

Image Analysis

Image Enhancement:

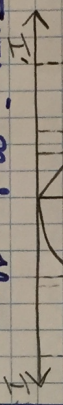
① Contrast - Histogram equalization:

- Problem: Image uses only small intensity range
- Goal: redistribute intensities to larger range
- Soln: Use random variable (intensity I) with uniform density ($\approx$ Histogram) $P(i)$.

Want map to uniform distribution
 $\rightarrow$ "Inversion method" (also very common)

Compute new intensities I' :

$$I' = i_{\max} \cdot CDF(I)$$



- Intuition: "Where $CDF(I)$ is flat: have only few pixels in this range, but nice CDF is flat: wide range of I mapped to small range of I' "

Read $Read\ I' \sim Uniform(0, i_{\max})$:

$$\begin{aligned} P(I' \leq i) &= P(I \leq CDF^{-1}(\frac{i}{i_{\max}})) \\ &= CDF(CDF^{-1}(\frac{i}{i_{\max}})) = \frac{i}{i_{\max}} \\ \Rightarrow P(i) &= \frac{d}{di} P(I' \leq i) = \frac{1}{i_{\max}} \quad \square \end{aligned}$$

② Intensity Normalization:

- Problem: Different images from same object can have different appearance!

WDG? Wrong of camera, protocol, then him

$\rightarrow$ In particular MRI/US $\rightarrow$ the double intensities
 less important for CT $\rightarrow$ found field with!

Estimate CDF via "cumulative histogram"

(*) Different approach:
 Recall how densities are transformed! (Place some mass on corresponding sets)
 $P(I) = \frac{dP(I')}{dI}$

- Given two images I & $J \rightarrow$ different "linear maps" needed to map them to histograms:

$$J' = \frac{(i_{\max} - i_{\min})}{(j_{\max} - j_{\min})} (J - j_{\min}) + i_{\min}$$

Note: Similar to earlier! Can use e.g. $\frac{1}{2}k(1-t)!$ values of CDF!

$$J' = \frac{\sigma_I}{\sigma_J} (J - \mu_J) + \mu_I$$

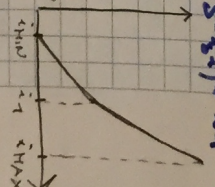
- Standardization: (Myul's method)

"Reserve linear map", can use different standard

$$J' = \frac{(i_{\max} - i_{\min})}{(j_{\max} - j_{\min})} (J - j_{\min}) + i_{\min}$$

- Double check when working with histograms:

- Make use of image contents (corner info)
- Does not know object size (we made relative for different)
- Does not use about histograms



③ Noise Suppression:

- Different models, e.g. $J = I + \epsilon(x)$, $\epsilon(x) \sim \mathcal{N}(0, \sigma^2)$

$$J = \epsilon(x) \cdot I$$

- Linear filter: Convolution with filter kernel

Problem: edges are lost
 in e.g. segmentation: Noise uniform over frequencies, linear filter distinguishes but signal (due in e.g. I) makes noise from edges!

Examples: $\frac{1}{2}k(1-t)$ in low frequencies $\rightarrow$ less noise
 $\frac{1}{2}k(1-t)$ in high frequencies $\rightarrow$ more noise
 APPROX. TO GAUSSIAN

Gaussian filtering is like running diff. eqn. with constant

add ③ Noise Suppression:

↳ Rank-order "Filter": (e.g. median)

• For each pixel, create rank order of neighboring values → pick median

• Non-linear → not commutative

• May create artifacts

↳ Anisotropic diffusion:

• PDE Gaussian Equation: $\frac{\partial I}{\partial t} = \nabla \cdot (c \cdot \nabla I)$

Solution: $I(x, t) = I_0 * \frac{1}{\sqrt{2\pi} \cdot \sigma^2} \exp(-\frac{1}{2} \frac{x^2}{\sigma^2})$

$$\sigma^2 = 4ct$$

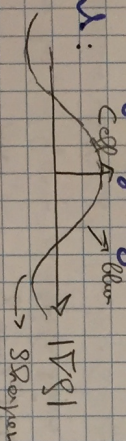
• Idea: Make Gaussian filter context aware

by making high (c) dependant on ∇I

$$e.g.: c = \exp(-|\nabla I|^2)$$

Plug into diff. eqn. gives an "effective"

constant:



Solution only valid for $c > 0$ (issue with FT)

For $c < 0$: Put $\ominus$ into time → Run diffusion in "reverse time" = Sharpen

• Solve numerically:

$$\frac{\partial I}{\partial t} \approx \frac{I^{(t+1)} - I^{(t)}}{\Delta t} = \nabla \cdot (c \cdot \nabla I) = \nabla c \cdot \nabla I + c \cdot \Delta I$$

$$I^{(t+1)} = I^{(t)} + \Delta t [\nabla c \cdot \nabla I + c \Delta I]$$

• Need to set Δt small ($CFL: \Delta t \leq \frac{1}{2} \Delta x^2$)

• To ensure converging results (and not just converge to homogeneous image) → restraining for

$$\frac{\partial I}{\partial t} = \nabla \cdot (c \nabla I) + \frac{1}{\sigma^2} (I - S)$$

Preserve edges
May preserve noise (e.g. salt & pepper)

average neighborhood

↳ Global filter:

• Typical noise suppression method: $I(p) = \sum_{q \in S} w(p, q) I(q)$

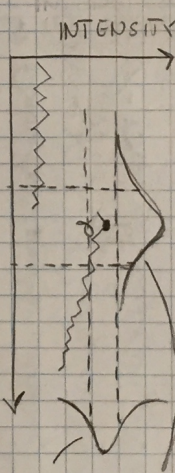
• E.g.: Gaussian blur: $I(p) = \sum_{q \in S} G_\sigma(\|p - q\|) I(q)$

→ Unless hybrid distance method

• Idea: Subdivide "edge term" → including range method

$$I(p) = \frac{1}{2} \sum_{q \in S} G_\sigma(\|p - q\|) G_\sigma(\|I_p - I_q\|) I(q)$$

• Subdivision in 1D: (e.g. a line of pixels)



$G_\sigma(\|p - q\|)$: "Only average over pixels, don't care about range"

$G_\sigma(\|I_p - I_q\|)$: "Only average over pixels with similar intensity"

• "Given individual kernel for each pixel"

↳ Non-local mean (NL-mean): (relation to bilateral filter)

• "Average neighbors with similar neighborhood"

• For every pixel p: Define V_p ... list of intensities

"Some kind of feature vector" in neighborhood of p → $V_p = S(V_p)$

• Compute weights $w(p, q)$ based on (dis-)similarity of neighborhood: $I(p) = \frac{1}{2} \sum_q G_\sigma(\|V_p - V_q\|) I(q)$

• No spatial term - providing average over pixels very far away

• Less redundancy in images ("local structure input")

• May create artifacts...

Non-local means
I, ..., I, ..., I
p, q, ..., p, q, ..., p, q, ...

(4) Bias Cancellation: (MRI artifacts)

↳ When does it come from?

- Field inhomogeneity ($\mu_{tissue} \neq 1$)
- Varying coil sensitivity

↳ Model: "Smooth" multiplicative factor

$$\text{Observed} \leftarrow \tilde{I} = I \cdot \tilde{\mu} + \tilde{n} \quad \begin{matrix} \text{Noise} \\ \text{Real image} \end{matrix}$$

↳ Cancellation methods:

- Additional measurement "at Receiver"
- "Spin in a bag" (Modelly placed Landmark)
- Joint regularization (Same structure $\Rightarrow$ Same index)
- $N3$ - algorithm

↳ Details of $N3$:

- Assume noise $\tilde{n} = 0$ & Acquire $\log(I)$

$$\tilde{I} = I \cdot \tilde{\mu} \quad \begin{matrix} \log \tilde{I} \\ \log I \\ \log \tilde{\mu} \end{matrix}$$

Pre requisite: Sum of independent RV

$$Z = X + Y \Rightarrow P(Z \leq z) = \int_{-\infty}^{\infty} \int_{-\infty}^{z-x} P_X(x) P_Y(y) dy dx$$

$$P_Z(z) = \frac{d}{dz} P(Z \leq z) = \int_{-\infty}^{\infty} \frac{d}{dz} \int_{-\infty}^{z-x} P_X(x) P_Y(y) dy dx$$

$$= \int_{-\infty}^{\infty} P_X(x) P_Y(z-x) dx$$

Assume pixels are iid samples of $I, \tilde{\mu}, \tilde{I}$

$$\tilde{I}, \tilde{\mu}, \tilde{I} \Rightarrow \tilde{I} = \tilde{\mu} \cdot \tilde{I}$$

Assume $\tilde{\mu} = N(0, \sigma^2)$ for small σ^2

$$\tilde{I}, \tilde{\mu}, \tilde{I} \Rightarrow \text{Pixel-wise} \Rightarrow \text{1D Conv}$$

SUMMARIZE: ASSUMPTIONS OF $N3$

- i) Zero Noise
- ii) Pixels iid
- iii) Bias is Gaussian

↳ and Details of $N3$:

- Procedure: i) Estimate $\tilde{I}$ from $\tilde{I}$ (Reconvolution)
- ii) Estimate $\tilde{\mu} = E[\log \tilde{I}]$

(i) Reconvolution of $\tilde{I}$ - Assume filtering

$$\tilde{I} = \tilde{\mu} * \tilde{I} \Rightarrow \hat{\tilde{I}} = \tilde{F}(\cdot) \Rightarrow \hat{\tilde{I}} = \tilde{\mu} * \tilde{I}$$

$$\Rightarrow \hat{\tilde{I}} = \frac{1}{\tilde{\mu}} \cdot \hat{\tilde{I}} = \frac{\tilde{\mu}^*}{\tilde{\mu}^2} \cdot \hat{\tilde{I}} \approx \frac{\tilde{\mu}^*}{\tilde{\mu}^2 + \epsilon} \cdot \hat{\tilde{I}}$$

ϵ - small $\rightarrow$ for better stability (no zero division)

(ii) Estimate of $\tilde{\mu} = E[\log \tilde{I}]$

Pre requisite: Conditional distribution of sum of indep. RV

$$Z = X + Y \Rightarrow P(Z = z | X) = P(X + Y = z | X = x)$$

$$\text{1D - Substitution: "Conv"} \quad = P(Y = z - x | X = x) = P_Y(z - x)$$

$$E[\log \tilde{I}] = E[\log \tilde{\mu} + \log \tilde{I}] = \log \tilde{\mu} + E[\log \tilde{I}]$$

$$E[\log \tilde{I}] = \int_{\mathbb{R}} \log(\tilde{\mu}) P(\tilde{\mu}) d\tilde{\mu} = \int_{\mathbb{R}} \log(\tilde{\mu}) P(\tilde{\mu}) d\tilde{\mu}$$

$$E[\log \tilde{I}] = \frac{1}{\tilde{\mu}} \int_{\mathbb{R}} \log(\tilde{\mu}) P(\tilde{\mu}) d\tilde{\mu}$$

↳ With $\tilde{\mu}$ above can in principle compute an estimate of the bias at every pixel! BUT: 2 Problems

i) Need for smoothing:

We ignored Noise + Don't really know $\tilde{\mu} \sim N(0, \sigma^2)$

Solution: Compute $\tilde{\mu}$ only on outer pixels

$\rightarrow$ Smooth interpolation

New: "SAFE" Deconvolution with small σ^2

Some as above:

Load values of N_3 :

(i) Need for iteration:

Deconvolution is unstable for large σ^2 of N_3

Trick: "Large Gaussian is convolution of small ones"

$$N_3 = N_{3_1} * N_{3_2} * \dots * N_{3_n}, \quad N_{3_i} \dots \text{small } \sigma^2$$

$$j = k_1 + \dots + k_n + i, \quad j = N_{3_1} * \dots * N_{3_n} * I$$

$$\text{Define: } x_0 = j, \quad x_0 = j = N_{3_1} * \dots * N_{3_n} * I$$

$$x_1 = x_0 - k_1, \quad x_1 = N_{3_2} * \dots * N_{3_n} * I$$

$$x_i = x_{i-1} - k_i, \quad x_i = N_{3_{i+1}} * \dots * N_{3_n} * I$$

$$\Rightarrow i\text{-th iteration: } x_i = F^{-1}(\hat{F} \hat{x}_{i-1})$$

$$k_i = E[k_i | x_{i-1}] = x_{i-1} - E[x_{i-1} | k_i]$$

Results in the following algorithm:

$$k = 0, \quad j = \log(\hat{z}) \dots \text{observed image}$$

$$\hat{F} = \frac{\hat{z}^*}{|\hat{z}|^2 + \epsilon} \quad (\hat{z} = F(k(0, \sigma^2)), \quad \sigma^2, \epsilon \ll 1)$$

while $\| \Delta k \| > \text{criterion}$

$$j = \text{DemodulateEstimate}(j)$$

$$I = F^{-1}(\hat{F} \cdot \hat{z})$$

$$\Delta k_{i+1} = j - E[j | i] \text{ @ } x_n \text{ or } \text{Gaussian}$$

$$\Delta k(x) = \text{SmoothFilter}(\Delta k(x_n))$$

$$\text{Update: } k = k + \Delta k$$

$$j = j - \Delta k$$

hidden without x_i

large σ^2 of N_3

$\Rightarrow N_3$ very narrow $\Rightarrow$ High divide

"many work with small number"

Image Transformations: (Registration Transforms)

$\hookrightarrow$ Registration will address the following problem:

Given two images $I(x)$, $J(x)$ of the same object

(but possibly under different conditions) find a transformation, such that

$$I(x) = J(T(x))$$

$\hookrightarrow$ First address problem of evaluating $J(T(x))$:

(i) Forward mapping:

Transform each pixel of input image onto output $x' = T(x)$

Problem: x' won't correspond to exact pixel coord.

• Multiple input pixels mapped to same output

• Some output pixels may be unmapped (gap)

Solution: Map corners of input pixels (see pic)

For all output pixels: Compute pixel results!

Result: Complex value depending on coverage of transformed input image!

input image!

(ii) Backward mapping:

Transform each output pixel $x = T^{-1}(x')$ and

assign value depending on position x in input!

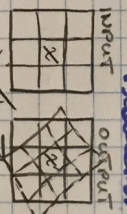
Problem: $T^{-1}(x')$ won't correspond to exact input pixel coordinates!

input pixel coordinates!

Solution: Interpolation

For x :

Results $T^{-1}(\cdot)$ not always existing/unique



NN - Interp.:
Single possible choice for categorical data

Recall: Piecewise linear interpolator for band-limited signals:
→ sinc(x) → (infinite support)

Interpolation Methods:

① Nearest-Neighbor Interpolation:

* Sets pixel value to intensity of nearest pixel $A_T^{-1}(x)$

+ Fast, no near-neighbor values - artifacts

② Linear/Bilinear/Trilinear Interpolation:

(1D): $g(x) \approx \frac{g(x_2) - g(x_1)}{x_2 - x_1} (x - x_1) + g(x_1)$

$$= \frac{1}{x_2 - x_1} [(x_2 - x)g(x_1) + (x - x_1)g(x_2)]$$

$\Delta x = x_2 - x_1$
 $\Delta x = 1$
 $\Delta x = 1$

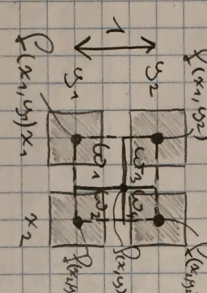
(2D): Very natural extension of above formula

$$g(x_1, y) \approx w_1 g(x_1, y_1) + w_2 g(x_1, y_2) + w_3 g(x_2, y_1) + w_4 g(x_2, y_2)$$

$$w_1 = (x - x_1)(y_2 - y_1)$$

$$w_2 = (x_2 - x)(y_2 - y_1)$$

$$\vdots$$



③ Cubic Convolutional Interpolation:

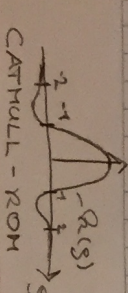
For equally spaced data (spacing = 1) often used interpolation: $g(x) \approx g(x) = \sum_a c_a Q(\frac{x - x_a}{\Delta x})$

$Q(x) \dots$ Kernel Function

Piecewise cubic kernel for kernel, e.g.

$$Q(x) = \begin{cases} A_1 |g|^3 + B_1 |g|^2 + C_1 |g| + D_1 & |g| \leq 1 \\ 0 & |g| > 1 \end{cases}$$

How to determine c_a ?
Interpolated has to pass through sample points $g(x_a) = g(x_a)$



Could increase support to $[-3, 3]$ → would get 1 order better approx.

Example: Designing the Catmull-Rom kernel

Given the asymmetric kernel, we need 8 conditions:

- $Q(0) = 1$ & $Q(1) = Q(2) = 0$
- (Because then $c_a = g(x_a) \dots$ Efficiency)
- Continuous & Continuous 1st-derivative

$$Q'(0) = Q'(2) = 0$$

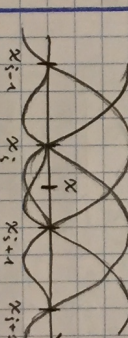
$$Q''(0) = Q''(1) \quad \& \quad Q''(1) = Q''(2)$$

Can use 8 conditions to rewrite kernel in form of a single parameter, say $A_2 = a$:

$$Q(x) = \begin{cases} (a+2)|g|^3 - (a+3)|g|^2 + 1 & |g| \leq 1 \\ a|g|^3 - 5a|g|^2 + 8a|g| - 4a & |g| > 1 \end{cases}$$

Goal: Choose a, s.t. find 3 terms of Taylor-expansion of $Q(x)$ & $g(x)$ match!

Consider interpolation of an arbitrary point x



$$g(x) = c_{i-1} \cdot Q(1+s) + c_i \cdot Q(s) + c_{i+1} \cdot Q(1-s) + c_{i+2} \cdot Q(2-s)$$

Taylor-expansions: $[g(x) \approx g(x_i) + g'(x_i)(x - x_i) + \frac{1}{2}g''(x_i)(x - x_i)^2 + \dots]$

$$c_{i-1} = g(x_{i-1}) - a g'(x_{i-1}) + \frac{1}{2} a^2 g''(x_{i-1}) + O(a^3)$$

$$c_i = g(x_i) = g(x_i) + a g'(x_i) + \frac{1}{2} a^2 g''(x_i) + O(a^3)$$

$$c_{i+1} = g(x_{i+1}) = g(x_i) + a g'(x_i) + \frac{1}{2} a^2 g''(x_i) + O(a^3)$$

$$c_{i+2} = g(x_{i+2}) = g(x_i) + 2a g'(x_i) + 2a^2 g''(x_i) + O(a^3)$$

$$\text{Plug in: } g(x) - g(x_i) = (2a+1)[a g'(x_i) + \frac{1}{2} a^2 g''(x_i)] + \dots$$

$$\Rightarrow a = -\frac{1}{2}$$

Choosing a transformation den for registration is like:

"Defining the function type we go for"

Classification of Transformations:

① Rigid-body ("Rigid-body Transformation")

"Rigid distance"

3 DOF: Rotation & Translation

Variables: Angles, Angles, Angles

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

② Similarity

4 DOF: Rotation & Translation + Isotropic Scaling

Variables: Angles, Translation, Ratio of lengths/area

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

③ Affine

6 DOF: Rot. & Transl. + Scaling in x/y + "Shear"

Variables: Rotated lines, Ratio of areas

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

④ Projective

8 DOF

Variables: "Conic ratio"

(Ratio of Ratio) of 4 collinear points

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \\ w \end{pmatrix}$$

"Projective linear group"

Variables: Similarity + Affine + Projective

Now for non-rigid transformation:

⑤ Quadratic Transformation Model

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & \dots & a_{16} \\ a_{21} & \dots & a_{26} \\ 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ x^2 \\ y^2 \\ xy \\ 1 \end{pmatrix}$$

12 DOF

TPS are a non-parametric method for interpolation! Given a set of known transformations $(x'_i) = T(x_i)$ interpolate over other points?

Typically interpolate x & y independently

③ Thin-plate splines (TPS)

Can think about it as of extension to quadratic model, i.e.

just there in more additional degrees of freedom, but actually TPS can be derived from a different perspective!

TPS are a generalization of cubic splines in 1D

$\Rightarrow$ They interpolate the data while minimizing a "bending energy"

$$E(g) = \int_{\mathbb{R}^2} |D^2 g|^2 dx$$

$D^2 g \dots$ Hessian of 2nd-order derivatives, $| \cdot |^2$ Frobenius norm

SPQR of derivation in 1D:

Assume there were no constraints (no sample points).

The variational calculus to minimize $E(g) = \int_{-\infty}^{\infty} g''^2 dx$

$$\frac{d}{dt} E(g + \epsilon \varphi) \Big|_{\epsilon=0} = \frac{d}{dt} \int_{-\infty}^{\infty} (g'' + \epsilon \varphi'')^2 dx \Big|_{\epsilon=0}$$

$$\xrightarrow{\text{Integration by parts}} = \int_{-\infty}^{\infty} 2 g'' \cdot \varphi'' dx = \int_{-\infty}^{\infty} 2 g^{(4)} \cdot \varphi dx = 0$$

$$\Rightarrow g^{(4)} = 0 \quad (\text{Euler-Lagrange equation})$$

Now include constraints! Just? Treat $g^{(4)}$ in a distributional sense $\rightarrow$ zero everywhere except for

discontinuities of sample locations: $g^{(4)} = \sum_i \delta(x - x_i)$

Use linearity: "Solve for a single sample point"

(= "Green's function"), then superimpose.

$$\frac{\partial g^{(4)}(x)}{\partial x_i} g(x_i, s) = \delta(x - s) \Rightarrow g(x, s) = \frac{1}{4!} |x - s|^3$$

$$\Rightarrow g(x) = \sum_i a_i \cdot g(x, x_i)$$

Sample random

$b_0 + b_1 x \dots$ does in orthogonal complement of function space containing $G(x, x_i)$

→ Maybe this is why we can only get "natural splines"?

ad ③ (TPS)

ad 1D deviation model - cubic splines:

Note that the solution is not unique, since for

any $g(x) = b_0 + b_1 x \rightarrow E(g) = 0$ Cubic-Spline

(linear polyom. in $\text{Range of } E(\cdot)$) Natural spline

Full solution: $g(x) = \sum_{i=1}^N a_i G(x, x_i) + b_1 x + b_0$

Problem: Have $N+2$ parameters, but only N

equations $g(x_i) = y_i \mid (X)$ (***)

Solution: "Orthogonality constraint" $\begin{cases} \sum a_i = 0 \\ \sum a_i x_i = 0 \end{cases}$

Write (X) & $(**)$ as linear system:

Now in equation $g(x_i) = y_i$

$$\begin{pmatrix} G(x_1, x_1) & G(x_1, x_2) & \dots & G(x_1, x_N) & x_1 & 1 \\ G(x_2, x_1) & G(x_2, x_2) & \dots & G(x_2, x_N) & x_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ G(x_N, x_1) & G(x_N, x_2) & \dots & G(x_N, x_N) & x_N & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \\ 0 \\ 0 \end{pmatrix}$$

Orthogonality constraint

Note: Different deviation & even different system

to be solved, but results in same interpolation as

in CHEA I

In 2D: dimension, e.g. 2D for nice images:

Same procedure: $E(g) = \int \Omega^2 \Delta g \, dx \, dy$

→ Euler Lagrange: $\Delta \cdot \Delta g = 0 \rightarrow \text{HARMONIC EQUATION}$

→ Green function: $G(x, \tilde{x}) = G(x, 1 - \tilde{x}) = G(x, 1 - \tilde{x})$

$= x^2 \cdot \log(x)$

→ Adding the kernel of $E(g)$ gives full solution

$g(x, y) = \sum_{i=1}^N a_i G(x, y, \tilde{x}_i) + b_2 x + b_1 y + b_0$

→ Constraints: $g(\tilde{x}) = g_i^{\text{true}}, \sum a_i = \sum a_i x_i = \sum a_i y_i = 0$

ad ③ (TPS)

To use thin plate harmonization: assume we're given

a set of homologous points & correspondences, e.g.

$(g_i^{\text{true}}, g_i^{\text{true}}) = (x_i, y_i), \text{ then } T(x) \approx (g(x, y)) \text{ where}$

$g(x, y) \approx g(x, y)$ are naturally fitted TPS or dense!

Advantages: + $\tilde{x}$ can be arbitrarily spaced (but uniform better)

+ Can be combined with regularization $\rightarrow$

Disadvantages: - $G(x, x_i) \dots$ infinite support (global)

⑦ Two-term regularization (FTD) via B-Splines

"Similar to TPS in the way that it is also non-

parametric"

Difference: fixed number (also called control-

SHOWS CONTROL points) on regular grid! $\rightarrow$ disadvantage (-)

GRID + REGION OF INFLUENCE $\rightarrow$ advantage (+)

WHEN EVALUATING $\rightarrow$ Control points only have local effects, "compensation" $\rightarrow$

in 2D: $u(x, y) = \sum_{i=1}^N \sum_{j=1}^N \tilde{u}_{i,j}^{\text{true}} \cdot \tilde{u}_i(x) \cdot \tilde{u}_j(y)$

$\tilde{u}_i(x), \dots$ "B-spline functions"

$\tilde{u}_{i,j}^{\text{true}} \dots$ known displacements of control

points (i, j)

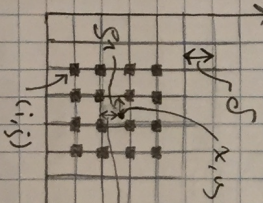
$i = \lfloor \frac{x}{\delta} \rfloor - 1, j = \lfloor \frac{y}{\delta} \rfloor - 1, \delta_1 = \frac{x}{\delta} - \lfloor \frac{x}{\delta} \rfloor, \delta_2 = \frac{y}{\delta} - \lfloor \frac{y}{\delta} \rfloor$

$\delta \dots$ "Grid spacing"

Note: from the definition it follows $\rightarrow$ Control exclude

$\tilde{u}(x, y)$ for points $x \leq \delta$ or $y \leq \delta$ ($\delta \times \delta$ "true" d)

$\rightarrow$ "Region of influence" would extend beyond grid...



Landmark Based Registration:

↳ Goal of Registration:

Given two images $I(x)$ & $I(x')$ of the same object,

Find a transformation $x' = T(x)$, s.t. $I(x') = I(T(x))$

↳ Idea of "Landmark - Based registration":

i) Parameter $T(x)$ (e.g. using a set of previous steps)

ii) Find some corresponding points (a sparse set)

iii) Estimate parameters of $T(x)$ using these "samples"

↳ Guided Steps: "finding correspondences"!

Approach: i) Finding interest points (DETECTOR)

ii) Positive region around these points

iii) Describe the content within regions

(given "Global representation of pixels")

iv) Match descriptors/features

(i) Detector - Finding interest points:

↳ Want some good interest points? + Corners!

↳ How can we detect them:

"SARF"ing around a window $w(x,y)$ in an arbitrary direction (u) would result in

large intensity changes

Want $E(u,v) = \sum_{x,y} w(x-x',y-y') \| \nabla I(x'+u,y'+v) - \nabla I(x',y') \|^2$

large!

Target: $I(x'+u,y'+v) \approx$

$\approx I(x',y') + I_x u + I_y v$

$\approx I(x',y') + I_x u + I_y v$

Goal (i) Detector:

↳ How can we detect them

$\Rightarrow E(u,v) = \sum_{x,y} w(x-x',y-y') \| \nabla I(x'+u,y'+v) - \nabla I(x',y') \|^2$

M is given by a convolution

$w(x,y) * \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$

$= \sum_{x,y} w(x-x',y-y') \cdot (u \ v) \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$

$= (u \ v) \cdot \left[\sum_{x,y} w(x-x',y-y') \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \right] \begin{pmatrix} u \\ v \end{pmatrix}$

$E(u,v) = u^T M u$

We want M in quadratic form so the large for any direction u . Since M is symmetric & yield it will

have $\lambda_1, \lambda_2 > 0$. Can understand this meaning as

align: $E(u,v)$ will be large for arbitrary (u,v) , if

$\lambda_1 \& \lambda_2$ are large

Avoid computing eigenvalues $\rightarrow R = \det(H) - \alpha \det(H)^2$

Notes: i) Cross terms in Equation

would not be $\Rightarrow$ How can we detect them in rotation invariant

How can we detect them: ii) Harris-corner detector is NOT rotation invariant

↳ How can we detect them:

"Corner Rate" of Harris-Hough large $\det(H)$

$H_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} I(x)$ $\Rightarrow \det(H) = \det \begin{pmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{pmatrix}$

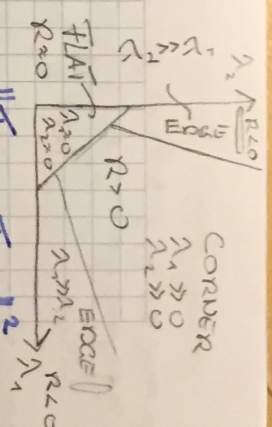
↳ Some modeling:

Take interest points as uniform grid of pixels over

the entire image!

How can we detect them:

How can we detect them:



"PATCH SELECT"

② Region Selection (Scale insensitive):

↳ Detector gave us an interest points! The descriptor will be defined over a small region centered at this interest point. How do we define this region in a scale independent manner? (Scale does not end up with the same region in differently scaled images)

↳ Main Exhaustive Search:

"Compare descriptor while varying patch size"
 Significant: Works Good Ring / Redundant for recognition!

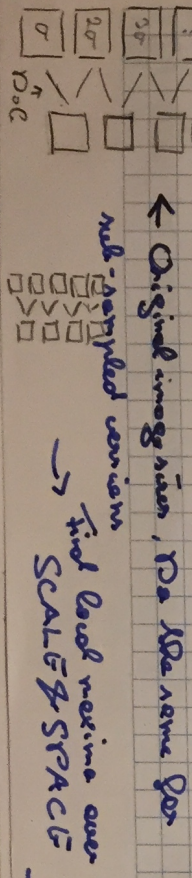
↳ Automatic Scale Selection:

Create a function, that depends on the size of the patch $g(s)$. Create the size s which is a local extremum of $g(s)$! Why? $\rightarrow$ True if $g(s)$ is $\cap$ ("At extremum g does not change with s " $\rightarrow$ "Independent of scale")

APPROX. Examples for $g(s)$: Average Intensity; Difference of Gaussian $\rightarrow$ Laplacian of Gaussian (LoG)

Note: Can use multiple function either in combination

with detection, or on their own - SIFT:



③ Description - Creating the feature:

↳ Have some "interest points" + "patch" $\rightarrow$ How to describe the content of this patch with a simple vector (will make it easy for matching)

↳ Population of good features:

- i) Invariant to illumination / scale / rotation / affine / ...
- ii) Local: Robustness to clutter / occlusion / ...
- iii) Distinct (for easy matching)
- iv) Efficient to compute

↳ Interesting bits: Simple & portable feature

Very sensitive to shifts $\rightarrow$ Solution: Use relative over patch

↳ SIFT - Feature: $\rightarrow$ Compute RELATIVE for "differential detection"

Divide patch in 4×4 grid & compute Histogram of gradient direction in each cell (8 bins) $\Rightarrow 4 \times 4 \times 8 = 128$ (Total approx.: SURF)

↳ Self-Similarity Descriptor: $\rightarrow$ MULTI-MODAL "approx. good for interest points"

i) Compute correlation of small image patch with larger region around $\rightarrow$ "Correlation map" (local appearance can be important)

ii) "Bag-Of-Features" Transform & subsampling $\rightarrow$ Scale invariance

④ Feature Matching:

↳ 2nd NN criterion:

Matched features based on their Euclidean distance

• Accept only on patch if ratio between distances of nearest & 2nd nearest neighbor is small enough, e.g. $\frac{d_1}{d_2} < 0.8$

"Nearest neighbor is significantly closer than 2nd nearest neighbor"

Can scale all Q_i 's by same amount $\rightarrow$ Same Trade.
For optim. problem below $\rightarrow$ Need a constraint, e.g. $\|Q\|_1 = 1$

$$A = UDV^T, Q = \sum_{i=1}^n b_i \vec{v}_i$$

$$\|A\vec{Q}\|_2^2 = \vec{Q}^T V D^2 V^T Q$$

$$\|A\vec{Q}\|_2^2 = \sum_{i=1}^n b_i \vec{v}_i^T \sum_{j=1}^n a_{ij}^2 \vec{v}_j^T \sum_{k=1}^n b_k \vec{v}_k$$

$$= \sum_{i=1}^n b_i \sum_{j=1}^n a_{ij}^2 \sum_{k=1}^n b_k \vec{v}_i^T \vec{v}_j \vec{v}_k$$

$$= \sum_{i=1}^n b_i \sum_{j=1}^n a_{ij}^2 \sum_{k=1}^n b_k \delta_{ijk}$$

$$= \sum_{i=1}^n b_i \sum_{j=1}^n a_{ij}^2 b_i$$

$$= \sum_{i=1}^n b_i^2 \sum_{j=1}^n a_{ij}^2$$

$$= \sum_{i=1}^n b_i^2 \sigma_i^2$$

Choose $b_i = 1$
for smallest singular value σ_n

5) Estimate the Transform:

$\hookrightarrow$ Assume we found a set of correspondences (2-4).
How to estimate the transform $T(\vec{x})$ from X and Y ?
Depends on the class we chose for T !

$\hookrightarrow$ Least-Squares Estimation for Registration:

Transformation model: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

$$\Rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{Q_{11}x + Q_{12}y + Q_{13}}{Q_{21}x + Q_{22}y + Q_{23}}$$

Build linear system for OLS:

$$\vec{x}'(Q_{11}x + Q_{12}y + Q_{13}) = Q_{21}x + Q_{22}y + Q_{23}$$

$$\vec{y}'(Q_{11}x + Q_{12}y + Q_{13}) = Q_{21}x + Q_{22}y + Q_{23}$$

$$-Q_{21}x - Q_{22}y - Q_{23} + Q_{11}x' + Q_{12}y' + Q_{13}x' = 0$$

$$-Q_{21}x - Q_{22}y - Q_{23} + Q_{11}x' + Q_{12}y' + Q_{13}x' = 0$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$\vec{Q}^T = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{21} & Q_{22} & Q_{23} \end{pmatrix}$$

$$D = \{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)}) \}$$

Non-Rigid Registration:

$\hookrightarrow$ Recall the problem definition for registration

Given two images I & J of the same object, find transformation, map I to J : $I(\vec{x} + \vec{t}) \sim J(\vec{x})$

I : Static Image / Reference Image

J : Moving / Moving Image

$\hookrightarrow$ General Transform:

Fixed $\rightarrow$ Metric $\rightarrow$ Optimizer

Modeling $\rightarrow$ Subpr. $\rightarrow$ Transform $\rightarrow$ Model

$\hookrightarrow$ TPS & B-Splines can be used for non-rigid registration (no previous relation). TPS minimizes bending energy, where optimization is done analytically, resulting in a linear system of equations (58 control points fixed, considered parametric)

1) Elastic Deform Registration:

$\hookrightarrow$ Deform align can be interpreted (approximatively) in terms of the above general formula.

Modeling: - Metric: SSR = $\sum_i (I - J)^2$

- Optimizer: 2nd-order method; Grid

- Deform: 2nd-order method; Grid

- Transform: Deformation; Grid

- Transform: Deformation; Grid

- Transform: Deformation; Grid

- Transform: Deformation; Grid

- Transform: Deformation; Grid

What's non-rigid?
- Compensate for some deformation
- Subpr. - Subpr. Subpr.
- Non - Segmentation

ad (1) Elastic Net:

↳ But can also derive Netom update differently, similar to "Optical Flow" $\Rightarrow$ Recall ∇T around

invariant would stay constant $\frac{d}{dt} I = 0$
 $\frac{d}{dt} I = 0 = I_t + I_x u_x + I_y u_y \Rightarrow \nabla I \cdot \vec{u} = -I_t$

Find $\vec{u}$ via linearization of Min eqn. - let's do a

picture derivation instead!

Note: $\vec{u} \parallel \nabla I \Rightarrow \vec{u} = \frac{\nabla I}{\|\nabla I\|} \cdot \frac{\|\nabla I\|}{\|\vec{u}\|}$

Additionally $\frac{(I_t - I)}{\|\vec{u}\|} = \|\nabla I\|$
 $\Rightarrow \|\vec{u}\| = \frac{(I_t - I)}{\|\nabla I\|}$

$\Rightarrow \vec{u} = \frac{(I_t - I) \nabla I}{\|\nabla I\|^2}$

↳ Problem: "Aperture Problem"

Update rule ill-defined for $\|\nabla I\|^2 \approx 0$

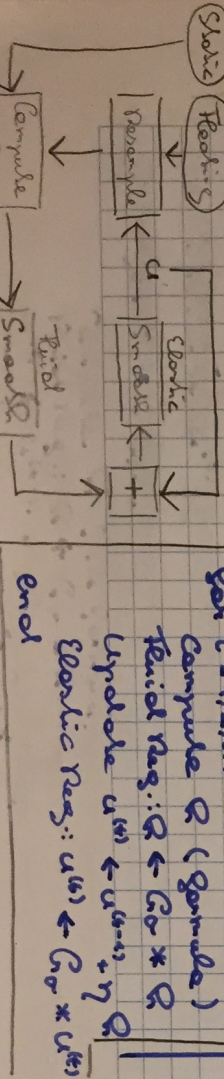
Multiplying with $\frac{\nabla I}{\|\nabla I\|^2 + (I_t - I)^2}$ for stability!

↳ Netom update step:

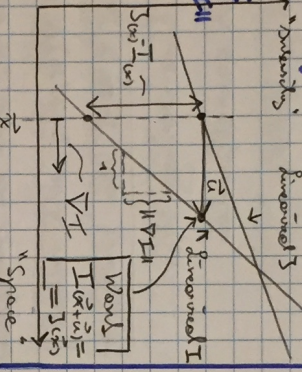
$$Q = \frac{(I_t(x) - I(x+u)) \nabla I}{\|\nabla I\|^2 + (I_t(x) - I(x+u))^2}$$

↳ Regularize via simple smoothing of displacement field with Gaussian kernel!

Find Algorithm:



We derived displacement field using linearization, ...
 improve accuracy of the approach by making smaller linear steps & iterate!



Consider focusing image I on non-linear at later time

ad (1) Elastic Net:

↳ Trier's original description of the Netom

• Netom algo is based on a rigid motion model:

- (i) Take column of an digit in (static) img I
- (ii) (Focusing) img I is deformable - vectors can be classified as "inside" / "outside"
- (iii) Netom: Egghorn riding on carban in I

Concept of POLARITY

Netom loading push vectors of I depending on where they are labeled as "inside" or "outside"

• I_{in} is in vector the attraction models:

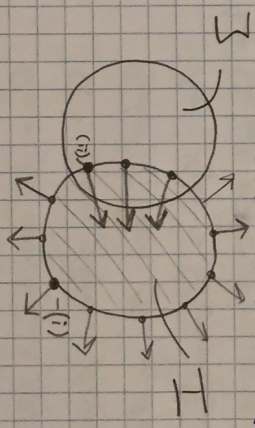
"Forces in image I pointing in direction of next similar points in image I"

↳ Quick derivation for Q from above formula implements "Netom". This reminds formula for

Force displacement from I to I word. any form:

$$Q = \frac{(I_t(x) - I(x+u)) \nabla I}{\|\nabla I\|^2 + (I_t(x) - I(x+u))^2} \propto (I_t(x) - I(x+u)) \nabla I$$

in Netom model of 'Definite' points in colour



(i) Here $(I_t(x) - I(x+u)) > 0 \Rightarrow$ force in same dir as ∇I

(ii) Here $(I_t(x) - I(x+u)) < 0 \Rightarrow$ force in opposite dir to ∇I

Assumed 'insulating' on edges between obj. & bg.

The derived OT spread is only one multiplication of I & not general idea

Make use of the degree, which is associated to our LIE-group... (Why didn't I study more...)

Subspace in group of diffeomorphisms

Isomorphisms: Diffeomorphisms

→ Diffeomorphic Transformation:

- the invertible (one to one)
- Measure Theory → the field even

→ Diffeomorphic Transformations don't form a vector space. (Not closed under addition) ? Rather they form a Lie-group (closed under composition) !

→ How do constraints transformation not to leave the "space of diffeomorphisms" : $\downarrow$ constraints

Define displacement field u via a velocity field v , namely the points x_0 will be transformed to the solution of $\dot{x}(t) = v(x, t)$, with initial condition $x(t_0) = x_0$, at time $t = 1$ we have $u = x(t_1) - x_0$

→ Problem: Need to integrate time varying $v(t)$...
 → Constraints ourselves the "stationary velocity fields"

② Diffeomorphic Deformations:

→ Clonic deformations for arbitrary transformation
 Can lead to non-invertibility & often non-smooth displacement fields

→ Can combine the "optimization problem" of clonic deformations to the group of diffeomorphisms

More properties are:
 - Sometimes a MUST HAVE (computational accuracy)
 - Sometimes they do NOT reflect reality (e.g. upon tumor removal)

- Often a reasonable starting point if there's no prior info.

→ Computationally cheap & intuitive approach

How do constraints the problem → Interpret "Q" from clonic deformations or stationary velocity field! Then the update field is given or solution to $\dot{x}(t) = Q(x(t))|_{t=t_0} \Rightarrow \tilde{Q} = \exp_t(Q)$

→ We get the following diffeomorphic deformation:

Initialize $u^{(0)}$
 for $t = 1, 2, \dots$

Compute Q (formula)

Fluid regularization: $Q \leftarrow (Q \circ \kappa) \circ Q$

Update $u^{(t)} \leftarrow u^{(t-1)} \circ \exp_t(Q)$

Elastic regularization: $u^{(t)} \leftarrow (Q \circ \kappa) \circ u^{(t)}$

end

→ Only difference in the field update:

• Clonic: $u^{(t)} = u^{(t-1)} + \eta Q$

• Diffeomorphic: $u^{(t)} = u^{(t-1)} \circ \exp_t(Q)$

Note: What's the difference between " + " & " o " for hybrid transformations?

" + " is faster but does not really make sense

" o " requires additional warping of old transformation

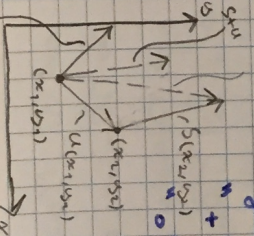
Addition: $(S + u)|_{(x_0, y_0)} = S(x_0, y_0) + u(x_0, y_0)$

Composition: $S \circ (I + u)|_{(x_0, y_0)} = S(x_0, y_0) \circ (I + u)|_{(x_0, y_0)}$

$= S|_{(x_0, y_0)} + u(x_0, y_0) \neq S|_{(x_0, y_0)}$

$\neq S|_{(x_0, y_0)} + u(x_0, y_0)$

$S(x_0, y_0)$



Only thing that changed...

Can be interpreted as:
 "Think back to the update Q in the fluid regularization, we have to do the group (space of diffeomorphisms) via the group (space of velocity fields), then map it to the space of diffeomorphisms via the exponential map."
 CLASSIC DIFFEOMORPHISMS SHOULD USE

Very much alike

Creates non-invertible trees

(3) Policy office Registration Model: (LEPTs)

→ Policy office Transformation are another class of Transformation (not discussed previously)

→ Goal: In different local regions describe Trade via affine map (but ONLY locally).

Obtain the global Transformation by non-linear joining together these local Trades.

→ Main approach for Quasi:

To each affine Transformation T_i assign a weight function $w_i(\tilde{x})$! This $w_i(\tilde{x})$ describes, where T_i is active. Then simply average Transformation:

$$T(\tilde{x}) = \sum_i w_i(\tilde{x}) T_i(\tilde{x}) \quad (\text{omitted } \sum_{i=1}^n)$$

→ Problem: This leads to non-invertibility.

→ Better approach: (LEPTs) Robust - Euclidean Policy office Transformations

Use diffeomorphism & the good old Moreau's "nonextended" via velocity fields!

→ 3 Steps to achieve this:

(i) Assign a velocity field to T_i :

The velocity field needs to be "consistent", i.e. the Trade $T_i(x_0)$ is given by the solution to the ODE $\dot{x}(t) = V_i(x(t))$ at time $t=1$ with initial condition x_0 !

LEPTs are one possible method to do Policy office

→ all 3 Steps to LEPTs:

(i) all assign a velocity field to T_i :

Write affine Trade in homogeneous coord.:

$$\tilde{T}_i(\tilde{x}) = \begin{pmatrix} A_i & t_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{x}_i \\ 1 \end{pmatrix} \quad A_i \dots \text{linear part}, t_i \dots \text{translation}$$

Define V_i via "principal matrix logarithm" of $\tilde{T}_i$:

$$\tilde{V}_i = \log(\tilde{T}_i) \tilde{x}_i = \log \begin{pmatrix} A_i & t_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{x}_i \\ 1 \end{pmatrix}$$

Instead this velocity field is consistent! Since

$$\tilde{x}_{(t)} = \log(\tilde{T}_i) \tilde{x} \Rightarrow \tilde{x}_{(t)} = \exp(t \cdot \log(\tilde{T}_i)) \cdot \tilde{x}_0$$

$$\text{At time } t=1: \tilde{x}_{(t=1)} = \tilde{T}_i \tilde{x}_0 \quad \checkmark$$

Note: You can compute $\log(\tilde{T}_i)$ efficiently!

"Square Rooting & Squaring" → Solve:

$$\log(\cdot) \& \exp(\cdot) \text{ can be computed accurately for matrices close to identity}$$

use e.g. Padé approx. or even Taylor 1st ord.!

$$\text{Use } \log(\tilde{T}_i) = 2^{-N} \cdot \log(\tilde{T}_i \cdot 2^N)$$

(ii) Turn the velocity fields:

$$\tilde{V}_{LEPT} = \sum_i w_i(\tilde{x}) \cdot \tilde{V}_i = \sum_i w_i(\tilde{x}) \log(\tilde{T}_i) \tilde{x}$$

Note: Since all V_i are linear independent, $\tilde{V}_{LEPT}$ will be linear independent

(iii) Compute global Transformation:

Since $\tilde{V}_{LEPT}$ is linear independent → Solve autonomous ODE: $\dot{\tilde{x}} = (\sum_i w_i \log(\tilde{T}_i)) \tilde{x} \quad | t=1$ & $\tilde{x}_{(0)} = \tilde{x}_0$

$$\Rightarrow \tilde{T}_{global}(\tilde{x}) = \exp\left(\sum_i w_i \log(\tilde{T}_i)\right) \cdot \tilde{x}_0 \quad \leftarrow O(N \times N)$$

$w_i \neq w_i(\tilde{x})$

EFFICIENT COMPUTATION VIA INVERSE SCALING & SQUARING

$$\text{Use } \exp(M) = \exp\left(\frac{M}{2^N}\right)^{2^N}$$

→ "INVERSE SCALING & SQUARING"

SVFs (only) have given by $\int_{\Gamma} \omega(x) \log(\tilde{\Gamma}) \tilde{x}$
 $\Rightarrow$ LERTs (probably) are a subgroup of homeomorphisms
 special through ordinary SVFs

LERTs are defined through holomorphic velocity fields (SVFs), or such they are diffeomorphisms (as discussed)

Actually existence of such a $v(t)$ is not clear...
 But can show approximation is possible

Goal ③ Polygenic Registration Models

$\hookrightarrow$ Can combine this transformation model with any Metric & Optimizer to obtain a descent & potentially fast registration algorithm!

Advantages of LERTs

- + They give smooth & invertible Transformations
- + They have few parameters
- + They can be applied in a Grid-based manner

Example: Registration of mandibles

Use multivariate B-splines to model the deformation bumps or decided through $\omega(x)$!
 They also tells us how the fine levels of different levels!

④ Log-Deform Algorithm:

$\hookrightarrow$ "Diffeomorphic Deform" or described in (2) only under the concept of holomorphic velocity fields for the cycle rule, i.e. $u^{(k)} \leftarrow u^{(k-1)} \circ \exp(Q)$
 $\hookrightarrow$ Problem: "Complete inverse" expensive to compute & also still not symmetric wrt $\Gamma \Leftrightarrow \tilde{\Gamma}$!

$\hookrightarrow$ Reduce by expressing complete displacement via SVF:

$$\exp(v^{(k)}) \leftarrow \exp(v^{(k-1)}) \circ \exp(Q)$$

Goal ④ Log-Deform:

$\hookrightarrow$ SVF represents complete transformation via SVF - called the "log-domain" since we

$$Q = \text{"SVF"} = \log(\text{"Displacement"})$$

$$\text{"Displacement"} = \exp(\text{"SVF"})$$

Then we also need a consistent regularization technique!
 $\hookrightarrow$ One possibility: "Poisson into the space of LERTs"

⑤ Diffeomorphism via solving PDEs:

(i) Elastic Registration:

Compute displacement field as solution to the Navier-Cauchy (aka Navier-Lamé) equation:

$$\mu \cdot \Delta \tilde{u} + (\lambda + \mu) \nabla(\nabla \cdot \tilde{u}) + \tilde{f} = \tilde{0}$$

(Derived from Cauchy's momentum equation)

$\tilde{u}$... Displacement, $\tilde{f}$... External force

μ, λ ... Lamé elasticity const. (related to E, ν)

Model extend given with weak minimality measures!

Solve using FEM & FEM

Disadvantage: Only small deformations

(ii) Fluid Registration: (Useful for large deformations)

Solve linearized Navier-Stokes for a velocity field!

(Replace $\tilde{u}$ by $\tilde{v}$ in above equation)

How to get $\tilde{u}$? Similar to domain on diffeomorphisms!

$$x(t) = v(x(t)) \Rightarrow x(t) = x_0 + \int_0^t v(x(s)) ds$$

$$\Rightarrow x(t=1) = x_0 + \int_0^1 v(x(s), s) ds$$

Usually good results!
 RCT: Consistency
 expensive (time & memory)