《数字图像处理》
第13讲 配准

冯建江

清华大学 自动化系

2017.12.28

期末考试

考试时间：

2018.1.4（周四）第2大节（9:50-12:15）

考试教室：三教3300

半开卷：可带一张有内容的A4纸

图像配准

定义：给定同一个物体/场景的两幅图像，将对应点的坐标对齐，得到两幅图像之间的空间变换

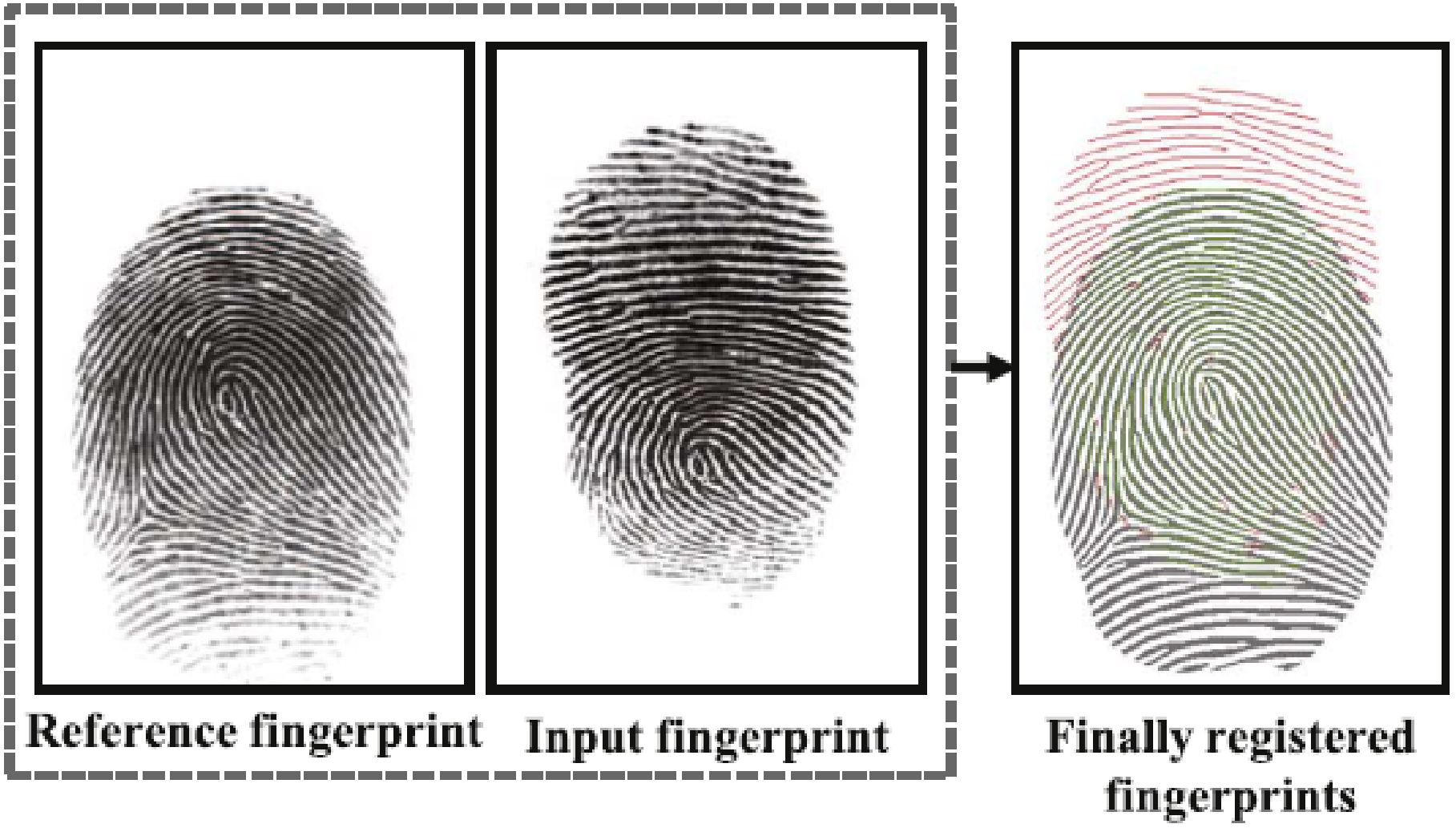
应用：全景图像



全景图

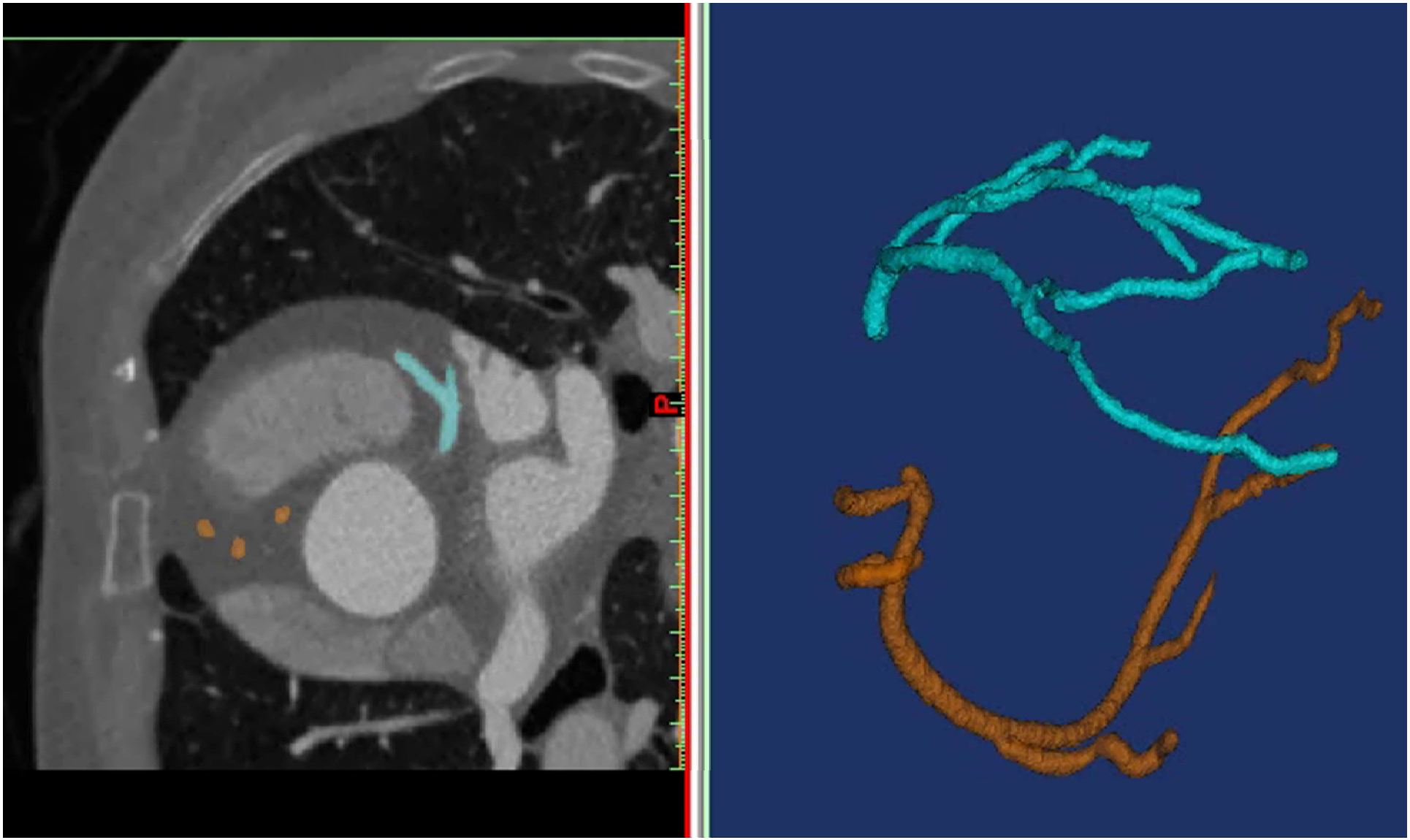
http://blog.csdn.net/yang_xian521/article/details/7589431

应用：指纹识别



Xuanbin Si, Jianjiang Feng, Bo Yuan, Jie Zhou:
Dense registration of fingerprints. Pattern Recognition 63: 87-101 (2017)

应用：冠心病诊断



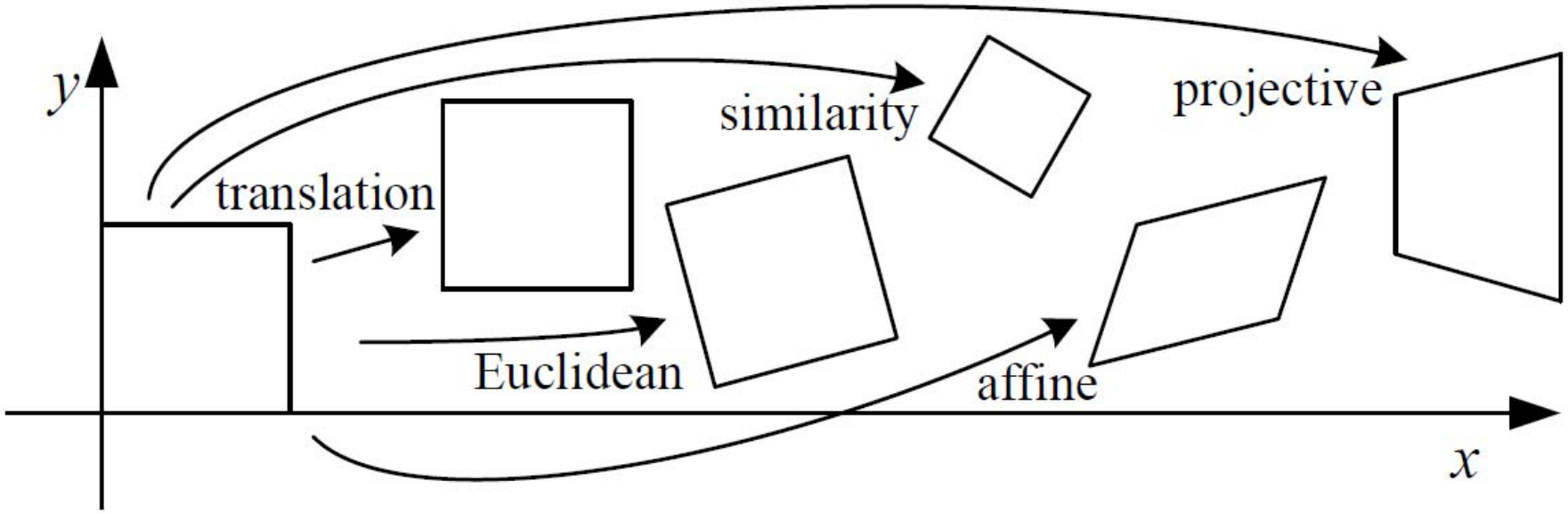
内 容

- 基本空间变换
- 刚性图像配准
- 图像渐变
- 指纹稠密配准

内 容

- 基本空间变换
- 刚性图像配准
- 图像渐变
- 指纹稠密配准

各种空间变换



Richard Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

平移

- 二维空间的点$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
- 增广向量$\bar{\boldsymbol{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$，便于矩阵级联相乘
- $\boldsymbol{x}' = [I \quad t]\,\bar{\boldsymbol{x}} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- $\bar{\boldsymbol{x}}' = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

刚体变换（rigid transformation）

- $\boldsymbol{x}' = [R \quad t]\,\overline{\boldsymbol{x}} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- $\overline{\boldsymbol{x}}' = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

相似变换（similarity transformation）

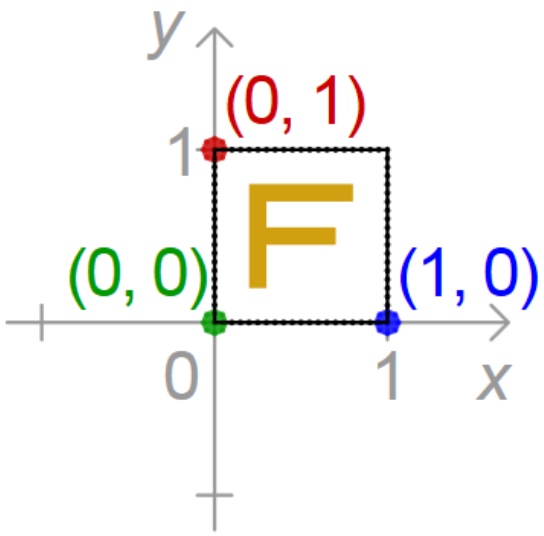
- $\boldsymbol{x}' = [sR \quad t]\,\overline{\boldsymbol{x}} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- $\overline{\boldsymbol{x}}' = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

仿射变换（affine transformation）

- $\boldsymbol{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- $\overline{\boldsymbol{x}}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

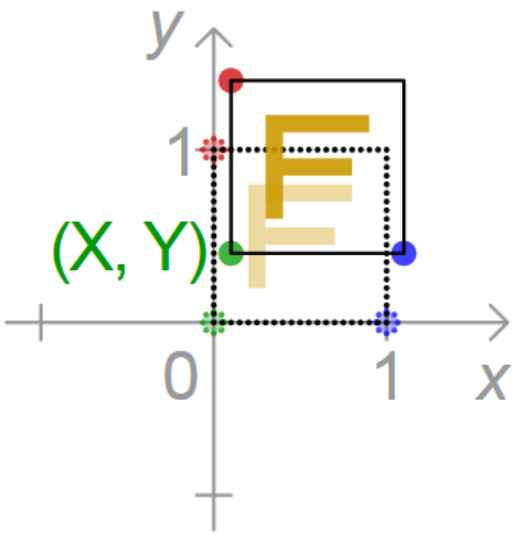
No change

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



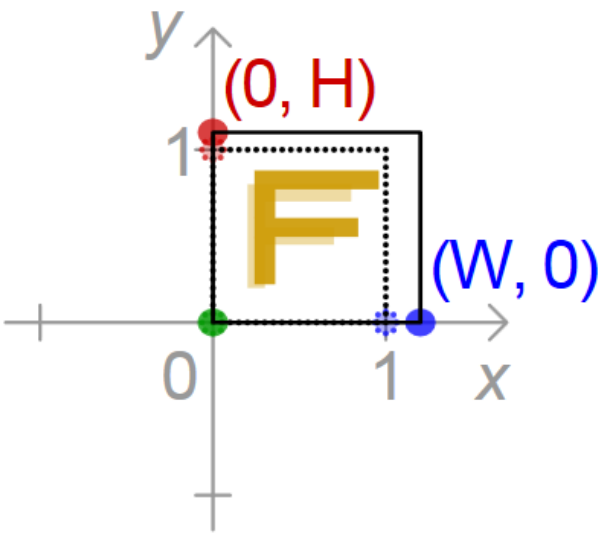
Translate

$$\begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix}$$



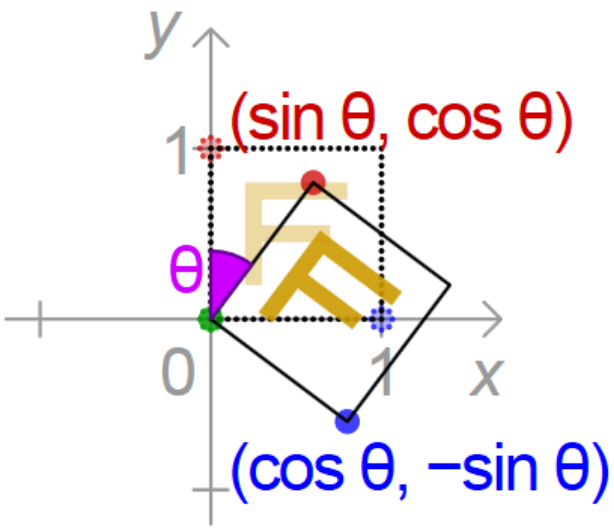
Scale about origin

$$\begin{bmatrix} W & 0 & 0 \\ 0 & H & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



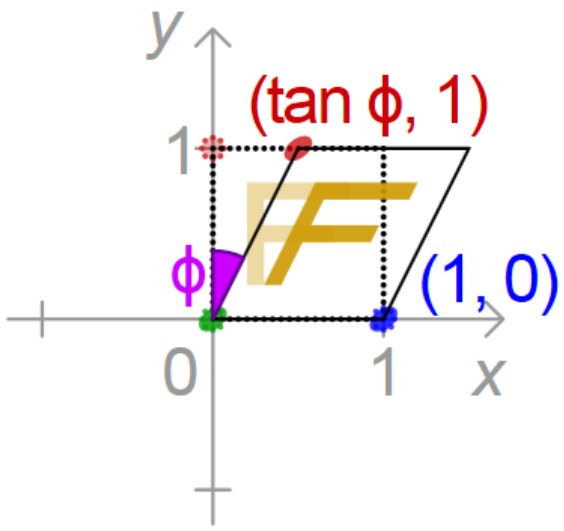
Rotate about origin

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



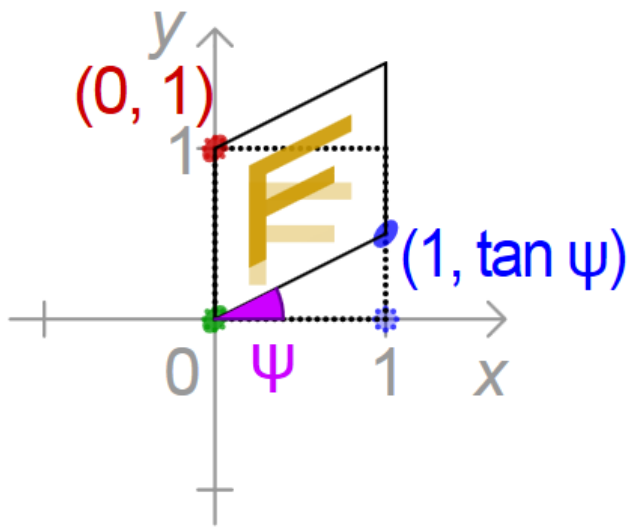
Shear in x direction

$$\begin{bmatrix} 1 & \tan\phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



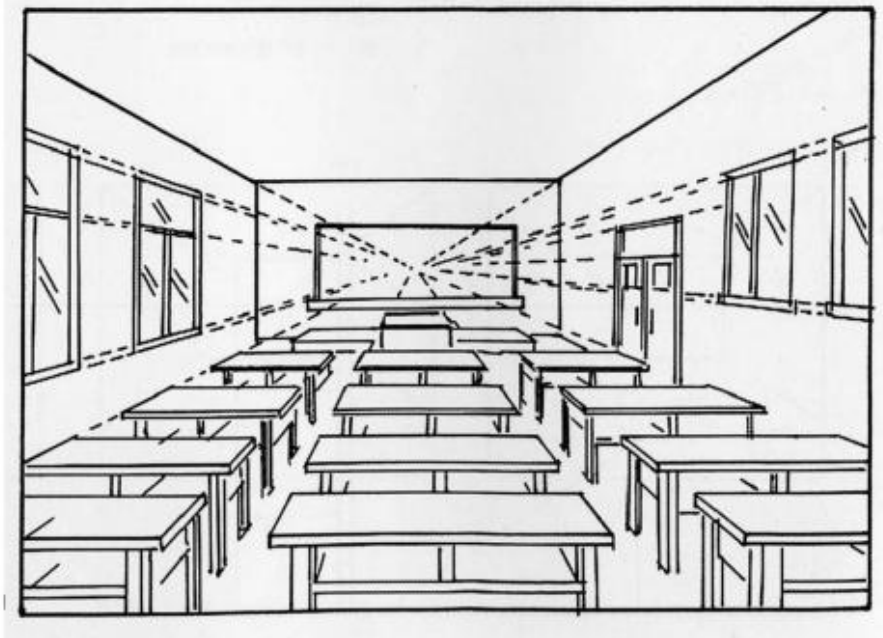
Shear in y direction

$$\begin{bmatrix} 1 & 0 & 0 \\ \tan\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

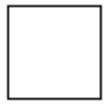


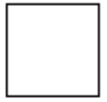
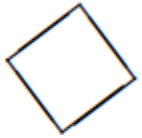
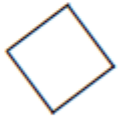
射影变换（Perspective transformation）

- $\overline{\boldsymbol{x}}' = \overline{\boldsymbol{H}}\overline{\boldsymbol{x}} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- 变换矩阵$\overline{\boldsymbol{H}}$是齐次的，有8个自由度，$\overline{\boldsymbol{H}}$和$a\overline{\boldsymbol{H}}$等价
- $x' = \frac{h_{00}x+h_{01}y+h_{02}}{h_{20}x+h_{21}y+h_{22}}, y' = \frac{h_{10}x+h_{11}y+h_{12}}{h_{20}x+h_{21}y+h_{22}}$



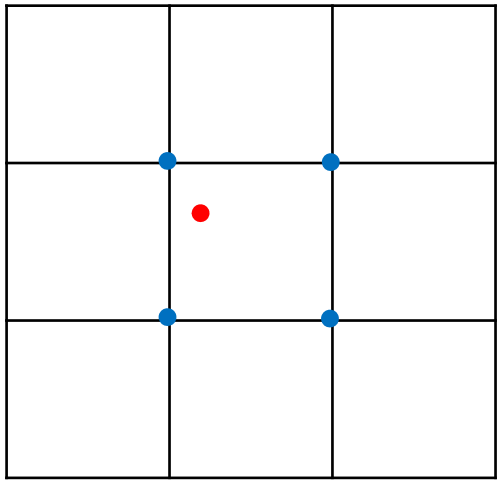
空间变换的层次性



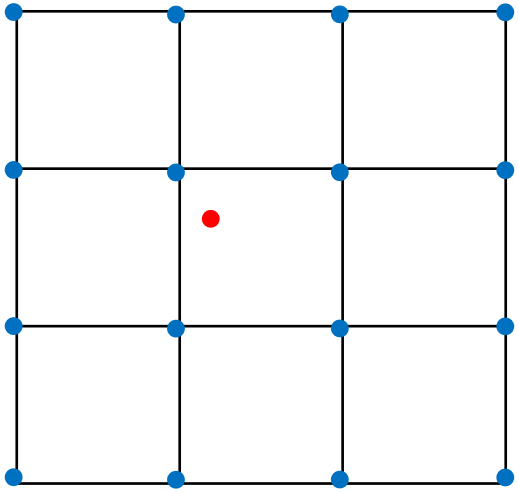
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

图像插值

- 在计算机上对图像进行空间变换，不得不进行插值
- 为什么？因为变化后的坐标很可能不是整数
- 为了计算方便，通过后向变换做插值（从输出图像的像素倒推输入图像的对应位置）
- 常用的插值方法：最近邻、双线性、双三次等插值方法（第二节课讲过）

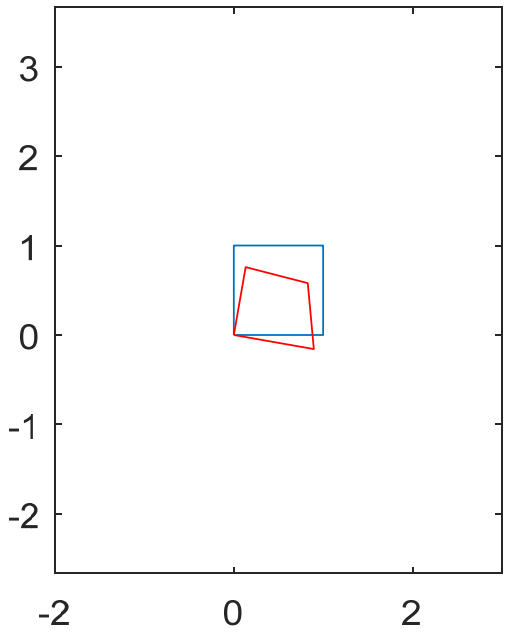


双线性



双三次

Demo



transform_points.m

transform_image.m

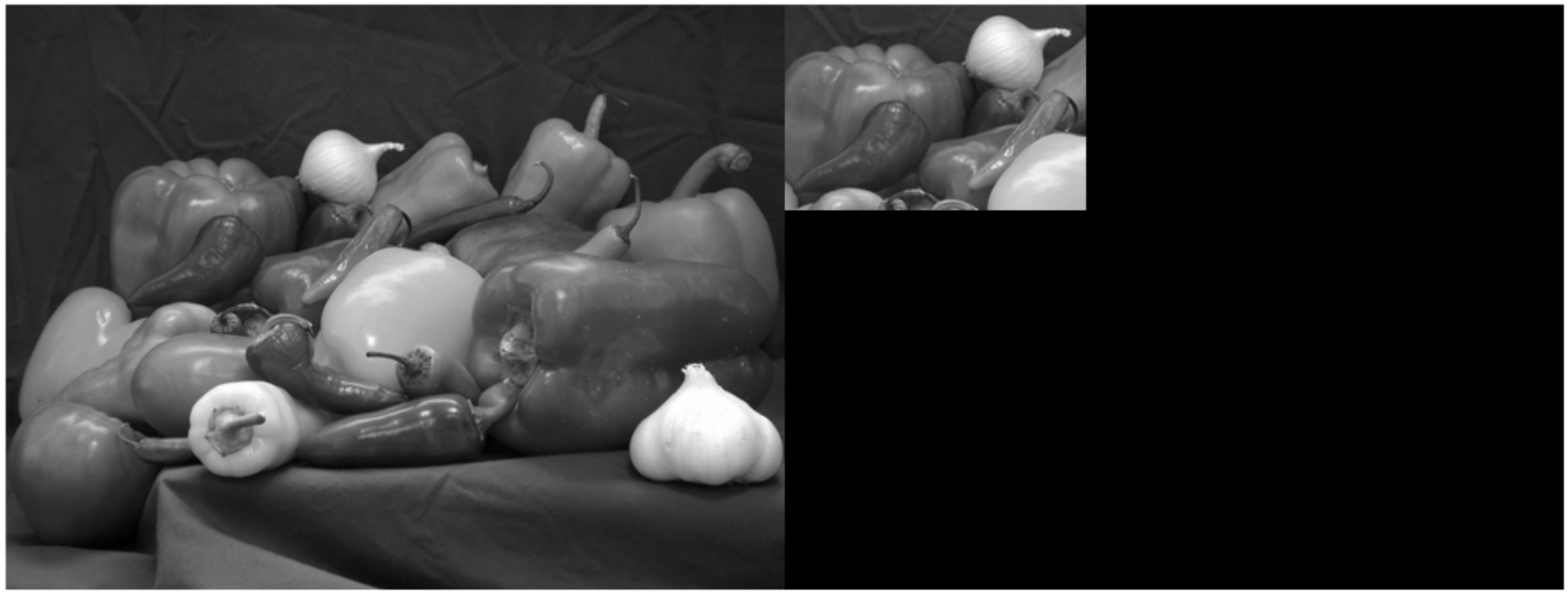
内 容

- 基本空间变换
- 刚性图像配准
- 图像渐变
- 指纹稠密配准

刚性图像配准

- 基于图像归一化互相关的方法
- 基于点匹配的方法

基于归一化互相关（normalized cross correlation）的平移估计



大图　大图均值　模板图像　模板均值

$$\gamma(u,v)=\frac{\sum_{x,y}\left[f(x,y)-\overline{f}_{u,v}\right]\left[t(x-u,y-v)-\overline{t}\right]}{\left\{\sum_{x,y}\left[f(x,y)-\overline{f}_{u,v}\right]^2\sum_{x,y}\left[t(x-u,y-v)-\overline{t}\right]^2\right\}^{0.5}}$$

归一化互相关

- 前一个例子是从大图中抠出来的小图，一模一样，过于简单
- 一个指纹的两幅不同图像

估计空间变换参数

- 已知一组对应点，估计空间变换（类型和参数）
- 类型与具体应用相关
- 类型确定后，写出线性方程组，估计参数
- 例如，仿射变换有6个参数，如果已知3对对应点（非共线），可以得出唯一解；如果多于3对，按照最小均方差方法求解。

$$\boldsymbol{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{aligned}
x'_1 &= a_{00}x_1 + a_{01}y_1 + a_{02} \\
y'_1 &= a_{10}x_1 + a_{11}y_1 + a_{12} \\
x'_2 &= a_{00}x_2 + a_{01}y_2 + a_{02} \\
y'_2 &= a_{10}x_2 + a_{11}y_2 + a_{12} \\
x'_3 &= a_{00}x_3 + a_{01}y_3 + a_{02} \\
y'_3 &= a_{10}x_3 + a_{11}y_3 + a_{12}
\end{aligned}$$

Demo

找综合作业4的灵感？http://photofunia.com

PhotoFunia Home Apps Blog Help Sign In

Search effects

- All effects 564
- Christmas 26
- Valentine's Day 20
- Halloween 18
- Filters 31
- Lab 94
- Cards 5
- Posters 71
- Galleries 33
- Photography 36
- Faces 92
- Billboards 53
- Celebrities 24

Celebrities

New Popular

Trump

Einstein

The Mortal Instrume...


Vampire


Marilyn Autograph


Obama

The Three Musketeers


Football Players

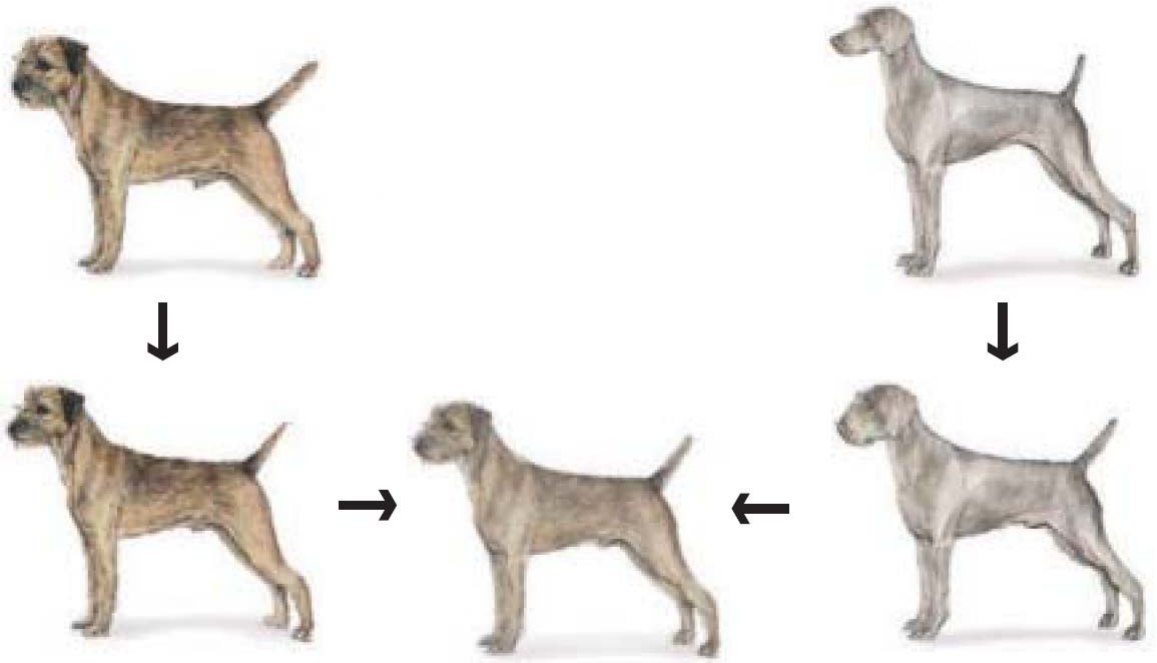
自动寻找对应点

- 给定两组点，点匹配算法确定对应点
- 经典的点匹配方法有：
  - 推广霍夫变换（Generalized Hough transform）
  - 对齐的方法（Alignment based ）
  - RANSAC方法
  - 迭代最近点（Iterative Closest Point）

内 容

- 基本空间变换
- 刚性图像配准
- 图像渐变
- 指纹稠密配准

Idea #3: Local warp, then cross-dissolve



- Morphing procedure:

- *for every t,*

1. Find the average shape (the “mean dog”☺)
   - local warping
2. Find the average color
   - Cross-dissolve the warped images

Image Morphing, ***select features***

1) Generating correspondence points (by hand)
    1) In matlab, use “cpselect” function, or write your own with “ginput” + “plot” (with “hold on”, “hold off”)

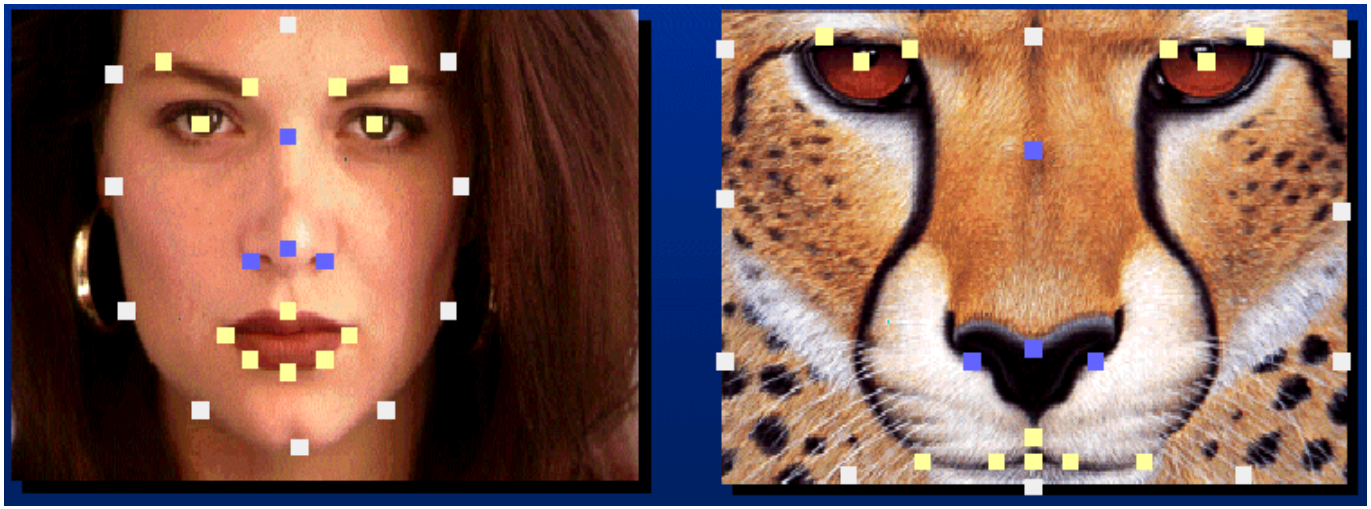


Image Morphing, ***triangulation***

2) Compute triangulation between the feature points

a) Use Delaunay triangulation (build in matlab)

b) Make sure both images have the same triangulation!

   a) Could use the mean shape to generate the triangulation, and copy the triangulation back.

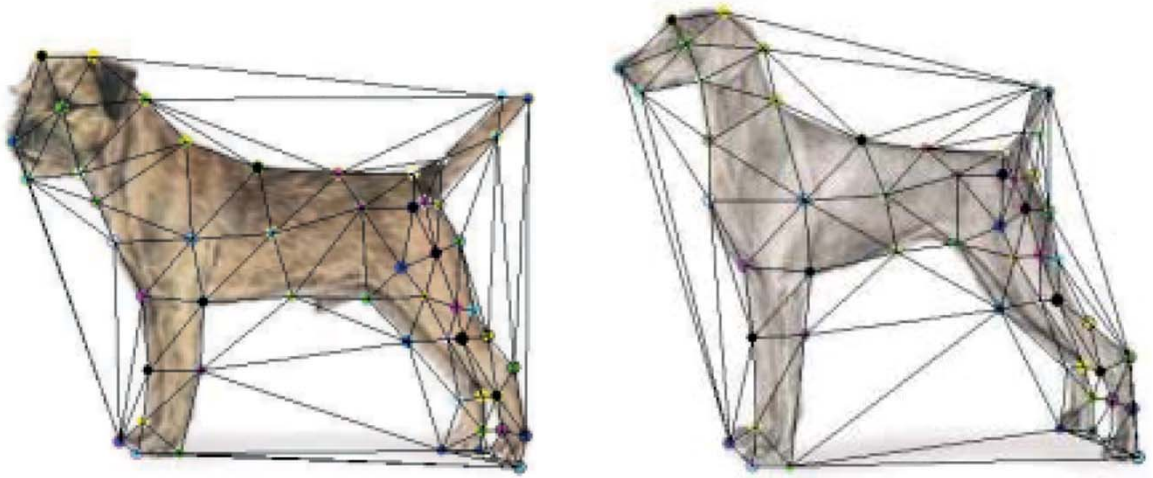
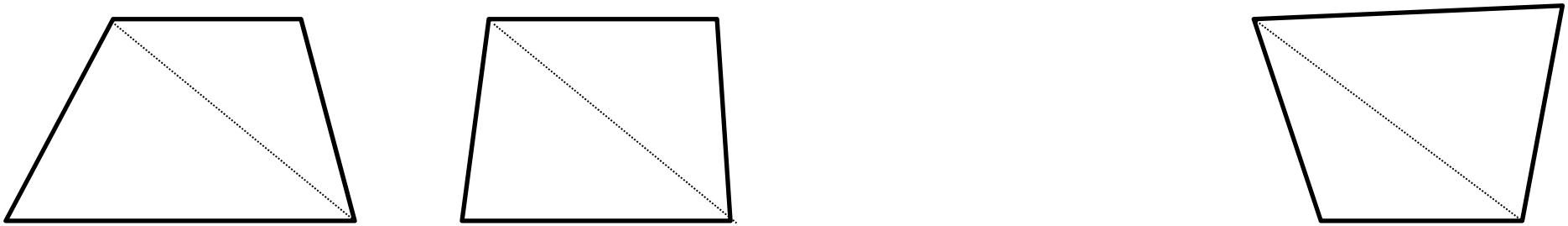


Image Morphing, generating ***intermediate shape*** (t-average)

2) Compute a weighted average shape

- Assume t = [0,1]
- Simple linear interpolation of each feature pair
- (1-t)*p1+t*p0 for corresponding features p0 and p1

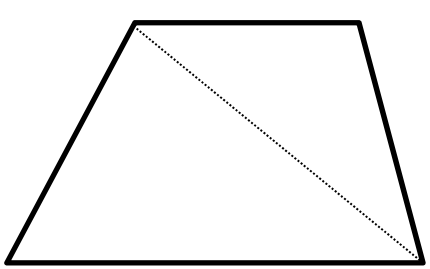


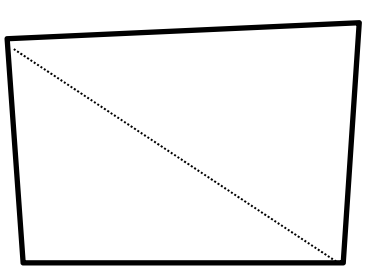
t=0.4

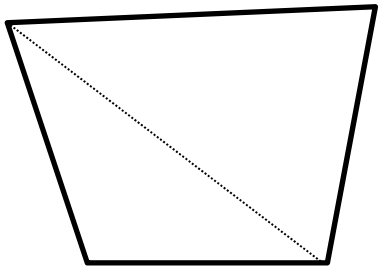
Image Morphing, generating ***intermediate shape***

2) Compute a weighted average shape

- Assume t = [0,1]
- Simple linear interpolation of each feature pair
- (1-t)*p1+t*p0 for corresponding features p0 and p1

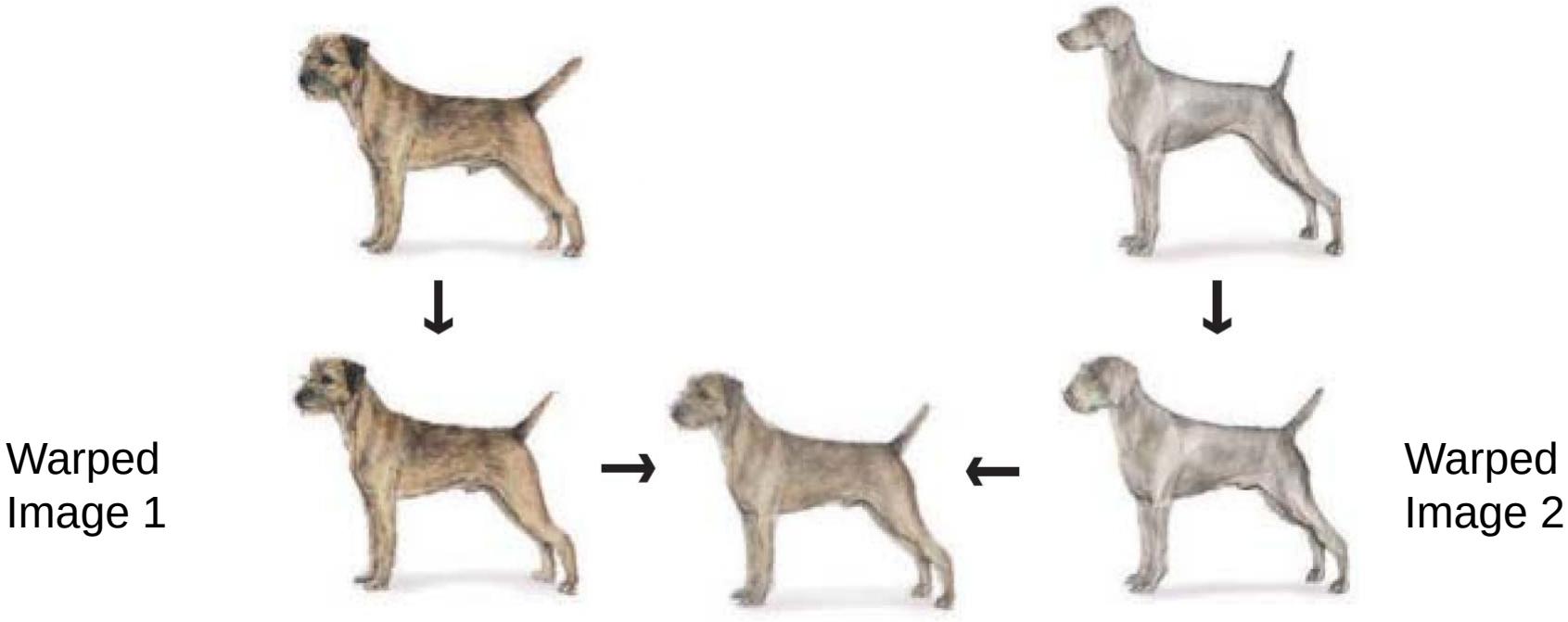






t=0.7

Image Morphing, ***generate warped image***



3) Generate warped image 1 and 2:
source: original image 1, and 2
target: ***t***-intermediate shape

We need to write the following matlab function:

morphed_im = morph(im_source, source_pts, target_pts, triangulation);

Image Morphing, ***generate warped image with Triangulation***

To find out where each pixel in new image comes from in old image

- Determine which triangle it is in
- Compute its barycentric co-ordinates
- Find equivalent point in equivalent triangle in original image, and copy over its intensity value

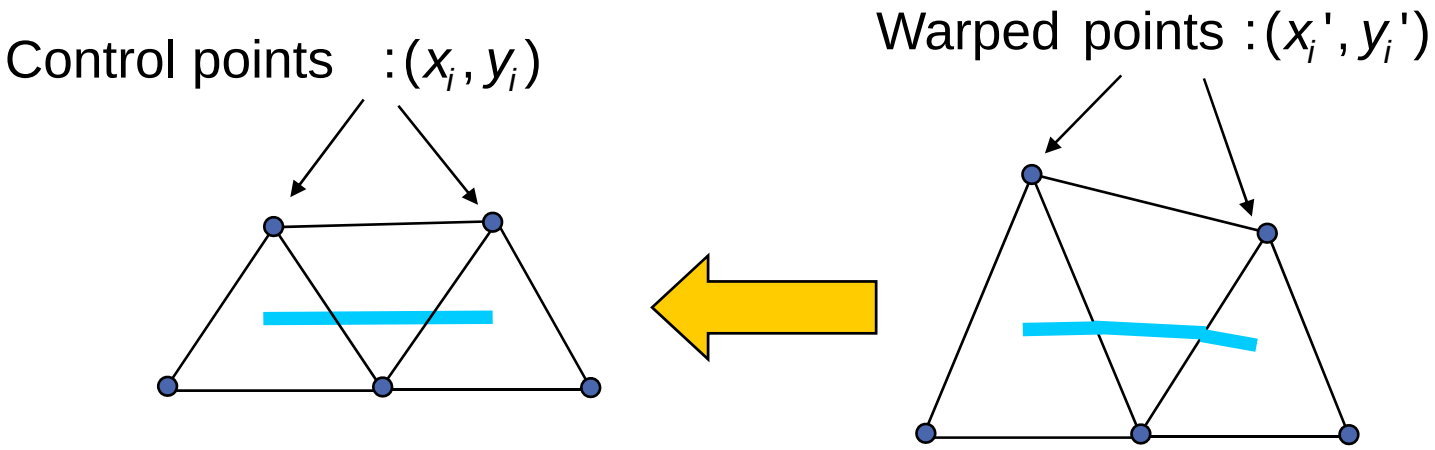
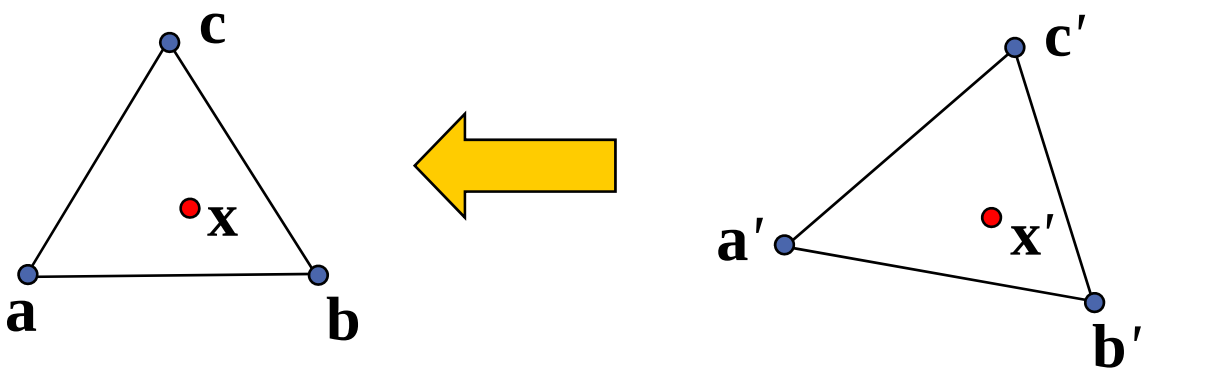


Image Morphing, ***generate warped image with Triangulation***

To find out where each pixel in new image comes from in old image

- Determine which triangle it is in
- Compute its barycentric co-ordinates
- Find equivalent point in equivalent triangle in original image, and copy over its intensity value

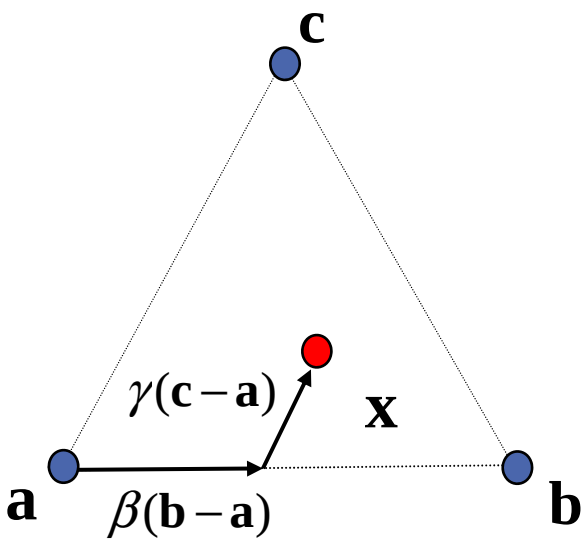


$$\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

$$\alpha + \beta + \gamma = 1$$

$$\mathbf{x}' = \alpha\mathbf{a}' + \beta\mathbf{b}' + \gamma\mathbf{c}'$$

Barycentric Co-ordinates



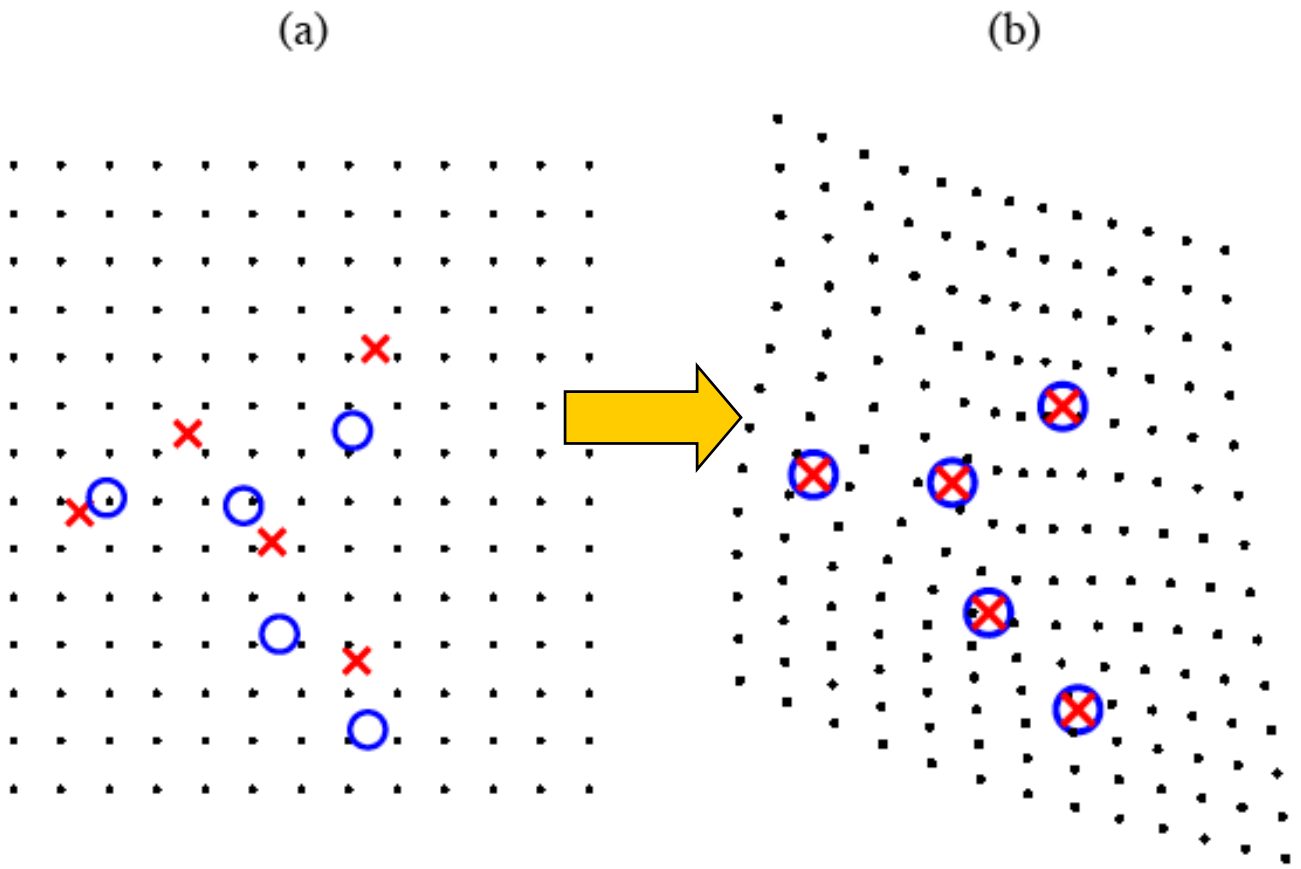
$$\begin{aligned}\mathbf{x} &= \mathbf{a} + \beta(\mathbf{b}-\mathbf{a}) + \gamma(\mathbf{c}-\mathbf{a}) \\ &= (1-\beta-\gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \\ &= \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}\end{aligned}$$

$$\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$
$$\alpha + \beta + \gamma = 1$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

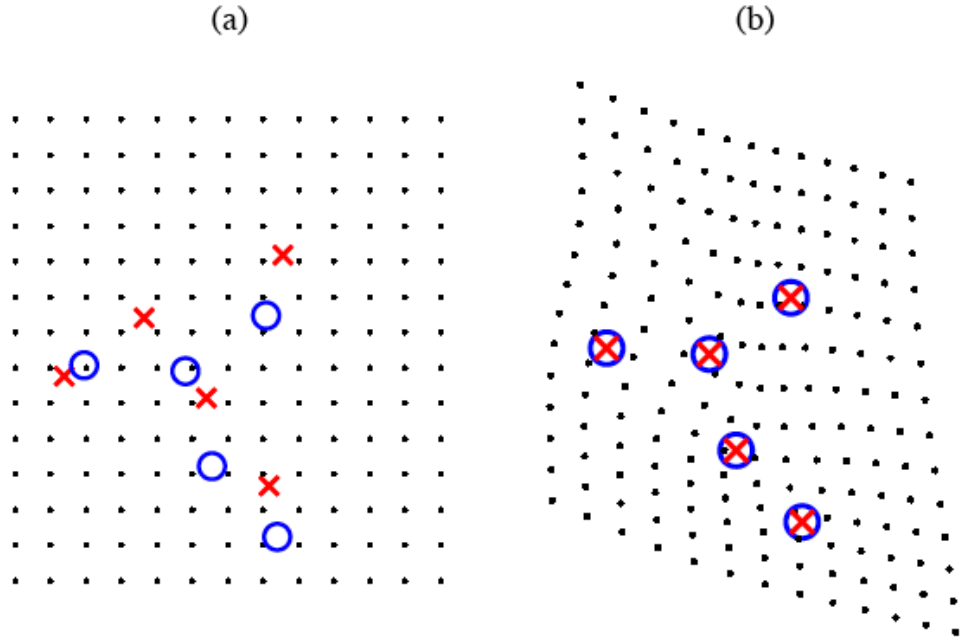
Three linear equations in 3 unknowns

Thin-plate spline TPS



Sparse and irregular positioned feature points, and smooth interpolation

Simple example of coordinate transformation using TPS

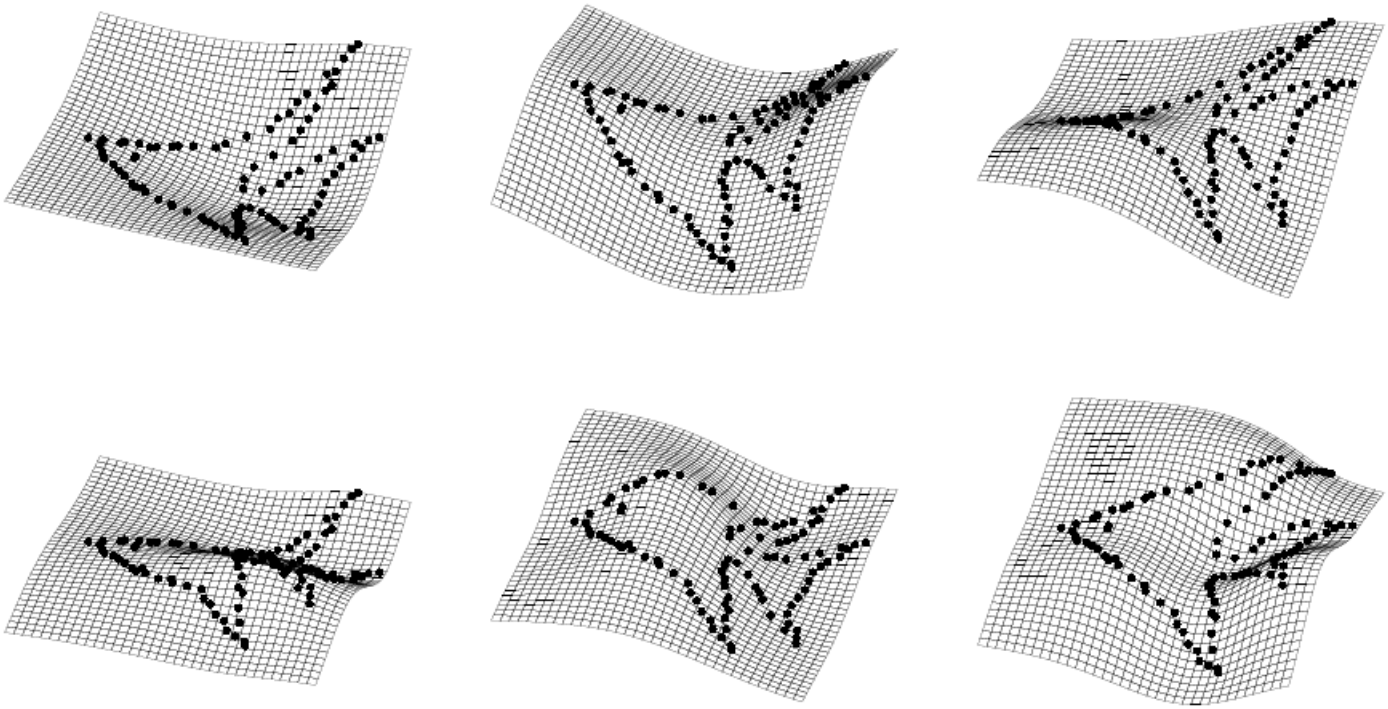


Let’s consider two sets of points for which we assume the correspondences to be known (a). The TPS warping allows a perfect alignment of the points and the bending of the grid shows the deformation needed to bring the two sets on top of each other (b). Note that in the case of TPS applied to coordinate transformation we actually use two splines, one for the displacement in the x direction and one for the displacement in the y direction. The displacement in each direction is considered as a height map
for the points and a spline is fit as in the case of scattered points in 3D space. And finally the two resulting transformations are combined into a single mapping.

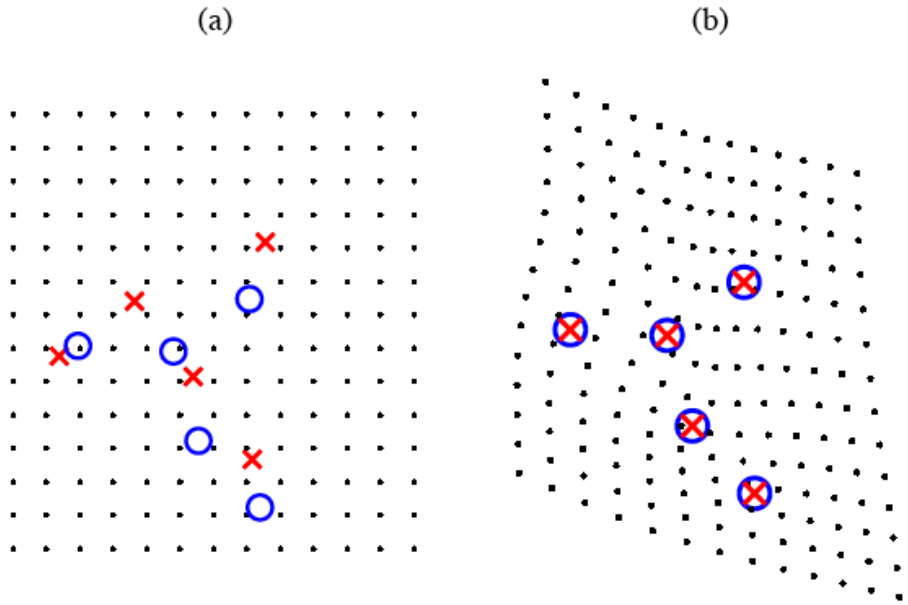
TPS model:

First the equation:

$$f(x,y) \quad = \quad a_1 + a_x x + a_y y + \sum_{i=1}^{p} w_i U\left(\|(x_i,y_i)-(x,y)\|\right)$$



how do we estimate the TPS parameters?



Say the blue circles are the source image features, and red crosses are the target image features, how do we “back-warp” all the image features in the target image back to the source image?
We could compute a TPS model that maps “red” to “blue”, and apply it to the rest of the target image pixels.

$$f(x,y) \quad = \quad a_1 + a_x x + a_y y + \sum_{i=1}^{p} w_i U\left(\|(x_i,y_i) - (x,y)\|\right)$$

how do we estimate the TPS parameters?

We would need two functions:

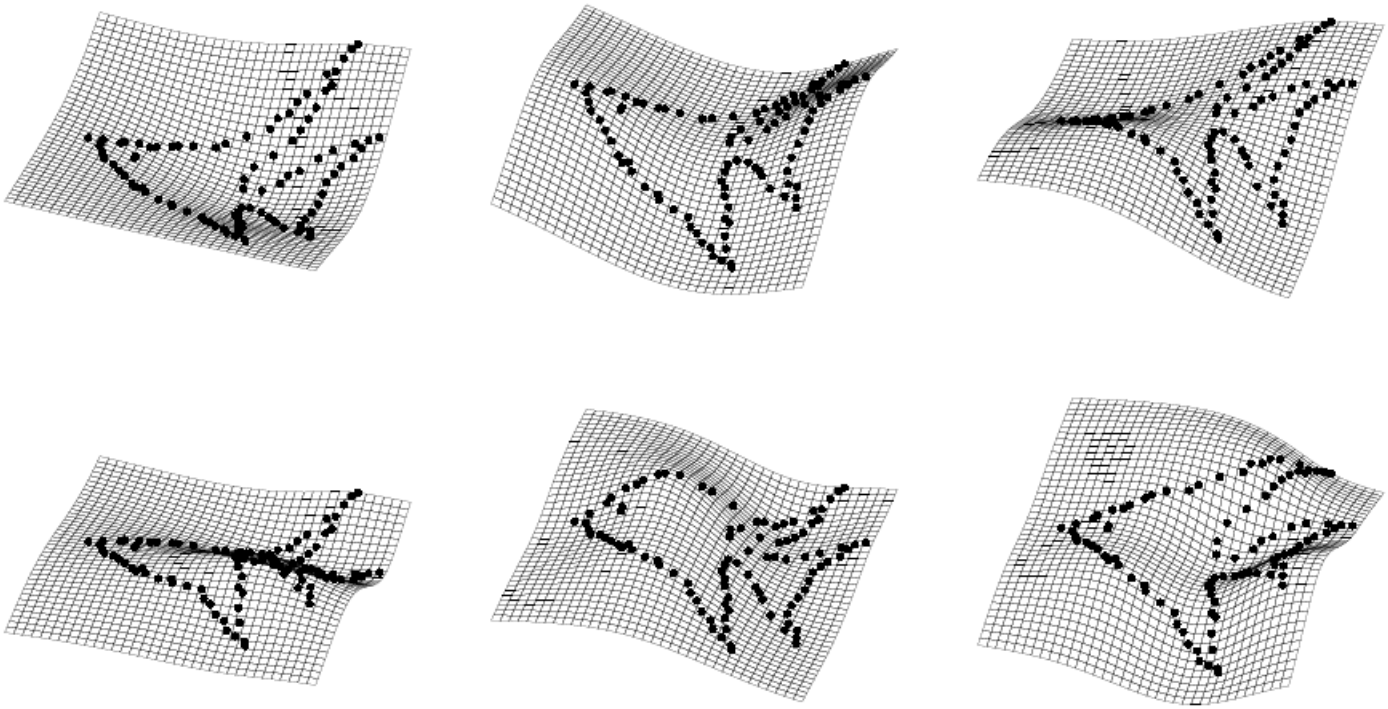
1) **tps_model = est_tps(source_pts, target_pts);**
2) **morphed_im = morph(im_source, tps_model);**

tps_model = (a1, ax, ay, w1, …, wp)

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{p} w_i U\left(\|(x_i,y_i)-(x,y)\|\right)$$

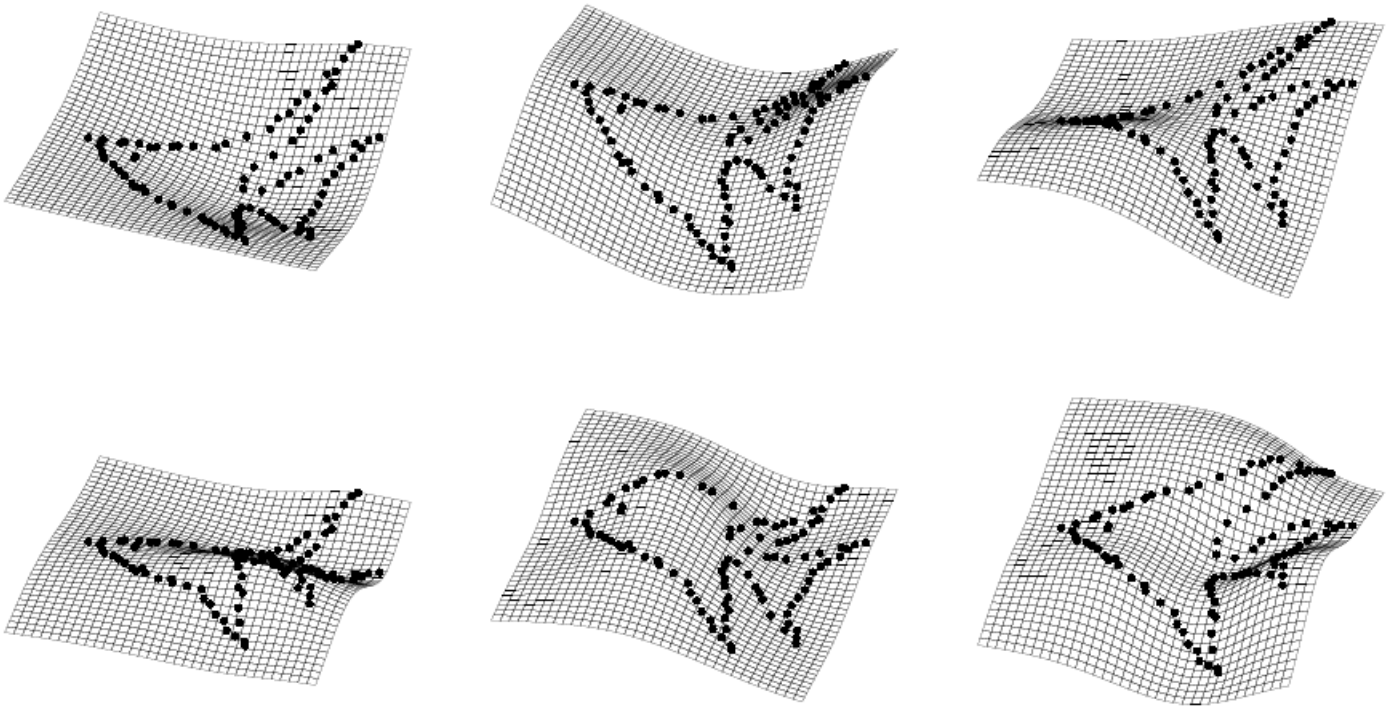
TPS model, special case 1, translation only

$$f(x,y) = \boxed{a_1} + a_x x + a_y y + \sum_{i=1}^{P} w_i U\left(\|(x_i,y_i)-(x,y)\|\right)$$



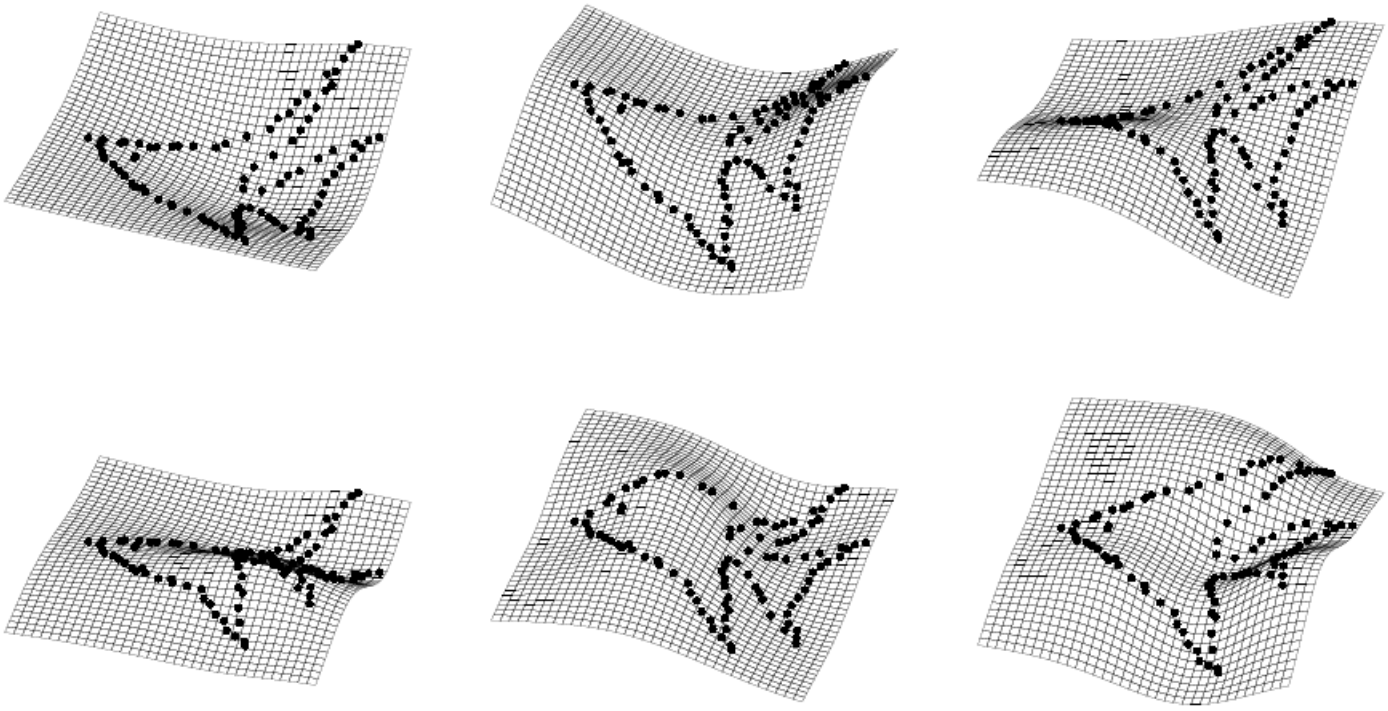
TPS model, case 2: Affine parameters only

$$f(x,y) \quad = \boxed{a_1 + a_x x + a_y y} + \sum_{i=1}^{P} w_i U\left(\|(x_i, y_i) - (x, y)\|\right)$$



TPS model: Full general case:

$$f(x,y) \quad = \quad a_1 + a_x x + a_y y + \sum_{i=1}^{P} w_i U\left(\|(x_i,y_i)-(x,y)\|\right)$$



TPS model:

$$f(x,y) \quad = \quad a_1 + a_x x + a_y y + \sum_{i=1}^{p} w_i U\left(\|(x_i,y_i)-(x,y)\|\right)$$

$$K_{ij} = U(\|(x_i,y_i)-(x_j,y_j)\|),$$

ith row of P is $(1,x_i,y_i)$,

$$\begin{bmatrix} K & P \\ P^T & O \end{bmatrix} \begin{bmatrix} w \\ a \end{bmatrix} \quad = \quad \begin{bmatrix} v \\ o \end{bmatrix}$$

变脸DEMO

https://github.com/GabriellaQiong/CIS581-Project2-Image-Morphing