Analytic Transform Parameter Optimization using Intensity-Based Similarity Metric

Minyoung Chung

Jan, 2015

Computer Graphics and Image Processing Laboratory
Dept. of Computer Science and Engineering
Seoul National University

Contents

Formulation of Cost Function (SM + T)

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

- $C(\Theta) = \sum_{p\in\Omega} SM(I_R(p), I'_F(p))$
  - $\boldsymbol{\Omega}$: 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
    - $I_x \equiv \frac{\partial I}{\partial x} \approx \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x}$
    - $I_y \equiv \frac{\partial I}{\partial y} \approx \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y}$
    - $\frac{\partial I_F}{\partial T^{-1}}(p) = \left(I_{F_x}(T^{-1}(p;\ \Theta)) \quad I_{F_y}(T^{-1}(p;\ \Theta))\right)$
    - 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*) : $\frac{\partial T^{-1}}{\partial \Theta} = \begin{pmatrix} x & -y & 1 & 0 \\ y & x & 0 & 1 \end{pmatrix}$

Thank you!