(Rigid/Affine) Image registration

Simon Rit$^{12}$

$^1$CREATIS laboratory
$^2$Léon Bérard center

Master EEAP / SI - Module 5 - 2012
Module 5: Registration and Motion Estimation

- **Registration**
  - Rigid/Affine: 4h, S. Rit
  - Non-rigid: 2h, D. Sarrut
  - Practical exercises: 2h, S. Rit and D. Sarrut

- **Motion estimation**
  - Classical approaches, discontinuities: 4h, P. Clarysse
  - Application to ultrasound imaging: 2h, H. Liebgott
  - Practical exercises: 2h, P. Clarysse and S. Rit

- Evaluation: practical exercises and test
Outline

1. Introduction
2. Transformations
3. Similarity measures
4. Optimization
5. Validation
6. Conclusion
Definition

The process of aligning images so that corresponding features can easily be related.

- Registration
- Alignment
- Geometrical correspondence
- Matching
- Motion estimation
Motivation

Many applications
- Medical imaging
- Vision
- etc.

Basic elements are used in many applications
- Interpolation
- Similarity measure
- Optimization
- etc.
Goal

Find a spatial transformation $T$ that matches two images.
Example #1: panoramic photography
Example #1: panoramic photography
Example #1: panoramic photography
Example #2: satellite pictures (Google maps)
Example #2: satellite pictures (Google maps)
Example #2: satellite pictures (Google maps)
Example #2: satellite pictures (Google maps)
Example #3: multimodal 3D medical images

Registration is a prerequisite for adequate image fusion

MRI

PET
Example #3: multimodal 3D medical images

Registration is a prerequisite for adequate image fusion

Before registration

After registration
Example #4: image-guidance

CT scanner

In-room cone-beam CT
Example #4: image-guidance

CT

X-ray projection

Cone-beam CT

3D/2D (one projection) or 3D/3D (reconstruction) registration
The input images may differ in

- Resolution
- Time
- Space (2D, 2D+t, 3D, 3D+t...)
- Modality (Photo, Radiography, CT, PET, MRI, US...)
- Subject (Inter-patient registration)
- ...

There are many possible output transformations.

⇒ Many algorithms...
Formalism

\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(I, J, T) \]

Notation

- \( I \): reference / fixed image
- \( J \): moving / floating image
- \( T \): transformation
- \( \mathcal{T} \): search space
- \( S \): similarity measure
- \( \arg \max \): optimization
- \( \hat{T} \): solution
Digital images

Discrete set of pixels/voxels

Pixels = 2D

Voxels = 3D

We consider two images $I(i), J(j) \in \mathbb{R}$. $i, j \in \mathbb{Z}^3$ are the spatial indices of the lattice.
Digital images

- **Spatial resolution**
  - Ex: $512 \times 512$ pixels of $0.5 \times 0.5 \text{ mm}^2$

- **Pixel values**
  - Scalar (e.g. CT) or vector (e.g. RGB) or matrix (e.g. diffusion MRI) at each point
  - Digital number(s) (integer, float...) $\Rightarrow$ limited precision

- **Visualization according to a color scale**
  - Defined with window/level for linear gray scales

- **Coordinate system**
  - Origin
  - Orientation (e.g., with respect to anatomical orientation: cranio-caudal, left-right, antero-posterior).
Formalism

\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(I, J, T) \]

Notation

- \( I \) = reference / fixed image
- \( J \) = moving / floating image
- \( T \) = transformation
- \( \mathcal{T} \) = search space
- \( S \) = similarity measure
- \( \arg \max \) = optimization
- \( \hat{T} \) = solution
Transformations

- Cartesian coordinates: \( \mathbf{x} = (x, y, z, t)^T \)
- Transformation:

\[
T : \mathbb{R}^4 \rightarrow \mathbb{R}^4 \\
\mathbf{x} \rightarrow T(\mathbf{x}) = \mathbf{x}'
\]
Search space $\mathcal{T}$

One point of $\mathcal{T}$ is a transformation $T$. It depends on the
- Degrees of freedom = number $n$ of parameters needed to describe the transformation. $n$ is the dimension of the optimization problem.
- Boundaries of each parameter (if any).

The success of the registration requires that
- The solution $\hat{T}$ is in $\mathcal{T}$.
- $\mathcal{T}$ is small enough with respect to the input data.
Categories

- Linear registration: $T(ax_1 + x_2) = aT(x_1) + T(x_2)$
  (more exactly, composition of linear transformations with translations)
  - Rigid (translations + rotations)
  - Rigid + Scaling
  - Affine

- Non-linear registration
  - Non-rigid
  - Deformable
  - Elastic
  - Fluid
Rigid registration

A rigid transformation can be described with translations and rotations.

The dimensionality can be adapted

- 2D/2D $\rightarrow$ 3 parameters (2 translations, 1 translation)
- 3D/3D $\rightarrow$ 6 parameters (3 rotations, 3 translations)
- 2D/3D $\rightarrow$ 6 parameters (3 rotations, 3 translations + projection operator).
Rigid / affine transformations can be represented with matrices:

$$T_{linear}(x) = Rx + t$$

where $R$ is the rotation matrix and $t$ the translation vector ($R$ is $3 \times 3$ and $t \in \mathbb{R}^3$ in 3D).

For rigid transformations, $R$ is constrained to have only 3 parameters. If one uses all 9 parameters of $R$, $T$ is an affine transformation.
Homogeneous coordinates

For practicality, one can use a single matrix with homogeneous coordinates:

\[ T_{\text{linear}}(x) = Mx \]

where \( x \) is the point in homogeneous coordinates and \( M \) combines \( R \) and \( t \).

In 3D, we would have \( x = (x, y, z, 1) \) and the matrix

\[
M = \begin{bmatrix}
R_{00} & R_{01} & R_{02} & t_0 \\
R_{10} & R_{11} & R_{12} & t_1 \\
R_{20} & R_{21} & R_{22} & t_2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Also useful to apply projection transforms.
Matrix manipulation

- Application: \( T(x) = Mx \)

- Inverse (if invertible): \( T^{-1}(x) = M^{-1}x \)

- Composition: \( T_2(T_1(x)) = (T_2 \circ T_1)(x) = M_2M_1x \)

- etc.
Parameterization

- Affine = all possible matrices = 12 parameters in 3D

\[ M = \begin{bmatrix}
  R_{00} & R_{01} & R_{02} & t_0 \\
  R_{10} & R_{11} & R_{12} & t_1 \\
  R_{20} & R_{21} & R_{22} & t_2 \\
  0 & 0 & 0 & 1
\end{bmatrix} \]

- For registration, one can
  - Optimize the 12 parameters of the matrix
  - Decompose the matrix in more meaningful parameters, e.g. translations, rotations, ...
Translations

In 2D, 2 translation parameters

\[
M = \begin{bmatrix} 1 & 0 & t_0 \\ 0 & 1 & t_1 \\ 0 & 0 & 1 \end{bmatrix}
\]

In 3D, 3 translation parameters

\[
M = \begin{bmatrix} 1 & 0 & 0 & t_0 \\ 0 & 1 & 0 & t_1 \\ 0 & 0 & 1 & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]
2D rotation around the origin

One parameter, the angle $\theta$:

$$R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
2D rotation around another point

One can combine a translation (=change of coordinate system) with this rotation matrix to have a rotation around another point \( p = (p_0, p_1) \):

\[
R' = \begin{bmatrix}
1 & 0 & p_0 \\
0 & 1 & p_1 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
1 & 0 & -p_0 \\
0 & 1 & -p_1 \\
0 & 0 & 1 \\
\end{bmatrix}
\]
3D rotation

- Rotation = 3 × 3 matrix (Euler’s rotation theorem)

- 3 degrees of freedom (parameters)

- Constraints: orthogonal matrix ($RR^T = Id$) and $det(R) = 1$
  \[ \Rightarrow \text{Inverse: } R^{-1} = R^T \]
3D rotation

- Several decompositions from the 9 matrix values to 3 parameters, e.g.
  - Euler angles
  - Axis-angle
  - Quaternions
  - etc. (see http://en.wikipedia.org/wiki/Charts_on_SO(3) for others)

⇒ The parameters have a different meaning in each case
3D rotation – Euler angles

- Composition of 3 rotations around the axis of the coordinate system (parameters $\alpha, \beta, \gamma$ or $\theta, \varphi, \psi$ or Yaw, Pitch and Roll / Roulis, Tangage et Lacet):

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \quad R_Y = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \quad R_Z = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Order matters $\Rightarrow$ several conventions!
  - Euler angles: ZXZ (XYX, XZX, YXY, YZY, ZYZ)
  - Tait-Bryan angles: YXZ (XYZ, XZY, YZX, ZXY, ZYX)
3D rotation – The ZXY convention

http://en.wikipedia.org/wiki/Euler_angles
3D rotation – Issues with Euler angles

- Convention issues
- *Gimbal lock* = singularity

Can be solved by changing the convention (other order) or adding a 4th gimbal.

http://en.wikipedia.org/wiki/Gimbal_lock
3D rotation – Axis-angle

\[ \mathbf{n} = (n_x, n_y, n_z)^T \]
3D rotation – Axis-angle

- From Euler’s rotation theorem
- 4 parameters
  - Unit vector \( \mathbf{n} = (n_x, n_y, n_z)^T \)
  - Angle \( \theta \)
- but 3 degrees of freedom because unit vector (\( \|\mathbf{n}\| = 1 \)).
- Computation of the new position from parameters with Rodrigues’ rotation formula:
  
  \[ \text{http://mathworld.wolfram.com/RodriguesRotationFormula.html} \]

- Limitations
  - Representation not minimal
  - Difficult to combine two rotations

⚠️ Euler parameters refer to the axis-angle representation and are different from the Euler angles...
3D rotation – Quaternions

4 parameters \( \mathbf{q} = (q_0, q_1, q_2, q_3)^T \) but 3 DOF \( |\mathbf{q}|^2 = 1 \)

\[
\begin{bmatrix}
q_0 q_0 + q_1 q_1 - q_2 q_2 - q_3 q_3 \\
2(q_2 q_1 + q_0 q_3) \\
2(q_3 q_1 + q_0 q_2)
\end{bmatrix}
\begin{bmatrix}
2(q_1 q_2 - q_0 q_3) \\
q_0 q_0 - q_1 q_1 + q_2 q_2 - q_3 q_3 \\
2(q_3 q_2 + q_0 q_1)
\end{bmatrix}
\begin{bmatrix}
2(q_1 q_3 + q_0 q_2) \\
2(q_2 q_3 - q_0 q_1) \\
q_0 q_0 - q_1 q_1 - q_2 q_2 + q_3 q_3
\end{bmatrix}
\]

Link with angle-axis: \( \mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ n_x \sin(\theta/2) \\ n_y \sin(\theta/2) \\ n_z \sin(\theta/2) \end{bmatrix} \)

Advantage: combine rotations, stability

Inconvenient: 4 non-independent parameters
Scaling

In 3D, 3 parameters \( s_x, s_y, s_z \)

\[
R = \begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & s_z
\end{bmatrix}
\]

Useful if working in voxel coordinates, e.g.

- Image #1: \( dx_I \times dy_I \times dz_I \) mm\(^3\)
- Image #2: \( dx_J \times dy_J \times dz_J \) mm\(^3\)

\[
R = \begin{bmatrix}
  \frac{dx_I}{dx_J} & 0 & 0 \\
  0 & \frac{dy_I}{dy_J} & 0 \\
  0 & 0 & \frac{dz_I}{dz_J}
\end{bmatrix}
\]
Shear

In 3D, 6 parameters $s_{xy}, s_{yx}, s_{xz}, s_{zx}, s_{yz}, s_{zy}$

$$R = \begin{bmatrix}
1 & s_{yx} & s_{zx} \\
{s_{xy}} & 1 & s_{zy} \\
{s_{xz}} & s_{yz} & 1
\end{bmatrix}$$
Skew

Idem as shear but antisymmetric ($R^T = -R$)

$\Rightarrow$ In 3D, 3 parameters $s_{yx}, s_{zx}, s_{zy}$

$$R = \begin{bmatrix} 1 & s_{yx} & s_{zx} \\ -s_{yx} & 1 & s_{zy} \\ -s_{zx} & -s_{zy} & 1 \end{bmatrix}$$
Projection – Matrix

- \((n - 1) \times n\) matrix
- Parallel or perspective
- 2D/1D perspective projection in homogeneous coordinates:

\[ M = \begin{bmatrix} SDD & 0 & 0 \\ 0 & 1 & SID \end{bmatrix} \]

- SDD: Source to Detector Distance
- SID: Source to Isocenter Distance
Summary

Linear matrices can be decomposed in
http://mathworld.wolfram.org/
http://www.wikipedia.com/

- Translation parameters: straightforward
- Rotation parameters: need caution, several solutions
- Scaling parameters: straightforward
- Skew parameters: less used
- +Projection for 2D/3D registration
Summary

Only a subset of the parameters can (should?) be optimized.

- Rigid: 6 parameters (translations and rotations)
- Rigid + global rescaling: 7 parameters
- Rigid + independent rescaling: 9 parameters
- Affine: 12 parameters (translations, rotations, scaling, skew)
Tips and tricks

Carefully chose the search space

- As little parameters as possible...

- ... but make sure that the sought solution is in it!

- Bound the search space, e.g. translations $< 2$ cm and rotations $< 10^\circ$

- Scale the parameters which are heterogeneous, e.g. translations in mm and rotations in radians or degrees
Formalism

\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(I, J, T) \]

**Notation**
- \( I \) = reference / fixed image
- \( J \) = moving / floating image
- \( T \) = transformation
- \( \mathcal{T} \) = search space
- \( S \) = similarity measure
- \( \arg \max \) = optimization
- \( \hat{T} \) = solution
Similarity measure

Measure of the alignment quality of $I$ and $J$ with $T$.

Two families

1. Feature-based registration
2. Intensity-based registration
Features

Features are extracted from images prior to similarity measure

- Points
- Lines
- Vectors
- Surfaces
- Volumes
- ...
Feature-based registration formalism

\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(\mathcal{F}_I, T \circ \mathcal{F}_J) \]

Notation

- \( \mathcal{F}_I \) = feature set of reference / fixed image
- \( \mathcal{F}_J \) = feature set of moving / floating image
- \( T \) = transformation
- \( \mathcal{T} \) = search space
- \( S \) = similarity measure
- \( \arg \max \) = optimization
- \( \hat{T} \) = solution
Geometric primitives – Manual identification

Landmarks (recognizable points) clicked by an expert.
Manual contours.
Geometric primitives – Automatic identification

- Segmentation
  ⇒ Another problem which can be very difficult...

- Unpaired features
  ⇒ Unpaired similarity measure, e.g. pair to closest feature in other image
  ⇒ Or find an algorithm to pair features (another problem...)
Manual vs auto: pros and cons

- **Manual**
  - Reference
  - Long
  - Expensive (requires time of an expert, e.g. physician)
  - Inter- and intra- observer variability

- **Automatic**
  - Fast and cheap
  - Paired or unpaired?
  - Reproducible
  - Robust
(Dis-)similarity measures for features

- **Sum of distances**
  - $L_1$-norm: $|x|_1 = \sum_{r=1}^{n} |x_r|
  - $L_2$-norm: $|x|_2 = \sqrt{\sum_{r=1}^{n} x_r^2}$
  - Quadratic sum (faster): $\sum_{r=1}^{n} x_r^2$
  - Other distances...
Distance map

Map of the distance between each point and the closest feature
⇒ Fast computation of distances for unpaired features

Computed only once for the reference image.
Fast computations: Chamfer distance (not Euclidian), separable algorithms [Coeurjolly, PAMI, 2007].
S(\mathcal{F}_I, T \circ \mathcal{F}_J) \text{ or } S(\mathcal{F}_J, T \circ \mathcal{F}_I)\text{?}

- This is not identical since
  - with the distance map \( d_I \)
    \[
    S(I, J, T) = \sum_{F \in \mathcal{F}_J} d_I \circ T(F)
    \]
  - with the distance map \( d_J \)
    \[
    S(I, J, T) = \sum_{F \in \mathcal{F}_J} d_J \circ T(F)
    \]

⇒ Use the distance map of the image which has features easier to extract
Summary of feature-based registration

- Preliminary step to identify features
- Pairing algorithm maybe required
+ Generally fast (depending on the feature type and size)
+ Registration of some features only...

There is currently a renewed interest in feature-based registration, particularly for non-rigid registration...
Limitation of feature based registration

Multimodal images

⇒ Intensity-based registration
Intensity-based registration formalism

\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(I, J \circ T) \]

Notation

- \( I \) = reference / fixed image
- \( J \) = moving / floating image
- \( T \) = transformation
- \( \mathcal{T} \) = search space
- \( S \) = similarity measure
- \( \arg \max \) = optimization
- \( \hat{T} \) = solution
Intensity-based similarity measure

\[ S(I, J \circ T) \] is a measure which assumes a functional dependence between the pixel intensities.

- Intensity-based registration = iconic registration
- Choice depends on the modalities
- Functional dependence
Forward and backward warping

Required for computing $J \circ T$

Forward mapping

Backward mapping
Forward and backward warping

- Forward mapping requires an additional weight map + potential holes
  ⇒ Backward is usually preferred. Not a problem:
  - $T$ is easily invertible for affine registration
  - One can always optimize $T^{-1}$ and invert it at the end

- The resolution of the reference image is used

∀ voxel $x \in \Omega_I$:

$$M x = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' = ax + by + cz + d \\ y' = ex + fy + gz + h \\ z' = ix + jy + kz + l \end{pmatrix}$$

Use increment in innermost loop (along $x$ here):

$$\begin{cases} x'_{n+1} = x'_n + a \quad \text{and} \quad x'_0 = by + cz + d \\ y'_{n+1} = y'_n + e \quad \text{and} \quad y'_0 = fy + gz + h \\ z'_{n+1} = z'_n + i \quad \text{and} \quad z'_0 = jy + kz + l \end{cases}$$
Interpolation

- Nearest neighbor
- Linear
- More accurate: spline, sinc...

⇒ Compromise between speed and accuracy
Based on the Beer-Lambert law:

$$\Phi(u, v) = \Phi_0 \exp \left( -\int_{L_{\beta, u, v}} f(x) \, dx \right)$$
Intensity-based similarity measures

- **Sum of Squared Difference**
  \[
  SSD(I, J) = \sum_{x \in \Omega} (I(x) - J(x))^2
  \]

- **Correlation coefficient**
  \[
  CC(I, J) = N \frac{\sum (I(x) - m_I)(J(x) - m_J)}{\sqrt{\sum l(x) - m_l} \sum \sqrt{J(x) - m_J}}
  \]

- **Mutual information**
  \[
  MI(I, J) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{p_i p_j}
  \]

- **Correlation ratio**
  \[
  CR(I, J) = \eta^2 (I|J) = 1 - \frac{1}{\sigma_I^2} \sum_j p_j \sigma^2_{I|j}
  \]

- Many others...
Sum of Squared Difference

$$SSD(I, J) = \sum_{x \in \Omega} (I(x) - J(x))^2$$

- Dissimilarity measure
- Fast
- Needs to be normalized to $\Omega$ size (in voxels)
- Functional dependence between intensities: identity
  - Optimal if images only differ by Gaussian noise (monomodal)
  - Not robust otherwise (multimodal)

- Similar: Sum of Absolute Difference (SAD)
(Pearson product-moment) Correlation coefficient

\[ CC(I, J) = \rho(I, J) = \frac{\text{cov}(I, J)}{\sigma_I \sigma_J} = N \frac{\sum (I(x) - m_I)(J(x) - m_J)}{\sum \sqrt{I(x) - m_I} \sum \sqrt{J(x) - m_J}} \]

- Range \([-1, 1]\)
  - 1 is perfect increasing linear relationship
  - -1 is perfect decreasing linear relationship
  - 0 if independent
  \[ \Rightarrow \text{Similarity measure: } |CC(I, J)| \]

- Fast (with computational tricks)
- Linear dependence: \( I(x) = a J(T(x)) + b \quad \forall x \quad a, b \in \mathbb{R} \)
  - More robust than SSD, e.g. for 2 imaging systems of a same modality
  - Not good for multimodal
Functional dependence between intensities

- SSD: $l(x) = J(T(x)) \quad \forall x$
- CC: $l(x) = a J(T(x)) + b \quad \forall x$

⇒ Other functional relationship? ⇒ Information theory
  - Mutual information
  - Coefficient ratio

Both require joint histograms.
Joint histogram – Definition

- **$i$, $j$:** (binned) colors
- **$n_{ij}$:** number of pixels with color $i$ in $I$ and $j$ in $J$
- **$n_i$, $n_j$:** marginal values, i.e., histograms of $I$ and $J$
Joint histogram
Joint histogram
Joint histogram

- 2D distribution of the pair of intensities at each voxel location

- SSD and CC can also be understood with joint histograms

- ...but it is useless to build it for SSD and CC

- Recomputed at each step of the optimization
Joint histogram – Example #2

From [Roche, PhD, 2001]
Joint histogram – Computation

Two steps:

1. Count the $n_{ij}$ (accounting for $T$)
   - Compute the warped image $J \circ T$ on $I$ grid
   - Compute the joint histogram of $J \circ T$ and $I$
   - Or directly update the histogram pixel-by-pixel

2. Derive the probability $p_{ij}$
Joint histogram
Mutual information – Entropy

- Measure of information as a registration metric

⇒ Entropy (Shannon-Wiener):

\[ H = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i \]

![Entropy distributions](image)

- Entropy=3.00
- Entropy=2.99
- Entropy=2.11
- Entropy=-0.00
Mutual information – Joint entropy

- Two images $\Rightarrow$ two symbols at each grid point

$$H = - \sum_{i,j} p_{ij} \log p_{ij}$$

$\Rightarrow$ Joint entropy: the more similar the distributions, the lower the joint entropy compared to the sum of individual entropies

$$H(I, J) \leq H(I) + H(J)$$
Mutual information

\[ MI(I, J) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{p_i p_j} \]

\[ MI(I, J) = H(I) - H(J|IJ) = H(J) - H(I|J) = H(I) + H(J) - H(I, J) \]

- Positive
- Symmetric \( MI(I, J) = MI(J, I) \)
- \( H(J|I) = H(I, J) - H(I) \): conditional entropy
- Example [Pluim, TMI, 2003]:
Mutual information is a reference

- First articles: June 1995
  - Maes in Belgium
  - Viola in US

- First journal articles: 1997

- Currently: #1 similarity measure
  - Multimodal
  - Robust

- Among the most cited papers of the domain...
For mutual information, implementation matters (as usual!).
Survey of methods: [Pluim, TMI, 2003].
Specifically:

- Interpolation
- Binning
- Probability computation
- Normalization (less sensitive to overlap changes)
  e.g. Normalized Mutual Information [Studholm, Pattern Recognition, 1999]
MI implementation – Interpolation

- Conventional: nearest neighbor, linear...

- Partial volume interpolation (Collignon)

- The difference lies in the smoothness of the optimized function
MI implementation – Interpolation

NN

PV

LIN
MI implementation – Number of bins

\[ h = 2 \]

\[ h = 10 \]
Frequential approach (simplest): \( p_{ij} = \frac{n_{ij}}{N} \)

Parzen window: \( p_{ij} = \sum_k \varphi(n_{ij} - n_k) \)
- \( \varphi = \text{Gaussian [Viola, PhD, 1995]} \)
- \( \varphi = \text{Cubic spline [Thevenaz, 2000]} \)

Also Bayesian approach...
Correlation ratio

\[ RC(I, J) = \eta^2(I|J) = 1 - \frac{1}{\sigma_I^2} \sum_j p_j \sigma_{I|j}^2 \]

- Based on joint histogram as well
- Assume a functional dependence between intensities
- Variance instead of entropy
- Results comparable to MI, not symmetric

Summary of intensity-based similarity measures

The measures can be classified with respect to hidden variables [Malandain, HDR, 2006]

- SSD, SAD: 0 variable
- CC: 2 variables, slope and intercept
- CR: size of one side of the joint hist
- MI: product of the size of the sides of the joint hist

⇒ MI is more general, hence its success
⇒ But it is also more susceptible to fail in simple situations where SSD is sufficient, e.g., time series
Local affine registration

One can always use a region-of-interest (ROI) of the image for registration (e.g. rectangular box or shaped volume of interest)

Applied to the reference and/or the target image depending on the similarity measure.
Formalism

\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(I, J, T) \]

**Notation**
- \( I \) = reference / fixed image
- \( J \) = moving / floating image
- \( T \) = transformation
- \( \mathcal{T} \) = search space
- \( S \) = similarity measure
- \( \arg \max \) = optimization
- \( \hat{T} \) = solution
Find the global minimum... as fast as possible
1D illustration

![1D illustration](image)

- **SSD**
- **Diagonal displacement (mm)**

---

Image registration  
Master EEAP / SI - Module 5 - 2012  
Simon Rit
Optimization

- Optimization = maximize or minimize

- Search for global minimum ⇒ if the max is sought, then minimize $-f$

- Optimization 1D / nD (up to $n = 12$ for affine registration)

- With or without a gradient of $f$
  - The solution respects $\nabla f(\hat{x}) = 0$ (as well as other extrema)

- Multi-resolution
Optimization

- Iterative process: iteration $x_k / f(x_{k+1}) < f(x_k)$

- Stopping criterion
  - $|f(x_{k+1}) - f(x_k)| < \epsilon$
  - $\nabla f(\hat{x}) = 0$

- One iteration $\Rightarrow$ search $\delta x_k / x_{k+1} = x_k + \delta x_k$

- Initialization $x_0$
  - Identity
  - Align image centers
  - Align mass centers
  - ...
1D optimization: Golden search

- Without derivatives
- Based on the bisection method (root search)
  - Based on the intermediate value theorem
- Bracketed solution
  If \( a < b < c \) such that \( f(b) < f(a) \) and \( f(b) < f(c) \) then \( f \) has a minimum in the interval \((a, c)\).

\[ \Rightarrow \] 4 values at each iteration (position based on Golden number)
- \( a < b < x < c \) or
- \( a < x < b < c \)

\[ \Rightarrow \] \( x \) replaces either \( a \) or \( c \) depending on where is the minimum.
1D optimization: Brent’s method

- Without derivatives

- Based on Dekker’s method, 1969
  - Bisection / Golden search method
  - Secant method
  - Linear interpolation

- Brent’s method, 1973
  + Inverse quadratic interpolation
  + Additional tests
  ⇒ 6 values at each step
Brent’s method - Illustration

Three points bracket the interval

Inverse quadratic interpolation

Iterate and stop when $f(x_3) - f(x_1) < \epsilon$
Brent’s method - Illustration

Three points bracket the interval
Brent’s method - Illustration

- Three points bracket the interval
- Inverse quadratic interpolation
Brent’s method - Illustration

- Three points bracket the interval
- Inverse quadratic interpolation
Brent’s method - Illustration

- Three points bracket the interval
- Inverse quadratic interpolation
- Iterate and stop when \( f(x_3) - f(x_1) < \epsilon \)
## nD optimization

<table>
<thead>
<tr>
<th>Without gradient</th>
<th>With gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplex</td>
<td>Gradient descent</td>
</tr>
<tr>
<td>Powell</td>
<td>Conjugate gradient</td>
</tr>
<tr>
<td></td>
<td>(Newton)</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
</tr>
<tr>
<td></td>
<td>Levenberg-Marquardt</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\delta_{k+1} &= -\nabla f(x_k) \\
\delta_{k+1} &= -\nabla f(x_k) + \beta_k d_k \\
\delta_{k+1} &= -H^{-1} \cdot \nabla f(x_k) \\
\delta_{k+1} &= -A^{-1} \cdot \nabla f(x_k) \\
\end{align*}
\]

SD/QN
Downhill simplex

- Nelder and Meade, 1965
- Simplex = geometric object (nondegenerate)
- $n + 1$ vertices
- Based on pseudo-derivatives
  - ⚠️ Not to be mixed with the simplex method in linear programming

1. Compute the function (similarity measure) at each vertex
2. Conditional deformation (see next slide)
Downhill simplex

Types of Simplex Moves

- Original Simplex
- Reflection
- Contraction
- Reflection and Expansion
- Shrinkage

https://wiki.ece.cmu.edu/ddl/index.php/Downhill_Simplex
Downhill simplex

Algorithm

- Sort $f(x_i)$ with $i$ the vertex number
- Take worst and replace with (if better)
  1. Reflection
  2. Expansion
  3. Contraction
  4. Shrink

- Iterate until convergence...
Downhill simplex - Examples

https://wiki.ece.cmu.edu/ddl/index.php/Downhill_Simplex
Downhill simplex - Examples
Line search along unit vectors

(Numerical Recipes)
Line search

- Used by many nD strategies
  - 1D optimization in a direction of the nD space

⇒ Given a set of parameters \( x \in \mathbb{R}^n \) and a direction \( y \in \mathbb{R}^n \), find the best \( \alpha \in \mathbb{R} \) so that \( x + \alpha y \) is minimal.

- Typically: Brent’s method
Powell’s method

- Powell’s conjugate gradient descent method (1964)
- Set of $n$ directions, e.g. the unit vectors
- Move along one direction until minimum
  - Line search
- Loop over directions until minimum
- Update set of directions
  - Displacement vector becomes a (conjugate) direction
  - Remove direction which contributed the most

- Order matters, e.g. [Maes, 1999] $(t_x, t_y, \phi_z, t_z, \phi_x, \phi_y)$.
- Converge in $n(n + 1)$ iterations for quadratic functions
Conjugate directions

Directions which will not spoil previous minimizations

- Let $\delta_k$ and $\delta_{k+1}$ be two successive search directions
- $f$ has been minimized in the $\delta_k$ direction
  $\Rightarrow$ The projection of the gradient on $\delta_k$ is $\vec{0}$
  $\Rightarrow$ Chose a perpendicular direction
  $\Rightarrow$ $\delta_{k+1}$ will not interfere with the minimization along $\delta_k$. 

---

Image registration Master EEAP / SI - Module 5 - 2012 Simon Rit 108
Conjugate directions properties

- If $f$ is quadratic, one pass through the set of directions results in the exact minimum
- Else, quadratic convergence

- Powell’s method constructs a conjugate direction at each iteration
Powell - Example

http://www.mathworks.com/matlabcentral/fileexchange/authors/26509
### nD optimization

<table>
<thead>
<tr>
<th>Without gradient</th>
<th>With gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplex</td>
<td>Gradient descent</td>
</tr>
<tr>
<td>Powell</td>
<td>Conjugate gradient</td>
</tr>
<tr>
<td></td>
<td><em>(Newton)</em></td>
</tr>
<tr>
<td>Quasi-Newton</td>
<td>Quasi-Newton</td>
</tr>
<tr>
<td>Levenberg-Marquardt</td>
<td>Levenberg-Marquardt</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\delta_{k+1} &= -\nabla f(x_k) \\
\delta_{k+1} &= -\nabla f(x_k) + \beta_k d_k \\
\delta_{k+1} &= -H^{-1} \cdot \nabla f(x_k) \\
\delta_{k+1} &= -A^{-1} \cdot \nabla f(x_k)
\end{align*}
\]
Gradient optimization

Gradient vector:

\[ \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \end{bmatrix} \in \mathbb{R}^n \]

Hessian matrix:

\[ H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{n \times n} \]
Gradient optimization

- Expected gain: $O(n)$ instead of $O(n^2)$

- Problem: computing partial derivatives
  ⇒ Is it worth the effort?

- But:
  - there might be redundancies in $\nabla f$
  - line search might be faster
Gradient optimization - 1D example

![Graph showing SSD vs Diagonal displacement (mm)]
Gradient optimization - 1D example

Image registration

Master EEAP / SI - Module 5 - 2012
Simon Rit

114
Gradient optimization - 1D example

Image registration

Master EEAP / SI - Module 5 - 2012
Simon Rit
Gradient optimization - 1D example
Gradient optimization - 1D example
Gradient optimization - 1D example
Gradient optimization - 1D example
Gradient optimization - 1D example
Gradient optimization - 1D example

SSD

Diagonal displacement (mm)
Computing the gradient in nD

- Feature-based registration: almost impossible
  - Depends on the segmentation...
  - ⇒ Iterative Closest Point [Zhang, IJCV, 1994]?

- Intensity-based registration: composition of image and transform, e.g. $J \circ T \Rightarrow$ use chain rule.
  In 1D: $f \circ g'(x) = f'(g(x)) \cdot g'(x)$. In our case:

\[
\frac{\partial J \circ T(x)}{\partial p} = \frac{\partial J(T(x))}{\partial x} \frac{\partial T}{\partial p}
\]

- Result is a combination of
  - Image gradient
  - Derivatives of the transform for each parameter in each direction ($m \times n$ Jacobian matrix in $mD$ for $n$ parameters)
Image gradient

- Finite difference(s) (approximate), e.g. \((f(x + h) - f(x))/h\)
- Deriche recursive gaussian filters and its derivatives
  [Link](http://hal.inria.fr/inria-00074778/en/)
Derivation of similarity measures

- Measure specific
  - SSD: [Thevenaz, IEEE IP, 1998]
  - MI: [Viola, PhD, 1995], [Thevenaz, IEEE IP, 2000]
  - CR: [Cachier, PhD, 2002]
Jacobian matrix examples

3 translations in 3D

\[
\begin{bmatrix}
\frac{\partial T_x}{\partial t_0} & \frac{\partial T_x}{\partial t_1} & \frac{\partial T_x}{\partial t_2} \\
\frac{\partial T_y}{\partial t_0} & \frac{\partial T_y}{\partial t_1} & \frac{\partial T_y}{\partial t_2} \\
\frac{\partial T_z}{\partial t_0} & \frac{\partial T_z}{\partial t_1} & \frac{\partial T_z}{\partial t_2}
\end{bmatrix}
= \begin{bmatrix}
t_0 & 0 & 0 \\
0 & t_1 & 0 \\
0 & 0 & t_2
\end{bmatrix}
\]

2 translations, 1 rotation in 2D

\[
\begin{bmatrix}
\frac{\partial T_x}{\partial t_0} & \frac{\partial T_x}{\partial t_1} & \frac{\partial T_x}{\partial \theta} \\
\frac{\partial T_y}{\partial t_0} & \frac{\partial T_y}{\partial t_1} & \frac{\partial T_y}{\partial \theta}
\end{bmatrix}
= \begin{bmatrix}
t_0 & 0 & -x \sin \theta - y \cos \theta \\
0 & t_1 & x \cos \theta - y \sin \theta
\end{bmatrix}
\]
Gradient descent

- *steepest (gradient) descent = Cauchy, 1847*
- Minimize $f$ along the gradient, i.e.

$$\delta_{k+1} = -\nabla f(x_k)$$

**Step size?**
- Fixed $\alpha \in \mathbb{R}^+$
- Function of gradient norm, e.g. $\alpha/||\nabla f(x_k)||$
- Line search
  $$\Rightarrow \nabla f(x_k) \cdot \nabla f(x_{k-1}) = 0$$
  $$\Rightarrow$$ Orthogonal steps
Gradient descent - Examples
Gradient descent - Examples
Gradient descent - Examples (Wikipedia)
Conjugate gradient

- Line search in the direction of the gradient
- Construct a conjugate direction to previous gradient

\[ \delta_{k+1} = \nabla f(x_{n+1}) + \alpha_k \delta_k \]

- If \( f \) quadratic, \( \delta_{k+1} \) are mutually conjugate
  \Rightarrow Converge in \( n \) iterations
Conjugate gradient

- Let $g_k = -\nabla f(x_k)$.

- Fletcher and Reeves
  \[
  \alpha_k = \frac{g_{k+1} \cdot g_{k+1}}{g_k \cdot g_k} = \frac{||g_{k+1}||^2}{||g_k||^2}
  \]

- Polak and Ribiere (faster)
  \[
  \alpha_k = \frac{(g_{k+1} - g_k) \cdot g_{k+1}}{g_k \cdot g_k}
  \]

- Stopping criterion: $g_{k+1} = 0$
Newton’s method in 1D

- We seek the step $a$ to go to the optimal position. Taylor serie:

$$f(x + a) = f(x) + f'(x) \cdot a + \frac{1}{2} f''(x) \cdot a^2 + \ldots$$

- Extremum if quadratic (right part)

$$f'(x) + f''(x) a = 0$$

$\Rightarrow$ Go to the optimum in one step (if quadratic)

$$\hat{x} = x_n - \frac{f'(x)}{f''(x)}$$
Newton’s method in nD

- If quadratic, go to optimum in one step

\[ \hat{x} = x_n - H^{-1} \cdot \nabla f(x) \]

⇒ The search direction is \( \delta_k = H^{-1} \cdot \nabla f(x) \)

- In practice, not quadratic ⇒ step \( \alpha \)

\[ x_{n+1} = x_n - \alpha H^{-1} \cdot \nabla f(x) \]
Newton’s method in nD

- Hessian = matrix of second derivatives

- If quadratic, converges in one iteration

- But computing the Hessian matrix costs too much and it is not quadratic...

⇒ Approximate Hessian
⇒ Use as search direction
Quasi-Newton

- Also called *variable metric methods*

- Stores more information than conjugate gradient, e.g. \( n \times n \) instead of \( n \)

- Use \( f(x_n) \) and \( \nabla f(x_n) \) to approximate Hessian

- Different solutions
  - DFP (Davidon-Fletcher-Powell)
  - BFGS (Broyden-Fletcher-Goldfarb-Shanno) (superior?)
Levenberg-Marquardt

- Least-Squares method
- Varies smoothly between
  - Steepest-descent far from minimum
  - Quasi-Newton close to minimum

- Solve at each iteration Hessian approximation

\[(H_k + \lambda_k I) \cdot \delta x_k\]

- Far from minimum: \(\lambda_k \) large \(\rightarrow H_k + \lambda_k I \approx I\)
- Close to minimum: \(\lambda_k \) small \(\rightarrow H_k + \lambda_k I \approx H_k\)
Summary

Without gradient:
- Simplex
- Powell (conjugate directions)

With gradient:
- Gradient descent
- Conjugate gradient
- Quasi-Newton
- Levenberg-Marquardt

Also:
- Genetic algorithms
- Simulated annealing
- ...
**Multiresolution**

- Convergence
- Large deformations
- Robustness to noise and local minima
  (smooth upper levels)
Comparison [Maes, MedIA, 1999]

- High resolution images (256 × 256 × 100), < 1 mm
- Mutual information
- Partial volume interpolation
- 3 resolutions
- 1260 registrations
Comparison [Maes, MedIA, 1999]
Comparison [Maes, MedIA, 1999]
Comparison

- **Full resolution**
  - **Iterations:**
    - SMP, POW, LM, CG < SD, QN
  - **Time:**
    - POW, SMP < CG < LM < SD, QN

- **With multiresolution**
  - **Iterations:**
    - CG, SMP, LM < POW, SD, QN
  - **Time:**
    - SMP < CG, LM, POW < SD, QN

- **Acceleration factor:** 2 to 6 (5 – 15 min)
Numerical recipes: www.nr.com

... with proper care, sometimes outdated or bugged, always double-check validity
Outline

1. Introduction
2. Transformations
3. Similarity measures
4. Optimization
5. Validation
6. Conclusion
## Validation

<table>
<thead>
<tr>
<th></th>
<th>Clinical data</th>
<th>Cadavers</th>
<th>Physical phantoms</th>
<th>Realistic simulations</th>
<th>Numerical simulations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Easy Control</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td><strong>Clinical Realism</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Validation - Anthropomorphic phantom
Example: Retrospective Image Registration Evaluation  
Vanderbilt university (Fitzpatrick)

- Multimodal PET / CT / MRI
- Head
- Markers removed from images during registration
Outline

1. Introduction
2. Transformations
3. Similarity measures
4. Optimization
5. Validation
6. Conclusion
\[ \hat{T} = \arg \max_{T \in \mathcal{T}} S(I, J, T) \]

...or use existing solutions (elastix)
Potential research in each compartment

- Optimization
- Similarity measure
- Parametrization
- Implementation
- ...