Introduction to Image Registration

concepts, methods and optimizations

Feb, 2015

Minyoung Chung
Computer Graphics and Image Processing Laboratory
Seoul National University

Contents

- Introduction to Image Registration
- Feature-Based Registration
- Intensity-Based Similarity Measurements
- Basic Optimization Methods
- Analytic Transform Parameter Optimization

Introduction to Image Registration

- Image Registration Process
- Transformations
- Implementation of Transformations

Image Registration

- Definition
  - Determine a spatial transformation that relates positions in one image to corresponding positions in one or more other images
  - Align into the same spatial coordinate

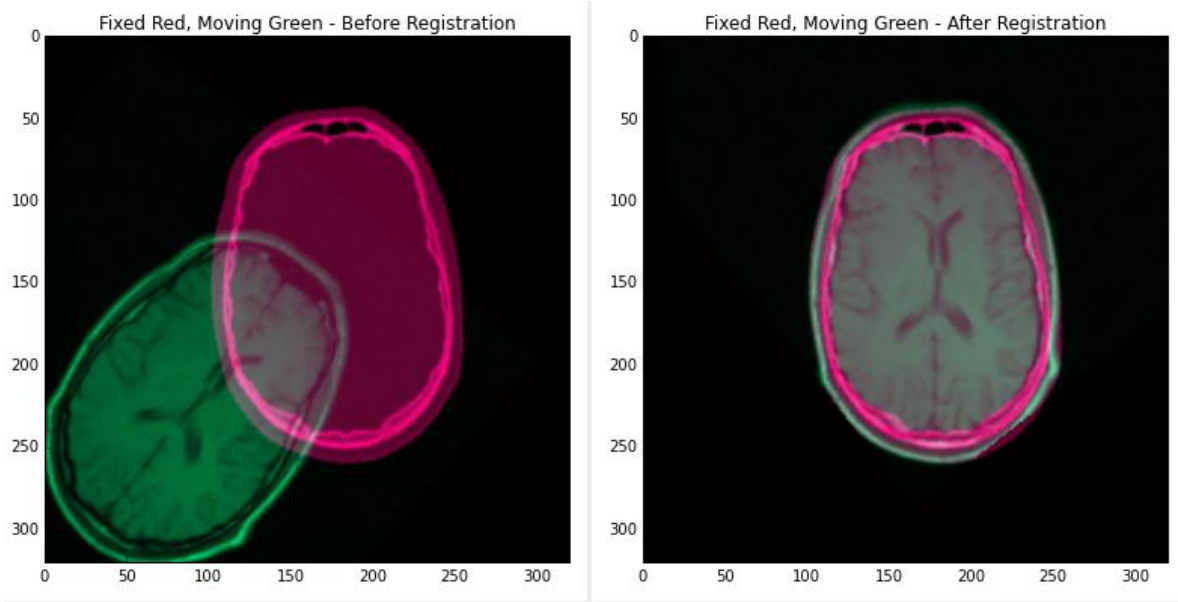
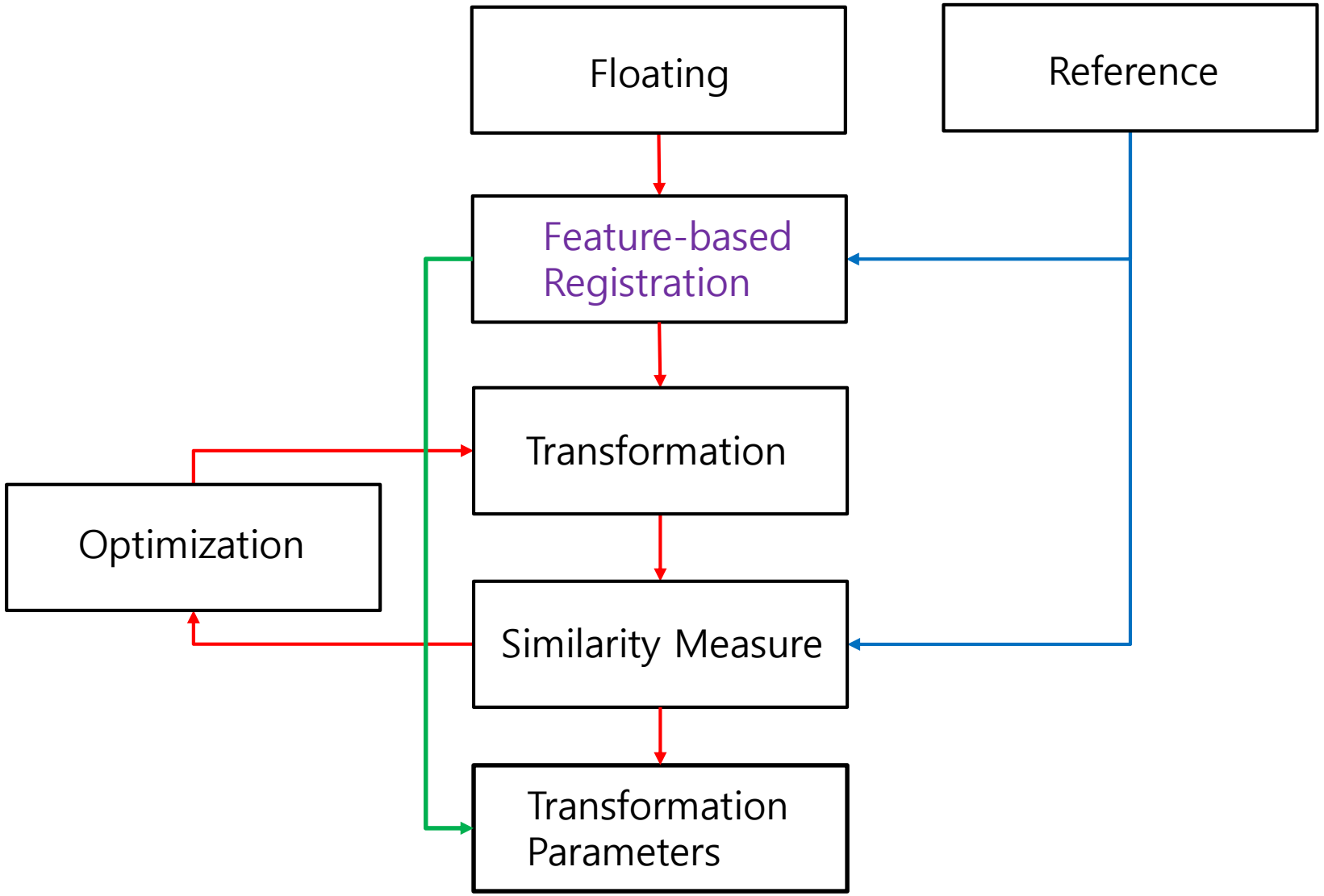


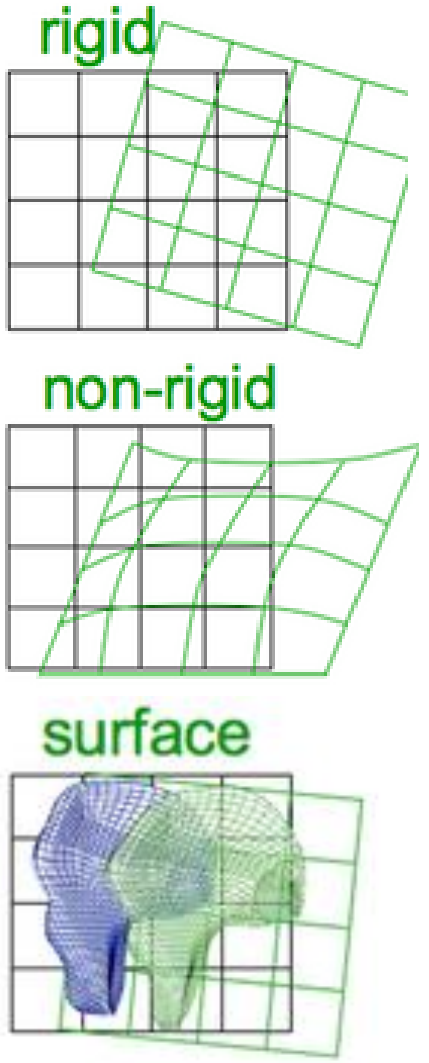
Image registration:
the CT scan (red) and the MRI scan (green) are registered in order to be combined in a single image.

Image Registration Process



Transformations

- Rigid registration
  - translation
  - rotation
- Non-rigid registration
  - affine transform (scaling, shearing)
  - curved transform
  - (free-deformations)
- Surface registration



Transformations (Rigid and Affine)

- $T(\boldsymbol{p};\ \boldsymbol{\Theta}) = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix} (\boldsymbol{p}) + \begin{pmatrix} \theta_{14} \\ \theta_{24} \\ \theta_{34} \end{pmatrix}$
  - where $\boldsymbol{p}$ is a vector of position,
  - $\boldsymbol{\Theta}$ parameterize the 12 degrees of freedom of transformation.
  - homogeneous form can be used.

Transformations (B-Spline Free-Form Deformation)

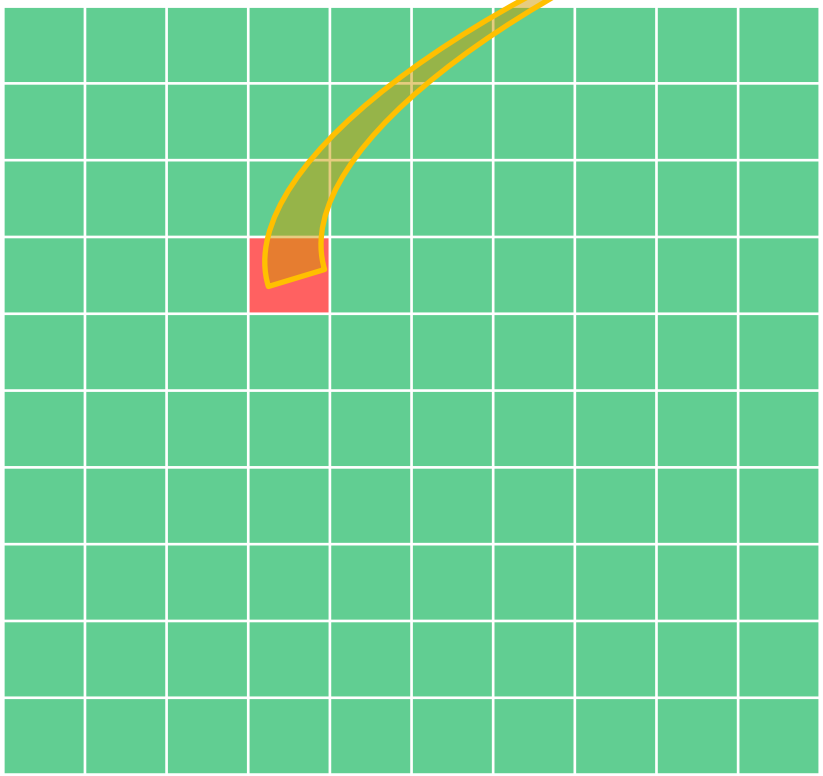
- $T(\boldsymbol{p};\ \boldsymbol{\Phi}) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(u)\, B_m(v) B_n(w) \boldsymbol{\phi}_{i+l,j+m,k+n}$
  - Domain of the image volume $\boldsymbol{\Omega} = \{(x,y,z) |\ 0 \le x < X, 0 \le y \le Y, 0 \le z < Z\}$
  - $\boldsymbol{\Phi}$ denote a $n_x \times n_y \times n_z$ mesh of control points $\phi_{i,j,k}$.
    - $n_x \times n_y \times n_z$: size of sub-voxels between control points.
  - where $i = \left\lfloor \frac{x}{n_x} \right\rfloor - 1, j = \left\lfloor \frac{y}{n_y} \right\rfloor - 1, k = \left\lfloor \frac{z}{n_z} \right\rfloor - 1,$
  - $u = \frac{x}{n_x} - \left\lfloor \frac{x}{n_x} \right\rfloor, v = \frac{y}{n_y} - \left\lfloor \frac{y}{n_y} \right\rfloor, w = \frac{z}{n_z} - \left\lfloor \frac{z}{n_z} \right\rfloor$
  - B_l represents the lth basis function of B-spline.
  - *local control property*

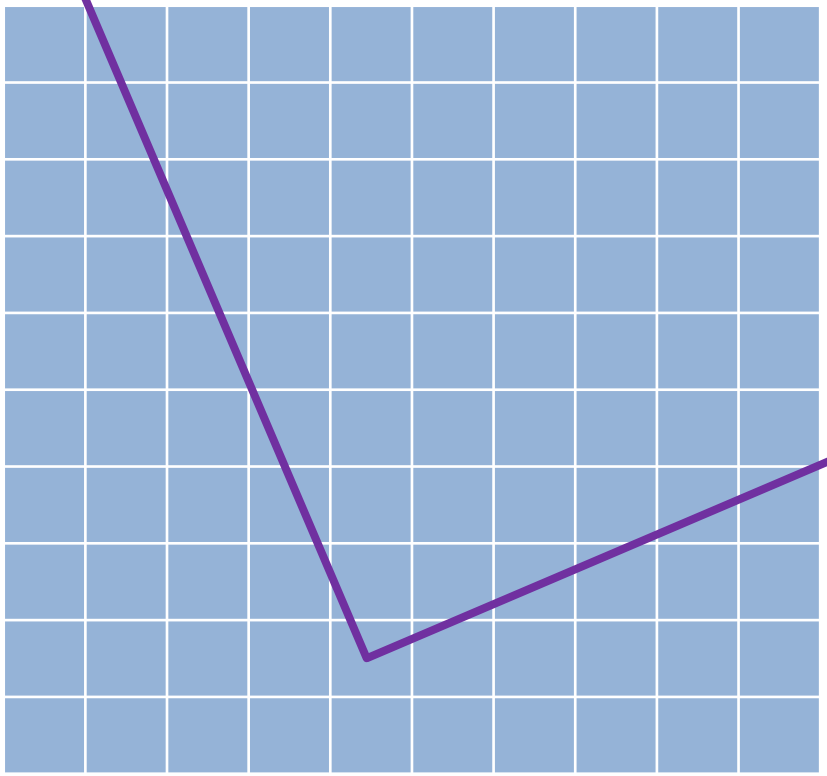
Implementation of Transformations

- Forward-mapping ?

$$I'_F(p) = I_F(T(p;\ \Theta))$$

Write to floating position ?

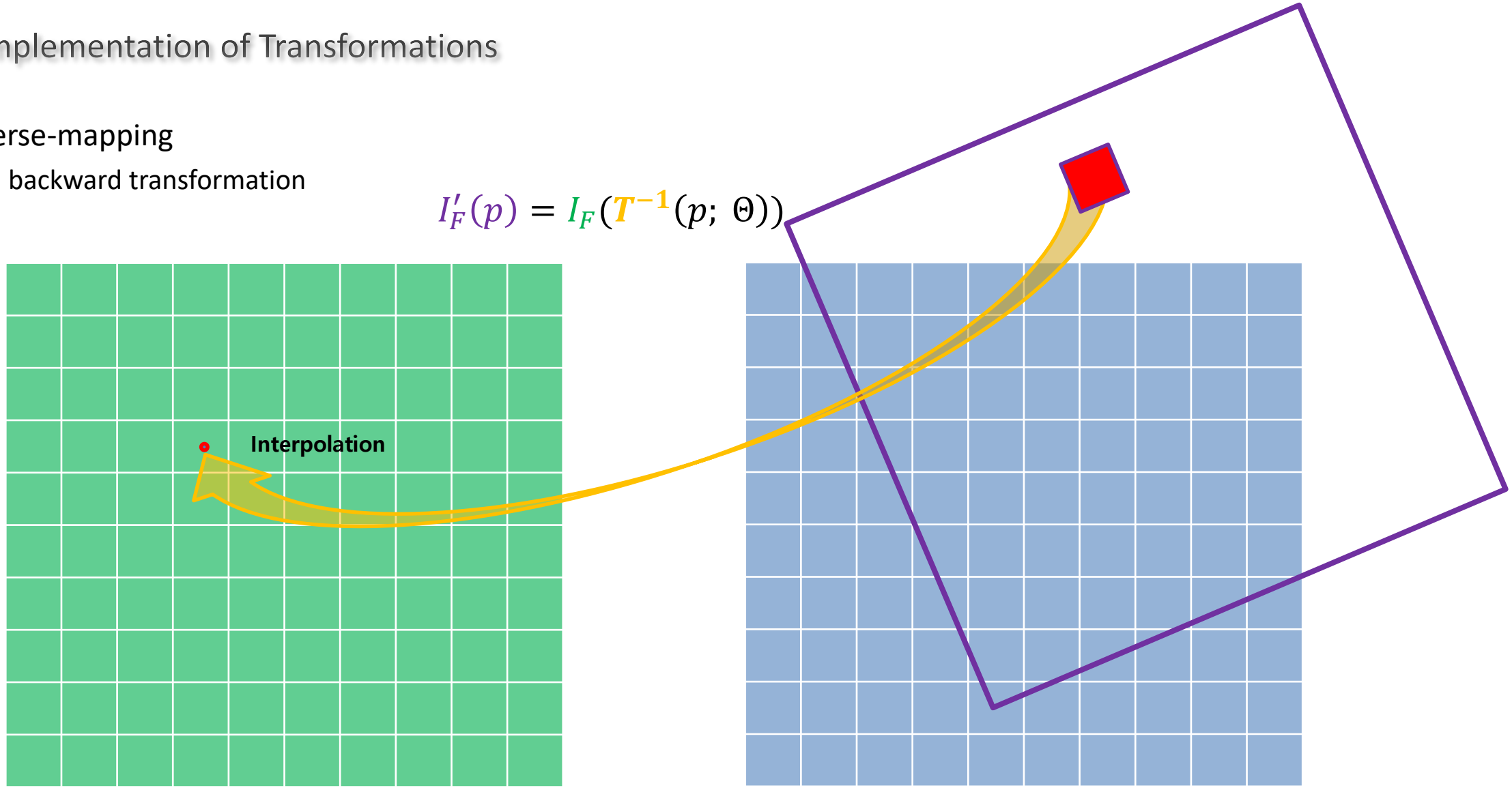




Implementation of Transformations

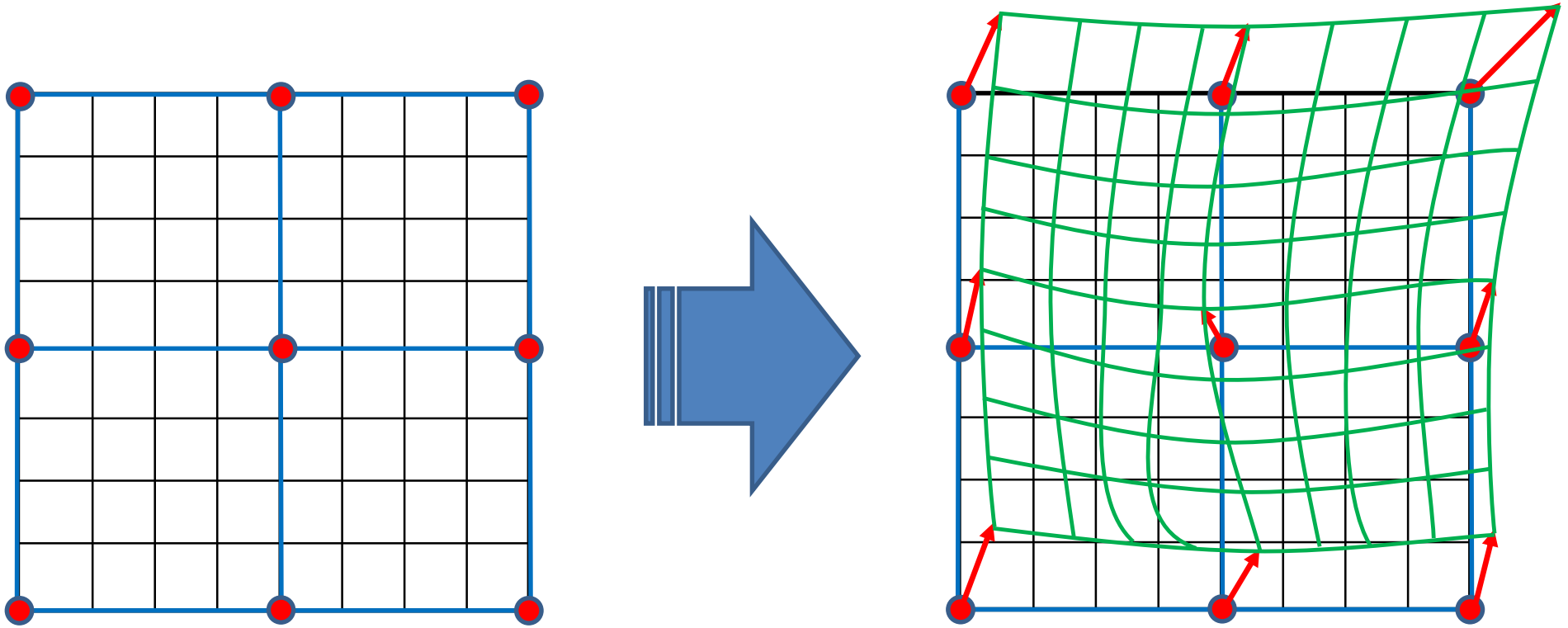
- Inverse-mapping
  - backward transformation

$$I'_F(p) = I_F(T^{-1}(p;\ \Theta))$$



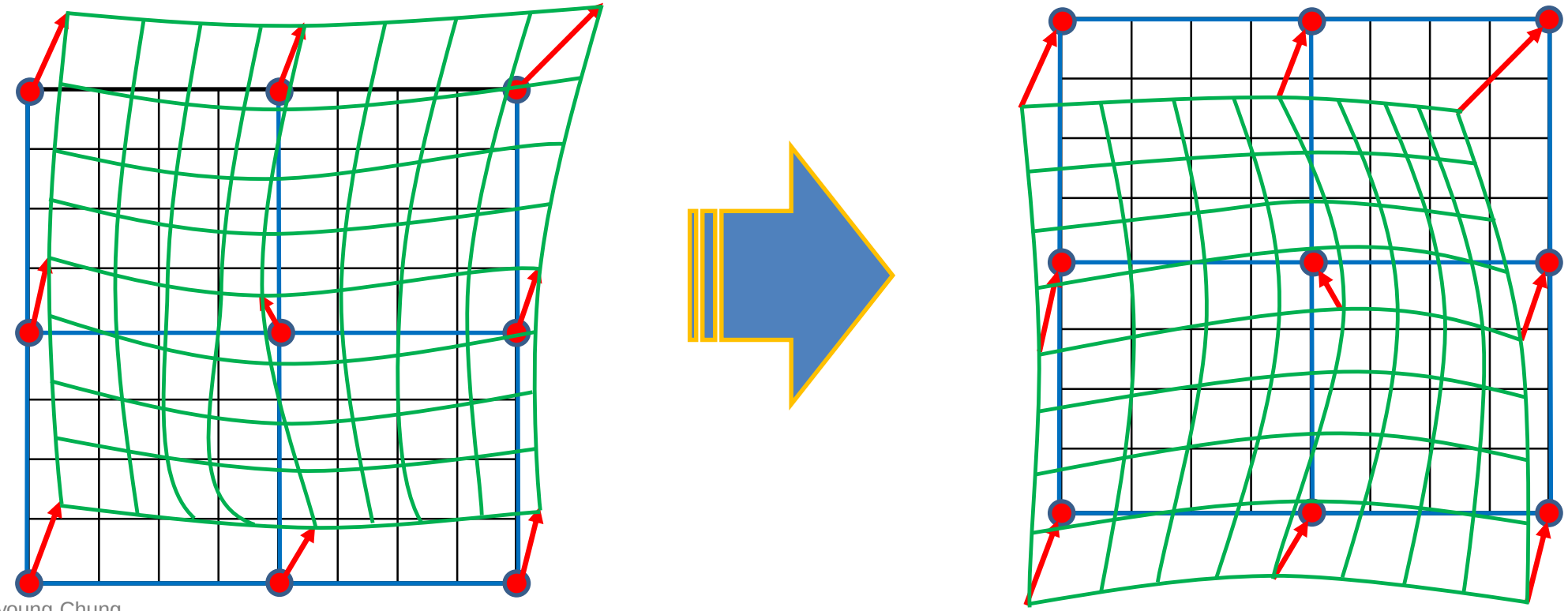
Implementation of Transformations

- B-spline deformation
  - *actually, four control points are needed to interpolate*



Implementation of Transformations

- Inverse of B-spline deformation
  - actual deformation is not calculated by these displacements (interpolated positions of voxels)
  - negative direction of control point's displacement is used for inverse-mapping
    - it means negative direction of every voxel's displacement



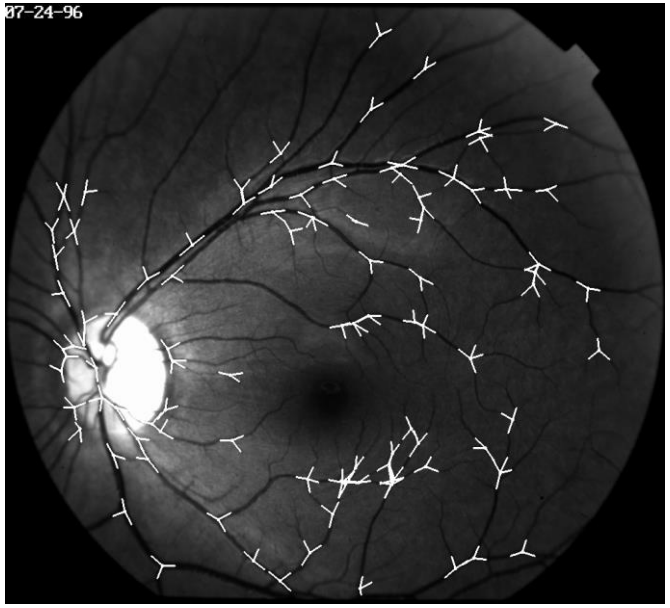
Feature-Based Registration

- Feature-Based Registration
- Point-based Method
- Iterative Closest Point
- Principal Component Analysis (PCA)
- Principal Axes Registration (PAR)

Feature-Based Registration

- Various features can be extracted from an image
  - discrete sets of point locations may have additional properties such as intensities and orientations
  - specific-featured points can be used as features
  - surface
  - segmented binary volume
  - …

- Feature-based registration problem :
  - Finding correspondences between features
  - Estimating the transformation parameters based on these correspondences
  - Reliably extracting features in all images
  - *Point-based or Surface-based registration*
    - commonly, matching feature positions
    - free transformations (rigid, non-rigid) : depend on parameters



Point-based Method

- Set of floating (moving) image features
  - $\mathcal{G} = \{g_i\}$
- Set of reference (fixed) image features
  - $\mathcal{F} = \{f_i\}$
- Each feature must include a point location in the coordinate system of its image
  - it may include more
- Set of correspondences
  - $\mathrm{C} = \{(g_k, f_k)\}$
- $E(\Theta;\ \mathrm{C}) = \sum_{(g_k, f_k)\in \mathrm{C}} [D(T(g_k;\ \Theta), f_k)]^2$
  - error objective function depends on
    - unknown transformation parameters,
    - unknown feature correspondences ?
  - transformation may include mapping of more than just locations
  - distance function, D, could be as simple as the Euclidean distance between location vectors
- Surface can be represented as set of large points (surface-based registration)

Least-Square Method (approximately solve $y = Ax$)

- Define residual or error $r = Ax - y$
- Find $x = x_{ls}$ that minimizes $\|r\|$
  - minimize norm of residual squared:
    - $\|r\|^2 = x^T A^T Ax - 2y^T Ax + y^T y$
    - square of $L^2 - norm$
  - set gradient w.r.t. x to zero:
    - $\nabla_x \|r\|^2 = 2A^T Ax - 2A^T y = 0$
  - normal equations:
    - $A^T Ax = A^T y$
  - assumptions imply $A^T A$ invertible,
    - $x_{ls} = \left(\boldsymbol{A^T A}\right)^{-1} \boldsymbol{A^T} y$
    - $\left(\boldsymbol{A^T A}\right)^{-1} \boldsymbol{A^T}$ = pseudo-inverse of A (A^+)
    - $A^+ = V\Sigma^+ U^T$ (using *SVD of* $A = U\Sigma V^T$, Σ^+ : reciprocal & transpose of Σ)
- x_{ls} is called least-squares (approximate) solution of $y = Ax$

Point-based Registration (assuming correspondences are given)

- 2D point locations: $g_k = (x_k, y_k), \quad f_k = (u_k, v_k)$
- Similarity transformation: $T(g_k;\ \Theta) = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} g_k + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$
- Euclidean distance: $D(T(g_k;\ \Theta), f_k) = \|T(g_k;\ \Theta) - f_k\|$
- $E(\Theta;\ C) = \sum_{(g_k, f_k)\in C} \left\| \begin{pmatrix} a & -b \\ b & a \end{pmatrix} g_k + \begin{pmatrix} t_x \\ t_y \end{pmatrix} - f_k \right\|^2$ (Least-square estimation)
  - $= \sum_{(g_k, f_k)\in C} \left\| \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} - \begin{pmatrix} u_k \\ v_k \end{pmatrix} \right\|^2$
  - $= \sum_{(g_k, f_k)\in C} \left\| \begin{pmatrix} ax_k - by_k + t_x - u_k \\ bx_k + ay_k + t_y - v_k \end{pmatrix} \right\|^2$
    - → matrix calculus
  - $= \sum_{(g_k, f_k)\in C} \left[(ax_k - by_k + t_x - u_k)^2 + (bx_k + ay_k + t_y - v_k)^2 \right]$
    - quadratic function of each parameter
    - → take the derivative with respect to each parameter & set the resulting gradient to 0 (vector)
    - → matrix inversion

$$Ax = b$$
$$x = pinv(A)B$$
$$x = (A^T A)^{-1} A^T B$$

Point-based Registration (Least-square estimation)

- $E(\Theta;\ C) = \sum_{(g_k,f_k)\in C} \left\| \begin{pmatrix} ax_k - by_k + t_x - u_k \\ bx_k + ay_k + t_y - v_k \end{pmatrix} \right\|^2$
- $E(\Theta;\ C) = \sum_{(g_k,f_k)\in C} \left\| \begin{pmatrix} x_k & -y_k & 1 & 0 \\ y_k & x_k & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ t_x \\ t_y \end{pmatrix} - f_k \right\|^2$
  - $= \sum \| \boldsymbol{X_k} \Theta - f_k \|^2$
  - $= \sum (X_k \Theta - f_k)^T (X_k \Theta - f_k)$
  - $= \Theta^T \sum {X_k}^T X_k \Theta - 2\Theta^T \sum X^T f_k + \sum {f_k}^T f_k$
- $\frac{\partial E}{\partial \Theta} = \Theta^T 2 \sum \left({X_k}^T X_k \right) - 2 \sum {f_k}^T X_k = 0$
- $\Theta = \left[\sum {X_k}^T X_k \right]^{-1} \left[\sum {X_k}^T f_k \right]$

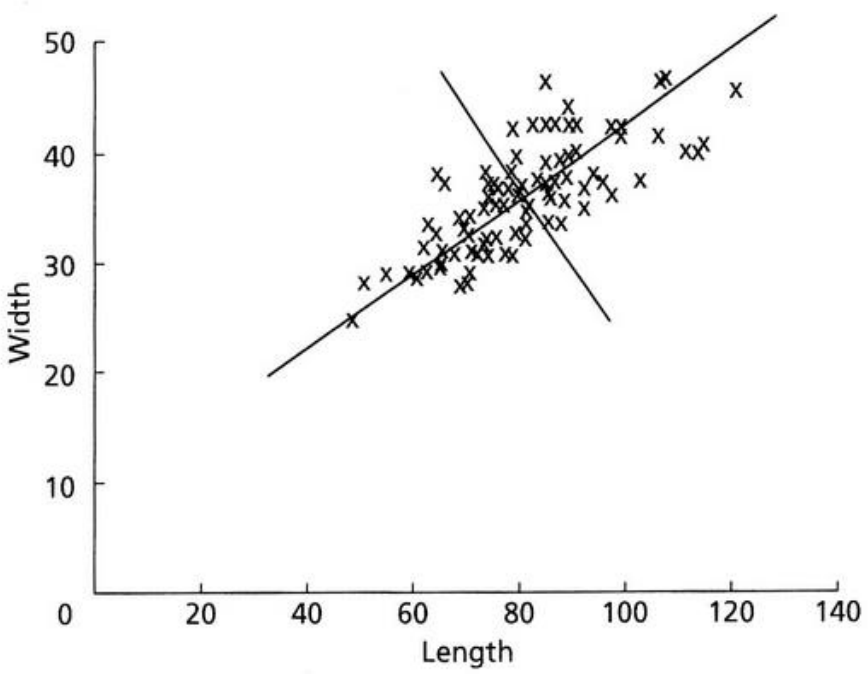
Iterative Closest Point (ICP)

- Minimize the Euclidean distance between point-correspondences
- Joint estimation of correspondences and parameters
  - unknown correspondences
- Given an initial transformation estimate Θ_0 (*very important*)
- $t = 0$
- Iterative procedure to converge to local minima:
  - Establish correspondences (find closest point):
    - $f_k = \underset{f \in \mathcal{F}}{\operatorname{argmin}} D(g'_k, \boldsymbol{f})$ // enforce unique correspondences
    - for fixed transformation parameter estimate, Θ_t, apply the transformation to each moving image feature
    - for each point in the moving image feature, find the closest fixed image feature
  - Estimate the new transformation parameters:
    - for the resulting correspondences, estimate Θ_{t+1}
  - Map feature into coordinate system of I_f
    - $g'_k = \boldsymbol{T}(g_k;\ \Theta_t)$
- Choose the best among found solutions for different initial positions

Principal Component Analysis

- PCA produces a single best line that satisfies the following conditions:
  - the sum of the squares of the perpendicular distances from the sample points to the line is a minimum
  - the first principal component indicates the greatest amount of variation
  - the second principal component is orthogonal with the first

- EMP (Equivalent Meridian Plane)
  - contains the first and the second principal axis
  - distances of points from the plane is a minimum
  - *EMP based MI registration method (EMP-MI)*
    - 2D-plane based registration



Principal Axes Registration

- Threshold
  - or various segmentations
- Point-features extraction
- Surface extraction
- (points, edges, contours …)
- others

Features

Good features need to be extracted for accurate result

Find Principal Axes by PCA

Principal Axes Registration

- $X = \{(x_i, y_i, z_i)^T \mid i = 1, \dots, n\}$
  - each pixel in the object(binary volume) is treated as a 3D-vector
  - n : total number of points
  - x_i, y_i, z_i : coordinate values of that pixel

- $u = \frac{1}{n}\sum_{i=1}^{n} X_i$
  - mean vector

- $C = \frac{1}{n}\sum_{i=1}^{n} X_i {X_i}^T - uu^T$
  - covariance matrix
  - symmetric matrix → Eigen decomposition (→ orthonormal basis of eigenvectors for R^n)
  - or perform *Singular Value Decomposition*

- $Cov(X, Y)$
  $= E[(X - E[X])(Y - E[Y])]$
  $= E[X \cdot Y] - E[X] \cdot E[Y]$
- $Cov(X, Y) = Cov(Y, X)$

orthonormal basis of eigenvectors for R^n → rotation matrix

Principal Axes Registration

- If **E** is arranged in the order so that the first row E is the eigenvector that corresponds to the largest eigenvalue of C and the last row corresponds its smallest eigenvalue,
  - $\boldsymbol{Y} = \boldsymbol{E}(\boldsymbol{X} - \boldsymbol{u})$
  - PCA transform
- The effect of the PCA transform
  - the centroid of the volume is translated to the origin of the global coordinate system (translation)
  - the principal axes of the volume will be coincident with the x-, y- and z- coordinates axes (rotation)
- $E = \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix}$
  - rotation angles with respect to the x-, y- and z- axes

$$\theta_y = \sin^{-1}(e_{31})$$
$$\theta_x = \sin^{-1}(-e_{32}/\cos\theta_y)$$
$$\theta_z = \sin^{-1}(-e_{21}/\cos\theta_y)$$

$$R = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_y\cos\theta_z & -\cos\theta_x\sin\theta_z+\sin\theta_x\sin\theta_y\cos\theta_z & \sin\theta_x\sin\theta_z+\cos\theta_x\sin\theta_y\cos\theta_z \\ \cos\theta_y\sin\theta_z & \cos\theta_x\cos\theta_z+\sin\theta_x\sin\theta_y\sin\theta_z & -\sin\theta_x\cos\theta_z+\cos\theta_x\sin\theta_y\sin\theta_z \\ -\sin\theta_y & \sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y \end{bmatrix}$$

Principal Axes Registration

- Determining the rotation matrix (R)

$$\boxed{R = E_r {E_f}^T}$$

- E_r : matrix of orthonormal eigenvectors (reference's principal axes)
- E_f : matrix of orthonormal eigenvectors (floating's principal axes)

Principal Axes Registration

- Fast computation
- Rigid transform invariant
- Work with small number of points
- Inaccurate when eigenvalues are similar (lengths of principal axes)

Feature Vector Extraction (why eigenvectors ?)

- $X = \{s_1, s_2, \cdots, s_N\}$
- u : arbitrary axis (variable)

- $\bar{\hat{x}} = \frac{1}{N}\sum_{i=1}^{N} \hat{x}_i = \frac{1}{N}\sum_{i=1}^{N} u^T s_i = u^T \left(\frac{1}{N}\sum_{i=1}^{N} s_i\right) = u^T \bar{s}$
  - mean of projection points to $\boldsymbol{u}$

- $\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} \left(\hat{x}_i - \bar{\hat{x}}\right)^2 = \frac{1}{N}\sum_{i=1}^{N} (u^T s_i - u^T \bar{s})^2$
  - variance of $\hat{x}_i$

- $\underset{u}{\operatorname{argmax}}\, \hat{\sigma}^2$

Feature Vector Extraction (why eigenvectors ?)

* Method of Lagrange multiplier
  - find local maxima (or minima) of a function *subject to* equality constraints

- maximize $f(x, y)$
- subject to $g(x, y) = c$

- $\Lambda(x, y, \lambda) = f(x, y) + \lambda(g(x, y) - c)$
  - λ : Lagrange multiplier

- $\nabla f = -\lambda \nabla g$
  - $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$
  - $\nabla g = \left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right)$
  - gradients of f and g are parallel
  - magnitude may differ (λ)

Feature Vector Extraction (why eigenvectors ?)

- maximize $f(u) = \hat{\sigma}^2$
- subject to $g(u) = u^T u = 1$
  - equality-constrained optimization problem → method of Lagrange multipliers

- $L(u) = \frac{1}{N}\sum_{i=1}^{N}(u^T s_i - u^T \bar{s})^2 + \lambda(1 - u^T u)$
  - λ : Lagrange multiplier

- $\underset{u}{\operatorname{argmax}}\, L(u)$

Feature Vector Extraction (why eigenvectors ?)

- $\frac{\partial L(u)}{\partial u} = \partial \left(\frac{1}{N} \sum_{i=1}^{N} (u^T s_i - u^T \bar{s})^2 + \lambda(1 - u^T u) \right) / \partial u$

  $= \frac{2}{N} \sum_{i=1}^{N} (u^T s_i - u^T \bar{s})(s_i{}^T - \bar{s}^T) - 2\lambda u^T$

  $= 2u^T \left(\frac{1}{N} \sum_{i=1}^{N} (s_i - \bar{s})(s_i{}^T - \bar{s}^T) \right) - 2\lambda u^T$

  $= 2u^T \boldsymbol{C} - 2\lambda u^T$ covariance matrix (**C**)

  $= 0$
- $\boldsymbol{Cu = \lambda u}$
- u is eigenvector of C
- λ is eigenvalue of C
- principal component vectors(axes) are eigenvectors of **C**
  - C : covariance matrix of input data

Intensity-Based Similarity Measurements

- Sum of Absolute Difference (SAD)
- Sum of Squared Difference (SSD)
- Normalized Cross-Correlation (NCC)
- Joint Entropy
- Mutual Information (MI)
- Normalized Mutual Information (NMI)
- *Histogram Binning*

Sum of Absolute Difference & Squared Difference

- $SAD = \frac{1}{N}\sum_i^N |R(i) - F'(i)|, \forall i \in R \cap F'$
- $SSD = \frac{1}{N}\sum_i^N |R(i) - F'(i)|^2, \ \forall i \in R \cap F'$
- R : reference image
- F : floating image
  - $F' = T(F)$
- Voxel-similarity measures are invariably calculated for the set of voxels in the overlapping region of R and F
  - $R \cap F'$
- If the images R and F being registered are identical, except for the misalignment
- Different intensity range ?

Normalized Cross-Correlation (Correlation Coefficient)

- $NCC = \frac{\sum_i (R(i)-\bar{R})(F'(i)-\bar{F'})}{\sqrt{\sum_i (R(i)-\bar{R})^2 \sum_i (F'(i)-\bar{F'})^2}}, \forall i \in R \cap F'$

  $= \frac{1}{n^2} \sum_i \frac{(R(i)-\bar{R})(F'(i)-\bar{F'})}{\sigma_r \sigma_{f'}}$

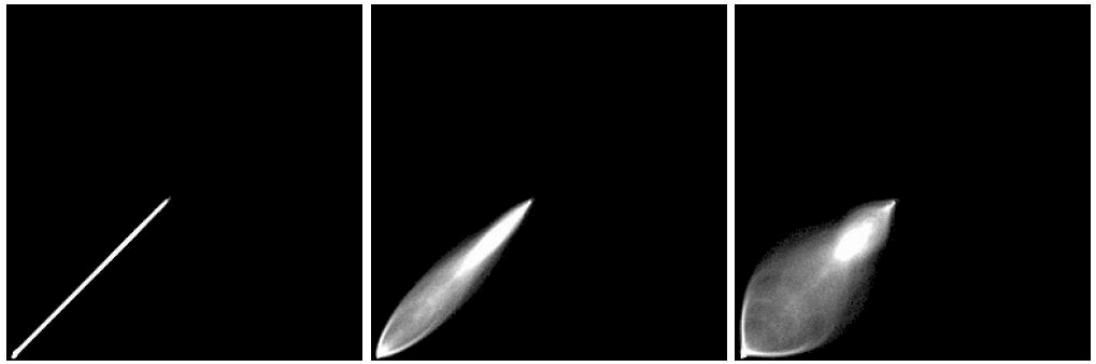
- $\bar{R}$ and $\bar{F}'$ are the mean values of voxels in image R and the transformed image F
- σ_r and $\sigma_{f'}$ are the standard deviation of the respective volumes
- calculate $\bar{R}, \sigma_r$ only with overlapped regions → more accurate result
- $-1 \leq NCC(R, F') \leq 1$
- good accuracy for mono-modal registration (linear relationship with intensity range)
- works bad in multi-modal registration

Joint Entropy

- Joint Probability Distribution Function (joint PDF) in Image
  1. Allocate $HIST[j,k]$
  2. Initialize the histogram: $HIST[j,k] = 0\ for\ all\ j,k$
  3. For each voxel $i \in R \cap F'$, increment $HIST[R(i), F'(i)]$
  4. Calculate $\sum_{j,k} HIST[j,k]$
  5. Normalize the histogram to calculate the PDF:

$$PDF[j,k] = \frac{HIST[j,k]}{\sum_{j,k} HIST[j,k]}$$

  - feature space
  - PDF changes with transformations
  - misregistration → dispersion ⇧



Identical MR images of the head

Joint Entropy

- $H = -\sum_{j,k} PDF[j,k] log PDF[j,k]$
  - from Shannon entropy (information theory)
    - measure of the uncertainty in a random variable (unpredictability)
  - dispersion of a probability distribution
  - ***Minimize*** H

$$H = -\sum_{s} p(s) log p(s)$$

- Available to multi-modal registration
  - not dependent on the actual values (only on the probabilities)

- Joint entropy is defined only for the overlapped region
  - contains large amount of air(noise) around the outside of the subject ?
  - → inter-background registration (very low joint-entropy)
  - → **Mutual Information**

Mutual Information (MI)

- ***Maximize*** $MI(R, F')$ as follows:
  1. Calculate $PDF[j,k]$ from images R and F'
  2. Calculate $H(R, F')$
  3. Calculate marginal entropies $H(R)$ and $H(F')$
     - $H(R) = -\sum_j PDF_R[j] log PDF_R[j]$
     - $H(F') = -\sum_j PDF_{F'}[j] log PDF_{F'}[j]$
  4. Calculate the mutual information $MI(R, F')$

$$MI(R, F') = H(R) + H(F') - H(R, F')$$

- Maximize marginal entropies for each image
- Minimize joint entropy
- Avoid *inter-background* registration
- High ratio of background in overlapped region ?
  - Entropy is a *relative* value
  - MI measure may actually increase while misregistration increase

Normalized Mutual Information (NMI)

- $NMI(R,F') = \frac{H(R)+H(F')}{H(R,F')}$

- $H(R) \rightarrow \frac{H(R)}{2}$
- $H(F') \rightarrow \frac{H(F')}{2}$
- $H(R,F') \rightarrow \frac{H(R,F')}{2}$

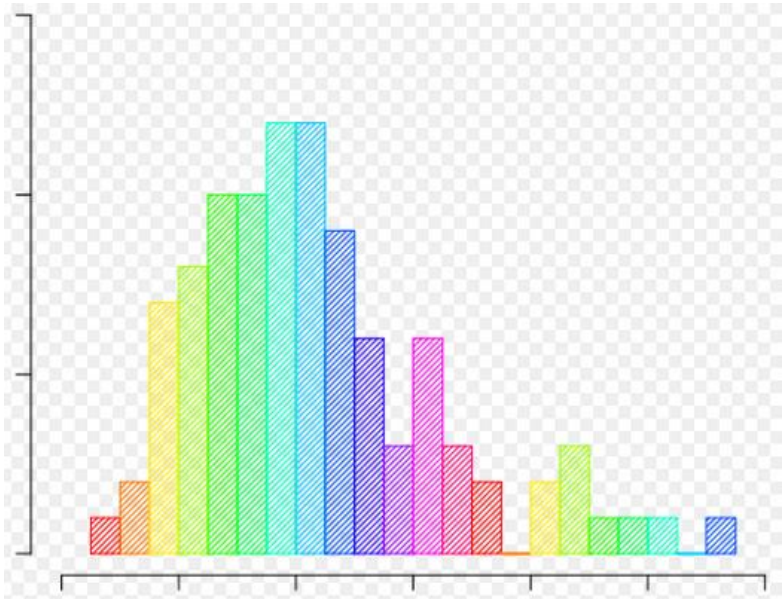
- $MI(R,F') \rightarrow \frac{MI(R,F')}{2}$
- $NMI(R,F') \rightarrow NMI(R,F')$

- Constant to ratio of object-to-background region
- Works at least as well as MI and in some cases performs better

Histogram Binning

Narrow bins

- high precision
- increase noise



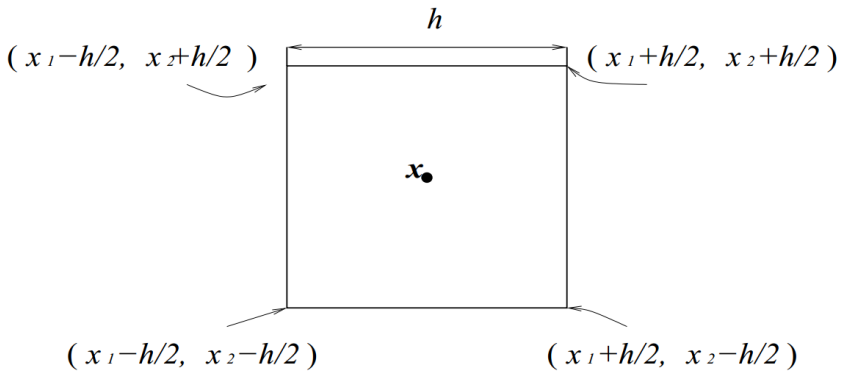
Wider bins

- reduce noise
- efficient
- sometimes more precise
- loss of information

- There is no “best” number of bins
- Different bin sizes can reveal different features of the data

Kernel Density Estimation (Parzen Windows Function)

- $P = \int_{\mathcal{R}} p(x)dx$
  - P is a probability that x is inside a region $\mathcal{R}$
  - If $\mathcal{R}$ is small, $P \approx p(x) \int_{\mathcal{R}} dx = p(x)V$. (where V is a "volume" of $\mathcal{R}$)
  - Let h be the length of the edge of a square region R. (in 2D case)
    - $V = h^2$



- $\phi\left(\frac{x_i - x}{h}\right) = \begin{cases} 1 & \frac{|x_{i_k} - x_k|}{h} < \frac{1}{2}, k = 1, 2 \\ 0 & otherwise \end{cases}$
  - window function
- k out of n samples falling within the region R,
  - $P = k/n$
  - $p(x) = \frac{k/n}{V} = \frac{k/n}{h^2}$
  - $k = \sum_{i=1}^{n} \phi\left(\frac{x_i - x}{h}\right)$

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^2} \phi\left(\frac{x_i - x}{h}\right)$$

✓ **window width h,**
✓ **kernel function ϕ**

Basic Optimization Methods

- Downhill simplex method
- Powell's method
- Steepest descent method

Parameter Optimization in Rigid Registration

- Multidimensional minimization (or maximization) problem
  - finding the minimum (or maximum) of a function of more than one independent variable

6-dimensional vector → **F (similarity measure)** → **output (scalar value)**

- Function INPUT : 6-dimensional vector (3 translation, 3 rotation)
- Function OUTPUT : result of similarity measure (scalar-valued function)

Downhill Simplex Method (Nelder-Mead Simplex Method)

- Starting guess → goes downhill through the N-dimensional topography
  - until it encounters a (local, at least) minimum
- Maintains non-degenerate *simplex*
  - geometrical figure in N dimensions of nonzero volume that is the convex hull of N+1 vertices
- Completely unrelated to the algorithm of linear programming
- Requires only function evaluations, not derivatives
- Nonlinear unconstrained optimization

Downhill Simplex Method

- Algorithm starts with a *simplex* in N dimension
  - N+1 vertices and associated function values
- One or more test points are computed (iteration)
- Iteration terminates with bounded level sets

Downhill Simplex Method

- Four scalar parameters must be specified (coefficients)
  - reflection (ρ)
  - expansion (ω)
  - contraction (γ)
  - shrinkage (α)

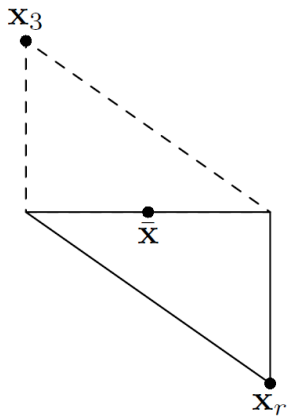
$$\rho > 0, \omega > 1, \omega > \rho, 0 < \gamma < 1, and\ 0 < \sigma < 1.$$

- Standard Nelder-Mead algorithm are (universal choices)
  - $\rho = 1, \omega = 2,\ \gamma = \frac{1}{2},\ and\ \ \sigma = \frac{1}{2}$

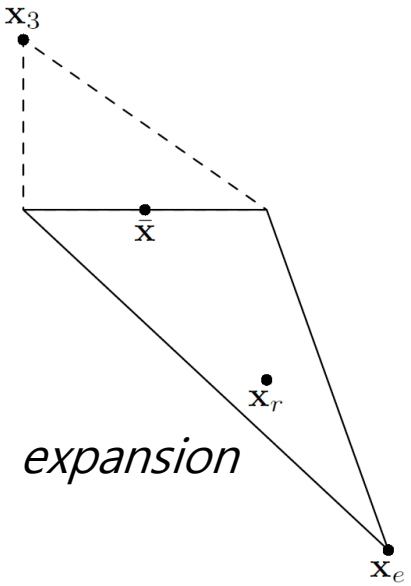
Downhill Simplex Method

1. Order
2. Reflect
3. Expand
4. Contract
   - outside
   - inside
5. Shrink

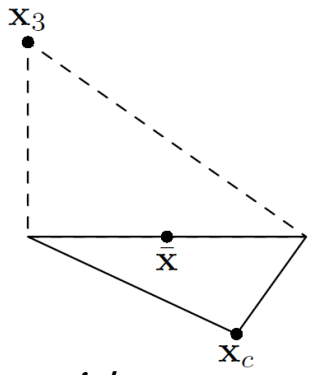
iteration



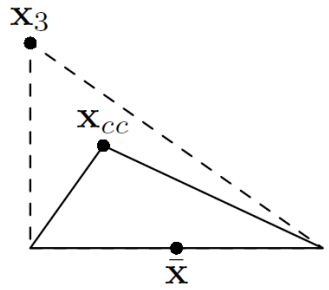
reflection



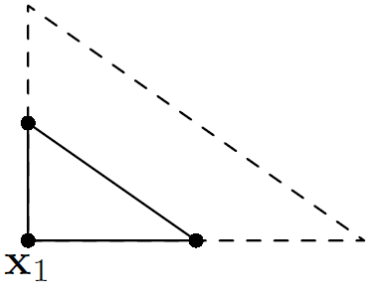
expansion



outside-contraction



inside-contraction

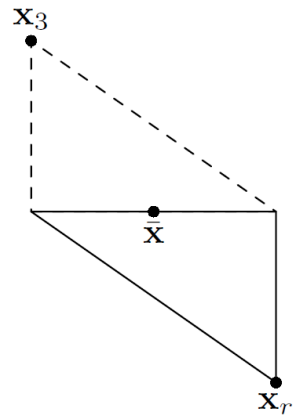


shrink

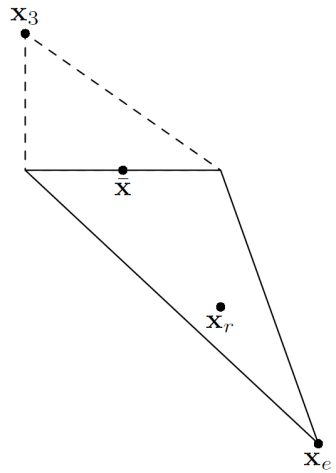
Downhill Simplex Method

- Order
  - order the n+1 vertices to satisfy $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$

- Reflect
  - compute the reflection point x_r
  - $x_r = \bar{x} + \rho(\bar{x} - x_{n+1}) = (1+\rho)\bar{x} - \rho x_{n+1}$
  - where $\bar{x} = \sum_{i=1}^{n} x_i / n$ is the centroid of the n best points (all vertices except for x_{n+1})
  - evaluate $f_r = f(x_r)$

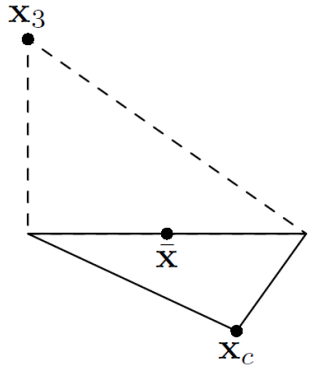


- Expand
  - compute the expansion point x_e
  - $x_e = \bar{x} + \omega(x_r - \bar{x}) = \bar{x} + \rho\omega(\bar{x} - x_{n+1}) = (1+\rho\omega)\bar{x} - \rho\omega x_{n+1}$
  - evaluate $f_e = f(x_e)$

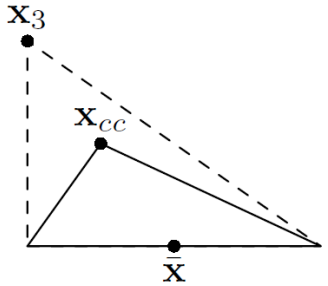


Downhill Simplex Method

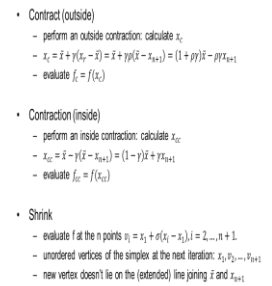
- Contract (outside)
  - perform an outside contraction: calculate x_c
  - $x_c = \bar{x} + \gamma(x_r - \bar{x}) = \bar{x} + \gamma\rho(\bar{x} - x_{n+1}) = (1 + \rho\gamma)\bar{x} - \rho\gamma x_{n+1}$
  - evaluate $f_c = f(x_c)$



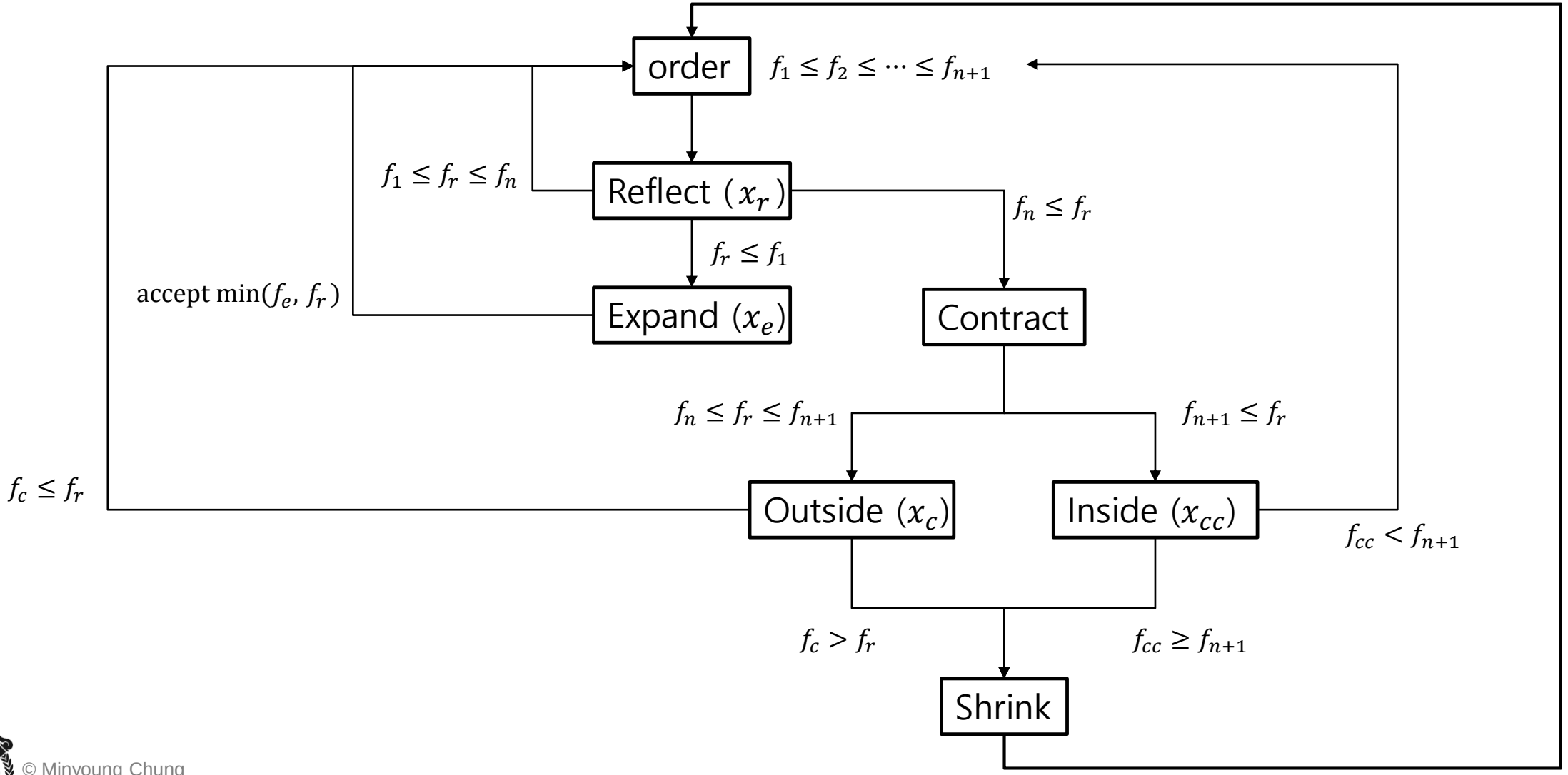
- Contraction (inside)
  - perform an inside contraction: calculate x_{cc}
  - $x_{cc} = \bar{x} - \gamma(\bar{x} - x_{n+1}) = (1 - \gamma)\bar{x} + \gamma x_{n+1}$
  - evaluate $f_{cc} = f(x_{cc})$



- Shrink
  - evaluate f at the n points $v_i = x_1 + \sigma(x_i - x_1), i = 2, \ldots, n + 1.$
  - unordered vertices of the simplex at the next iteration: $x_1, v_2, \ldots, v_{n+1}$
  - new vertex doesn't lie on the (extended) line joining $\bar{x}$ and x_{n+1}


• Contract (outside)
– perform an outside contraction: calculate x_c
– $x_c = \bar{x} + \gamma(x_r - \bar{x}) = \bar{x} + \gamma\rho(\bar{x} - x_{n+1}) = (1 + \rho\gamma)\bar{x} - \rho\gamma x_{n+1}$
– evaluate $f_c = f(x_c)$
• Contraction (inside)
– perform an inside contraction: calculate x_{cc}
– $x_{cc} = \bar{x} - \gamma(\bar{x} - x_{n+1}) = (1 - \gamma)\bar{x} + \gamma x_{n+1}$
– evaluate $f_{cc} = f(x_{cc})$
• Shrink
– evaluate f at the n points $v_i = x_1 + \sigma(x_i - x_1), i = 2, \ldots, n + 1.$
– unordered vertices of the simplex at the next iteration: $x_1, v_2, \ldots, v_{n+1}$
– new vertex doesn't lie on the (extended) line joining $\bar{x}$ and x_{n+1}

Downhill Simplex Method



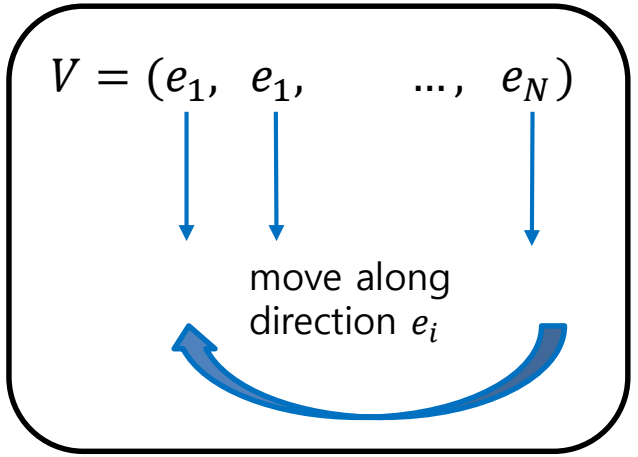
Downhill Simplex Method

- Non-shrink ordering rule
  - the worst vertex ${x_{n+1}}^{k}$ is discarded
  - accepted point created during iteration k ($v^{(k)}$), becomes a new vertex
  - new vertex takes position $j+1$ in the vertices of Δ_{k+1}
    - $j = \max_{0 \le l \le n} \{ l \mid f(v^{(k)}) < f({x_{l+1}}^{(k)}) \}$

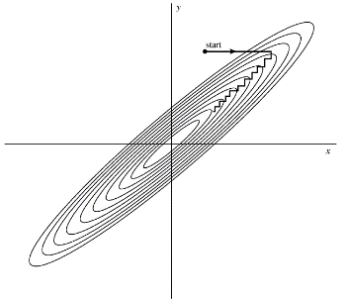
- Shrink ordering rule
  - only ${x_1}^{(k)}$ is carried over from Δ_k to Δ_{k+1}
  - IF (${x_1}^{(k)}$ and one or more of the new points are tied as the best point)
    - if $\min\{f({v_2}^{(k)}, \dots, f({v_{n+1}}^{(k)})\} = f({x_1}^{(k)})$, $\boldsymbol{{x_1}^{(k+1)} = {x_1}^{(k)}}$
  - whatever rule is used to define ordering

Powell's Method

- Multidimensional minimization method → sequences of line minimizations
  - different methods possibly differ only by how, at each stage
- Choice of successive directions does not involve explicit computation of *gradient*
- Simple method:
  - take the unit vectors $e_1, e_2, \ldots, e_N$ as a set of directions.
  - move along the first direction to its minimum (local at least)
  - then *from there* along the second direction to its minimum, and so on.
  - cycling through the whole set of directions as many times as necessary



- Very inefficient for some other functions
  - narrow valley (at some angle to the coordinate basis vectors)
  - series of many tiny steps
  - *conjugate directions*



Powell's Method with Conjugate Directions

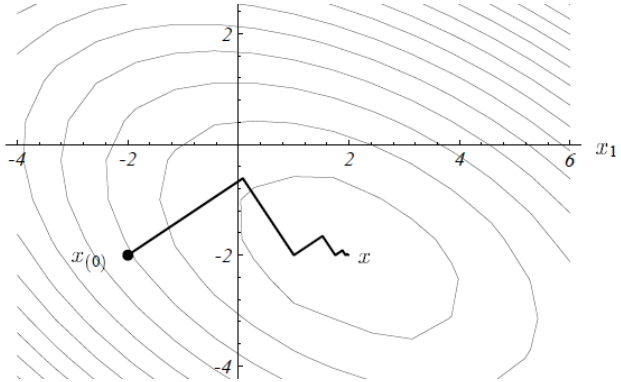
- Conjugate directions
  - directions which will not *spoil* previous minimizations

- Let δ_k and δ_{k+1} be two successive search directions
- f has been minimized in the δ_k direction
  - the projection of the gradient on δ_k is $\vec{0}$
  - choose a perpendicular direction
  - δ_{k+1} will not interfere with the minimization along δ_k
- If f is quadratic, one pass through the set of directions results in the exact minimum
  - else, quadratic convergence

- Construct a conjugate direction at each Powell's iteration

Steepest Descent Method

- Start at arbitrary point $x_{(0)}$
- Each step (i):
  - set direction where f decreases most quickly $\left(-\nabla f\left(x_{(i)}\right)\right)$
    - direction of steepest descent
  - $x_{(i+1)} = x_{(i)} - \alpha \nabla f\left(x_{(i)}\right)$
  - do : *line search*
- Line search : choosing α to minimize f along a line
  - α minimizes f when the *directional derivative* $\left(\frac{d}{d\alpha} f\left(x_{(i+1)}\right)\right)$ is equal to zero
  - $\frac{d}{d\alpha} f\left(x_{(i+1)}\right) = \nabla f\left(x_{(i+1)}\right)^T \frac{d}{d\alpha} x_{(i+1)} = -\nabla f\left(x_{(i+1)}\right)^T \nabla f\left(x_{(i)}\right) = 0$
  - $\nabla f\left(x_{(i)}\right)$ and $\nabla f\left(x_{(i+1)}\right)$ are orthogonal
- f is minimized where the gradient is orthogonal to the search line
  - *calculate* α
  - $\nabla f\left(x_{(i)} - \alpha \nabla f\left(x_{(i)}\right)\right)^T \nabla f\left(x_{(i)}\right) = 0$



Analytic Transform Parameter Optimization

- Formulation of Cost Function
- Parameterization of Transformations
- Derivatives of Transformations
- Derivatives of Cost Functions
- Other Optimizations

Formulation of Cost Function (SM + T)

- Cost function in Image Registration
  - similarity measurements(***SM***) are calculated between I_R(reference) and I_F(floating) images
  - I_F includes *transformations(**T**)*

- $C(\Theta) = \sum_{p \in \Omega} SM(I_R(p), I'_F(p))$
  - $\boldsymbol{\Omega}$: domain of the image volume
  - SM : similarity measure function

- $I'_F(p) = I_F\left(\boldsymbol{T}^{-1}(\boldsymbol{p};\ \boldsymbol{\Theta})\right)$
  - where $\boldsymbol{p}$ is a vector of position,
  - $\boldsymbol{\Theta}$ is transformation parameters

Parameterization of Transformations

- Rigid transformation → 6 parameters
  - 3 translations
  - 3 rotations

- Affine transformation → 12 parameters
  - 6 rigid parameters
  - 3 scaling
  - 3 shearing

- Non-rigid transformation (non-parametric) → # of control points in Spline-based deformation

Parameterization of Transformations

- $T(\boldsymbol{p};\ \boldsymbol{\Theta}) = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix}(\boldsymbol{p}) + \begin{pmatrix} \theta_{14} \\ \theta_{24} \\ \theta_{34} \end{pmatrix}$
  - where $\boldsymbol{p}$ is a vector of position,
  - $\boldsymbol{\Theta}$ parameterize the 12 degrees of freedom of transformation. (rigid and affine transformation)
  - homogeneous form can be used.
- $T(\boldsymbol{p};\ \boldsymbol{\Phi}) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(u)\, B_m(v) B_n(w) \boldsymbol{\phi}_{i+l,j+m,k+n}$
  - Domain of the image volume $\boldsymbol{\Omega} = \{(x,y,z) |\ 0 \le x < X, 0 \le y \le Y, 0 \le z < Z\}$
  - $\boldsymbol{\Phi}$ denote a $n_x \times n_y \times n_z$ mesh of control points $\phi_{i,j,k}$. (B-spline FFD transformation)
    - $n_x \times n_y \times n_z$: size of sub-voxels between control points.
  - where $i = \left\lfloor \frac{x}{n_x} \right\rfloor - 1, j = \left\lfloor \frac{y}{n_y} \right\rfloor - 1, k = \left\lfloor \frac{z}{n_z} \right\rfloor - 1,$
  - $u = \frac{x}{n_x} - \left\lfloor \frac{x}{n_x} \right\rfloor, v = \frac{y}{n_y} - \left\lfloor \frac{y}{n_y} \right\rfloor, w = \frac{z}{n_z} - \left\lfloor \frac{z}{n_z} \right\rfloor$
  - B_l represents the lth basis function of B-spline.

Derivatives of Transformations (affine)

- $\frac{\partial}{\partial \Theta} \boldsymbol{T}\ (\boldsymbol{p};\ \boldsymbol{\Theta})$
  - where $\boldsymbol{p}$ is three-dimensional vector (position),
  - $\boldsymbol{\Theta}$ is 12 transformation parameters
  - *[3x12] Jacobian matrix*

- $T\ (p;\ \Theta) = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix}$

- $\frac{\partial}{\partial \Theta} T\ (p;\ \Theta) = \begin{pmatrix} \frac{p'_x}{\partial \Theta_1} & \cdots & \frac{p'_x}{\partial \Theta_{12}} \\ \frac{p'_y}{\partial \Theta_1} & \cdots & \frac{p'_y}{\partial \Theta_{12}} \\ \frac{p'_z}{\partial \Theta_1} & \cdots & \frac{p'_z}{\partial \Theta_{12}} \end{pmatrix}$

Derivatives of Transformations (B-spline)

- $\frac{\partial}{\partial \boldsymbol{\Phi}} T(\boldsymbol{p};\ \boldsymbol{\Phi})$
  - where $\boldsymbol{p}$ is three-dimensional vector (position),
  - $\boldsymbol{\Phi}$ is 64 transformation parameters
  - consider $T(\boldsymbol{p};\ \boldsymbol{\Phi})$, as translation performed position by B-spline displacement
  - *[3x64] Jacobian matrix*
- Analytic computation of derivative is available with parametric B-spline deformation
  - other free-from deformation methods (non-parametric) are not able to compute analytically
  - central difference scheme with (a lot of) similarity measure are required
- Analytic computation of first-derivative
  - $T(\boldsymbol{p};\ \boldsymbol{\Phi}) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_l(u)\, B_m(v) B_n(w) \boldsymbol{\phi}_{i+l,j+m,k+n}$
  - $\frac{\partial}{\partial \boldsymbol{\Phi}} T(\boldsymbol{p};\ \boldsymbol{\Phi}) = \frac{\partial T(p;\, \Phi)}{\partial \phi_{i,j,k}} = B_l(u) B_m(v) B_n(w)$
    - where $l = i - \left\lfloor \frac{x}{n_x} \right\rfloor + 1, m = j - \left\lfloor \frac{y}{n_y} \right\rfloor + 1 \ and \ n = k - \left\lfloor \frac{z}{n_z} \right\rfloor + 1,$
    - $B_l(u) = 0 \ for \ l < 0 \ and \ l > 3.$
    - the derivative term are nonzero only in the neighborhood of a given point.

Derivatives of Cost Functions

- Cost function can be written as:
  - $C(\Theta)$, $C(\Theta, \Phi)$
  - where Θ is rigid | affine transformation parameters,
  - Φ is control points parameters
- Cost Function (object function) is a *similarity measure*
  - scalar-valued function with multiple variables
  - input vector : transformation parameters (rigid or affine)
  - transformation function (ex. matrix) is included
- SSD cost function
  - $C(\Theta) = \sum_{p \in \Omega}[I_R(p) - \boldsymbol{I'_F(p)}]^2$
  - where Ω is region of intersection (overlapped) between images
  - p : pixel location within region
  - I_R : reference image (target)
  - I_F : floating image
  - inverse mapping is actually used : $I'_F(p) = I_F\left(\boldsymbol{T^{-1}}(p;\ \Theta)\right)$

Derivatives of Cost Functions (SSD)

- $C(\Theta) = \sum_{p\in\Omega}\left[I_R(p) - I_F\left(\boldsymbol{T^{-1}}(p;\ \Theta)\right)\right]^2$
- $\nabla C(\Theta) = \frac{\partial C}{\partial \Theta}(\Theta)$
  - $\Theta = (\theta_1, \theta_2, \cdots, \theta_k)^T$
- $\frac{\partial}{\partial \Theta} C(\Theta) = -2\sum_{p\in\Omega}\left[I_R(p) - I_F\left(\boldsymbol{T^{-1}}(p;\ \Theta)\right)\right] \frac{\partial I_F}{\partial T^{-1}} \frac{\partial T^{-1}}{\partial \Theta}$
  - $I_R(p) - I_F\left(\boldsymbol{T^{-1}}(p;\ \Theta)\right)$: current error at pixel location $\Delta I(p)$
  - $\frac{\partial I_F}{\partial T^{-1}}$: intensity gradient in moving image
    - $I_x \equiv \frac{\partial I}{\partial x} \approx \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x}$
    - $I_y \equiv \frac{\partial I}{\partial y} \approx \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y}$
    - $\frac{\partial I_F}{\partial T^{-1}}(p) = \left(I_{F_x}(T^{-1}(p;\ \Theta))\ \ I_{F_y}(T^{-1}(p;\ \Theta))\right)$
    - pre-compute derivatives in floating image I_F
  - $\frac{\partial T^{-1}}{\partial \Theta}$: change in transformation w.r.t. change in parameters

Derivatives of Cost Functions (SSD)

- $\frac{\partial T^{-1}}{\partial \Theta}$: change in transformation w.r.t. change in parameters
- $T^{-1}(p;\ \Theta) = \begin{bmatrix} ax - by + t_x \\ bx + ay + t_y \end{bmatrix}$
  - *example of 2-dimensional rigid transformation*
  - $\Theta = (a, b, t_x, t_y)^T$
  - $p = (x, y)^T$
  - so derivative is 2x4 matrix (*Jacobian*) : $\frac{\partial T^{-1}}{\partial \Theta} = \begin{pmatrix} x & -y & 1 & 0 \\ y & x & 0 & 1 \end{pmatrix}$

Derivatives of Cost Functions (NCC, B-spline) – extra example

- Analytic computation of first-derivative

$$\frac{\partial SM}{\partial \phi_i} = -\frac{2}{|\Omega|}\sum_{x\in\Omega}\left(I_{Ref}\left(\mathbf{x}\right) - I_{Float}\left(\mathbf{T}\circ\mathbf{x}\right)\right)\cdot\frac{\partial\left(I_{Float}\left(\mathbf{T}\circ\mathbf{x}\right)\right)}{\partial\left(\mathbf{T}\circ\mathbf{x}\right)_{x_l}}\cdot\left(\prod_{k=1}^{3}B_{l_k}\left(u_k\right)\right)$$

- Approximating computation of first-derivative

$$\frac{\partial SM}{\partial \phi_i} = \frac{1}{2\Delta\phi_i|\Omega|}\left(\sum_{x\in\Omega}\left(I_{Ref}\left(\mathbf{x}\right) - I_{Float}\left(\mathbf{T}_i^{+}\circ\mathbf{x}\right)\right)^2 - \sum_{x\in\Omega}\left(I_{Ref}\left(\mathbf{x}\right) - I_{Float}\left(\mathbf{T}_i^{-}\circ\mathbf{x}\right)\right)^2\right)$$

References

- Rueckert, Daniel, et al. "Nonrigid registration using free-form deformations: application to breast MR images." *Medical Imaging, IEEE Transactions on* 18.8 (1999): 712-721.
- Xie, Zhiyong, and Gerald E. Farin. "Image registration using hierarchical B-splines." *Visualization and Computer Graphics, IEEE Transactions on* 10.1 (2004): 85-94.
- Pluim, Josien PW, JB Antoine Maintz, and Max A. Viergever. "Mutual-information-based registration of medical images: a survey." *Medical Imaging, IEEE Transactions on* 22.8 (2003): 986-1004.
- Lu, Zhentai, et al. "A fast 3-D medical image registration algorithm based on equivalent meridian plane." *Image Processing, 2007. ICIP 2007. IEEE International Conference on*. Vol. 5. IEEE, 2007.
- Bülow, Heiko, Laurence Dooley, and Diederich Wermser. "Application of principal axes for registration of NMR image sequences." *Pattern Recognition Letters* 21.4 (2000): 329-336.
- Shang, Lifeng, Jian Cheng Lv, and Zhang Yi. "Rigid medical image registration using PCA neural network." *Neurocomputing* 69.13 (2006): 1717-1722.
- Lagarias, Jeffrey C., et al. "Convergence properties of the Nelder--Mead simplex method in low dimensions." *SIAM Journal on optimization* 9.1 (1998): 112-147.

- Press, William H. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

Thank you!