computer vision
五章 多視点幾何

自己紹介

・中川 高志
・２年前に５人で独立して会社立ち上げました

会社のサイト
http://www.cflat-inc.com/

会社のブログ（週一回更新）
http://cflat-inc.hatenablog.com/

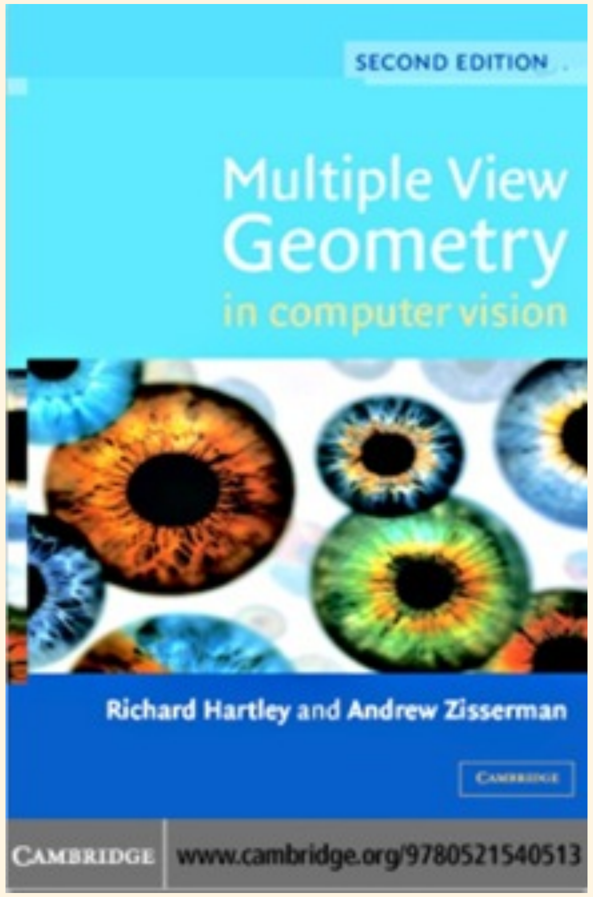
・趣味：娘の抱っこ
（先々週父になりました）



・courseraのmachine learningが今週から始まったので見始めました

目次

- エピポーラ幾何
- カメラと３D構造を使った計算
- 多視点による復元
- ステレオ画像



↑今回の内容は全て
ここに詳しく載ってます。

エピポーラ幾何

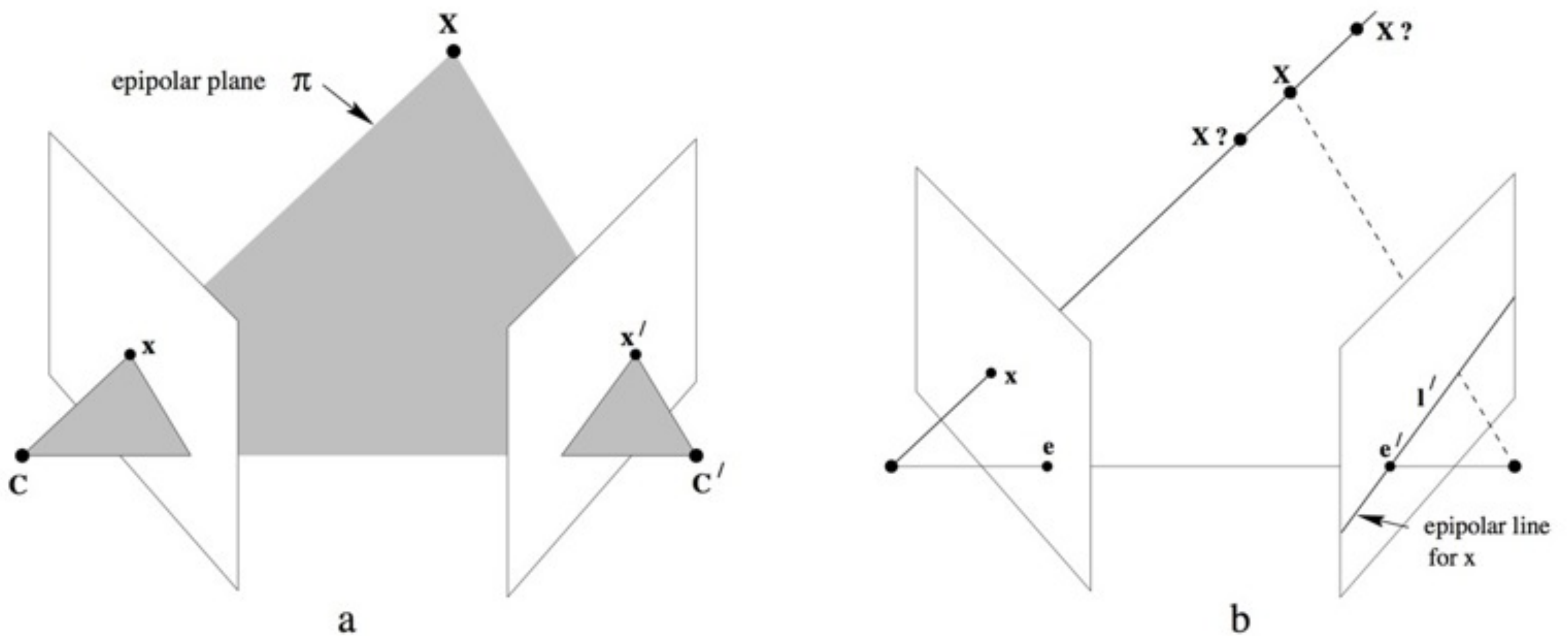
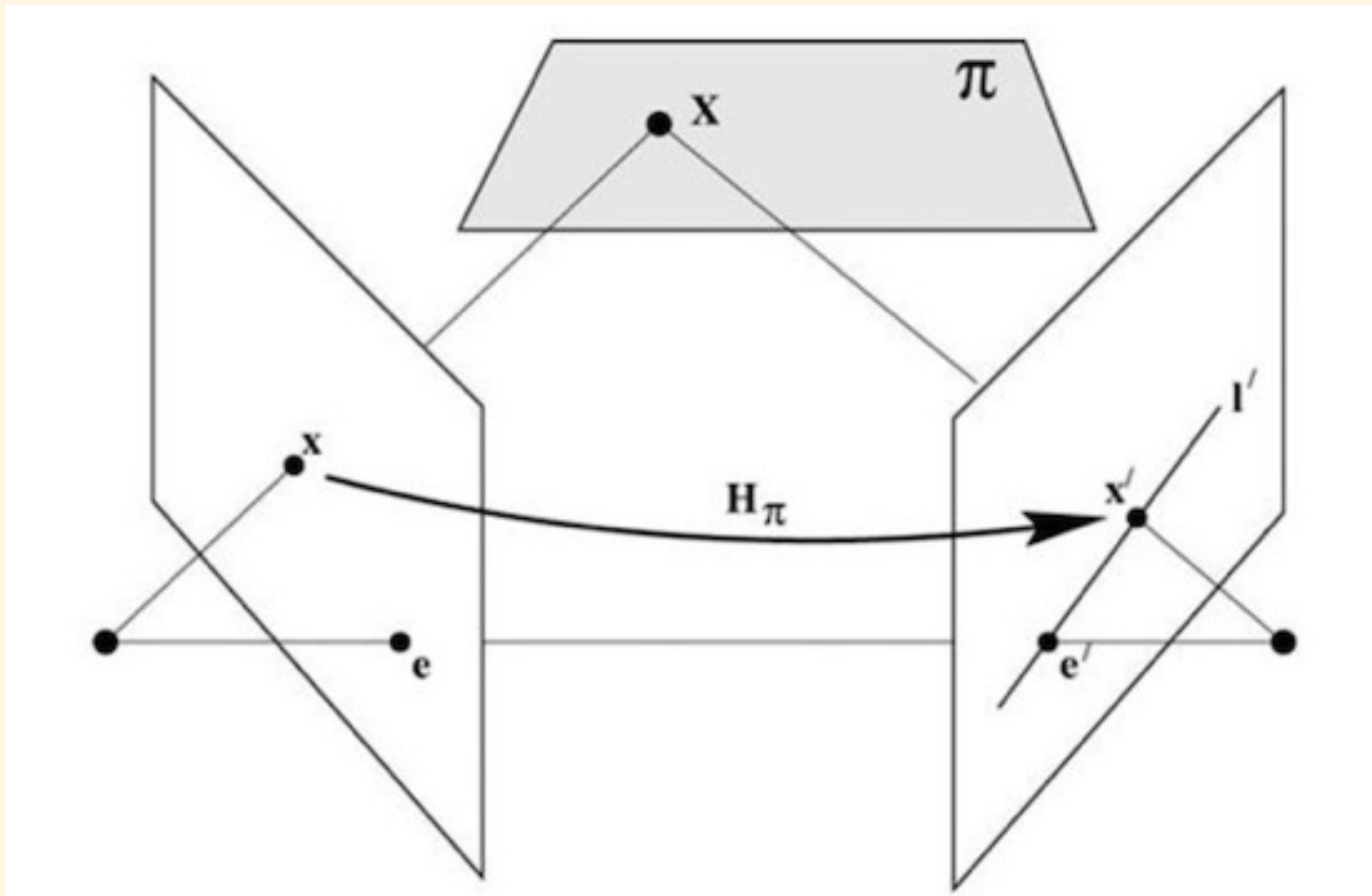


Fig. 9.1. **Point correspondence geometry.** *(a) The two cameras are indicated by their centres* **C** *and* **C′** *and image planes. The camera centres, 3-space point* **X**, *and its images* **x** *and* **x′** *lie in a common plane* π. *(b) An image point* **x** *back-projects to a ray in 3-space defined by the first camera centre,* **C**, *and* **x**. *This ray is imaged as a line* **l′** *in the second view. The 3-space point* **X** *which projects to* **x** *must lie on this ray, so the image of* **X** *in the second view must lie on* **l′**.

エピポーラ制約 $$x_2^T F x_1 = 0$$

エピポーラ幾何



$$x' = H_\pi x$$

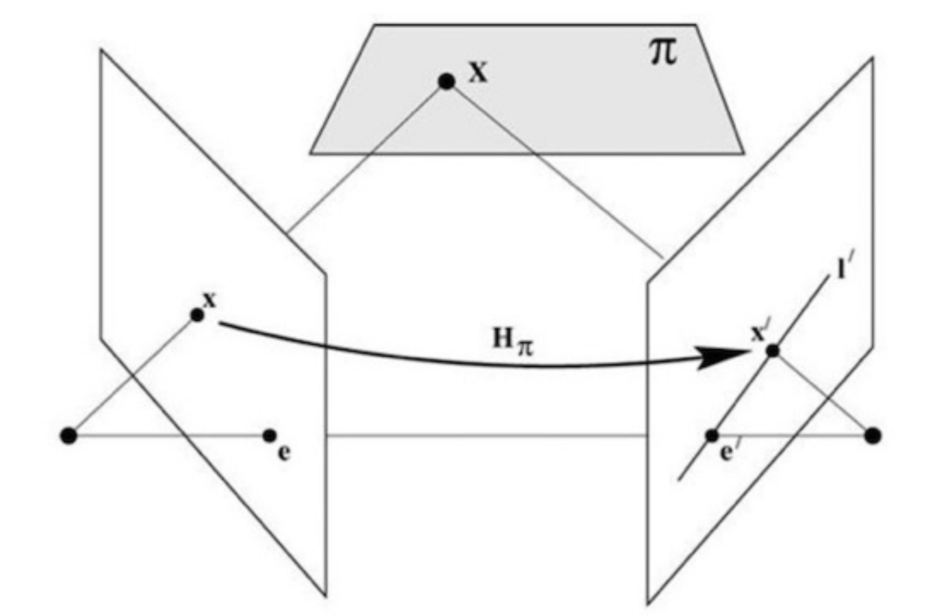
$$F = [e']_\times H_\pi$$

$$l' = e' \times x' = [e']_\times x' = \boxed{[e']_\times H_\pi} x = Fx$$

直線l'はxにある変換Fを施す事得られる

エピポーラ幾何

Fはランク２
※点から線への変換なのでランク２でなければならない



x’とl’は同一直線上であることから、

$$0 = x'^T l' = x'^T F x$$

また、以下の式が全てのxについて成り立つので

$$0 = e'^T l' = e'^T F x = \boxed{(e'^T F)}\, x$$

$$e'^T F = 0$$

$$x_2^T F x_1 = 0 \longrightarrow$$ F行列が分かれば、カメラ行列を求める事が出来る

エピポーラ幾何

エピポール線はエピ極で交わる

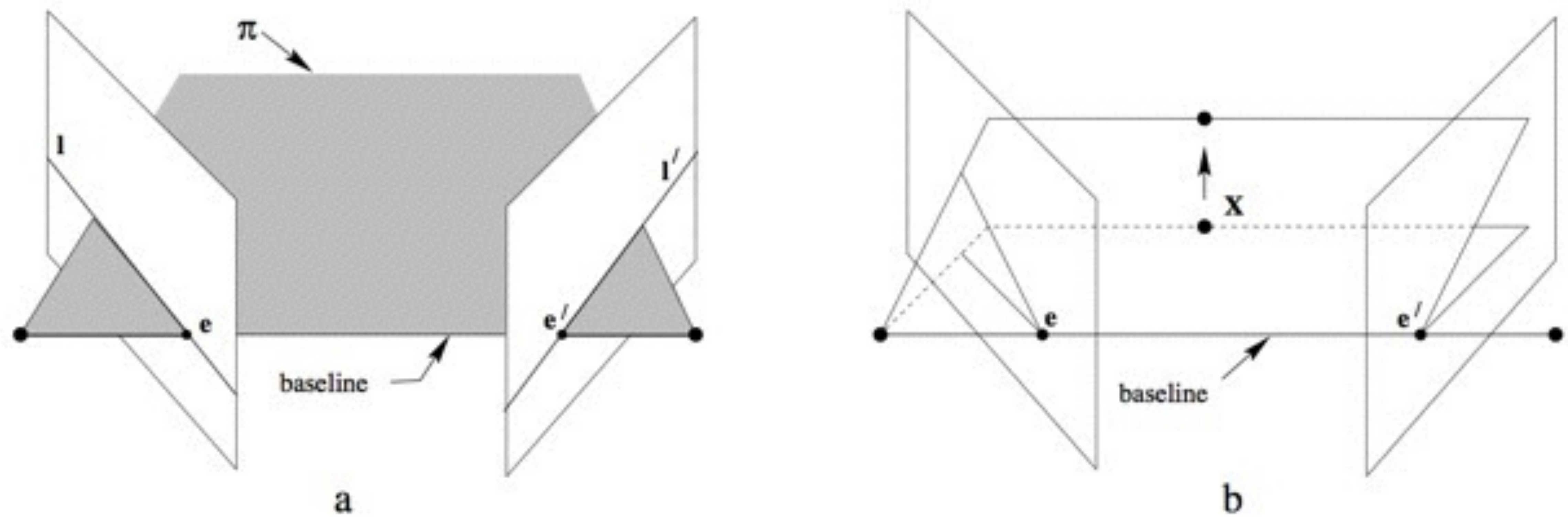
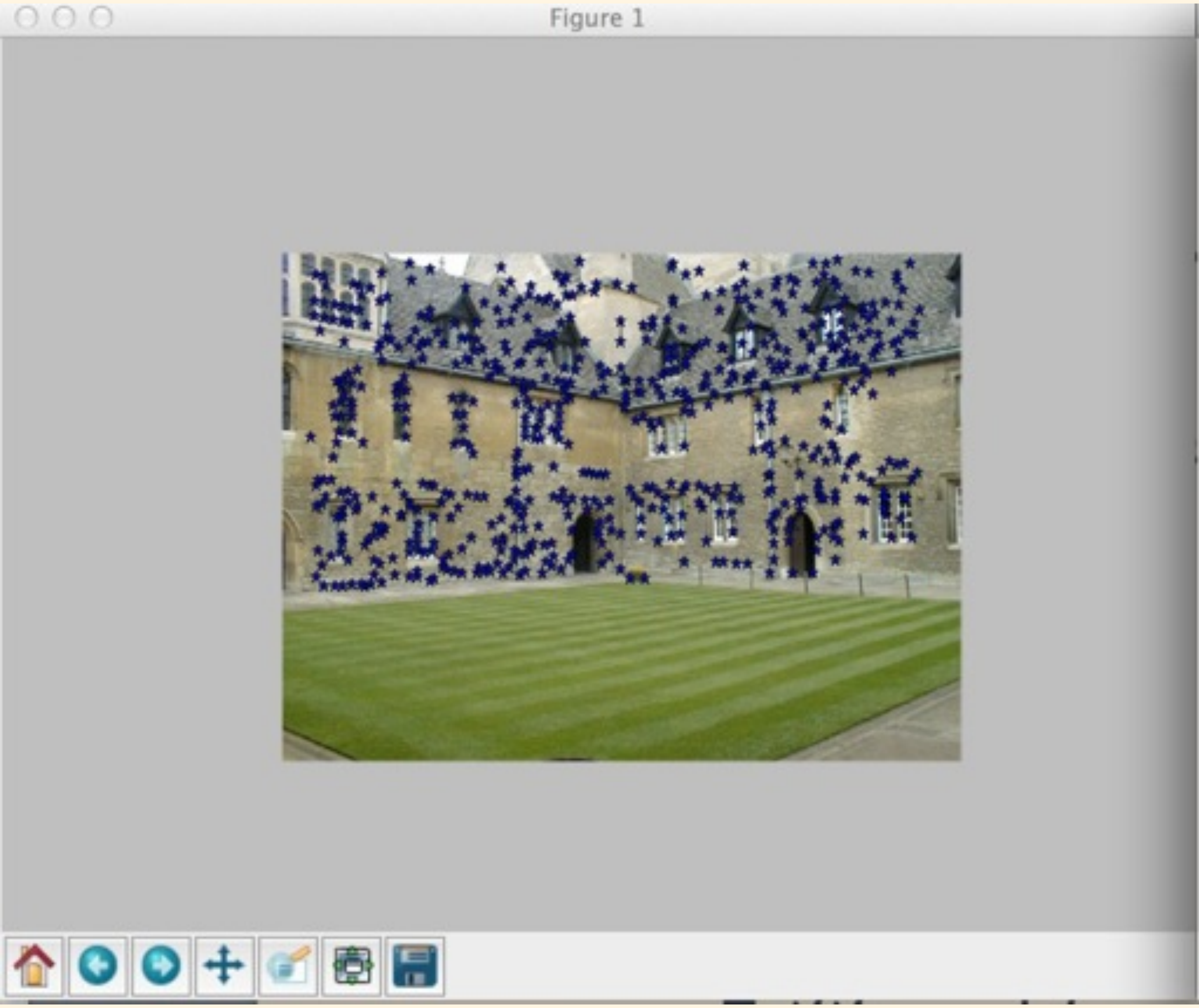


Fig. 9.2. **Epipolar geometry.** *(a) The camera baseline intersects each image plane at the epipoles* $\mathbf{e}$ *and* $\mathbf{e}'$. *Any plane* π *containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines* $\mathbf{l}$ *and* $\mathbf{l}'$. *(b) As the position of the 3D point* $\mathbf{X}$ *varies, the epipolar planes "rotate" about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines intersect at the epipole.*

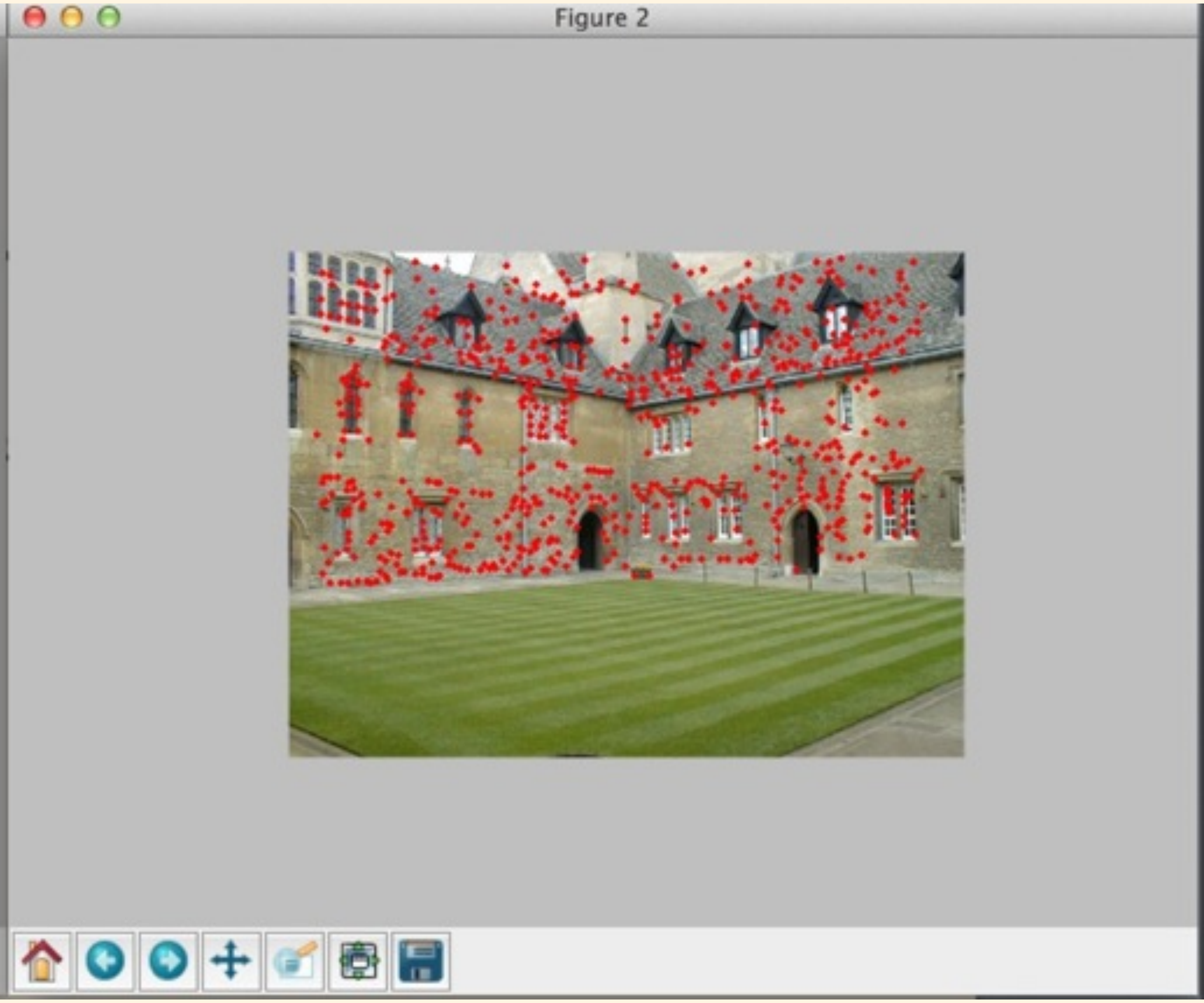
5.1.1.merton2d.pyを実行

用いるサンプルセットの中身を視覚的に確認

画像１の特徴点



3次元上の点群を画像１に射影



赤点の方が多い

∵ 画像２，画像３の特徴点から生成された点群もあるため

load_vggdata.pyの実行結果

list　array

```
ipdb> points2D
[array([[ 718.79 ,  676.587,  675.319, ...,  479.229,  475.771,  474.916], [ 263.096,  424.715,  394.656, ...,  441.16 ,  429.052,  419.315]]),
array([[ 689.792,  688.621,  951.71 , ...,  470.376,  466.728,  466.172],  [ 460.223,  428.371,  438.48 , ...,  475.694,  462.309,  452.315]]),
array([[ 755.058,  969.715,  748.762, ...,  501.171,  497.93 ,  497.132],  [ 302.399,  462.839,  125.158, ...,  501.598,  487.109,  478.137]])]
```

画像1のx座標　画像1のy座標　画像2のx座標　画像2のy座標　画像3のx座標　画像3のy座標

```
ipdb> points3D
array([[ 3.7585792 ,  1.0378863 ,  1.5606923 , ...,  2.7471349 , 0.73672803,  0.97395698],
       [-0.44845037, -0.54627892, -0.5211711 , ...,  1.4925585 , 0.6206597 ,  0.64403613],
       [ 4.4300374 ,  3.4601538 ,  3.4636809 , ...,  0.00678837,  -0.17002507, -0.23360821]])
```

x座標　y座標　z座標

```
ipdb> corr
array([[  0,  -1,   0], [  1,   0,  -1], [  2,   1,  -1], ..., [639, 665, 511], [640, 666, 512],  [641, 667, 513]])
```

```
ipdb> P
[<camera.Camera object at 0x105a14f90>, <camera.Camera object at 0x105a14190>, <camera.Camera object at 0x105a263d0>]
```

5.1.1.merton2d.pyの説明

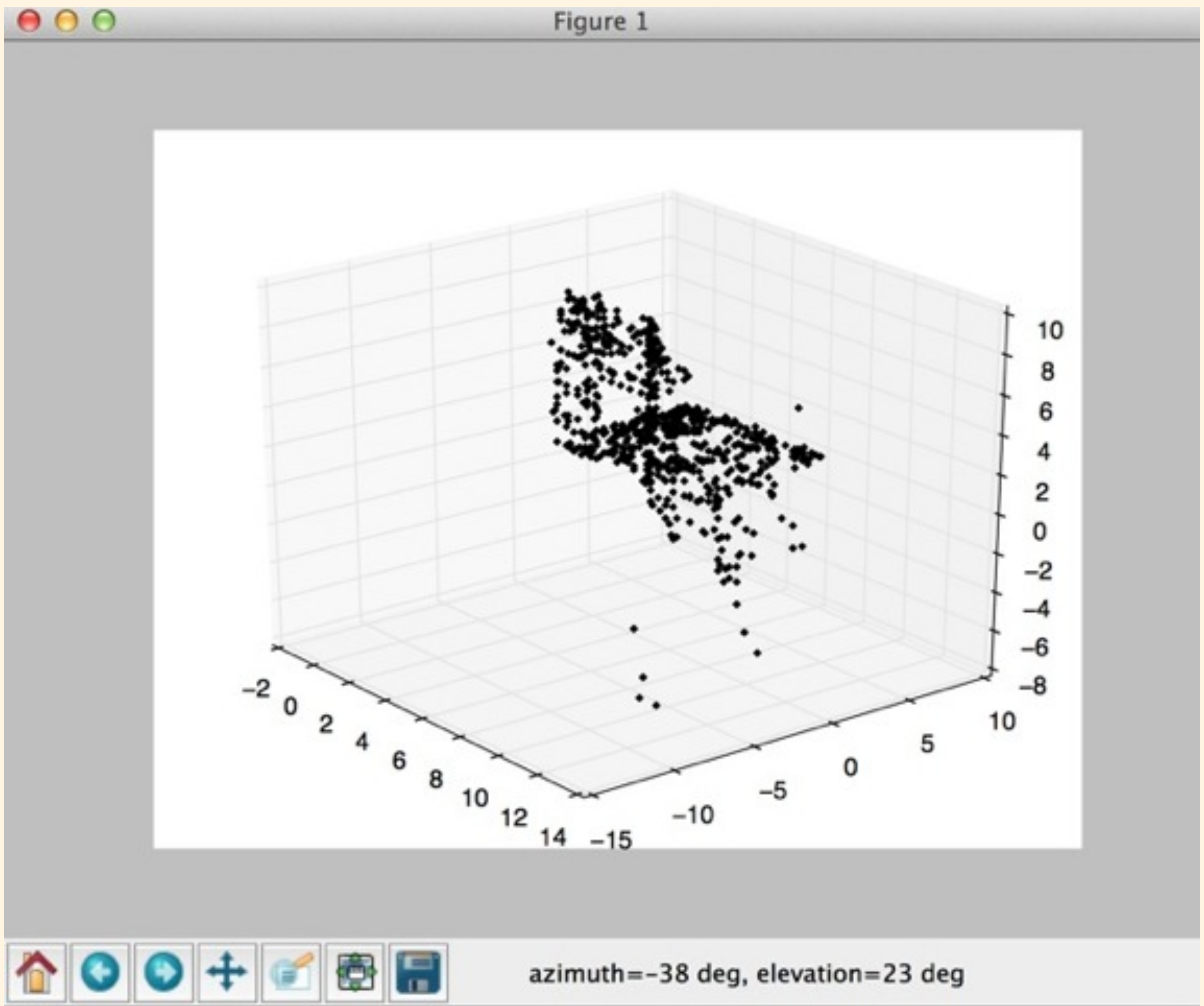
x = P[0].project(X)

$$x = \underbrace{\begin{bmatrix} p1 & p2 & p3 & p4 \\ p5 & p6 & p7 & p8 \\ p9 & p10 & p11 & p12 \end{bmatrix}}_{\text{カメラ行列}} \underbrace{\begin{bmatrix} x1 & x2 & x3 & \ldots \\ y1 & y2 & y3 & \ldots \\ z1 & y3 & z3 & \ldots \\ 1 & 1 & 1 & \ldots \end{bmatrix}}_{\text{各列が３次元上の点（同時座標）}}$$

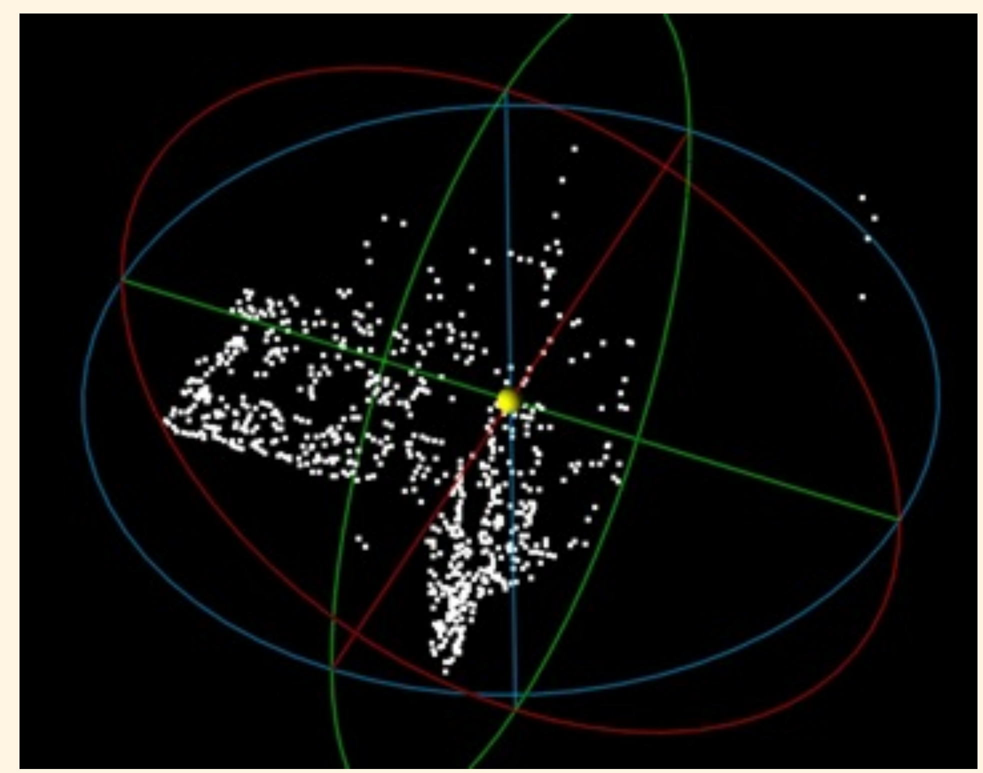
２次元上の点（同時座標）

Matplotlibによる3Dデータの描画

- 5.1.2.merton3d.pyを実行



CloudCompareでp3dを直接見た場合



F行列の計算 - 8点法

エピポーラ制約

$$x'^T F x = 0$$

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x'f_{11} + y'f_{21} + f_{31} & x'f_{12} + y'f_{22} + f_{32} & x'f_{13} + y'f_{23} + f_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

F行列の計算 - 8点法

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

$$\begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

F行列の計算 - 8点法

９成分だがスケールは任意なので、８つの式があれば良い

各対応点の列

$$
\underbrace{\begin{bmatrix}
x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\
x'_2x_2 & x'_2y_2 & x'_2 & y'_2x_2 & y'_2y_2 & y'_2 & x_2 & y_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1
\end{bmatrix}}_{\text{A}}
\underbrace{\begin{bmatrix}
f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33}
\end{bmatrix}}_{\text{f}} = 0
$$

$Af = 0$ を求める

→ $\|f\| = 1$ という条件で $\|Af\|$ を最小化

特異値分解を用いると解ける

$A = UDV^T$ と特異値分解すると $f = V$ の最後の列が解となる

今回は更にFがランク２という制約を利用する

$$F = U \begin{bmatrix} r & & \\ & s & \\ & & t \end{bmatrix} V^T$$

(r > s > t)

$$F' = U \begin{bmatrix} r & & \\ & s & \\ & & 0 \end{bmatrix} V^T$$

t=0とすることでランク２

更にこうする事でフルベニウスノルムが最小となる

$$\|F - F'\| = \left\|UDV^T - UD'V^T\right\| = \|D - D'\| = t$$

Fが特異行列（→ランク２）でないとエピポール線がエピ極で一点に交わらない

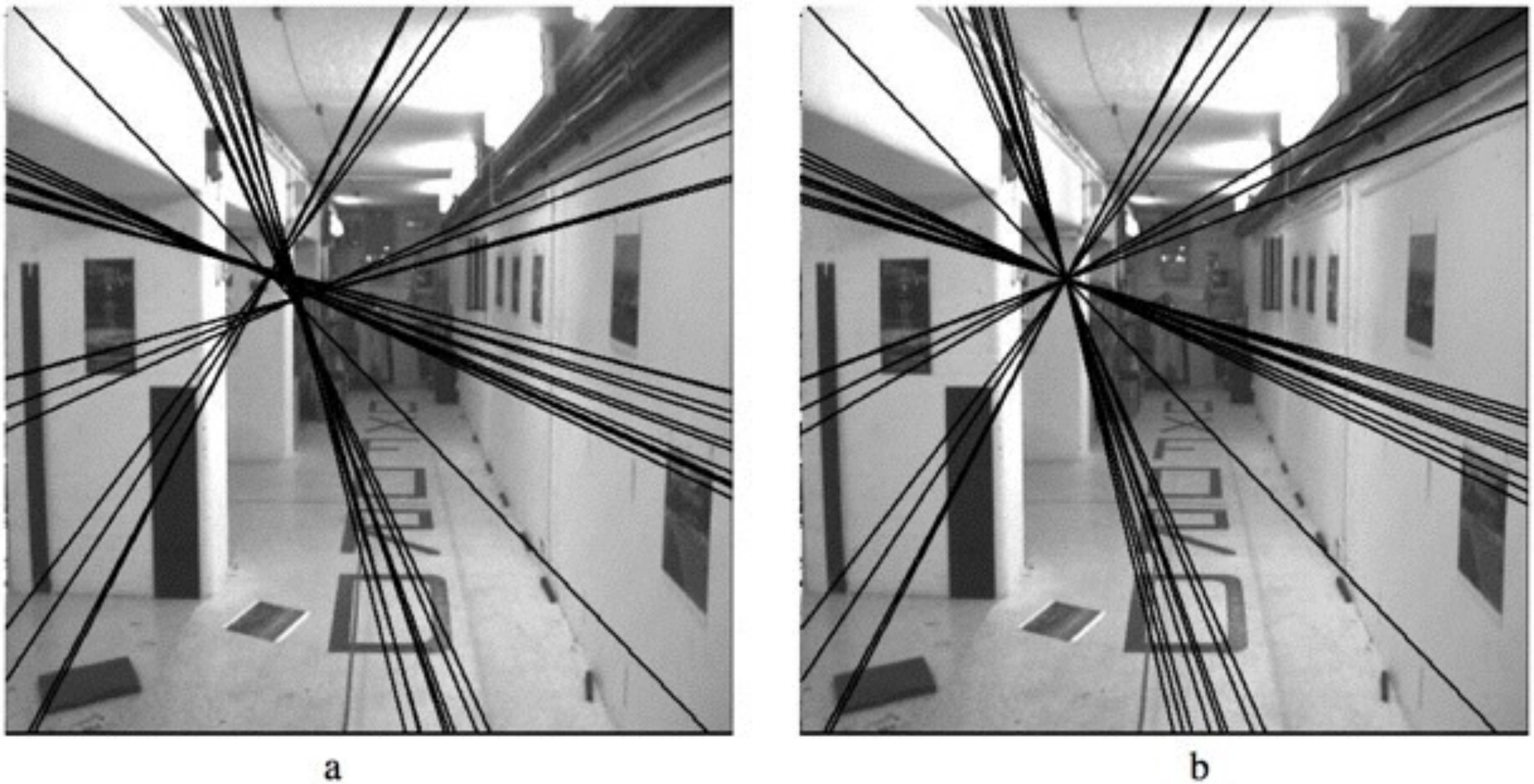


Fig. 11.1. **Epipolar lines.** *(a) the effect of a non-singular fundamental matrix. Epipolar lines computed as* $\mathbf{l}' = \mathtt{F}\mathbf{x}$ *for varying* $\mathbf{x}$ *do not meet in a common epipole. (b) the effect of enforcing singularity using the SVD method described here.*

エピ極とエピポーラ線

$$Fe = 0$$

Fの右側のゼロベクトルに対応するエピ極

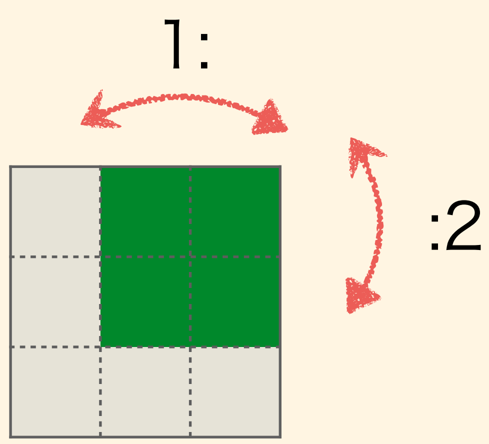
$$e^{'T}F = F^{T}e^{'} = 0$$

Fの左側のゼロベクトルに対応するエピ極

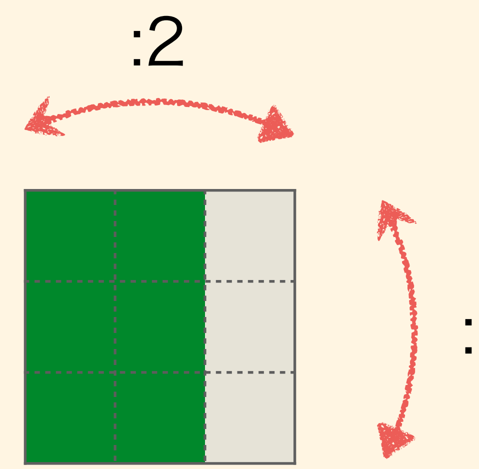
$$Ax = 0$$

の形なので特異値分解を利用してxを求める事が出来る

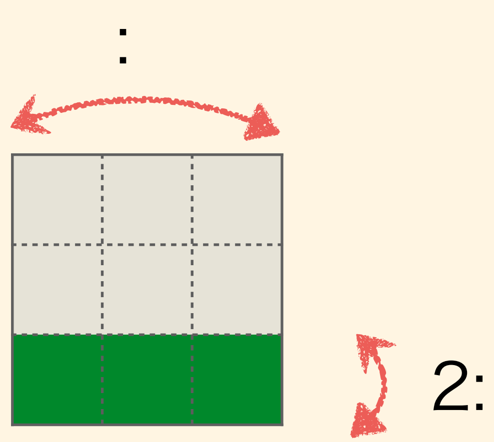
2次元配列のスライシング



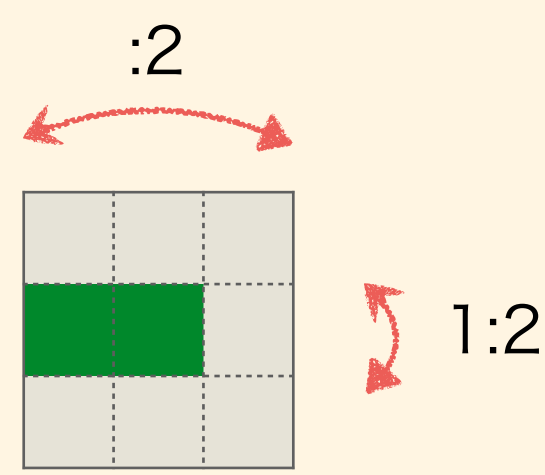
arr[:2, 1:]



arr[:, :2]



arr[2:, :]



arr[1:2:, :2]

5.1.4.sfm.py

```
# 最初の2枚の画像の点のインデクス番号
ndx = (corr[:,0]>=0) & (corr[:,1]>=0)# corr:対応関係 *でないものを選択
```

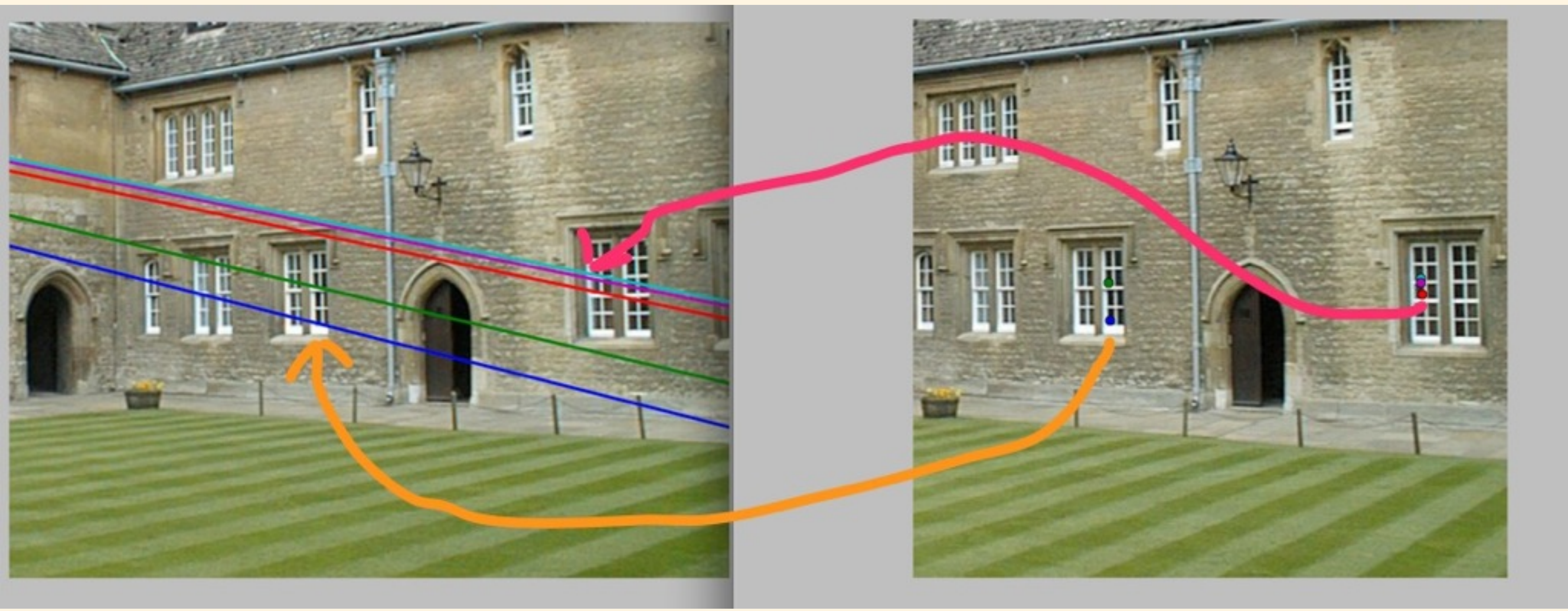
```
ipdb> ndx
array([False,  True,  True,  True,  True, False,  True, False,  True,
        True, False, False,  True,  True, False,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True,  True, False,  True,
```

```
# 座標値を取得し、同次座標系にする
x1 = points2D[0][:,corr[ndx,0]]
x1 = vstack( (x1,ones(x1.shape[1])) )
```

```
ipdb> points2D[0]
array([[ 718.79 ,  676.587,  675.319, ...,  479.229,  475.771,  474.916],
       [ 263.096,  424.715,  394.656, ...,  441.16 ,  429.052,  419.315]])
```

```
ipdb> corr[ndx,0]
array([  1,   2,   3,   4,   6,   8,   9,  12,  13,  15,  16,  17,  18,
        19,  21,  22,  23,  24,  25,  27,  28,  29,  30,  31,  32,  33,
        35,  36,  37,  38,  41,  43,  45,  46,  48,  49,  51,  52,  53,
```

5.1.4.sfm.py実行結果



三角測量
（カメラ行列から３次元の点を計算する）

$$\text{6列}\;\underbrace{\begin{bmatrix} P_1 & -x_1 & 0 \\ P_2 & 0 & -x_2 \end{bmatrix}}_{\text{6行}} \begin{bmatrix} X \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = 0$$

→ $Ax = 0$ の形

→ 特異値分解

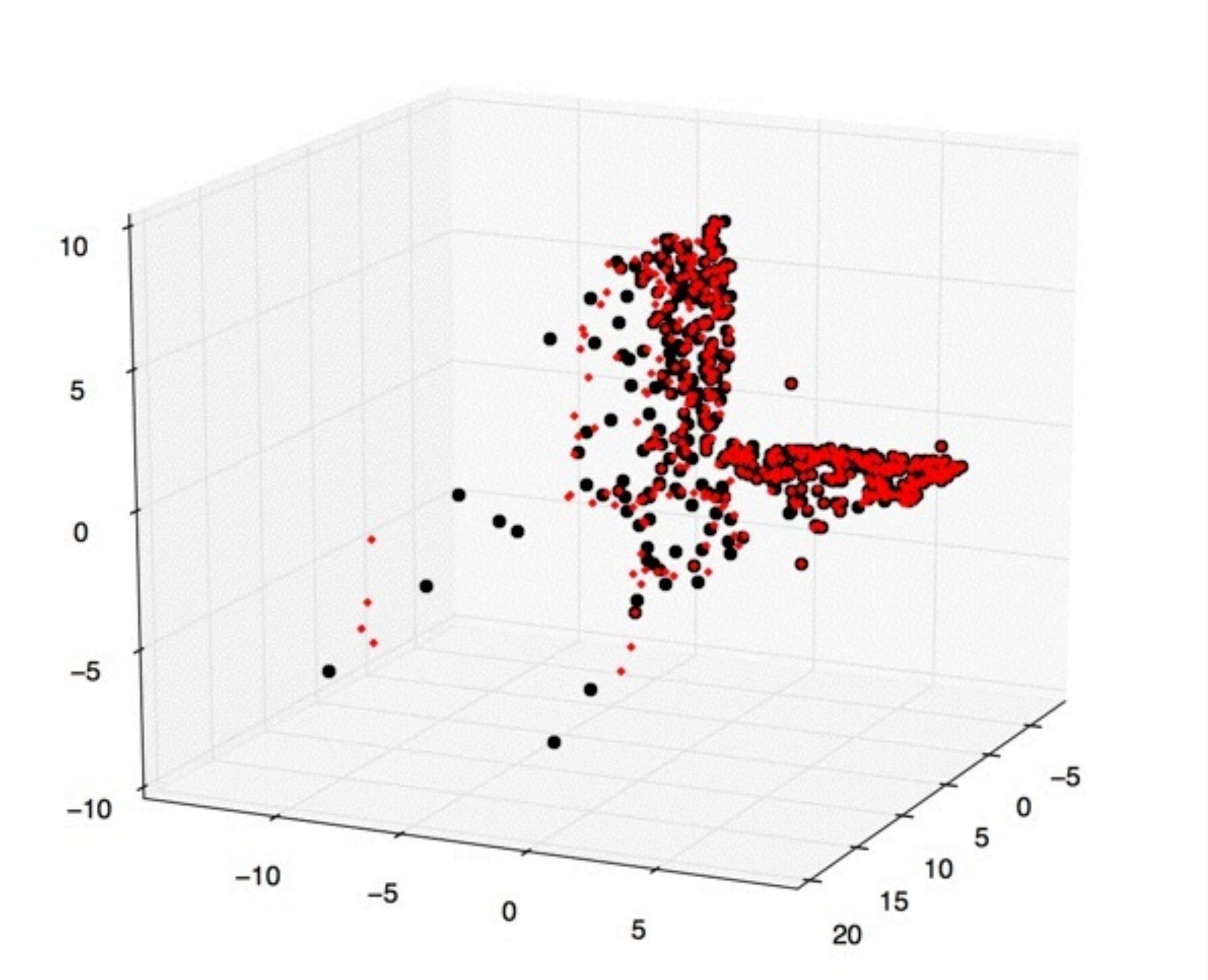
$$\lambda_1 x_1 = P_1 X$$

$$\lambda_2 x_2 = P_2 X$$

$$P_1 X - \lambda_1 x_1 + 0 \cdot x_2 = 0$$

$$P_2 X - 0 \cdot x_1 + \lambda_2 \cdot x_2 = 0$$

5.2.1.triangulate.pyを実行



赤丸：正解の点、黒丸：推定した点

3Dの点群からカメラ行列を計算

$$\lambda_1 x_1 = P X_1$$

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} X_1 - \lambda_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0$$

$$P_1 X_1 = \begin{bmatrix} P_{11} & P_{12} & P_{13} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \end{bmatrix} = X_1^T P_1^T$$

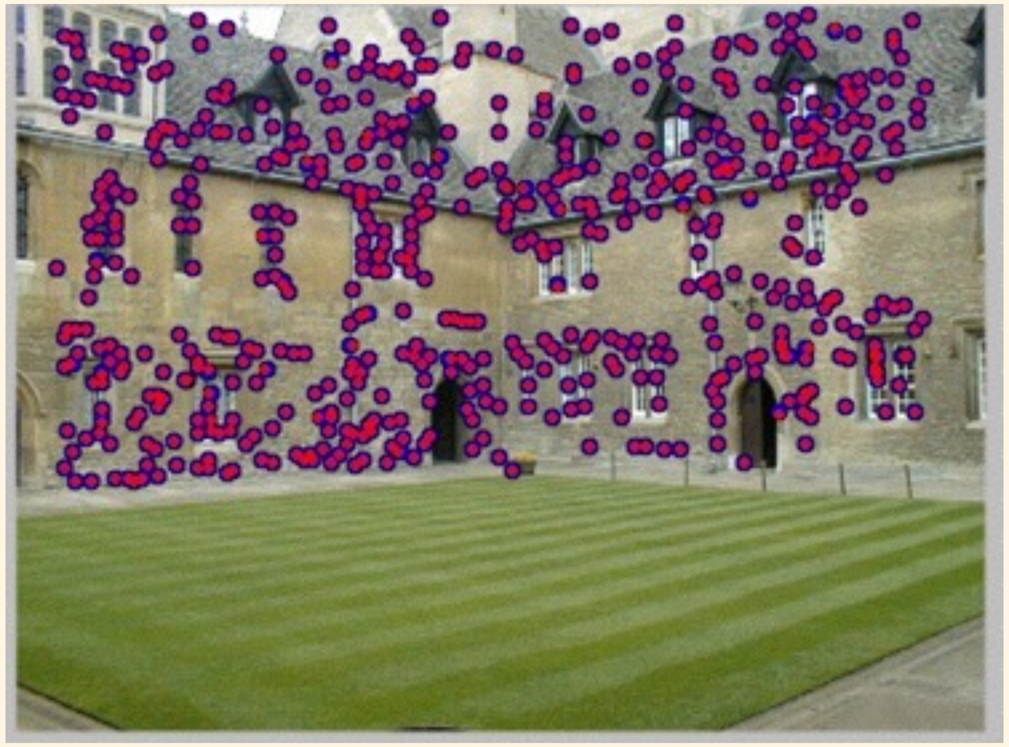
$$\begin{bmatrix} X_1^T & 0 & 0 & -x_1 \\ 0 & X_1^T & 0 & -y_1 \\ 0 & 0 & X_1^T & -1 \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \\ \lambda_1 \end{bmatrix} = 0$$

→ $Ax = 0$ の形

→ 特異値分解

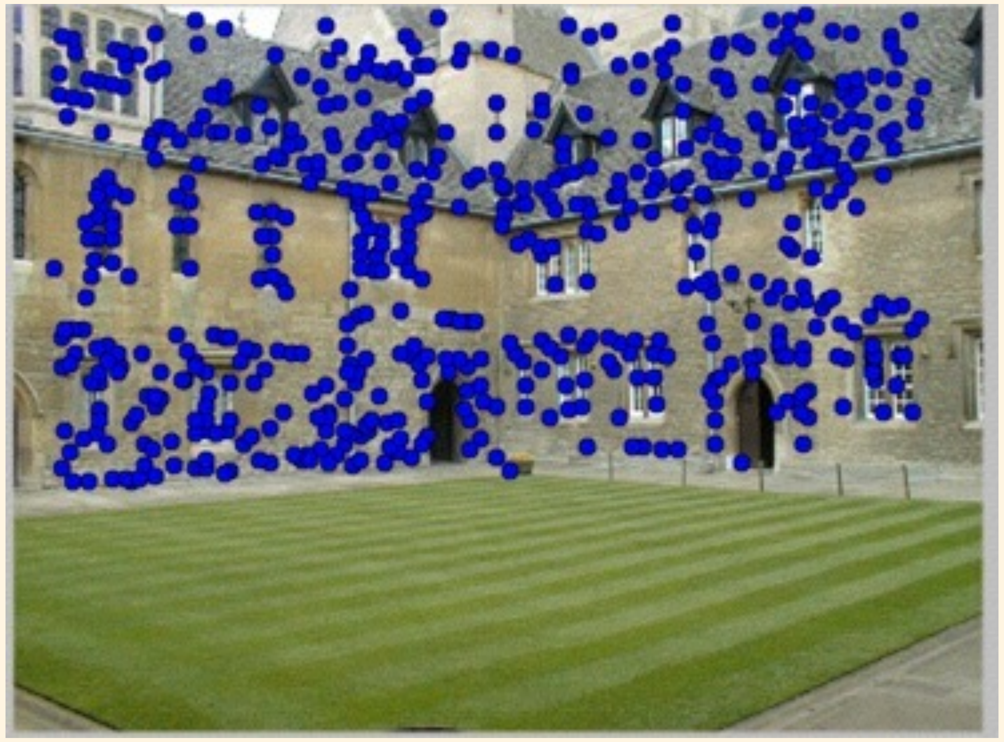
5.2.2.compute.pyを実行

赤＋青



赤：推定結果、青：オリジナル

青のみ



基礎行列からカメラ行列を計算する

内部パラメータが不明な場合
→復元は射影変換を求めるところまで

$$P_2 = [S_e F \mid e]$$

交代行列

$$A^T = -A$$

$$A = \begin{bmatrix} 0 & -a_2 & a_1 \\ a_2 & 0 & -a_0 \\ -a_1 & a_0- & 0 \end{bmatrix}$$

基礎行列からカメラ行列を計算する

内部パラメータが既知の場合
→スケール以外を正確に復元出来る

$$P = K\left[R \mid t\right]$$
$$x = PX = K\left[R \mid t\right]X$$
$$x_K = K^{-1}x = K^{-1}K\left[R \mid t\right]X = \left[R \mid t\right]X$$
$$x_{K_2}^T E x_{K_1} = 0$$

基本行列の4つの解

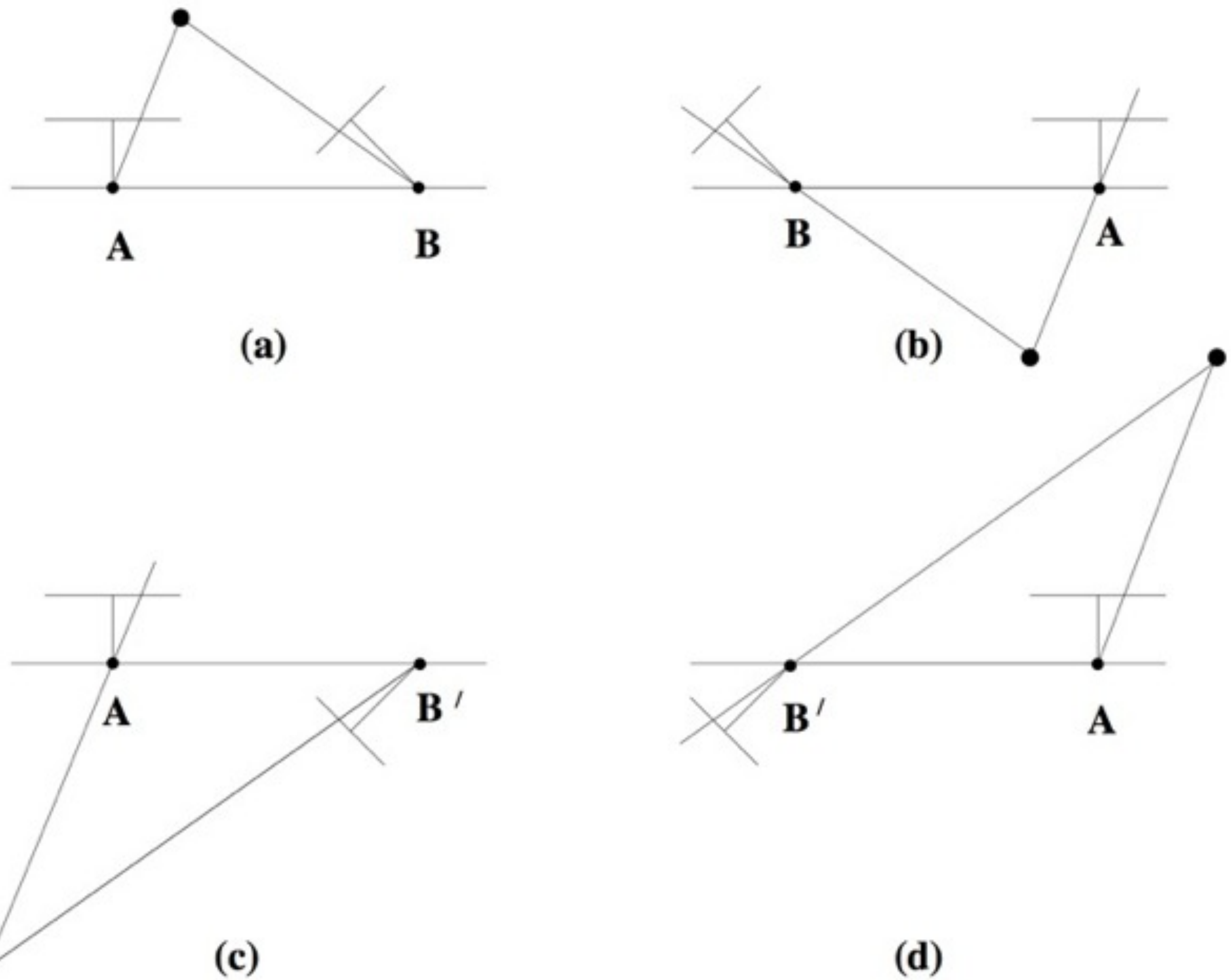


Fig. 9.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates* 180° *about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

ロバストな基礎行列推定

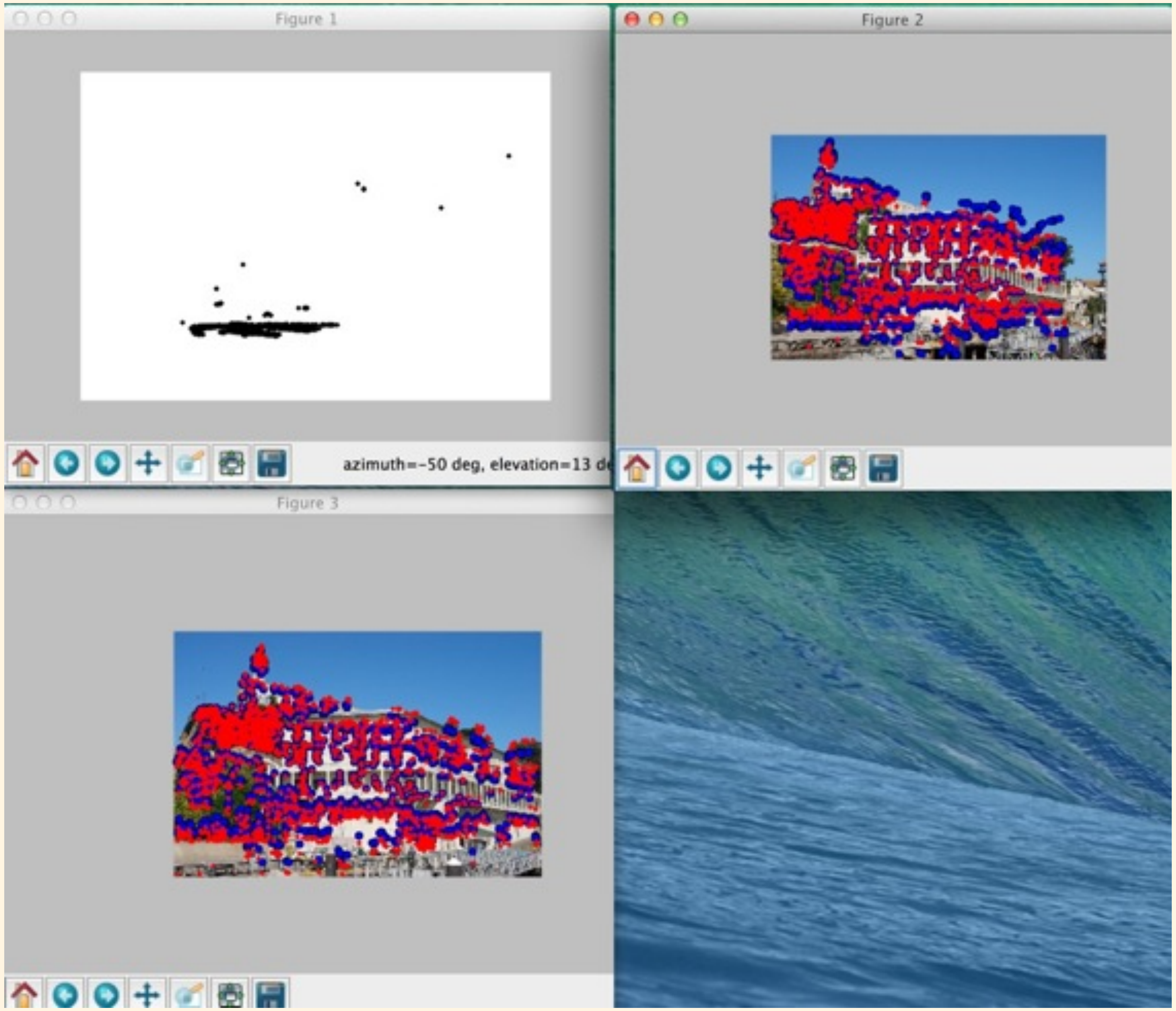
sampson距離

$$\sum \frac{\left(x_i^{\prime T} F x_i\right)^2}{(Fx_i)_1^2 + (Fx_i)_2^2 + \left(F^T x_i'\right)_1^2 + \left(F^T x_i'\right)_2^2}$$

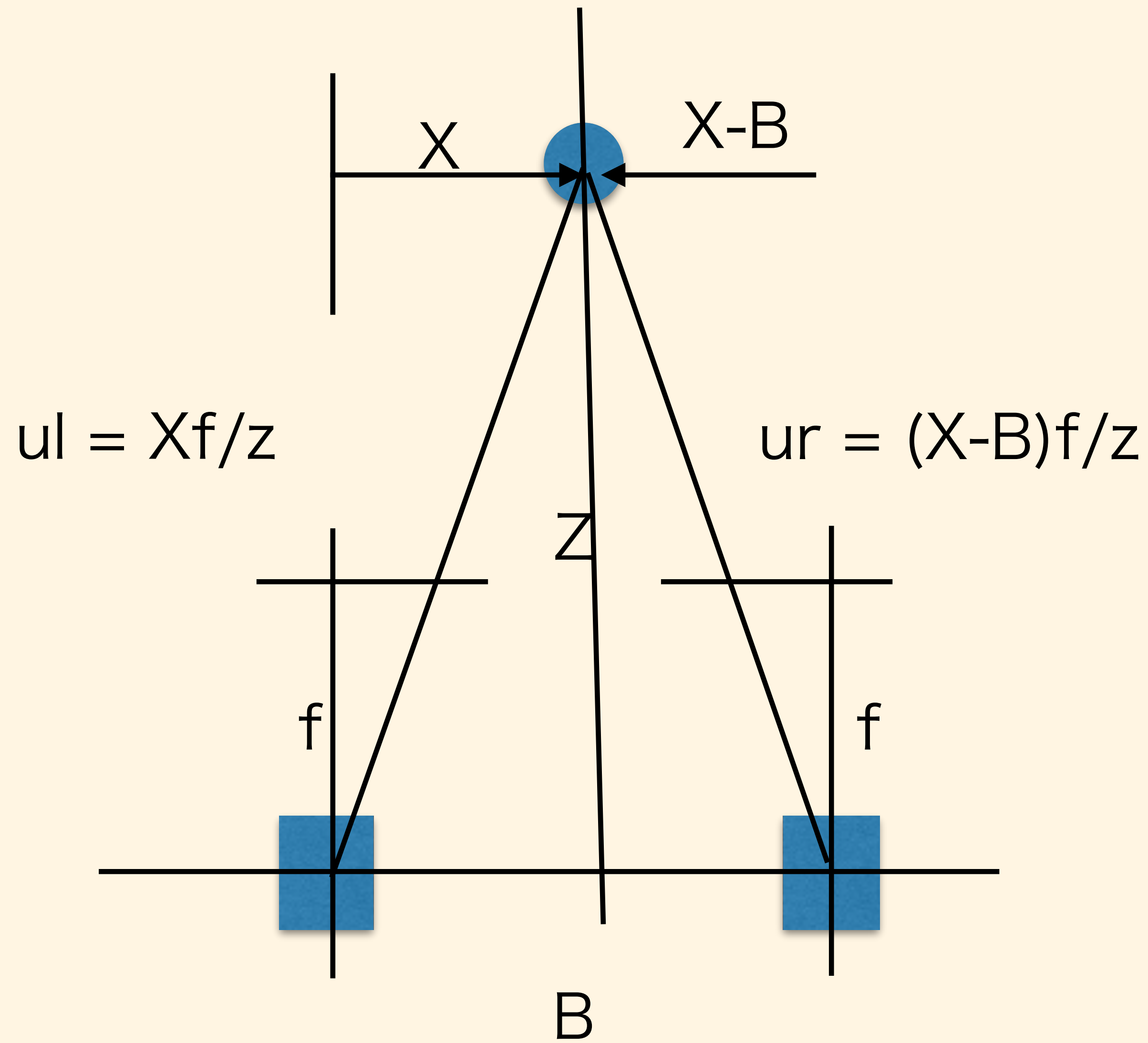
画像の正規化

画像の点群の重心が原点になるように移動し、
分散が１になるように正規化

5.3.2.recons3d.py実行結果



ステレオ原理

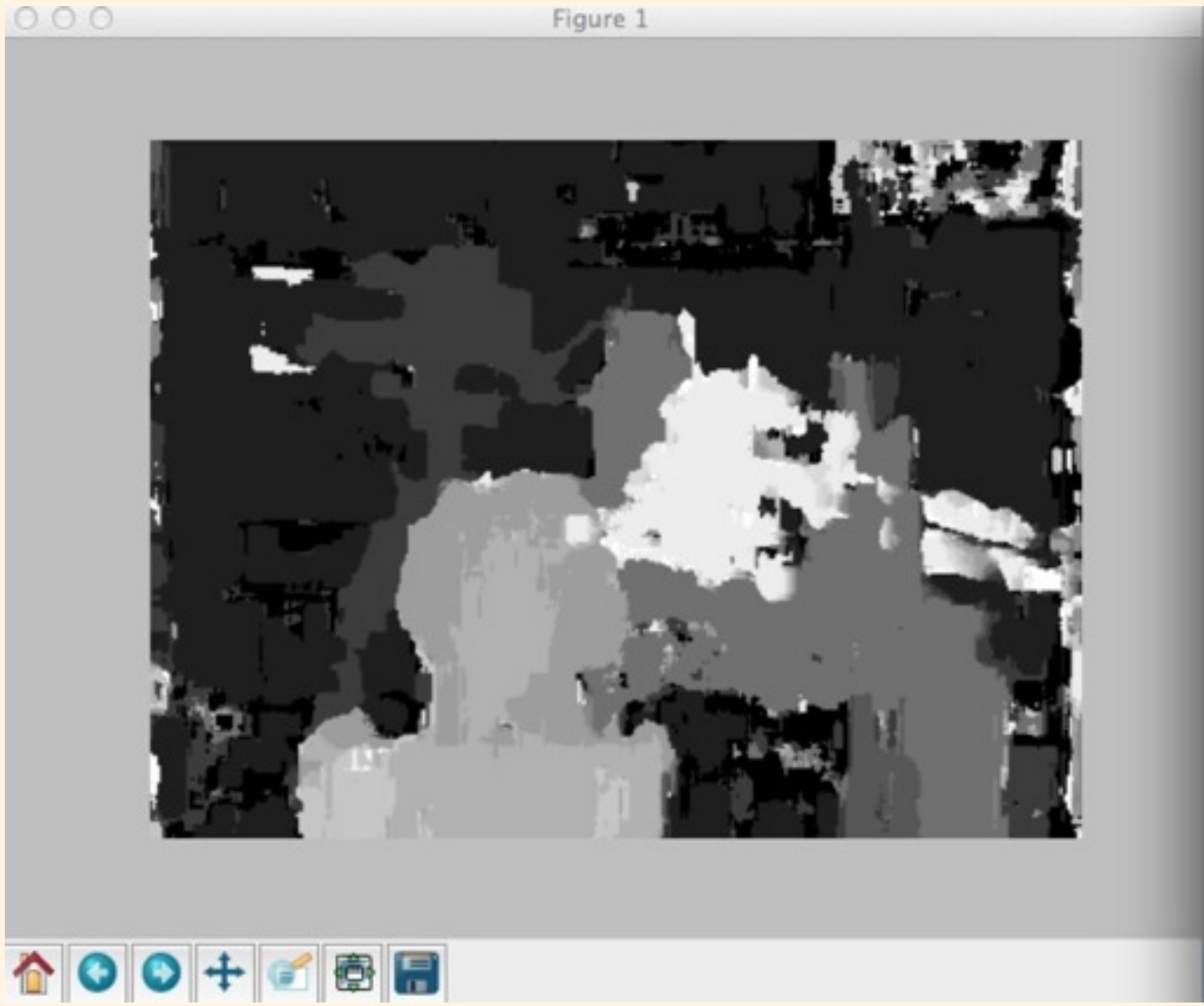


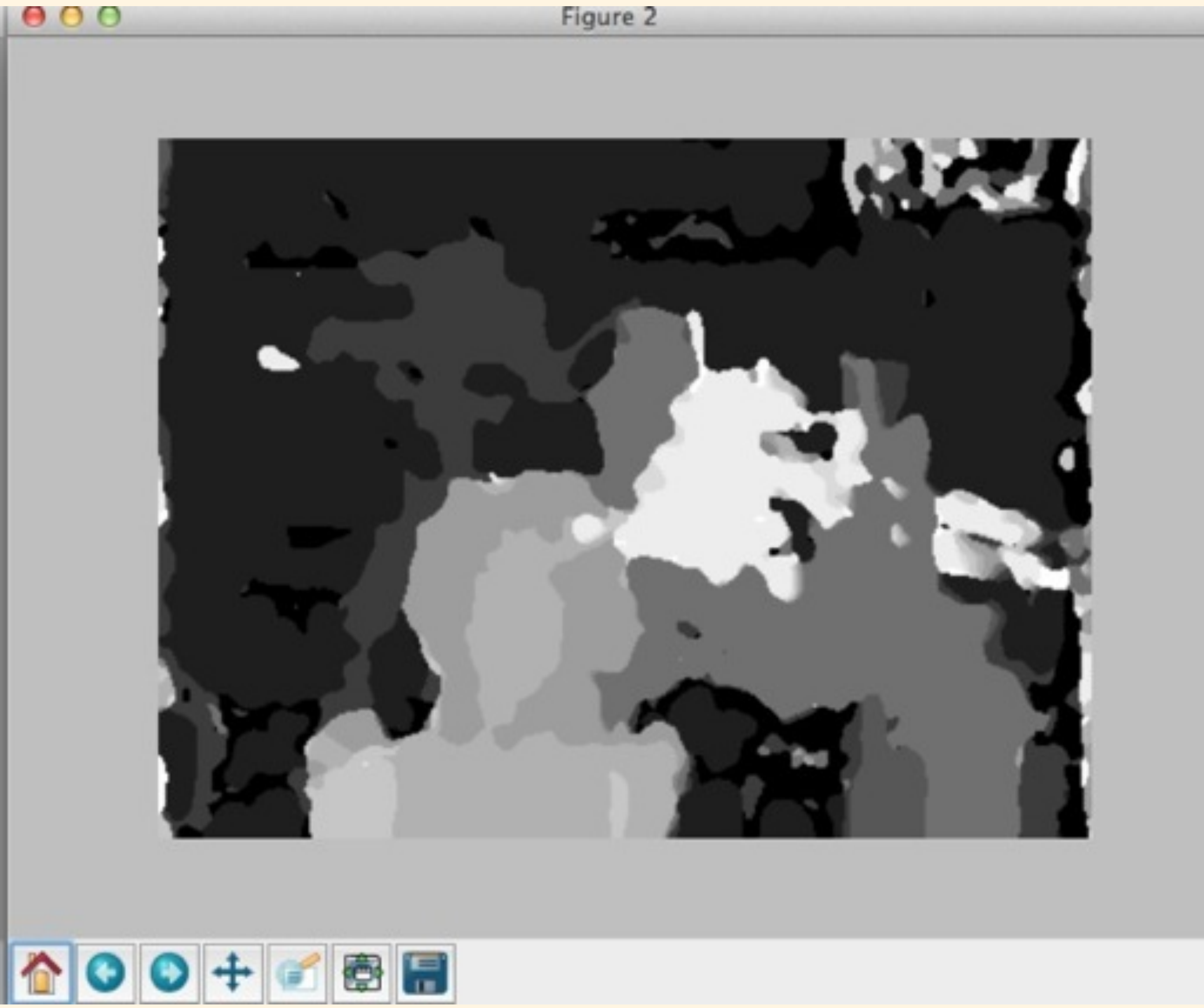
ul-ur=Bf/z

NCCとZNCC

http://imagingsolution.blog107.fc2.com/blog-entry-186.html

5.4.1.stereo.py実行結果





OpenMVGを使ったstructure from motionの紹介

http://cflat-inc.hatenablog.com/entry/2013/10/21/214859