コンピュータビジョン
多視点幾何

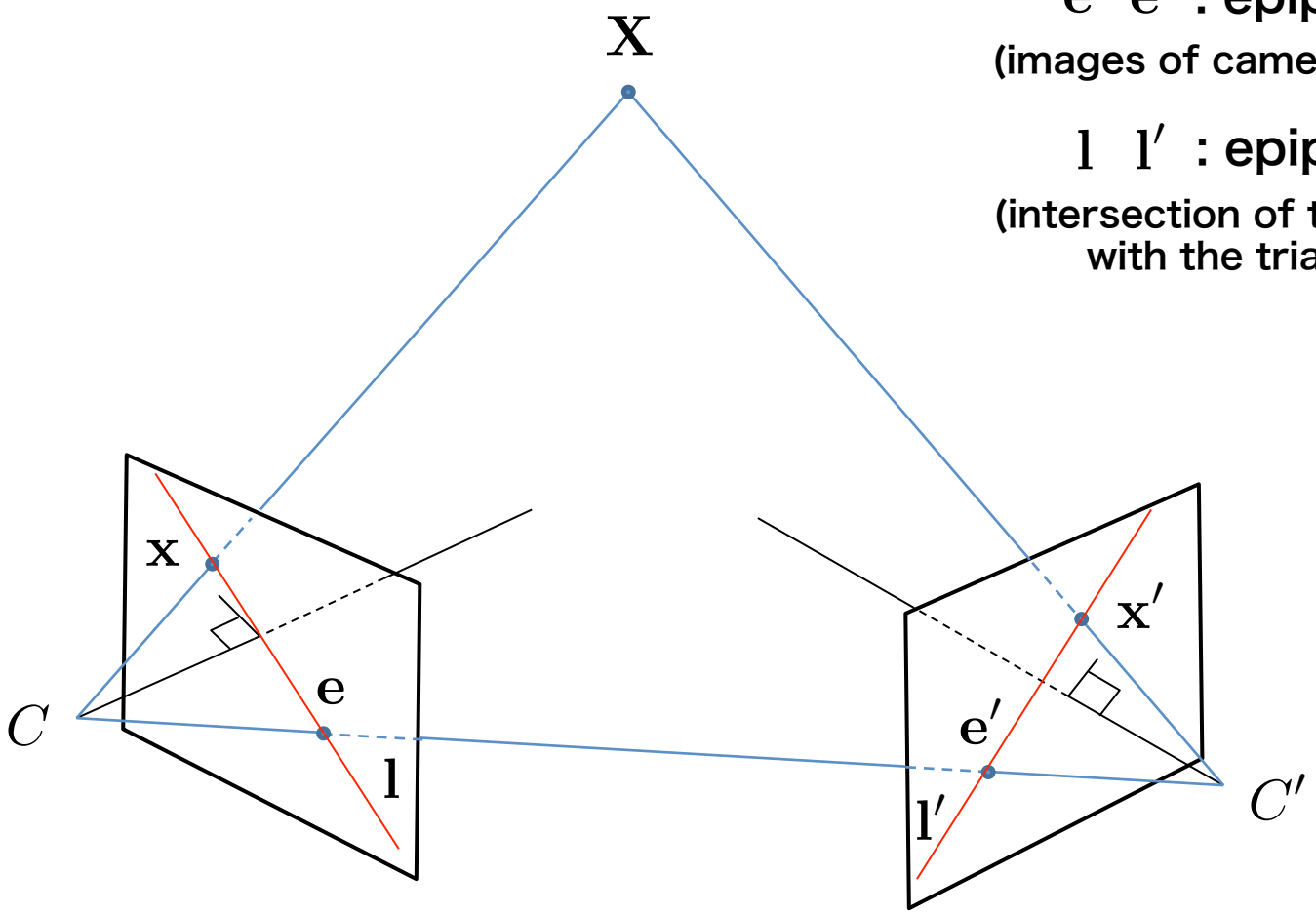
Epipolar geometry
(Two-view geometry)

$\mathbf{e}$ $\mathbf{e}'$: epipoles

(images of camera centers)

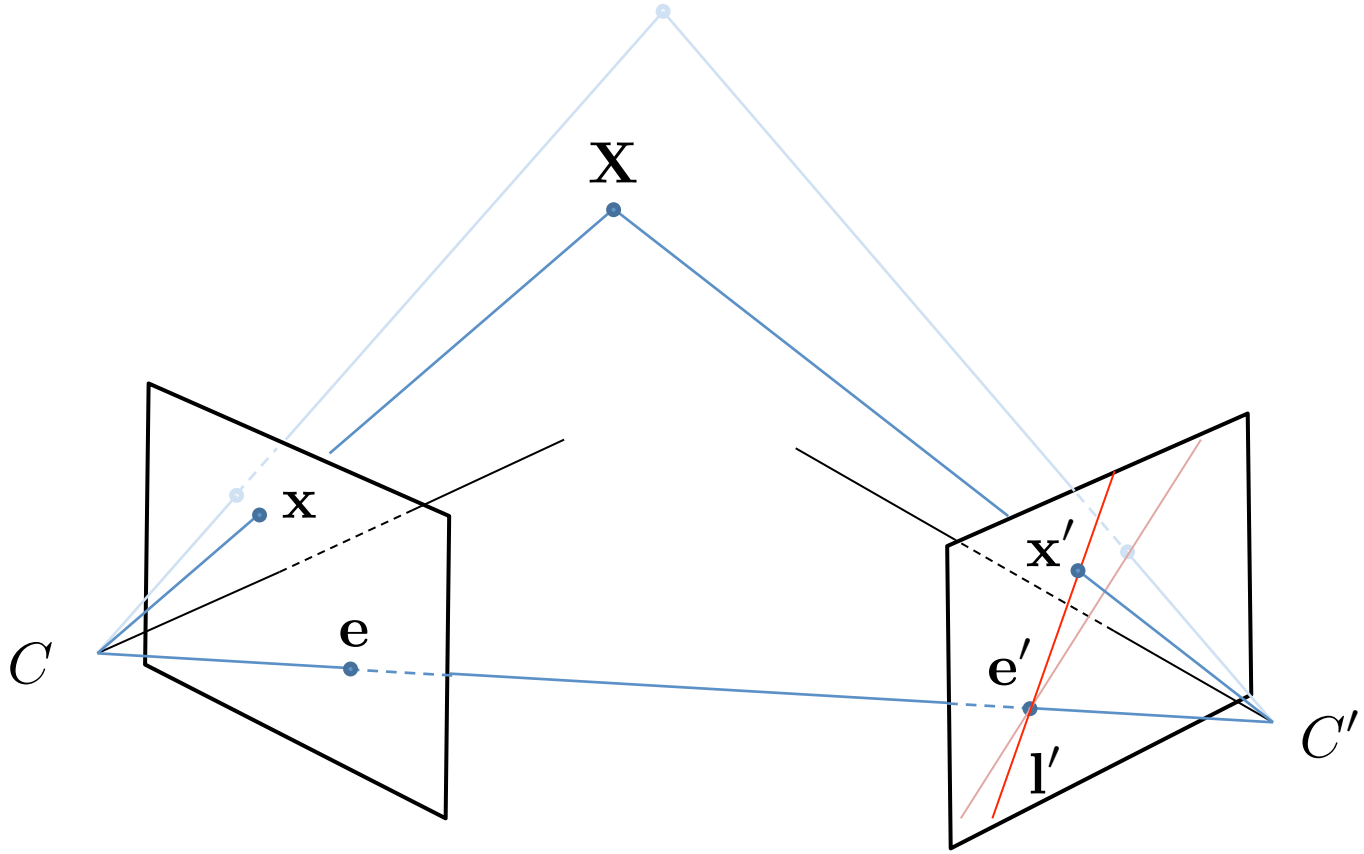
$\mathbf{l}$ $\mathbf{l}'$: epipolar lines

(intersection of the image plane with the triangle CC'X)



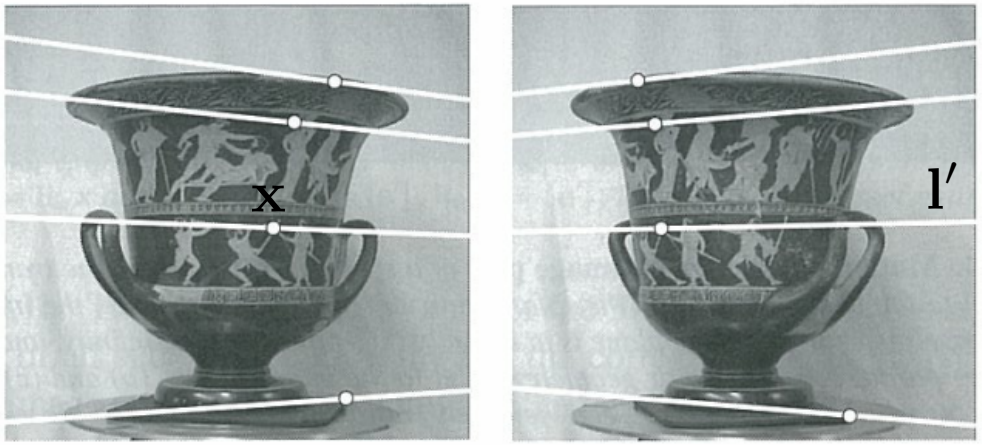
Epipolar geometry
(Two-view geometry)

- x を決めると l’ が決まる

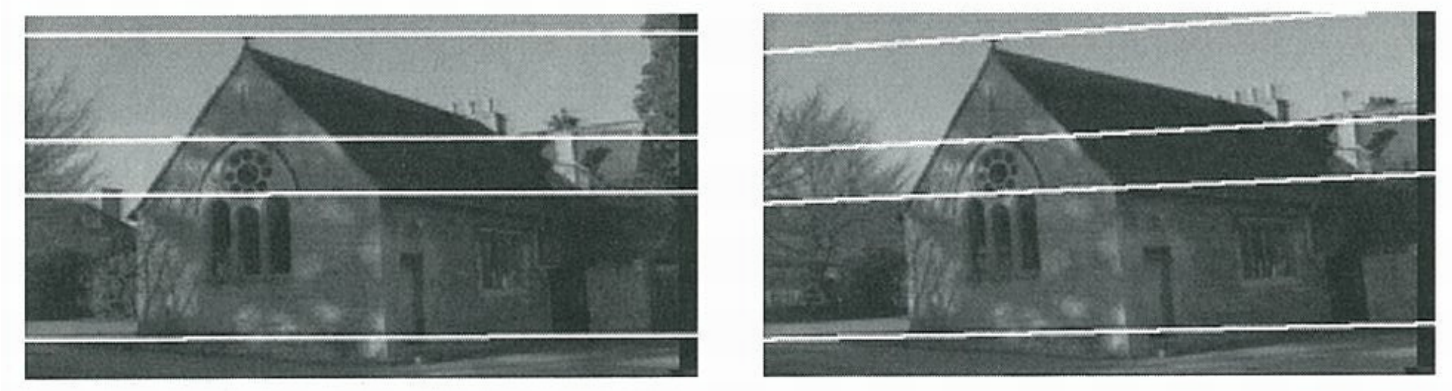


Epipolar geometry
(Two-view geometry)

- x を決めると l' が決まる



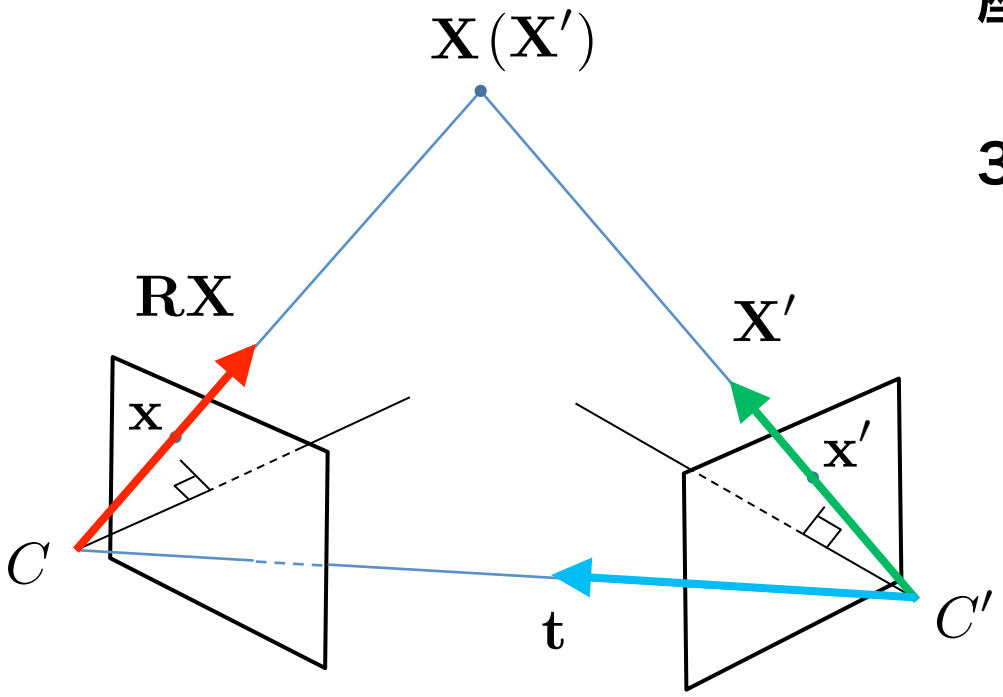
  - epipoleが無限遠にあるとき



[Hartley-Zisserman03]

Essential matrix

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$



E は自由度5

R(3)+t(3)-scale(1)=5

座標変換： $\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{t}$

3つのベクトルが同一平面上

$$\rightarrow \mathbf{X}'^{\top}(\mathbf{t} \times \mathbf{R}\mathbf{X}) = 0$$

$$\rightarrow \mathbf{X}'^{\top}([\mathbf{t}]_\times \mathbf{R}\mathbf{X}) = 0$$

$$\rightarrow \mathbf{X}'^{\top}([\mathbf{t}]_\times \mathbf{R})\mathbf{X} = 0$$

$$\rightarrow \mathbf{X}'^{\top}\mathbf{E}\mathbf{X} = 0$$

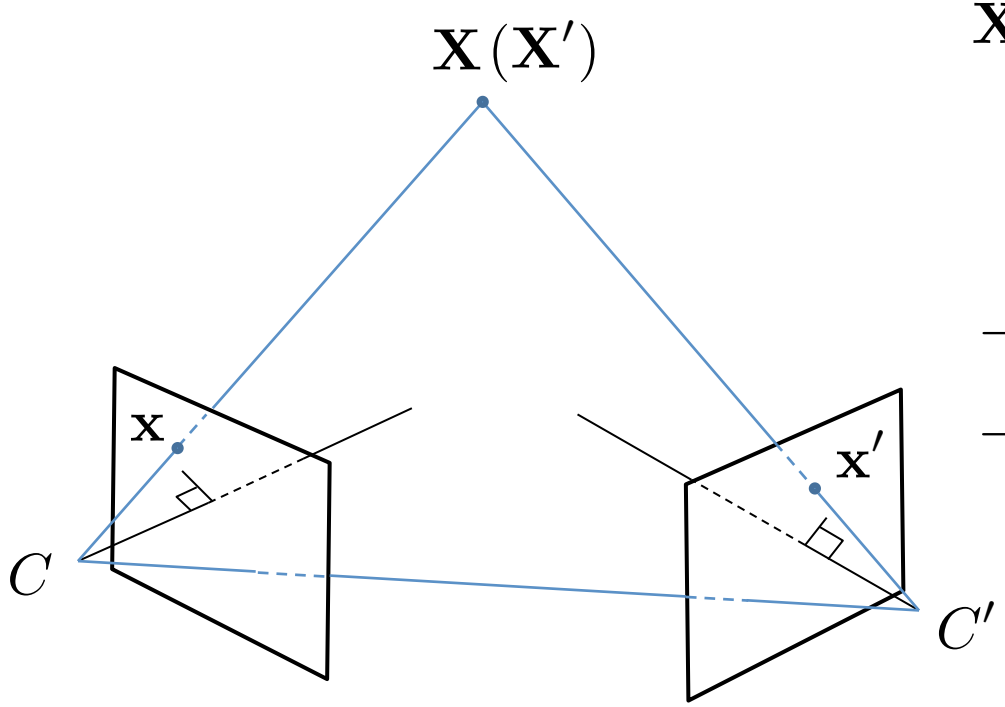
Fundamental matrix

$$\mathbf{F} = \mathbf{K}'^{-\top}\mathbf{E}\mathbf{K}^{-1}$$

essential mat.の「画像座標版」

F は自由度7

3×3 – scale(1) – “det=0”(1) = 7



$\mathbf{X}'^{\top}\mathbf{E}\mathbf{X} = 0$ に以下を代入

$$\mathbf{x} \propto \mathbf{K}\mathbf{X} \quad \rightarrow \mathbf{X} \propto \mathbf{K}^{-1}\mathbf{x}$$

$$\mathbf{x}' \propto \mathbf{K}'\mathbf{X}' \quad \rightarrow \mathbf{X}' \propto \mathbf{K}'^{-1}\mathbf{x}'$$

$$\rightarrow (\mathbf{K}'^{-1}\mathbf{x}')^{\top}\mathbf{E}(\mathbf{K}^{-1}\mathbf{x}) = 0$$

$$\rightarrow \mathbf{x}'^{\top}(\mathbf{K}'^{-\top}\mathbf{E}\mathbf{K}^{-1})\mathbf{x} = 0$$

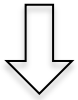
$$\rightarrow \mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$

Fundamental matrixの推定

8-point algorithm

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$

$$x'xf_{00} + x'yf_{01} + x'f_{02} + y'xf_{10} + y'yf_{11} + y'f_{12} + xf_{20} + yf_{21} + f_{22} = 0$$



$$9$$

$$n \begin{bmatrix} x'_0x_0 & x'_0y_0 & x'_0 & y'_0x_0 & y'_0y_0 & y'_0 & x_0 & y_0 & 1 \\ x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ & & & & & & \vdots & & \\ x'_{n-1}x_{n-1} & x'_{n-1}y_{n-1} & x'_{n-1} & y'_{n-1}x_{n-1} & y'_{n-1}y_{n-1} & y'_{n-1} & x_{n-1} & y_{n-1} & 1 \end{bmatrix} \begin{bmatrix} f_{00} \\ f_{01} \\ f_{02} \\ f_{10} \\ f_{11} \\ f_{12} \\ f_{20} \\ f_{21} \\ f_{22} \end{bmatrix} = \mathbf{0}$$

$$\mathbf{A}\mathbf{f} = \mathbf{0}$$

Fundamental matrixの推定

fundM.py

```
    if desc0 != None:
        # Match the saved and the latest kp's
        matches = bf.match(desc0, desc)
        src_pts = numpy.float32([ kp0[m.queryIdx].pt \
                                  for m in matches ]).reshape(-1,1,2)
        dst_pts = numpy.float32([ kp[m.trainIdx].pt \
                                  for m in matches ]).reshape(-1,1,2)
        F, mask = cv2.findFundamentalMat(src_pts, dst_pts,
                                                    \cv2.FM_LMEDS)
        …
    key = cv2.waitKey(10)

    if key == 0x1b: # ESC
        cv2.imwrite('img1.png', image)
        numpy.savetxt('fundM.txt', F)
        break
    elif key == 0x20:
        desc0 = desc.copy()
        kp0 = kp[:]
        cv2.imwrite('img0.png', image)
        print 'Target updated!'
```

Space → 第１視点・ESC → 第２視点，fundM.txt にセーブ

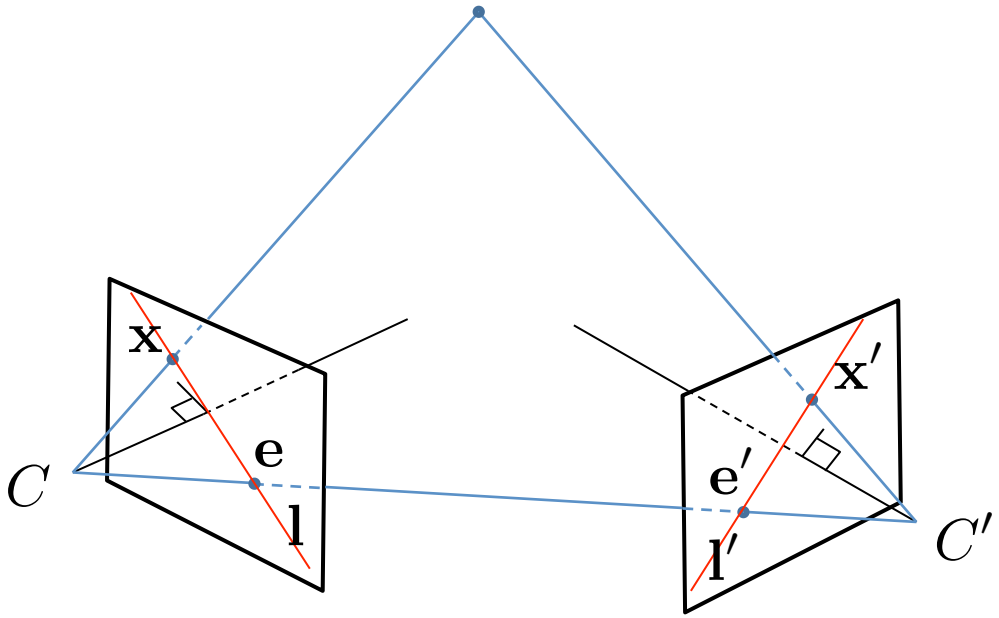
Fundamental matrix

x に対するepipolar line l' は $\mathbf{l}' \propto \mathbf{F}\mathbf{x} \quad (\mathbf{x}'^{\top}\mathbf{l}' = \mathbf{x}'^{\top}(\mathbf{F}\mathbf{x}) = 0)$

(x' に対するepipolar line l は $\mathbf{l} \propto \mathbf{F}^{\top}\mathbf{x}' \, (\mathbf{x}^{\top}\mathbf{l} = \mathbf{x}^{\top}(\mathbf{F}^{\top}\mathbf{x}') = 0)$ **)**

x に対するepipolar line l' は定まらない： $\mathbf{F}\mathbf{e} = \mathbf{F}^{\top}\mathbf{e} = \mathbf{0}$

どのepipolar line l も e を通る： $\mathbf{l}^{\top}\mathbf{e} = \mathbf{x}'^{\top}\mathbf{F}\mathbf{e} = \mathbf{x}^{\top}(\mathbf{F}\mathbf{e}) = 0$



直線の式： $\mathbf{l}^{\top}\mathbf{x} = 0$

$$\mathbf{l} = [a, b, c]^{\top}$$
$$\mathbf{x} = [x, y, 1]^{\top}$$
$$ax + by + c = 0$$

Epipolar line

epi_line.py

```
def on_mouse(event, x, y, flags, param):
    if event == cv2.EVENT_MOUSEMOVE:
        l = F.dot(numpy.array((x,y,1)))
        w = img1.shape[1]
        img1c = img1.copy()
        cv2.line(img1c, (0,int(-l[2]/l[1])),\
                 (w,int((-l[2]-l[0]*w)/l[1])), (0,0,255))
        cv2.imshow('Image 1', img1c)

cv2.namedWindow('Image 0')
cv2.setMouseCallback('Image 0', on_mouse)

img0 = cv2.imread('img0.png')
img1 = cv2.imread('img1.png')
img1c = img1.copy()
F = numpy.loadtxt('fundM.txt')
cv2.imshow('Image 0', img0)
cv2.imshow('Image 1', img1c)

while 1:
    key = cv2.waitKey(10)

    if key == 0x1b: # ESC
        break
```

Essential matrixの分解

- **E を分解して R と t を取り出せる**

  E の特異値分解は必ず次の形に：

$$\mathbf{E} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^\top$$

  次の４通りの (R, t) がある :

$$(\mathbf{R}, \mathbf{t}) = (\mathbf{U}\mathbf{W}\mathbf{V}^\top, \mathbf{u}_3)$$
$$(\mathbf{R}, \mathbf{t}) = (\mathbf{U}\mathbf{W}\mathbf{V}^\top, -\mathbf{u}_3)$$
$$(\mathbf{R}, \mathbf{t}) = (\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top, \mathbf{u}_3)$$
$$(\mathbf{R}, \mathbf{t}) = (\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top, -\mathbf{u}_3)$$

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Essential matrixの分解

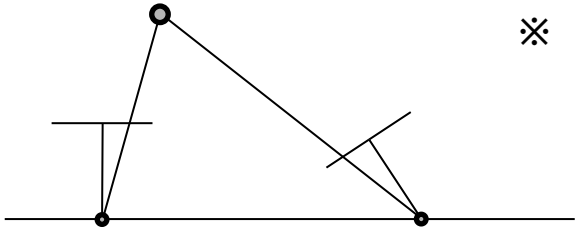
decompE.py

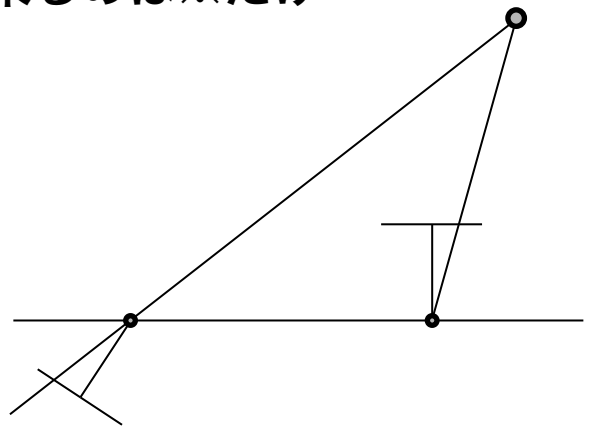
```
import numpy, cv2

def decompose_E(E):
    U, s, Vt = numpy.linalg.svd(E)
    print 'singluar value = ', s
    #print U.dot(numpy.diag(s)).dot(Vt), E
    W = numpy.array([[0,-1,0],[1,0,0],[0,0,1]])
    R0=U.dot(W).dot(Vt) # UWVt
    R1=U.dot(W.transpose()).dot(Vt) # UWVt
    t=U[:,2] # u3
    Rt_list=((R0,t),(R0,-t),(R1,t),(R1,-t))
    return Rt_list

if __name__=="__main__":
    F = numpy.loadtxt('fundM.txt')
    K = numpy.loadtxt('K.txt')
    E = (K.transpose().dot(F)).dot(K)
    Rt_list = decompose_E(E)
    print Rt_list
```

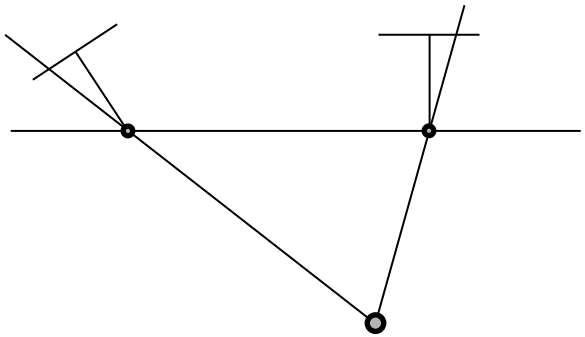
Essential matrixの分解

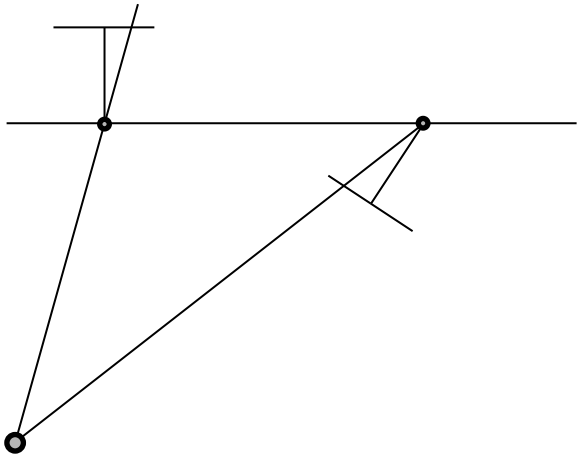
- ４通りの分解
  - ただし３次元点が両カメラの前に来るのは※だけ

※



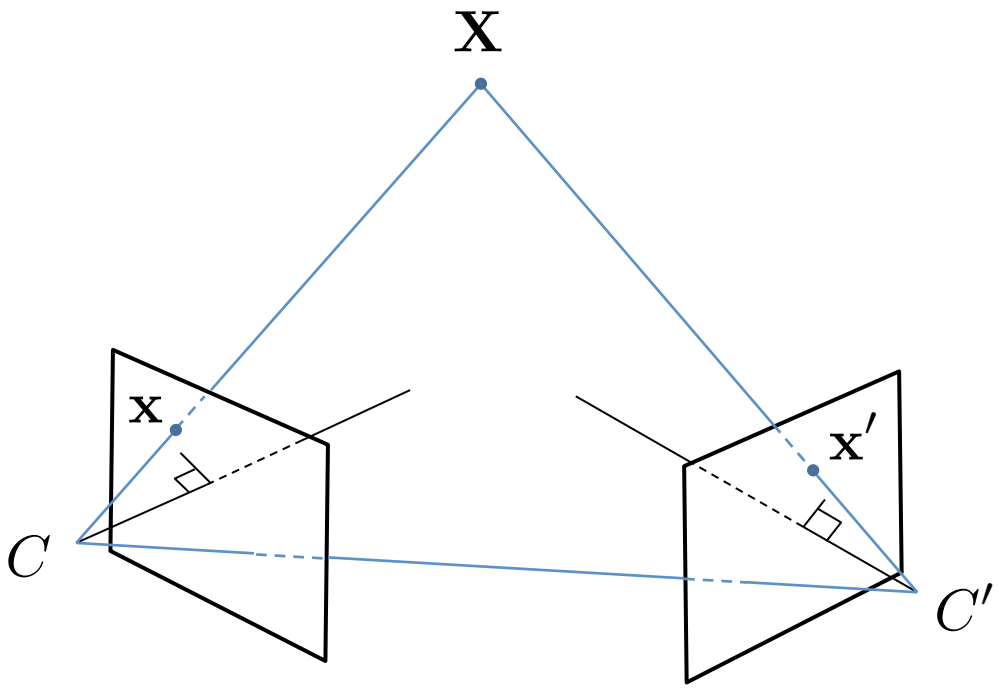






Triangulation

- ２つのカメラ行列 P, P’ が与えられたとき，x, x’ から X を求める



$$\mathbf{x} \propto \mathbf{P}\mathbf{X} \qquad \mathbf{x}' \propto \mathbf{P}'\mathbf{X}$$

$$\mathbf{P} = \mathbf{K}[\mathbf{I} \;\; \mathbf{0}] \qquad \mathbf{P}' = \mathbf{K}'[\mathbf{R} \;\; \mathbf{t}]$$

$$x = \frac{p_{00}X + p_{01}Y + p_{02}Z + p_{03}}{p_{20}X + p_{21}Y + p_{22}Z + p_{23}}$$

$$y = \frac{p_{10}X + p_{11}Y + p_{12}Z + p_{13}}{p_{20}X + p_{21}Y + p_{22}Z + p_{23}}$$

$$x' = \frac{p'_{00}X + p'_{01}Y + p'_{02}Z + p'_{03}}{p'_{20}X + p'_{21}Y + p'_{22}Z + p'_{23}}$$

$$y' = \frac{p'_{10}X + p'_{11}Y + p'_{12}Z + p'_{13}}{p'_{20}X + p'_{21}Y + p'_{22}Z + p'_{23}}$$

⇩

$$\begin{bmatrix} \mathbf{a}_0^\top \\ \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \mathbf{a}_3^\top \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

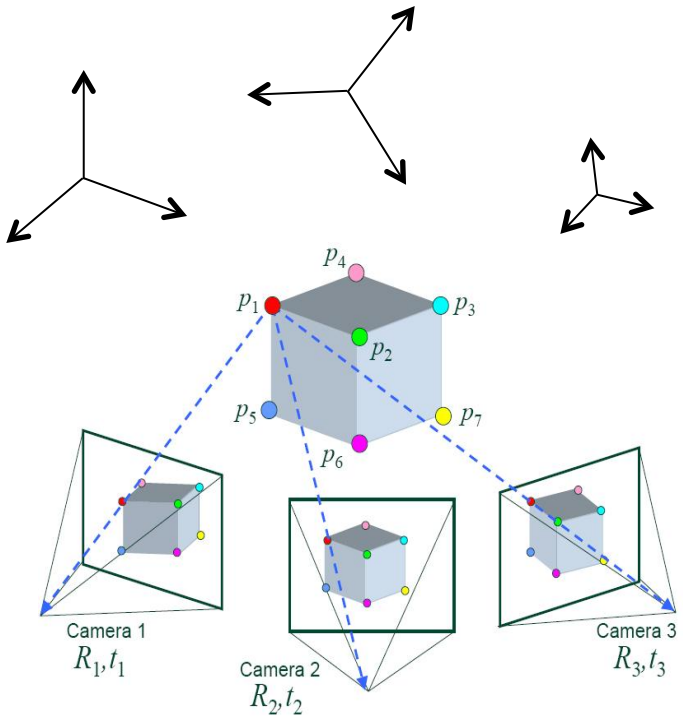
Triangulation

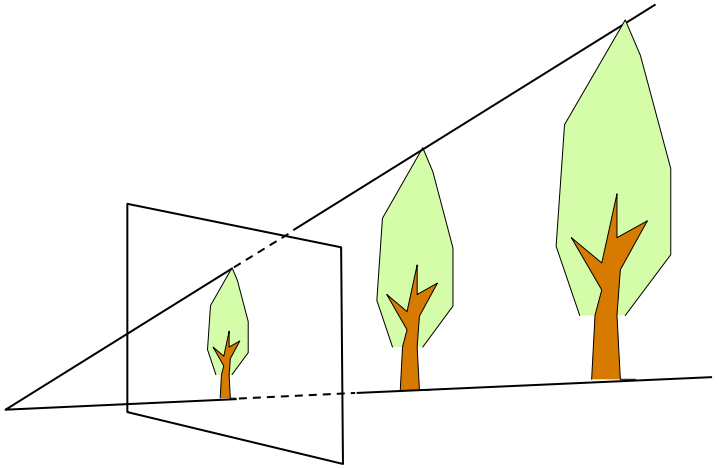
triangulate.py

```
F = numpy.loadtxt('fundM.txt')
K = numpy.loadtxt('K.txt')
pts0 = numpy.loadtxt('pts0.txt').transpose()
pts1 = numpy.loadtxt('pts1.txt').transpose()

E = (K.transpose().dot(F)).dot(K)
Rt_list = decompE.decompose_E(E)
P0 = numpy.array([[1,0,0,0],[0,1,0,0],[0,0,1,0]])
P0 = K.dot(P0)
for i in range(4):
    P1c = numpy.zeros((3,4))
    P1c[0:3,0:3] = Rt_list[i][0]
    P1c[:,3] = Rt_list[i][1]
    P1 = K.dot(P1c)
    print 'P1=', P1
    pts3d = cv2.triangulatePoints(P0, P1, pts0, pts1)
    pts3d = pts3d/pts3d[3,:] # normalize
    pts3d_1 = P1c.dot(pts3d) # X'=RX+t
    #print pts3d[2,:]
    print numpy.median(pts3d[2,:])
    print numpy.median(pts3d_1[2,:])
```

基本的な不定性

- 画像からの３次元復元では，世界座標系の取り方とスケールは不定
  - 視点が何枚あっても同じ





Structure-from-Motion (SfM)

- 問題：*n* 個のシーンの点を*m* 視点で撮影した画像（の点の座標）を元に, *n* 点の３次元座標と*m* カメラの姿勢を求めたい

INPUT:
m image coordinates
of the *n* points

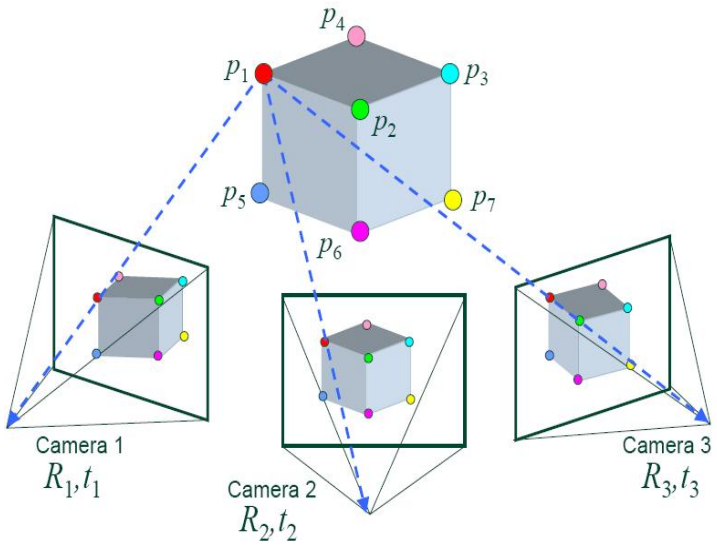
$$\mathbf{x}_{ij} = [x_{ij}, y_{ij}]$$

$$(i = 1, \ldots, m)$$

$$(j = 1, \ldots, n)$$

Image formation model

$$\mathbf{x}_{ij} \propto \mathbf{P}_i \mathbf{X}_j$$



OUTPUT:
n point coordinates
m camera info.

$$\mathbf{X}_j \ (j = 1, \ldots, n)$$

$$\mathbf{P}_i \ (i = 1, \ldots, m)$$

カメラモデル (revisited)

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image coordinates

$$\mathbf{x} \propto \mathbf{P}\mathbf{X}$$

$$\mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]$$

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

3D coordinates

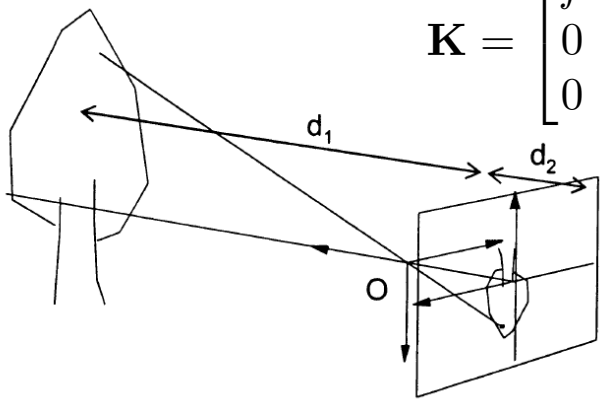
Internal parameters
（focal length etc.）

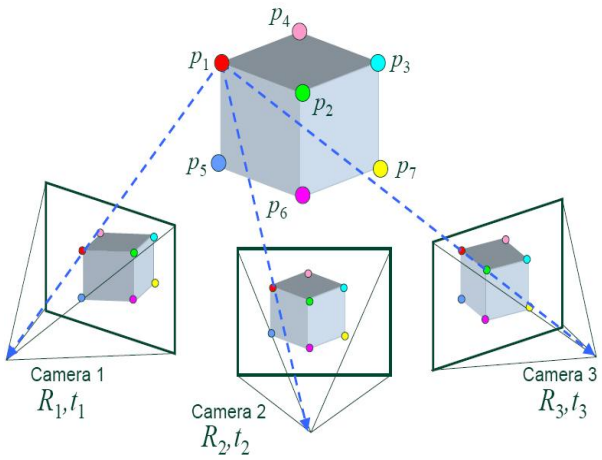
“What camera?”

External parameters
（＝camera poses）

“From where?”

$$\mathbf{K} = \begin{bmatrix} f & s & x_0 \\ 0 & \alpha f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$





解き方（Factorization）

Known　Unknown　Unknown

$$\mathbf{x}_{ij} \propto \mathbf{P}_i \mathbf{X}_j$$

$$(i = 1, \ldots, m, \quad j = 1, \ldots, n)$$

$$\underbrace{\begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \cdots & \mathbf{x}_{1n} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \cdots & \mathbf{x}_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{x}_{m1} & \mathbf{x}_{m2} & \cdots & \mathbf{x}_{mn} \end{bmatrix}}_{n} \sim \underbrace{\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_m \end{bmatrix}}_{4} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots \mathbf{X}_n \end{bmatrix}}_{n} \updownarrow 4$$

$3m$

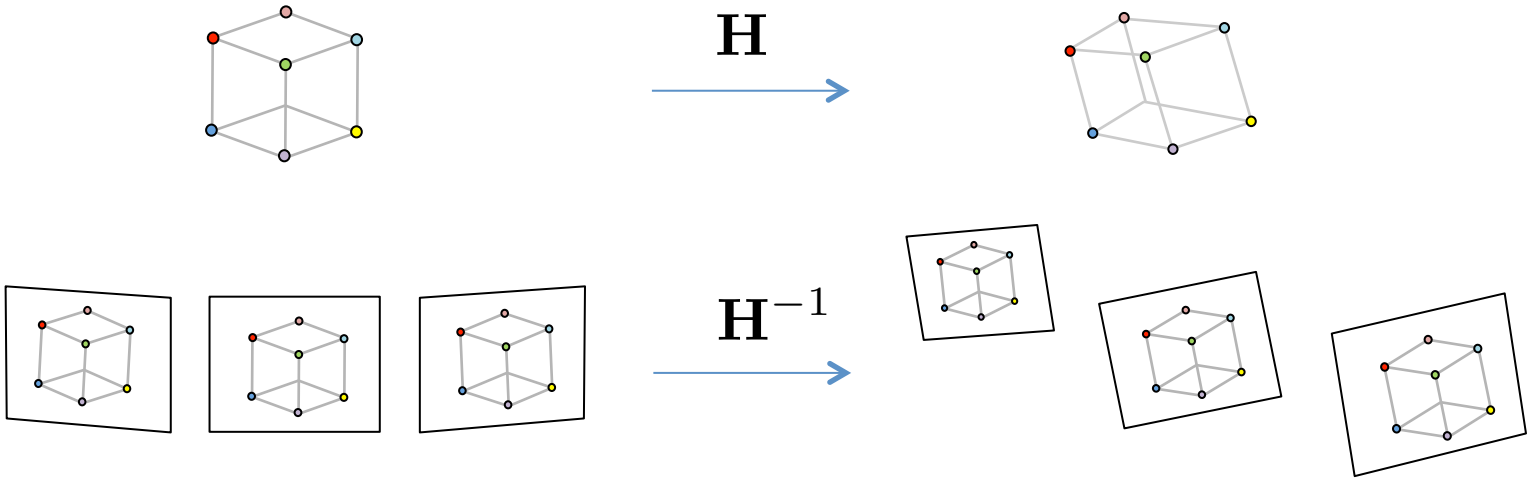
A large number of data　　Unknowns

射影的不定性：Projective ambiguity

- 復元の不定性
  - 画像だけでは一意には決まらない

$$\mathbf{x}_{ij} \propto \mathbf{P}_i\mathbf{X}_j = \mathbf{P}_i\mathbf{H}^{-1}\mathbf{H}\mathbf{X}_j = (\mathbf{P}_i\mathbf{H}^{-1})(\mathbf{H}\mathbf{X}_j) = \mathbf{P}'_i\mathbf{X}'_j$$

  - 任意の3D projective trans. H 分の自由度（15 DOF）

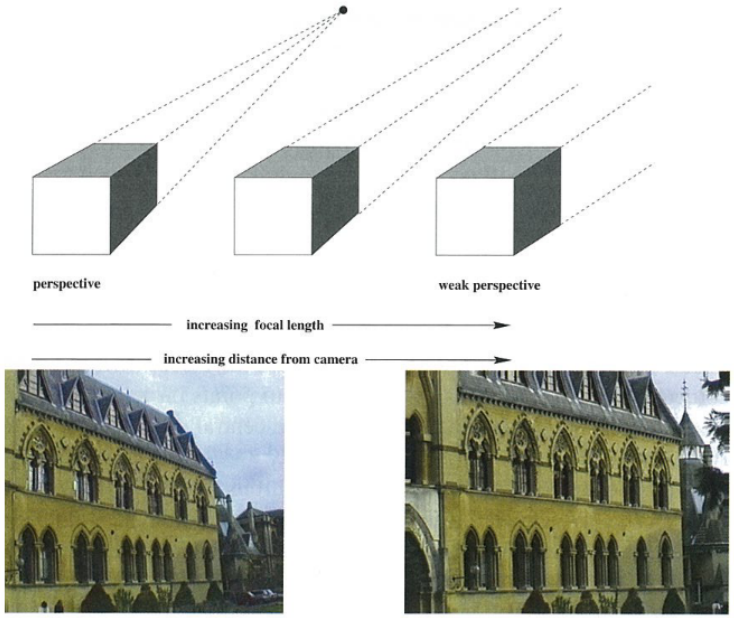


Affine camera

- カメラ行列が次の形のカメラをアフィンカメラと呼ぶ

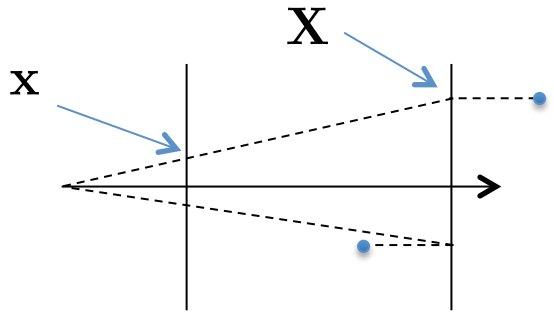
$$\mathbf{P} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 正射影カメラを表す他，長焦点レンズの近似にも
  - 正射影：orthographic projection，telecentric lens



“scaled orthographic projection”

$$\mathbf{P} = \begin{bmatrix} \alpha & & \\ & \alpha & \\ & & 1 \end{bmatrix} \begin{bmatrix} & \mathbf{r}_1^\top & & t_1 \\ & \mathbf{r}_2^\top & & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Affine camera

- アフィンカメラの投影の式は線形になる

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \propto \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

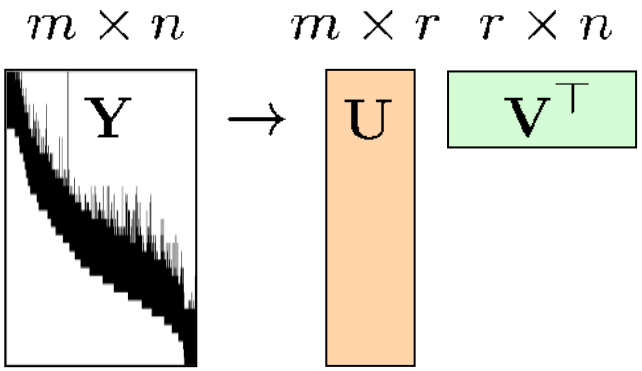
$$\tilde{\mathbf{x}} = \tilde{\mathbf{P}}\mathbf{X}$$

- SfMが行列分解の問題に帰着
  - Tomasi-Kanadeのfactorization method [1992]

$$\begin{bmatrix} \tilde{\mathbf{x}}_{11} & \tilde{\mathbf{x}}_{12} & \cdots & \tilde{\mathbf{x}}_{1n} \\ \tilde{\mathbf{x}}_{21} & \tilde{\mathbf{x}}_{22} & \cdots & \tilde{\mathbf{x}}_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \tilde{\mathbf{x}}_{m1} & \tilde{\mathbf{x}}_{m2} & \cdots & \tilde{\mathbf{x}}_{mn} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{P}}_1 \\ \tilde{\mathbf{P}}_2 \\ \vdots \\ \tilde{\mathbf{P}}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

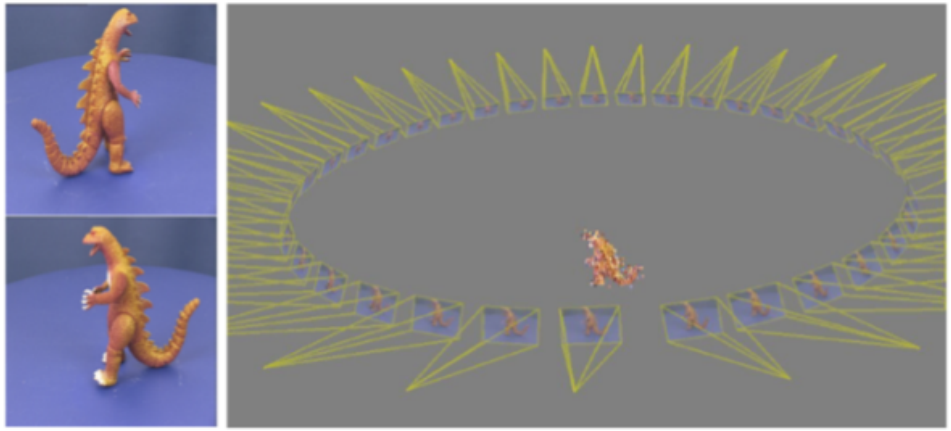
Affine factorization

- 全点が見えているわけではない
  - 非観測成分(missing components)を含む行列分解
  - Wibergのアルゴリズム
- 行列分解には不定性 → 射影的不定性のアフィンカメラ版
  - UとVには制約があるのでそれを使ってこの不定性を解消
  - PのP[0,0:3]とP[1,0:3]の長さが同じで互いに直交



$$\phi(\mathbf{U},\mathbf{V}) = \|\mathbf{Y} - \mathbf{U}\mathbf{V}^\top\|_F^2 \rightarrow \min$$

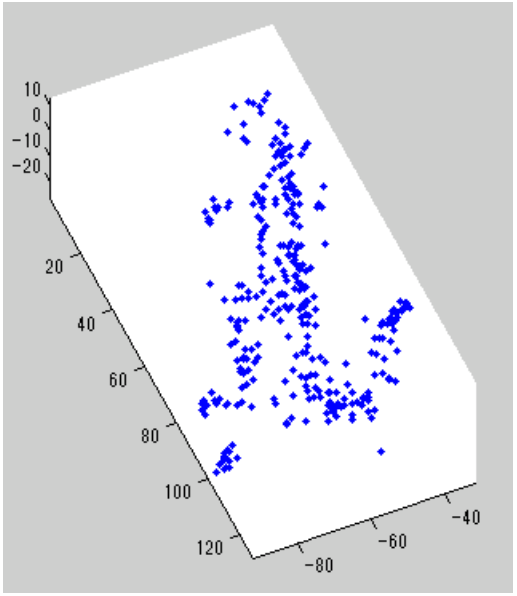
$$\mathbf{P} = \begin{bmatrix} \alpha & & \\ & \alpha & \\ & & 1 \end{bmatrix} \begin{bmatrix} & \mathbf{r}_1^\top & & t_1 \\ & \mathbf{r}_2^\top & & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

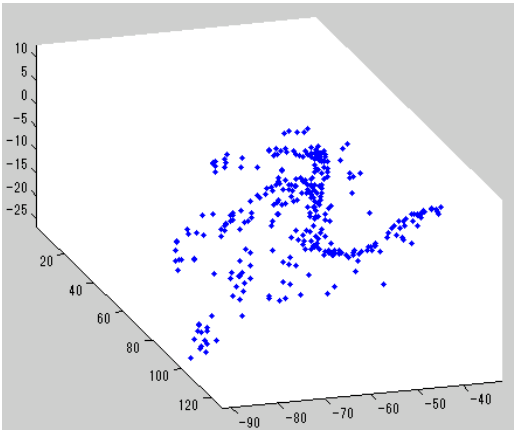


Affine factorization

- Matlabでの実行内容

```
>> Y=load('M.txt');
>> mask=load('mask.txt');
>> [U,V]=damped_wiberg(Y, mask, 4)
[001] 1.000000e-01 4.574572e+08
[002] 1.000000e-02 9.673768e+07
[003] 1.000000e-03 6.650256e+07
...
[125] 1.000000e-04 6.263656e+03
[126] 1.000000e-05 6.237914e+03
[127] 1.000000e-06 6.237883e+03
[128] 1.000000e-07 6.237882e+03
>> plot3(U(:,1),U(:,2),U(:,3),'.')
>> axis equal
```





※ 分解不定性を解消していないので，結果は偶然