I_{\mathbf{T}}(u, v; \theta_k)^T \frac{\partial \mathbf{T}}{\partial \theta}(u, v; \theta_k)\right)^T (J(u, v) - I_{\mathbf{T}}(u, v; \theta_k))$
      $H \mathrel{+}= \left(\nabla I_{\mathbf{T}}(u, v; \theta_k)^T \frac{\partial \mathbf{T}}{\partial \theta}\right)^T \left(\nabla I_{\mathbf{T}}(u, v; \theta_k)^T \frac{\partial \mathbf{T}}{\partial \theta}\right)$
    end for
  end for
  $\Delta\theta = H^{-1}\Delta\theta$
  $k = k + 1$
  $\theta_k = \theta_{k-1} + \Delta\theta$
end while

Summary

Fourier based registration

- Instead than working in the spatial domain, we could work in the Fourier domain
- This is particular effective when looking for similarity (rigid) transformations. Thus, translation + rotation + uniform scaling
- By using the Fourier transform, we can directly estimate the parameters with a closed form solution instead than **optimizing** a cost function based on a similarity measure as it is the case when using directly the intensity of the pixels
- Let's first recap some concepts about the Fourier transform

Definition 2D continuous Fourier transform

Let $f(x, y)$ be a real-valued function with spatial coordinates (x, y) defined in $\mathcal{R}^2$, $F(p, q)$ a complex-valued function of frequency (p, q), defined in $\mathcal{R}^2$, representing the Fourier transform of f, $\mathcal{F}\{\cdot\}$ the Fourier transform operator and $\mathcal{F}^{-1}\{\cdot\}$ its inverse, we define the 2D Fourier transform as:

$$\mathcal{F}\{f\}(p, q) = F(p, q) = \int_{\mathcal{R}} \int_{\mathcal{R}} f(x, y) e^{-i2\pi(px+qy)} dxdy \tag{21}$$

and the inverse 2D Fourier transform as:

$$\mathcal{F}^{-1}\{F\}(x, y) = f(x, y) = \int_{\mathcal{R}} \int_{\mathcal{R}} F(p, q) e^{i2\pi(px+qy)} dpdq \tag{22}$$

Definition 2D discrete and periodic Fourier transform

Let $f(x,y)$ be a real-valued discrete and periodic image of size $[M$ x $N]$, $F(p,q)$ a complex-valued image of size M x N representing the Fourier transform of f, $\mathcal{F}\{\cdot\}$ the Fourier transform operator and $\mathcal{F}^{-1}\{\cdot\}$ its inverse, we define the 2D discrete Fourier transform as:

$$\mathcal{F}\{f\}(p,q) = F(p,q) = \frac{1}{\sqrt{NM}} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-i2\pi(\frac{px}{N} + \frac{qy}{M})} \tag{23}$$

and the inverse 2D Fourier transform as:

$$\mathcal{F}^{-1}\{F\}(x,y) = f(x,y) = \sum_{p=0}^{N-1} \sum_{q=0}^{M-1} F(p,q) e^{i2\pi(\frac{px}{N} + \frac{qy}{M})} \tag{24}$$

2D Fourier transform - Properties

- $F(p,q)$ is complex in general
- *Rectangular coordinates*: $F(p,q) = F_R(p,q) + iF_I(p,q)$ where both F_R, $F_I \in \mathcal{R}$
- *Polar coordinates*: $F(p,q) = |F(p,q)|e^{i\angle F(p,q)}$ where $|F(p,q)| = \sqrt{F_R^2 + F_I^2} \in \mathcal{R}$ is called **magnitude** and $\angle F(p,q) = \arctan(F_I(p,q)/F_R(p,q)) \in \mathcal{R}$ is called **phase**

$$
\begin{aligned}
f(x,y) &= \int_{\mathcal{R}}\int_{\mathcal{R}} F(p,q)e^{i2\pi(px+qy)}dpdq \\
&= \int_{\mathcal{R}}\int_{\mathcal{R}} |F(p,q)|e^{i(\angle F(p,q)+2\pi(px+qy))}dpdq
\end{aligned}
\tag{25}
$$

- The magnitude $|F(p,q)|$ determines the relative importance of the frequency components and the phase $\angle F(p,q)$ the relative phase of the frequency components at the origin ($x,y=0$)

2D Fourier transform - Properties

- **Linearity**: $\mathcal{F}\{af+bg\}(x,y) = aF(p,q)+bG(p,q)$ where $\mathcal{F}\{g\}=G$
- **Time reversal**: $\mathcal{F}\{f\}(-x,-y) = F(-p,-q)$
- **Similarity** (scaled signal): $\mathcal{F}\{f\}(ax,by) = \frac{1}{ab}F(\frac{p}{a},\frac{q}{b})$ with $a,b>0$
- **Rotation**: Let $[\hat{x},\hat{y}]^T = R[x,y]^T$ and $[\hat{p},\hat{q}]^T = R[p,q]^T$ where R is a rotation matrix. It results $\mathcal{F}\{f\}(\hat{x},\hat{y}) = F(\hat{p},\hat{q})$. A rotation of f by an angle θ implies that also F is rotated by the same angle.
- **Translation (circular shift)**: $\mathcal{F}\{f\}(x-a,y-b) = F(p,q)e^{-i2\pi(ap+bq)}$. When using DFT (as for images) this is a circular shift !
- **Convolution theorem**: $\mathcal{F}\{f*g\}(x,y) = F(p,q)G(p,q)$ where $*$ means convolution
- **Cross-correlation theorem**: $\mathcal{F}\{f\star g\}(x,y) = F^{*}(p,q)G(p,q)$ where * means complex conjugate (equal real part, opposite sign for the imaginary part)

2D Fourier transform - Only translation

- Let's see how to use the previous properties to compute the parameters of a rigid registration ($\Theta = \{t_x, t_y, \theta, s\}$)
- Let's start from the translation
- Let f and g be two $[N \times M]$ images that differ only by a displacement in both dimensions $a, b \in \mathcal{R}$:

$$g(x,y) = f(x-a, y-b) \tag{26}$$

- From the translation (shift) property we know that their corresponding Fourier transforms F and G are related by:

$$G(p,q) = F(p,q)e^{-i2\pi(ap+bq)} \tag{27}$$

2D Fourier transform - Only translation

- If we compute the cross correlation between the two images, we obtain:

$$\begin{aligned}\mathcal{F}\{g \star f\}(x,y) &= G^*(p,q)F(p,q) = F^*(p,q)F(p,q)e^{i2\pi(ap+bq)} = \\ &= |F(p,q)|e^{-i\angle F(p,q)}|F(p,q)|e^{i\angle F(p,q)}e^{i2\pi(ap+bq)} = \\ &= |F(p,q)|^2 e^{i2\pi(ap+bq)}\end{aligned} \tag{28}$$

- where we have used the fact that $F^*(p,q) = |F(p,q)|e^{-i\angle F(p,q)}$
- From Eq.28, we can see that the phase difference term $e^{i2\pi(ap+bq)}$ is weighted by the magnitude. This can bias the analysis. We would like to only have the phase difference term $\rightarrow$ Need for a normalization term !

2D Fourier transform - Only translation

- Instead than using the cross correlation, we consider the normalized cross power spectrum (or phase correlation):

$$\text{PC} = \frac{G(p,q)F^*(p,q)}{|G(p,q)F^*(p,q)|} = e^{-i2\pi(ap+bq)} \tag{29}$$

- In this way, we can normalize the estimate and eliminate the bias of the magnitude.
- By computing the inverse Fourier transform we obtain

$$\mathcal{F}^{-1}\{\text{PC}\} = \mathcal{F}^{-1}\{e^{-i2\pi(ap+bq)}\} = \delta(x-a, y-b) \tag{30}$$

- where we used the shift property and the fact that $\mathcal{F}\{\delta\}(p,q) = \int_{\mathcal{R}}\int_{\mathcal{R}} \delta(x,y)e^{-i2\pi(px+qy)}dxdy = 1$

2D Fourier transform - Only translation

- Theoretically, $\mathcal{F}^{-1}\{\text{PC}\}$ is a matrix that should be equal to 1 exactly at $x = a$ and at $y = b$ and 0 elsewhere
- From a practical point of view, due to noise or other transformations, we usually do not have only one peak and therefore we look for the **location in the matrix with the greatest peak**

$$(a, b) = \arg\max_{(x,y)} \mathcal{F}^{-1}\{\text{PC}\} \tag{31}$$

- Interpolation schemes can then be used to estimate the peak location at non-integer locations. This is called *subpixel registration*
- **WARNING**: Note that in practice we use DFT and thus g is modeled as a circularly-shifted version of f, that is, the part of g that are translated after one side, reappears on the opposite side

2D Fourier transform - Only translation

- Recipe for estimating only the translation given $f(x,y)$ and $g(x,y) = f(x-a, y-b)$
  1. Compute $F(p,q)$ and $G(p,q)$ using FFT
  2. Compute PC using Eq.29
  3. Compute the inverse FFT of PC
  4. Use Eq.31 to find the translation parameters a and b
- Please note that, due to the circular shift property, a negative value of a (resp. b) is equivalent to $N + a$ (resp. $M + b$)
- N.B. The described method work even when we apply a linear shift (not circular) even though we always assume a circular shift ! This means that, if you don't know whether g has been transformed using a circular or linear shift, you will have to check that. We'll see that in the TP !

2D Fourier transform - Rotation

- For rotations, it's easier to switch to polar coordinates. We have:

$$\text{Spatial} \begin{cases} x = r\cos(\theta) \\ y = r\sin(\theta) \end{cases} \Leftrightarrow \text{Polar} \begin{cases} r = \sqrt{x^2+y^2} \\ \theta = \text{arctan2}(y,x) \end{cases}$$

$$\text{Frequency} \begin{cases} p = w\cos(\phi) \\ q = w\sin(\phi) \end{cases} \Leftrightarrow \text{Polar} \begin{cases} w = \sqrt{p^2+q^2} \\ \phi = \text{arctan2}(q,p) \end{cases}$$

- where $arctan2$ is a variation of $\tan^{-1}$. From the Rotation property, we have:

$$\mathcal{F}\{f\}(r,\theta+\alpha) = F(w,\phi+\alpha) \tag{32}$$

- thus a rotation applied in the spatial coordinates implies the same rotation in the frequency coordinates

2D Fourier transform - Rotation

- Let f and g be two $[N \times M]$ images that differ only by a rotation of angle α wrt the origin $(x = 0, y = 0)$:

$$g(x, y) = f(x\cos(\alpha) + y\sin(\alpha), -x\sin(\alpha) + y\cos(\alpha)) \tag{33}$$

- or in polar coordinates

$$g(r, \theta) = f(r, \theta - \alpha) \tag{34}$$

- From the rotation property we know that their corresponding Fourier transforms F and G are related by:

$$G(w, \phi) = F(w, \phi - \alpha) \tag{35}$$

- where also the rotation of an angle α in the frequency domain is computed wrt to the origin $(p = 0, q = 0)$

2D Fourier transform - Rotation

- If we consider only the magnitudes of F and G, we obtain:

$$|G(w,\phi)| = |F(w,\phi-\alpha)| \tag{36}$$

- in polar coordinates, a rotation can actually be seen as a translation !
- Use the phase correlation method as before by computing the Fourier transform of $|G(w,\phi)|$ and $|F(w,\phi-\alpha)|$
- Be careful that we compute the angle with respect to the origin in both coordinates (spatial and frequency!) $\rightarrow$ we usually center images in both domains to have the origin in the middle of the image
- WARNING: if f is a real image, the spectral magnitude in polar coordinates is a periodic (cyclical) function of the angle θ with a period of π. The real rotation angle might thus be α or $\alpha+\pi$. Need to check both images !

- What about scaling ? Remember the scaling property of the Fourier transform:

$$\mathcal{F}\{f\}(ax, by) = \frac{1}{ab} F(\frac{p}{a}, \frac{q}{b}) \quad \text{with } a, b \in \mathcal{R}^+ \tag{37}$$

- Let f and g be two $[N \times M]$ images that differ only by a scaling $a, b \in \mathcal{R}^+$:

$$g(x, y) = f(ax, by) \Leftrightarrow G(p, q) = \frac{1}{ab} F(\frac{p}{a}, \frac{q}{b}) \tag{38}$$

- if we convert the coordinates to a logarithmic scale, we obtain

$$G(\ln(p), \ln(q)) = F(\ln(p) - \ln(a), \ln(q) - \ln(b)) \tag{39}$$

- where we ignored the multiplication factor $\frac{1}{ab}$

2D Fourier transform - Scaling

$$G(\ln(p), \ln(q)) = F(\ln(p) - \ln(a), \ln(q) - \ln(b)) \tag{40}$$

- Using $z = \ln(p)$, $t = \ln(q)$, $c = \ln(a)$ and $d = \ln(b)$, we obtain:

$$G(z, t) = F(z - c, t - d) \tag{41}$$

- and, again, we can obtain the scaling parameters $a = \exp(c)$ and $b = \exp(d)$ using the phase correlation technique on the magnitudes of $G(z, t)$ and $F(z - c, t - d)$. Note that the multiplication factor $\frac{1}{ab}$ would disappear when computing the phase correlation, so we can ignore it.

2D Fourier transform - Rotation and Scaling

- What if two images are actually both rotated and scaled ? Can we retrieve the correct parameters ? It turns out that we can only if the image $g(x,y)$ has been scaled using a uniform scaling ($a=b$). Let's see why !
- First, let's define $g(x,y)=f(ax,by)$ in polar coordinates

$$r=\sqrt{(ax)^2+(by)^2}$$
$$\theta=\text{arctan2}(by,ax)$$

- Now let's see how it would change if we were using a uniform scaling

$$r_g=\sqrt{(ax)^2+(ay)^2}=a\sqrt{x^2+y^2}=ar_f$$
$$\theta_g=\text{arctan2}(ay,ax)=\text{arctan2}(y,x)=\theta_f$$

- We can see that using a uniform scaling, there is a simple linear relationship between the polar coordinates of g and f which is not the case when using $a\neq b$

2D Fourier transform - Rotation and Scaling

- Let f and g be two $[N \times M]$ images that differ by a rotation of angle α wrt the origin $(x = 0, y = 0)$ and a uniform scaling $s \in \mathcal{R}^+$:

$$g(x, y) = f(sx\cos(\alpha) + sy\sin(\alpha), -sx\sin(\alpha) + sy\cos(\alpha)) \quad (42)$$

- which can be rewritten in polar coordinates as:

$$g(r, \theta) = f(sr, \theta - \alpha) \quad (43)$$

- Using the rotation and scaling property of the Fourier transform, transforming the first coordinate in the log scale and ignoring $\frac{1}{s^2}$, we obtain:

$$G(w, \phi) = F(\frac{w}{s}, \phi - \alpha) = F(\ln(w) - \ln(s), \phi - \alpha) \quad (44)$$

2D Fourier transform - Rotation and Scaling

$$G(w, \phi) = F(\ln(w) - \ln(s), \phi - \alpha) = F(\xi - c, \phi - \alpha) \tag{45}$$

- We can thus still use the phase correlation technique on the magnitudes of G and F and retrieve the rotation and scaling parameters α and $s = \exp(c)$
- Please note that using a uniform scaling s, this affects only the distance r (and thus w) and not the angle θ (thus ϕ) and thus it does not interfere wit the estimate of α. This would not be the case if we were using two different scaling for x and y

- Let's see now how to estimate all parameters together ($\Theta = \{a, b, \theta, s\}$).
- Let f and g be two $[N \times M]$ images that differ by a rotation of angle α wrt the origin $(x = 0, y = 0)$, a uniform scaling $s \in \mathcal{R}^+$ and a translation in both directions $a, b \in \mathcal{R}$:

$$g(x, y) = f(sx\cos(\alpha) + sy\sin(\alpha) - a, -sx\sin(\alpha) + sy\cos(\alpha) - b) \quad (46)$$

- Using the rotation, scaling and shift property of the Fourier transform, we obtain:

$$G(p, q) = \frac{1}{s^2} F(\frac{p\cos(\alpha) + q\sin(\alpha)}{s}, \frac{-p\sin(\alpha) + q\cos(\alpha)}{s})e^{-i\angle G(p,q)} \quad (47)$$

2D Fourier transform - Rotation + Saling + Translation

$$G(p,q) = \frac{1}{s^2} F\left(\frac{p\cos(\alpha) + q\sin(\alpha)}{s}, \frac{-p\sin(\alpha) + q\cos(\alpha)}{s}\right) e^{-i\angle G(p,q)} \tag{48}$$

- Please note that $\angle G(p,q)$ is the phase of g which depends on translation, scaling and rotation
- We can then notice that the translation does not affect the magnitude. If we first work only with the magnitude we can estimate the scaling and rotation parameters as before
- Once estimated, we can first transform f into g_1 which will take into account only scaling and rotation differences and then use g_1 to estimate the translation parameters

2D Fourier transform - Rotation + Scaling + Translation

- Please note that the polar-log mapping of the spectral magnitude corresponds to the **Fourier-Mellin transform**
- WARNING: if f is a real image, the spectral magnitude in polar coordinates is a periodic (cyclical) function of the angle θ with a period of π. The real rotation angle might thus be α or $\alpha + \pi$. Need to check both images !
- One should therefore first scale f using the estimated s then rotate the resulting image using both α and $\alpha + \pi$ and use both images to estimate the translation.
- The magnitude of the peaks or the final transformed images can then be used to choose the correct set of parameters for rotation and translation
- For more information, please refer to [10,11,12]

2D Fourier transform - Considerations

- Using the FFT (Fast Fourier Transform) computational efficiency is higher than using optimization of the correlation coefficient based on spatial coordinates
- Robust to variations/disturbances in intensity (i.e. contrast, brightness, illumination, etc)
- Robust to frequency-dependent noise
- Sensitive to:
    1. frequency-independent noise/deformations spread across all frequencies
    2. aliasing due to rotation (*solution*: when transforming the magnitude into log-polar coordinates, convert only frequencies close to the center (e.g. $-N/2 - N/2$))
    3. aliasing due to rotation at the image borders (*solution*: use a Blackman-Harris window operation on the original images f and g before taking their Fourier transforms)

References

1. J. Modersitzki (2004). *Numerical methods for image registration*. Oxford university press
2. J. V. Hajnal, D. L.G. Hill, D. J. Hawkes (2001). *Medical image registration*. CRC press
3. G. Wolberg (1990). Digital Image Warping. IEEE
4. The Matrix Cookbook
5. S. Umeyama (1991). Least-squares estimation of transformation parameters.... IEEE TPAMI
6. S. Durrleman et al. (2014). Morphometry of anatomical shape complexes ... . NeuroImage.
7. J. Ashburner (2007). A fast diffeomorphic image registration algorithm. NeuroImage
8. J. P. W. Pluim (2003). Mutual information based registration of medical images: a survey. IEEE TMI
9. S. Baker, I. Matthews (2004). Lucas-Kanade 20 Years On: A Unifying Framework. IJCV

References

10. B. Srinivasa Reddy and B. N. Chatterji (1996). An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration. IEEE TIP
11. Qin-Sheng Chen et al. (1994) Symmetric Phase-Only Matched Filtering of Fourier-Mellin Transforms for Image Registration and recognition. IEEE TPAMI
12. Stone et al. (2004). Analysis of image registration noise due to rotationally dependent aliasing. Journal of Vis Comm and Im Repr
13. Ruben Gonzalez (2011). Improving Phase Correlation for Image Registration. IEEE IVC