

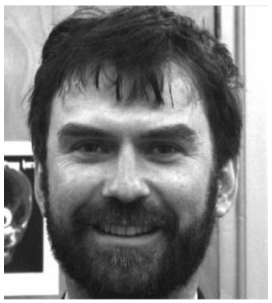


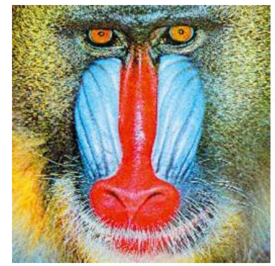
数据可视化
图像数据基本变换5
几何变换

庄吓海

大数据学院

图像的几何变换

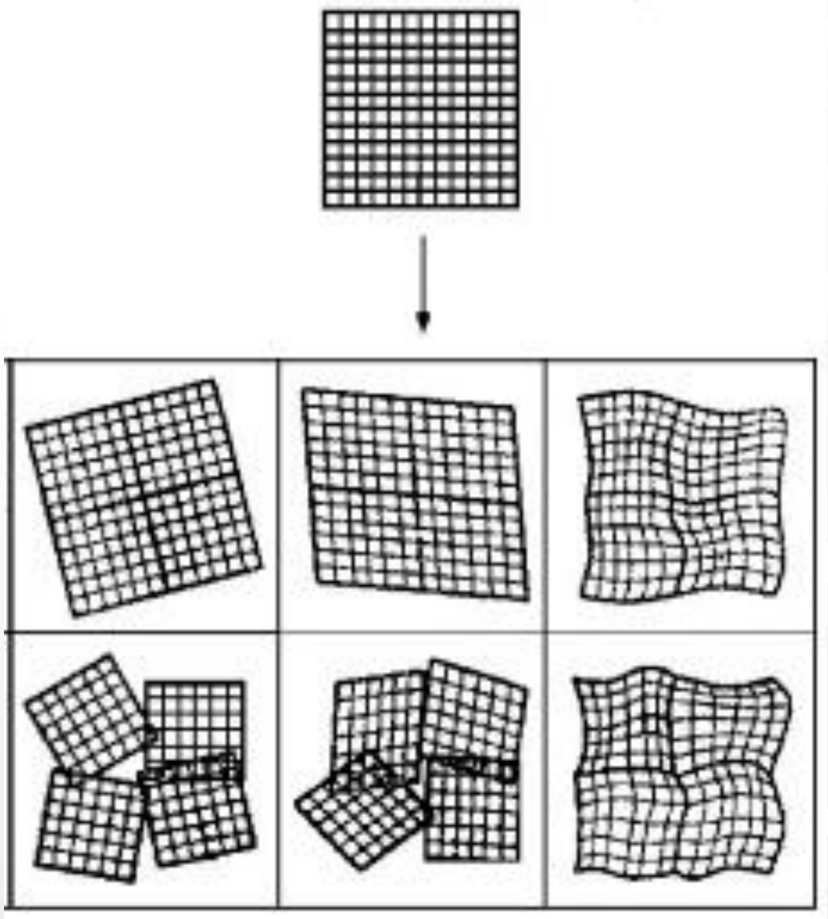




[Images courtesy of David J. Hawkes]

图像的几何变换

- Linear transformation
  - Rigid transformation
  - Similarity transformation
  - Affine transformation
- Nonlinear transformation
  - FFD
  - Locally affine
- Global VS Local

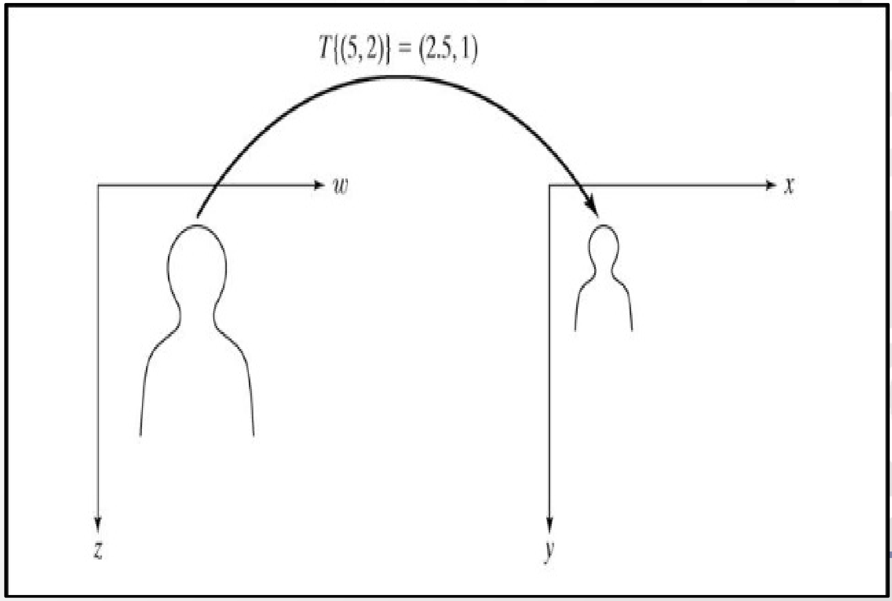


图像的几何变换

大数据学院 School of Data Science

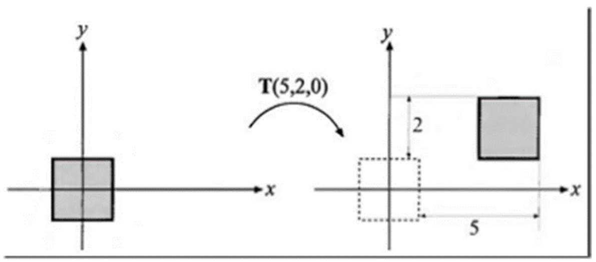
- $X' = T(X)$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \left[\begin{bmatrix} & T & \end{bmatrix} \right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

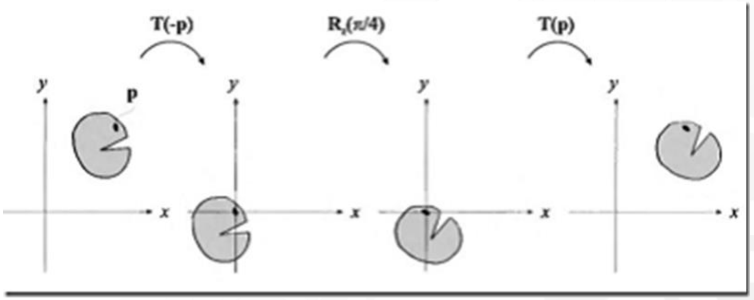


大数据学院 School of Data Science

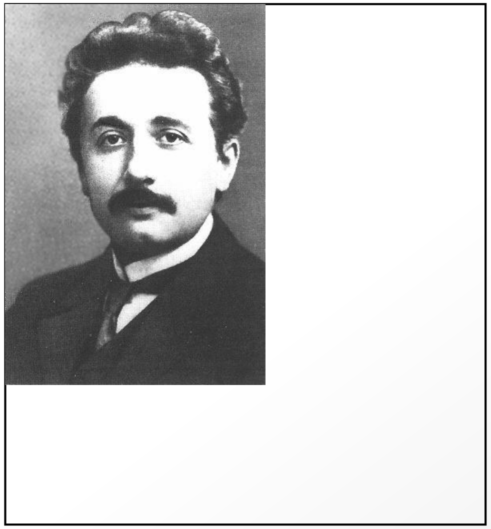
Rigid transformation preserves the angels and distances within the model

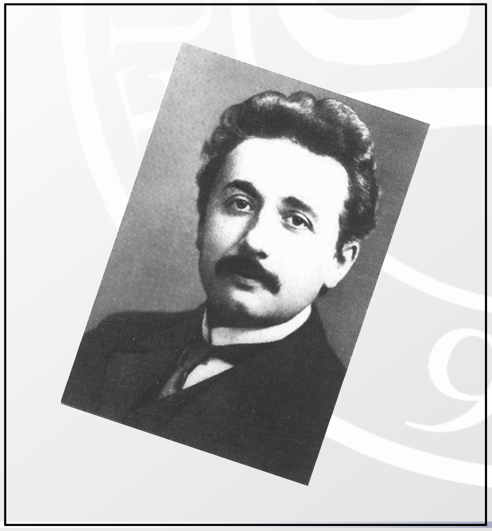


Translation



Rotation





线性变换-刚性变换

大数据学院 School of Data Science

- A 3D rigid body transform is defined by:

  --3 translations - in X, Y & Z directions

  --3 rotations - about X, Y & Z axes

- The order of the operations matters

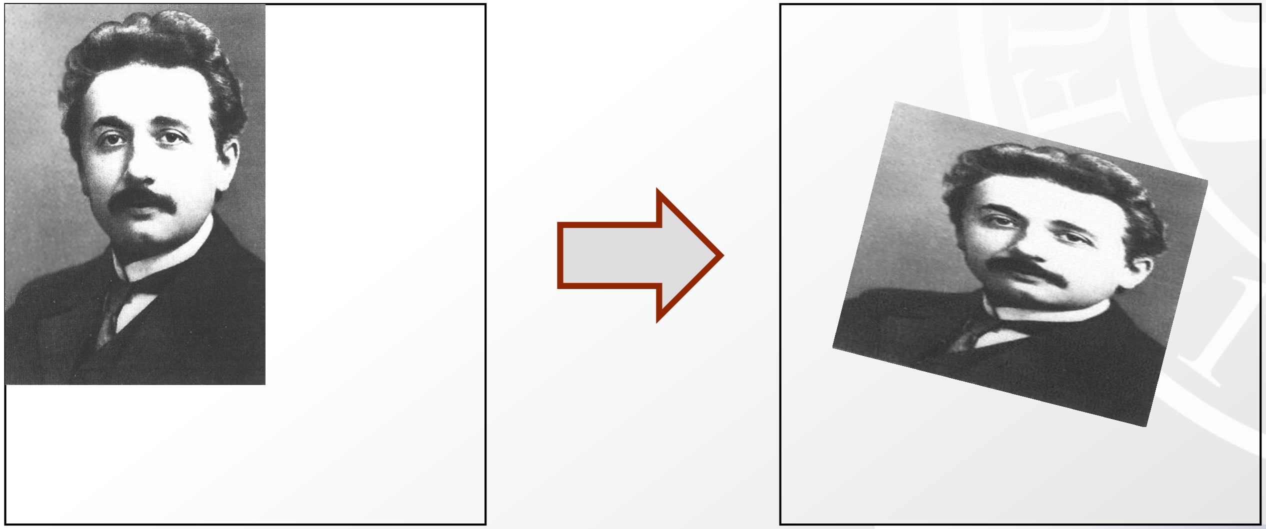
$$\begin{pmatrix} 1 & 0 & 0 & X_{trans} \\ 0 & 1 & 0 & Y_{trans} \\ 0 & 0 & 1 & Z_{trans} \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi & 0 \\ 0 & -\sin\Phi & \cos\Phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \cos\Omega & \sin\Omega & 0 & 0 \\ -\sin\Omega & \cos\Omega & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translations	Pitch about x axis	Roll about y axis	Yaw about z axis

线性变换–相似变换

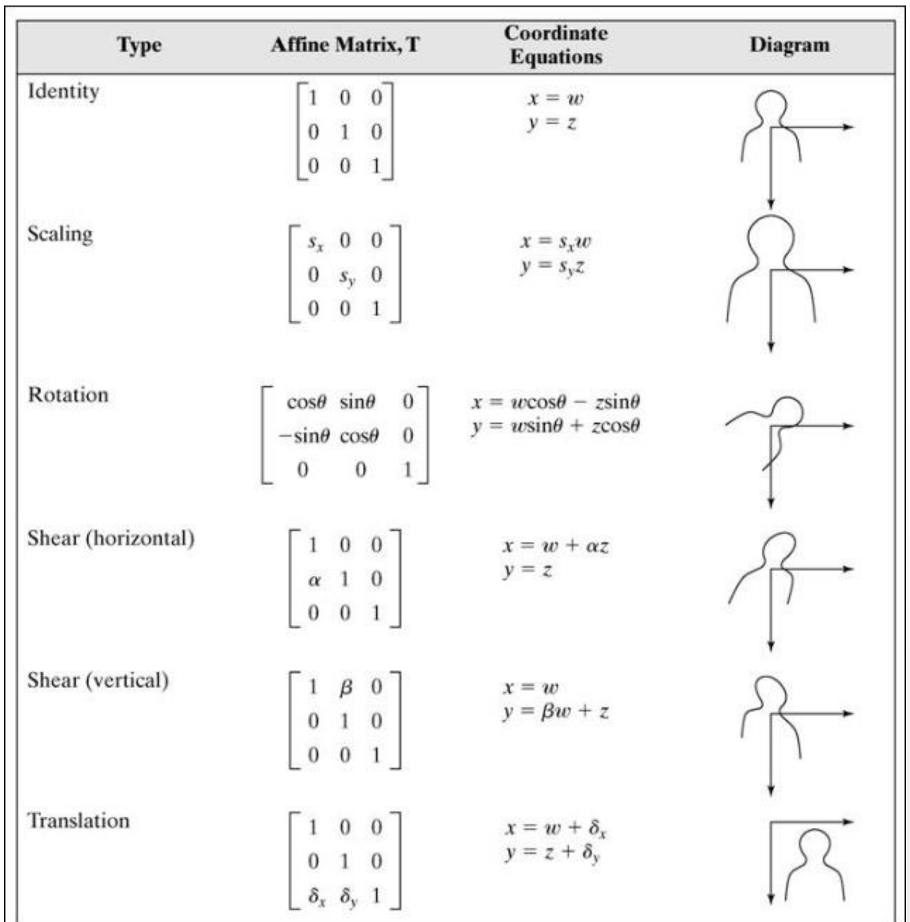
Similarity transformation adds scaling *{s}*

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_{rigid} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Affine transformation applies a function between affine spaces which preserves points, straight lines and planes.

Parallel lines remain parallel



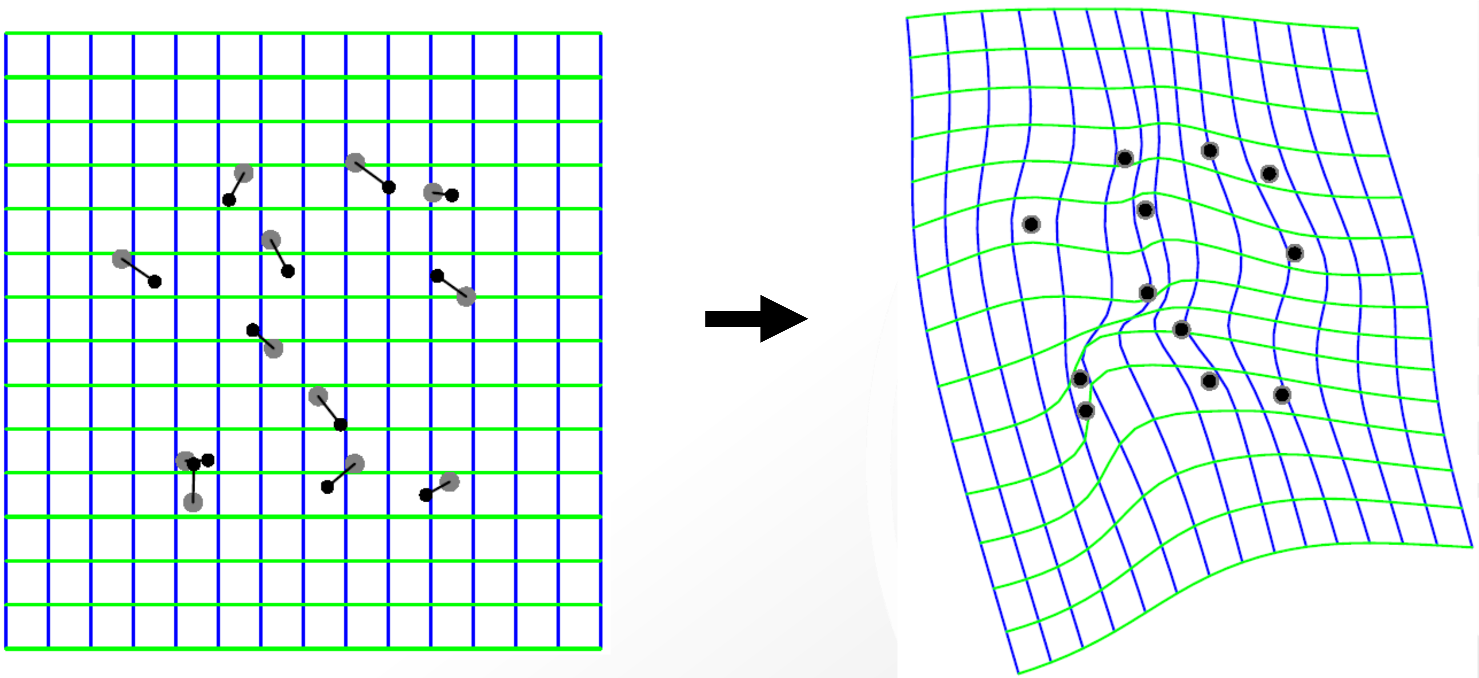
Type	Affine Matrix, T	Coordinate Equations	Diagram
Identity	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$x = w$ $y = z$	
Scaling	$\begin{bmatrix}s_x&0&0\\0&s_y&0\\0&0&1\end{bmatrix}$	$x = s_x w$ $y = s_y z$	
Rotation	$\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}$	$x = w\cos\theta - z\sin\theta$ $y = w\sin\theta + z\cos\theta$	
Shear (horizontal)	$\begin{bmatrix}1&0&0\\\alpha&1&0\\0&0&1\end{bmatrix}$	$x = w + \alpha z$ $y = z$	
Shear (vertical)	$\begin{bmatrix}1&\beta&0\\0&1&0\\0&0&1\end{bmatrix}$	$x = w$ $y = \beta w + z$	
Translation	$\begin{bmatrix}1&0&0\\0&1&0\\\delta_x&\delta_y&1\end{bmatrix}$	$x = w + \delta_x$ $y = z + \delta_y$	

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

非线性变换–FFD

大数据学院 School of Data Science

Thin-Plate Spline



F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell., 11(6):567–585, June 1989.

Thin-Plate Spline

给定N个自变量$\mathbf{x}_k$和对应的函数值$\mathbf{y}_k$，求插值函数

$$\Phi(\mathbf{x}) = \begin{bmatrix} \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \end{bmatrix},$$

使得

$$\mathbf{y}_k = \Phi(\mathbf{x}_k). \tag{2}$$

我们可以认为是求两个插值函数$\Phi_1(\mathbf{x})$和$\Phi_2(\mathbf{x})$。

TPS插值函数形式如下：

$$\Phi_1(\mathbf{x}) = \mathbf{c} + \mathbf{a}^{\mathbf{T}}\mathbf{x} + \mathbf{w}^{\mathbf{T}}\mathbf{s}(\mathbf{x}) \tag{3}$$

其中c是标量，向量$\mathbf{a} \in \mathbb{R}^{2\times 1}$，向量$\mathbf{w} \in \mathbb{R}^{\mathbf{N}\times 1}$，函数向量

$$\mathbf{s}(\mathbf{x}) = (\sigma(\mathbf{x}-\mathbf{x}_1), \sigma(\mathbf{x}-\mathbf{x}_1), \cdots, \sigma(\mathbf{x}-\mathbf{x}_{\mathbf{N}}))^{\mathbf{T}}$$

$$\sigma(\mathbf{x}) = ||\mathbf{x}||_2^2 \log ||\mathbf{x}||_2.$$

$\Phi_2(\mathbf{x})$和$\Phi_1(\mathbf{x})$有一样的形式。看到这里可能会产生疑问？插值函数的形式千千万，怎么就选择公式(3)这种形式呢？我们可以把一个插值函数想象成弯曲一个薄钢板，使得它穿过给定点，这样会需要一个弯曲能量：

$$J(\Phi) = \sum_{j=1}^{2} \iint_{\mathbb{R}^2} \left(\frac{\partial^2 \Phi_j}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 \Phi_j}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 \Phi_j}{\partial y^2}\right)^2 dxdy$$

那么可以证明公式(3)是使得弯曲能量最小的插值函数。

非线性变换-FFD

大数据学院
School of Data Science

- Thin-Plate Spline
- Free-form

(From [S.Y. Lee, K.Y Chwa, and S.Y. Shin, SIGGRAPH, 1995])

非线性变换-FFD

自由形变变换（Free-form deformation, FFD）

Compute lattice coordinates
(u, v, w)

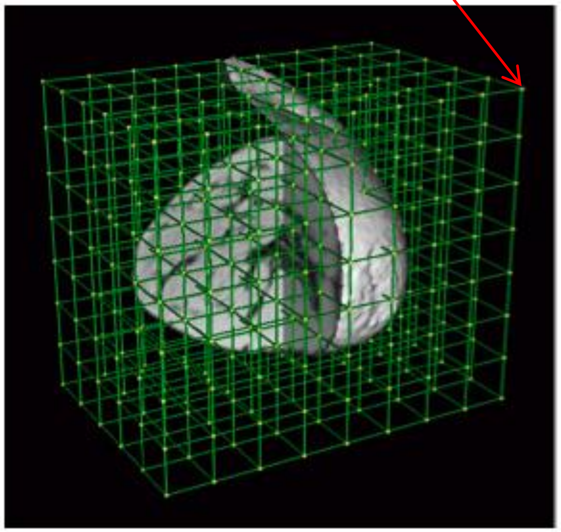
Alter the control points
$\mathbf{p}_{ijk}$

Compute the deformed points
$\mathbf{Q}(u, v, w)$

$$\mathbf{Q}(u, v, w) = \sum_{ijk} \mathbf{p}_{ijk} B(u) B(v) B(w)$$

$$X' = X + Q$$

控制点



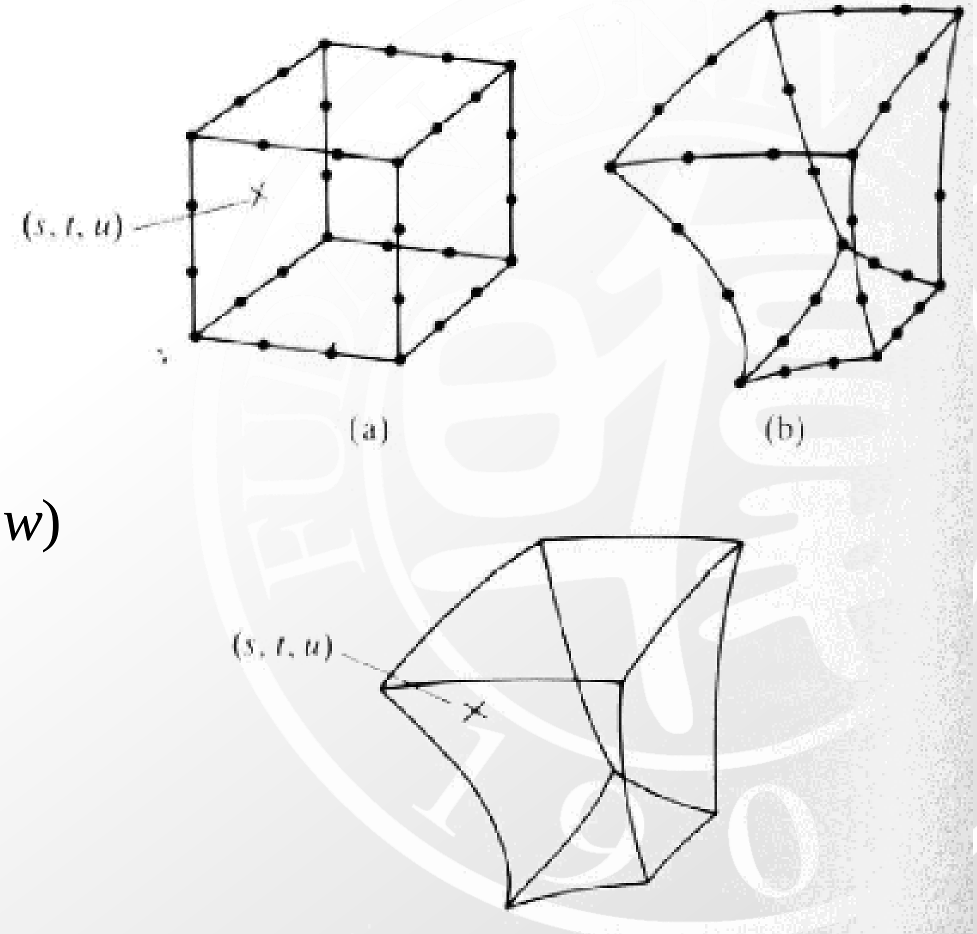
Free form deformation (FFD)

自由形变变换（Free-form deformation, FFD）

Compute lattice coordinates
(u, v, w)

Alter the control points
$\mathbf{p}_{ijk}$

Compute the deformed points
$\mathbf{Q}(u, v, w)$



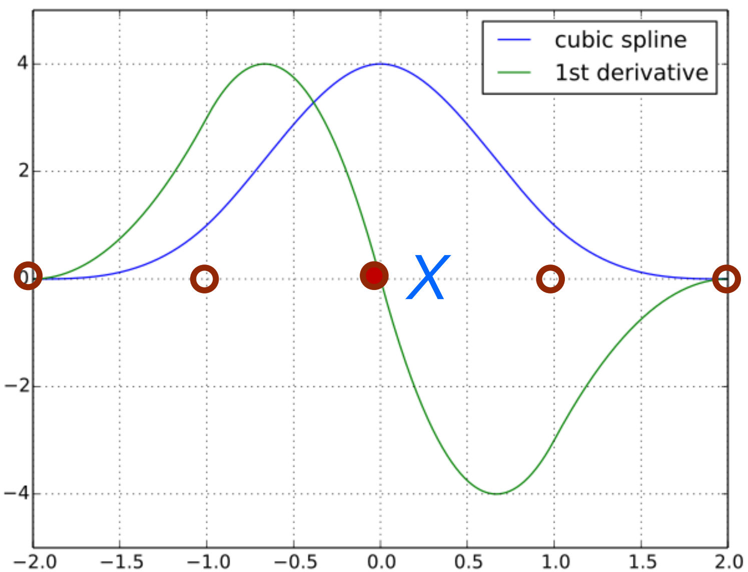
(a) (b) (c)

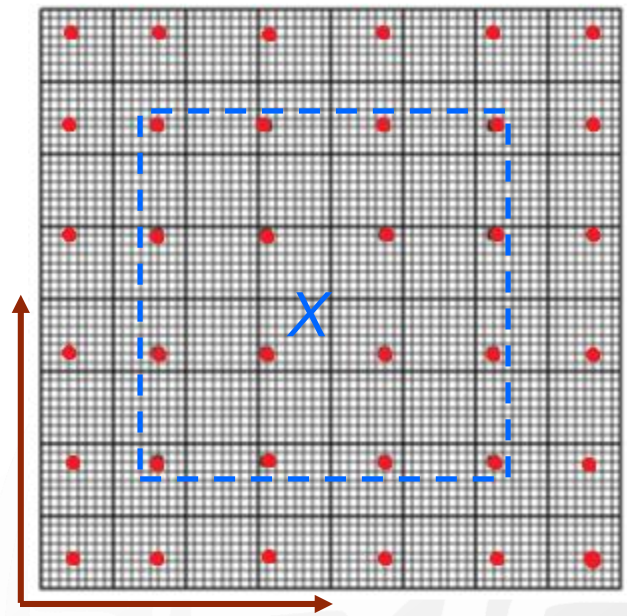
$$\mathbf{Q}(u, v, w) = \sum_{ijk} \mathbf{p}_{ijk} B(u) B(v) B(w)$$

$$X' = X + Q$$

非线性变换–FFD

- 使用三次B-样条核函数
- $X' = X + T_{\text{local}}$





$$\mathbf{T}_{\text{local}}(x, y, z) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(u)B_m(v)B_n(w)\phi_{i+l,\, j+m,\, k+n} \quad (3)$$

where $i = \lfloor x/n_x \rfloor - 1$, $j = \lfloor y/n_y \rfloor - 1$, $k = \lfloor z/n_z \rfloor - 1$, $u = x/n_x - \lfloor x/n_x \rfloor$, $v = y/n_y - \lfloor y/n_y \rfloor$, $w = z/n_z - \lfloor z/n_z \rfloor$ and where B_l represents the lth basis function of the B-spline [22], [23]

$$B_0(u) = (1-u)^3/6$$
$$B_1(u) = (3u^3 - 6u^2 + 4)/6$$
$$B_2(u) = (-3u^3 + 3u^2 + 3u + 1)/6$$
$$B_3(u) = u^3/6.$$

D Rueckert, LI Sonoda, C Hayes, DLG Hill, MO Leach, DJ Hawkes: Nonrigid registration using free-form deformations: application to breast MR images. IEEE transactions on medical imaging 18 (8), 712-721

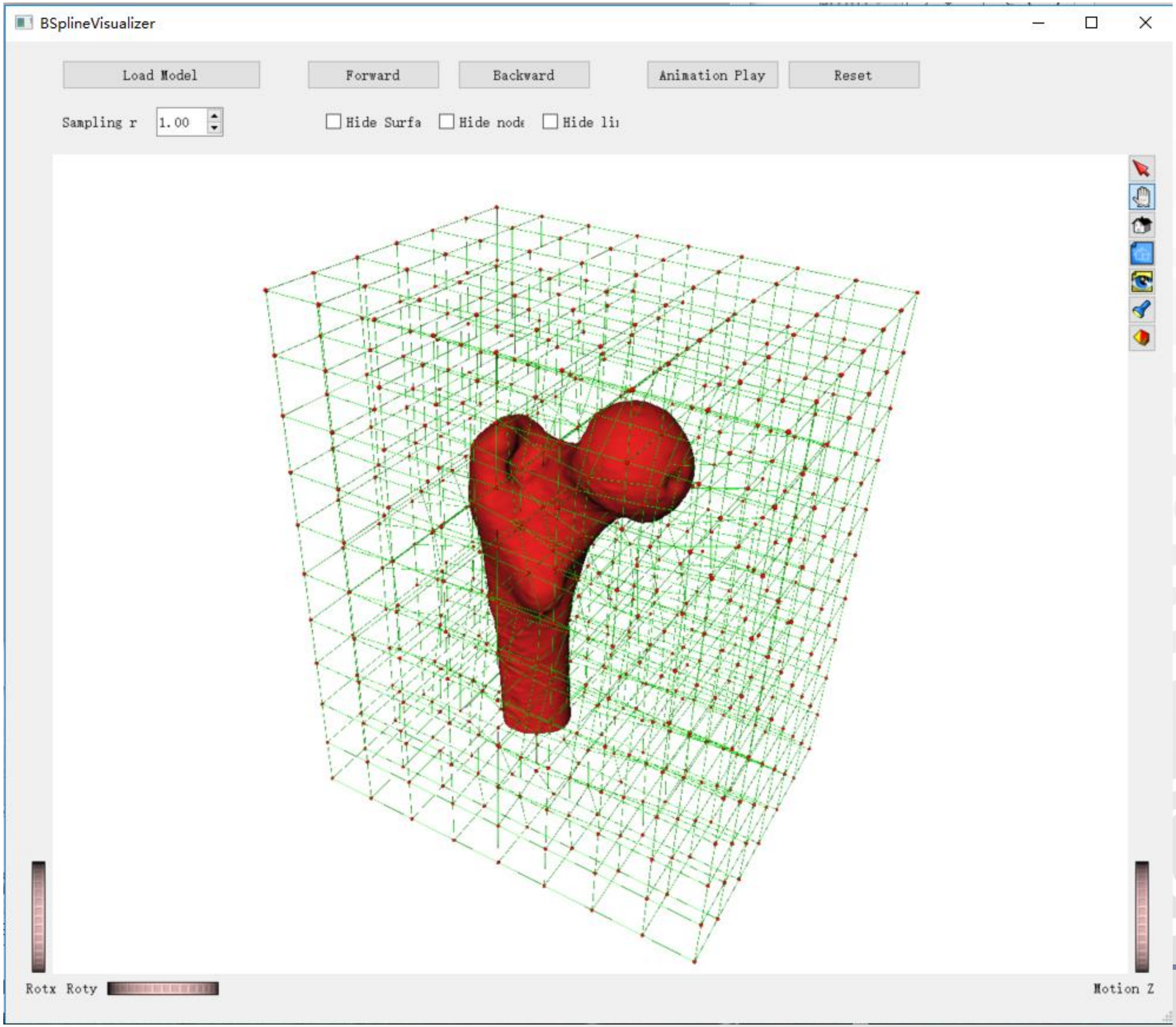
非线性变换-FFD

大数据学院 School of Data Science

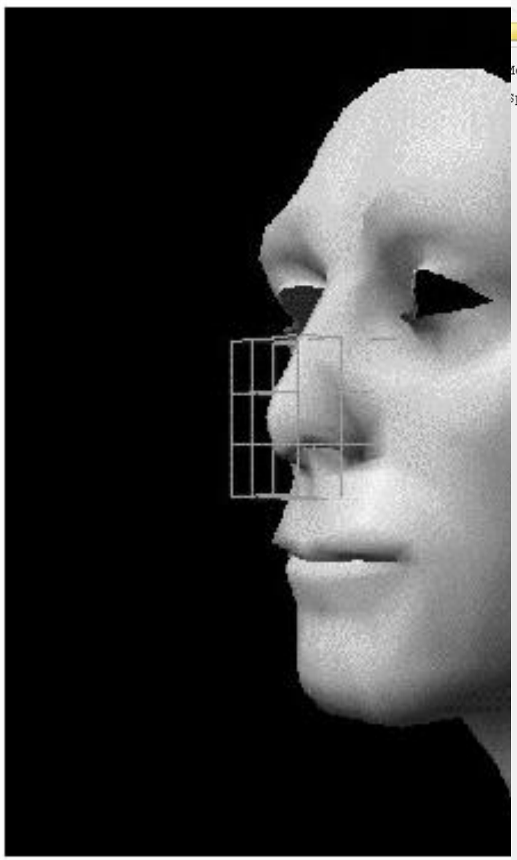
```
void  zxhTransformFFDBase::TransformPhysToPhys( const float fFrom[],float fTo[])const
{
  int iGrid[ImageDimensionMax] = {0,0,0,0} ;
  float fu[ImageDimensionMax]={0,0,0,0}, afOffset[ImageDimensionMax]={0,0,0,0};
  for(int index=0;index<m_iDimension;++index)
  {
    iGrid[index]= static_cast<int>(floor(fFrom[index]));
    fu[index]   = fFrom[index]-floor(fFrom[index]);
  }
  if(m_iDimension==3)
  {
    float fBSpline_u[4]={0,0,0,0},fBSpline_v[4]={0,0,0,0},fBSpline_w[4]={0,0,0,0};
    for(int ite=0;ite<4;++ite)
    {
      fBSpline_u[ite]  = BSplinei(ite-1,fu[0]);
      fBSpline_v[ite]  = BSplinei(ite-1,fu[1]);
      fBSpline_w[ite]  = BSplinei(ite-1,fu[2]);
    }
    for(int i=-1;i<=2;i++)
    for(int j=-1;j<=2;j++)
    for(int k=-1;k<=2;k++)
    {
      float* poff  = GetCtrPnt(iGrid[0]+i,iGrid[1]+j,iGrid[2]+k);
      if(poff!=0)
      {
        float w=fBSpline_u[i+1]*fBSpline_v[j+1]*fBSpline_w[k+1];
        for(int idx=0;idx<m_iDimension;++idx)
          afOffset[idx]  += poff[idx]*w;
      }
    }
  }else{}
  for(int idx=0;idx<m_iDimension;++idx)
    fTo[idx]  = fFrom[idx]+afOffset[idx] ;
  return  ;
}
```

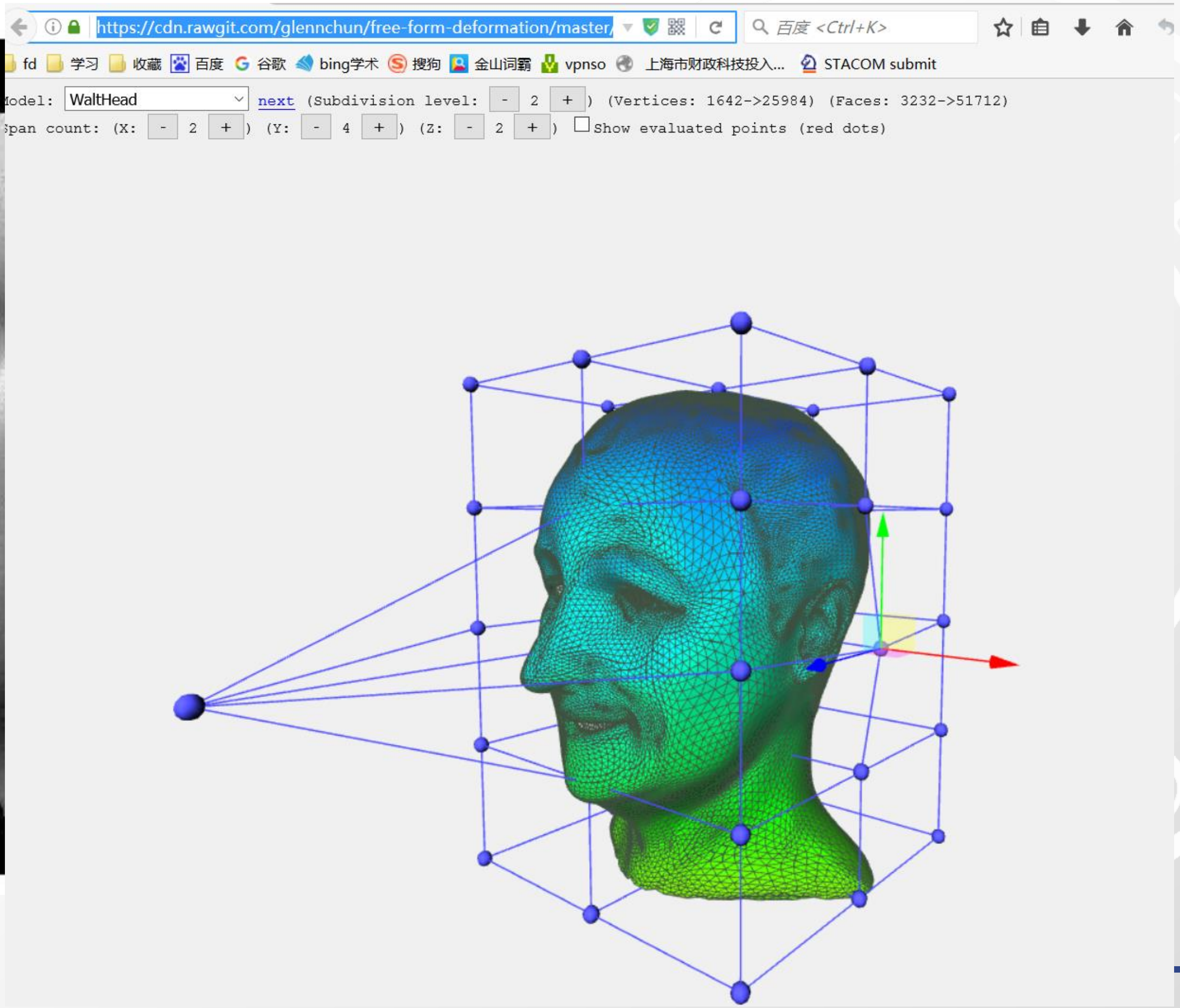
```
inline float  BSplinei(int iOrd,float u) const
{
  float v;
  switch(iOrd)
  {
  case -1:
    v=1-u;
    return v*v*v/6.0f;
  case 0:
    v=u*u*3.0f;
    return (v*u-v*2.0f+4.0f)/6.0f;
  case 1:
    v=3.0f*u*u;
    return (-v*u+v+3.0f*u+1.0f)/6.0f;
  case 2:return u*u*u/6.0f;
  }
  return 0;
};
```

非线性变换-FFD

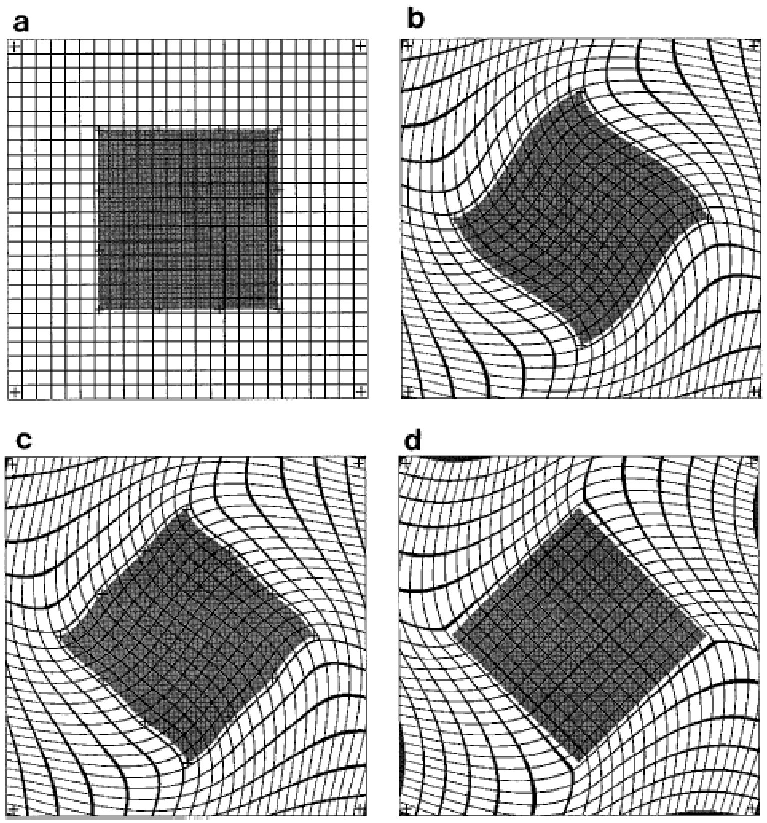

BSplineVisualizer
Load Model
Forward
Backward
Animation Play
Reset
Sampling r
1.00
Hide Surfa
Hide node
Hide li
Rotx Roty
Motion Z

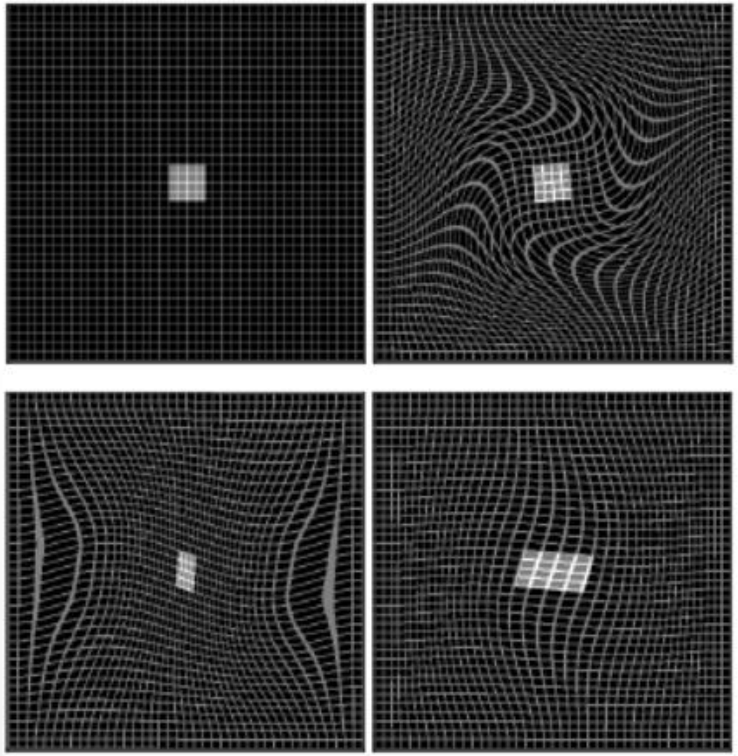
非线性变换-FFD



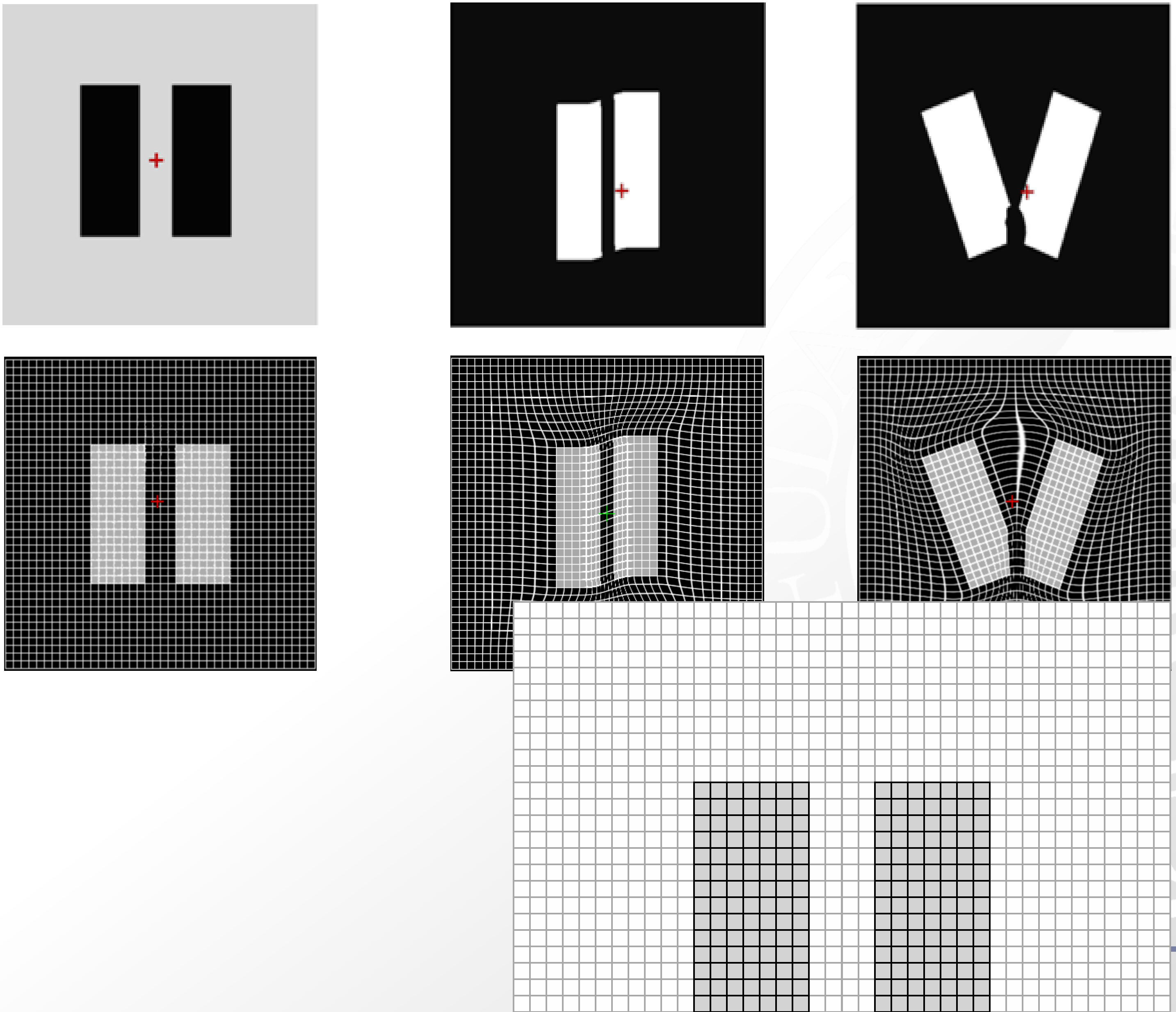

https://cdn.rawgit.com/glennchun/free-form-deformation/master/
百度 <Ctrl+K>
fd 学习 收藏 百度 谷歌 bing学术 搜狗 金山词霸 vpnso 上海市财政科技投入... STACOM submit
Model: WaltHead next (Subdivision level: - 2 +) (Vertices: 1642->25984) (Faces: 3232->51712)
Span count: (X: - 2 +) (Y: - 4 +) (Z: - 2 +) Show evaluated points (red dots)

局部仿射

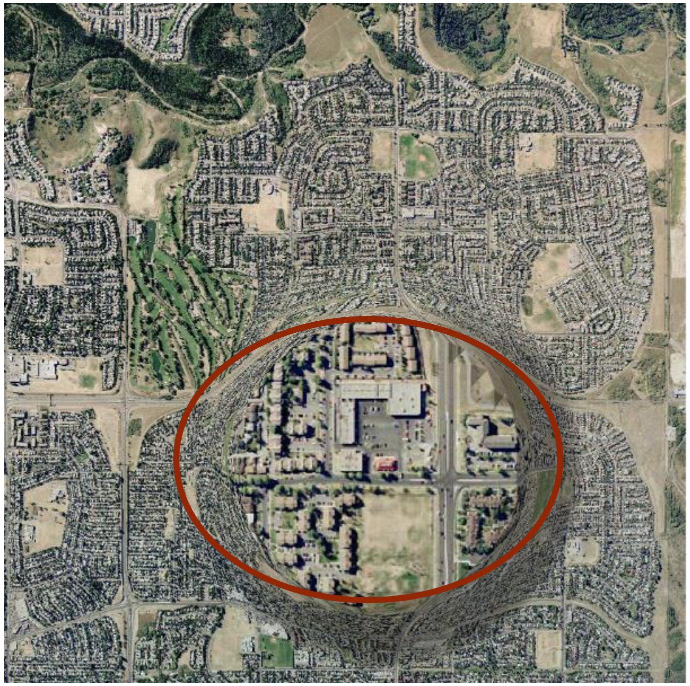




局部仿射

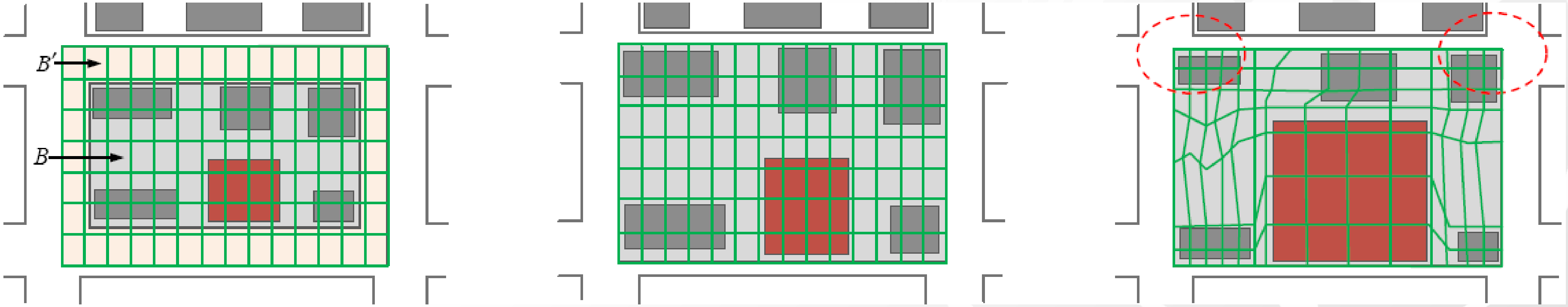


局部仿射：地图可视化



局部仿射：地图可视化

- 可能做法：
  - 放大整个街区，鱼眼放大
- 基于网格，有针对性的放大
  - 空间利用率更高，没有变形



局部仿射

- 假设有n个局部区域（坐标点）U_i，每个区域对应的局部仿射变换为G_i, 则局部仿射空间变换公式由下式计算

$$T(X)=\begin{cases} G_i(X), & X \in U_i,\ i=1..n \\ \sum_{i=1}^{n} w_i(X) G_i(X_i), & X \notin \bigcup_{i=1}^{n} U_i \end{cases}$$

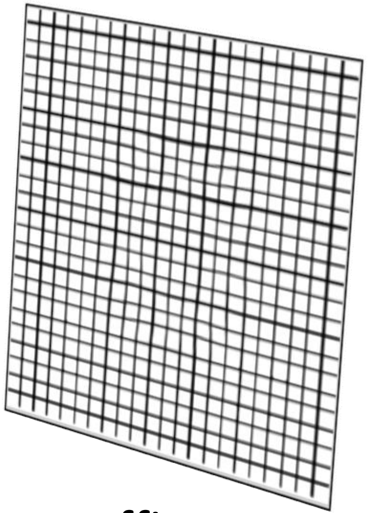
$$w_i(X)=\left(1/d_i(X)^e\right)\Big/\left(\sum_{i=1}^{n} 1/d_i(X)^e\right)$$

- 其中$d_i(X)$表示X到U_i的距离。

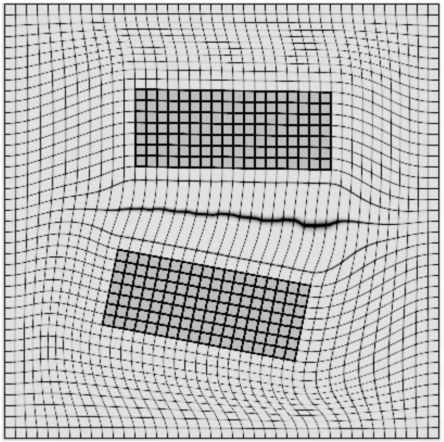
X Zhuang, KS Rhode, RS Razavi, DJ Hawkes, S Ourselin: A registration-based propagation framework for automatic whole heart segmentation of cardiac MRI. IEEE Transactions on Medical Imaging, 29 (9), 1612-1625

Summary of transforms

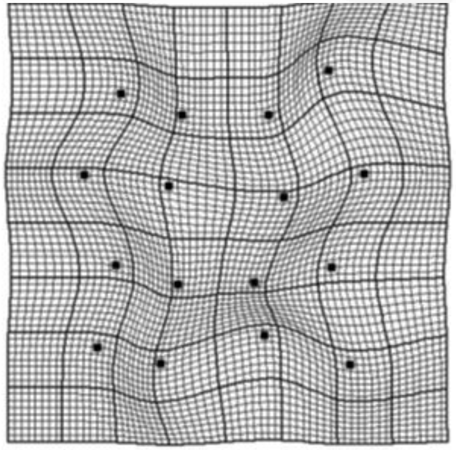
大数据学院 School of Data Science



affine



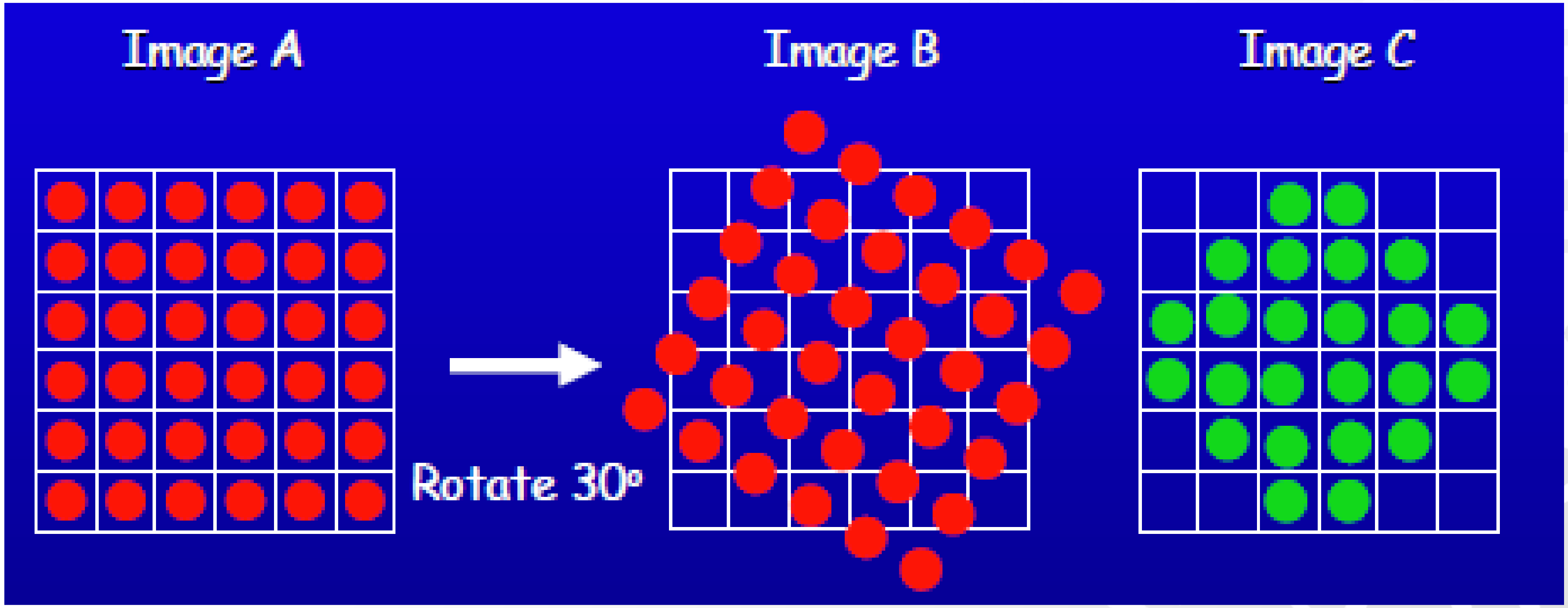
locally affine



deformable

Transform an image

- Forward VS Backward (inverse)
- Image grid
- Interpolation



Transform an image

前向图变换：

procedure *forwardWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

Algorithm 3.1 Forward warping algorithm for transforming an image $f(\boldsymbol{x})$ into an image $g(\boldsymbol{x}')$ through the parametric transform $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.

Transform an image

反向图变换:

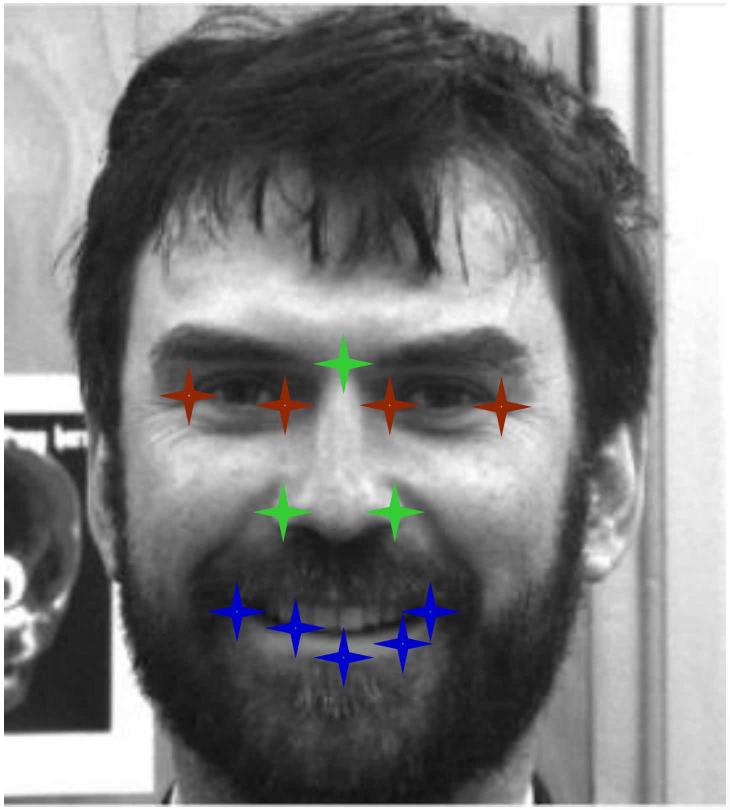
procedure $\mathit{inverseWarp}(f, \boldsymbol{h}, \textbf{out}\ g)$:

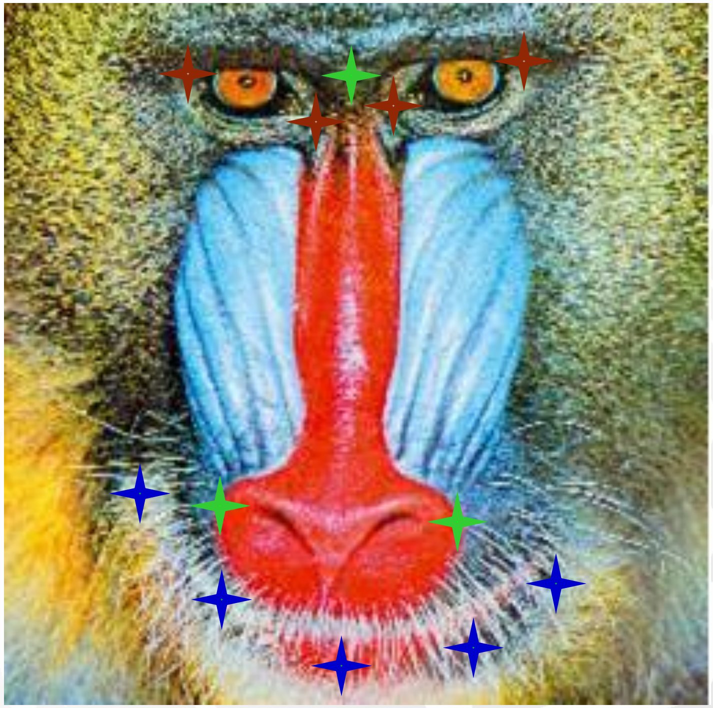
For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$

1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

Algorithm 3.2 Inverse warping algorithm for creating an image $g(\boldsymbol{x}')$ from an image $f(\boldsymbol{x})$ using the parametric transform $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.

Figure 3.47 Inverse warping algorithm: (a) a pixel $g(\boldsymbol{x}')$ is sampled from its corresponding location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$ in image $f(\boldsymbol{x})$; (b) detail of the source and destination pixel locations.





$$T(X)=\begin{cases} G_i(X), & X\in U_i,\ i=1\ldots n \\ \sum_{i=1}^{n} w_i(X)G_i(X_i), & X\notin \bigcup_{i=1}^{n} U_i \end{cases}$$

$$w_i(X)=\left(1/d_i(X)^e\right)\Big/\left(\sum_{i=1}^{n}1/d_i(X)^e\right)$$

[Images courtesy of David J. Hawkes]

復旦大學 FUDAN UNIVERSITY





Thank You！