静岡Developers勉強会
コンピュータビジョン
vol.4

実践コンピュータビジョン

第４章
カメラモデルと
拡張現実感

今回の概要

- ピンホールカメラモデル
- カメラキャリブレーション
- 平面とマーカーを使った姿勢推定
- 拡張現実感

ピンホール
カメラモデル

身近なもの

目、カメラ

原理

投影

投影の種類

平行投影

(正射影)

特徴

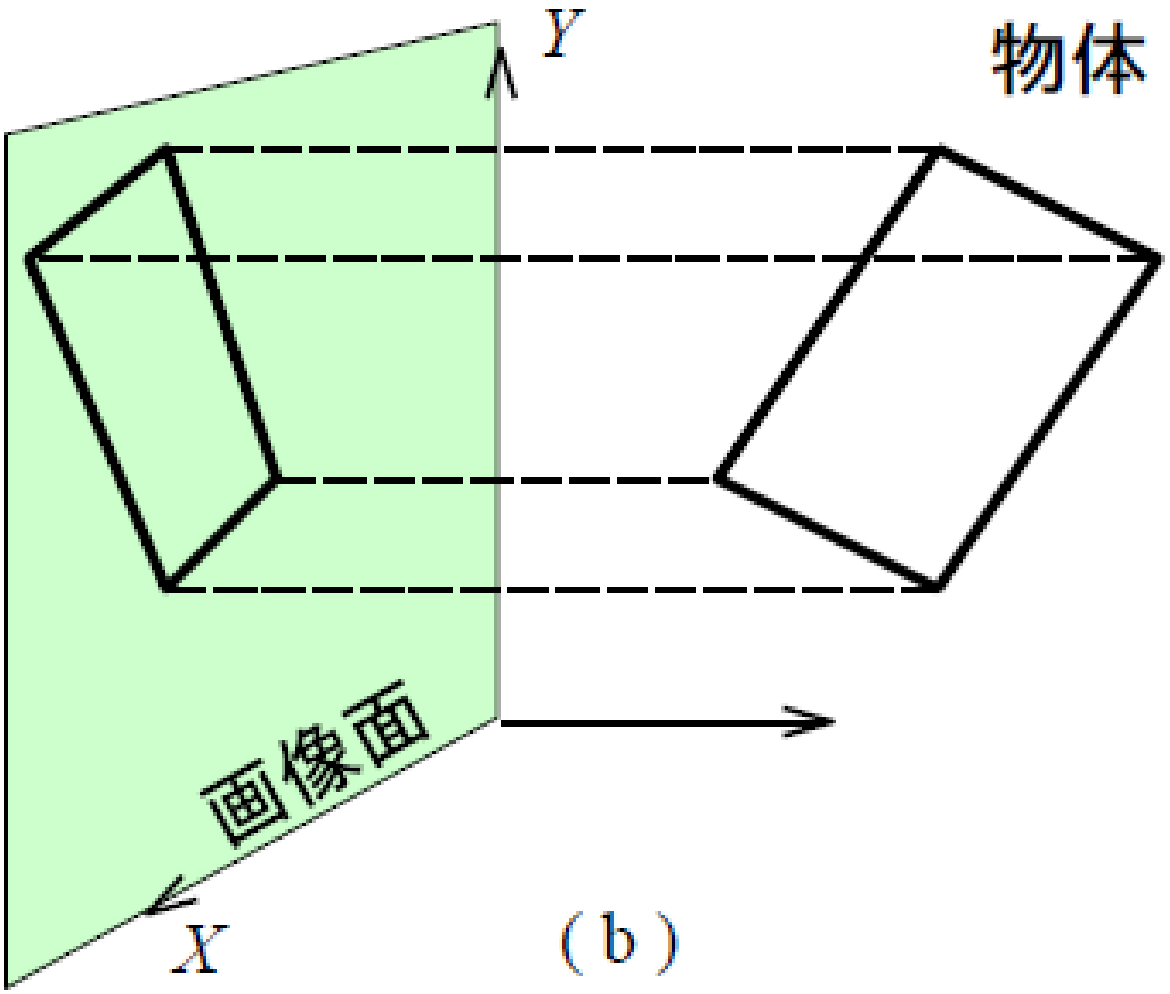
距離が離れていても座標が変わらない

物体と投影面の座標の関係

$$x' = x, y' = y$$

x', y' :スクリーン上の座標

x, y :実座標



平行投影

平行投影はカメラモデルとは関係がない

カメラモデルの射影方法

透視投影

(中心射影)

特徴

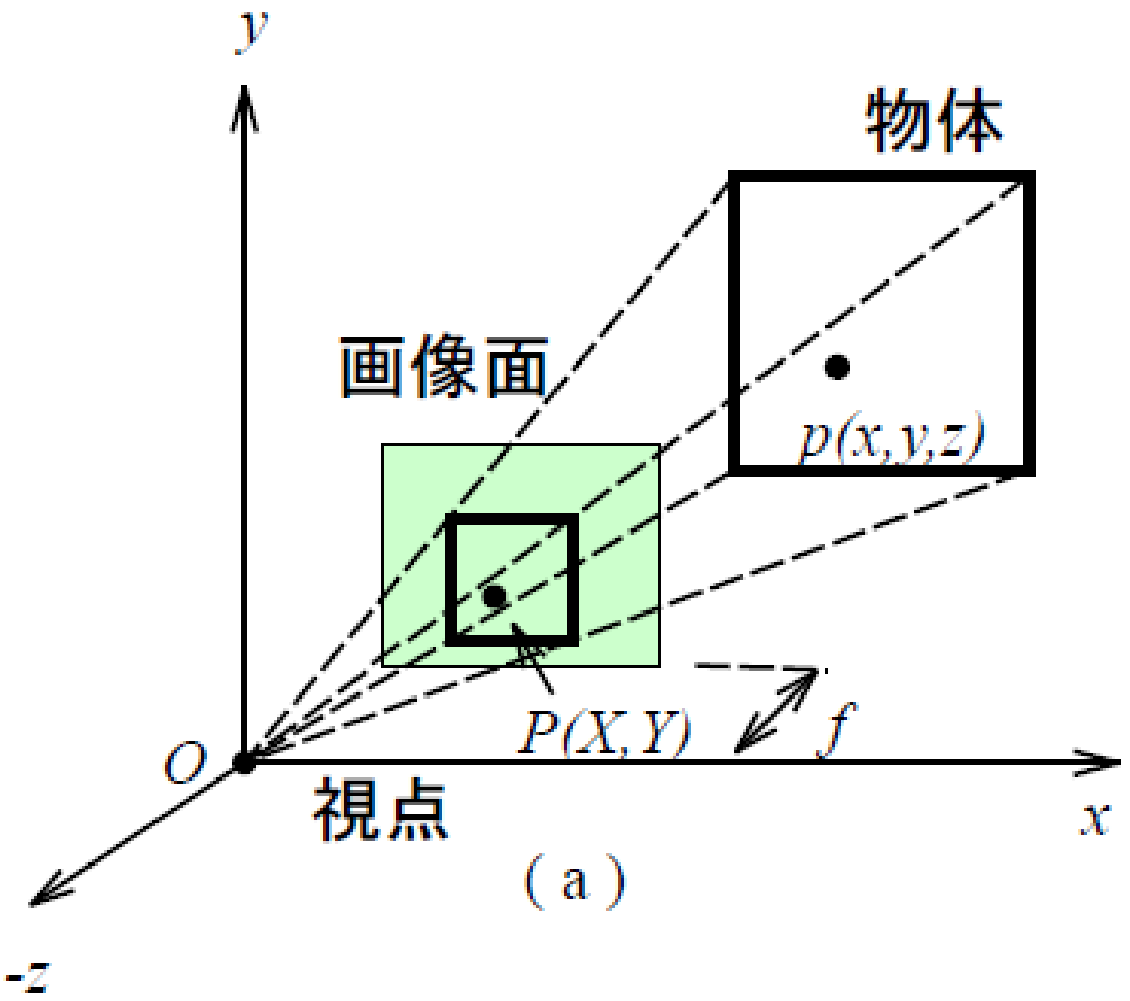
近くのものは大きく、遠くのものは小さく映る

物体と投影面の座標の関係

$$x' = \frac{x}{z}, y' = \frac{y}{z}$$

x', y' :スクリーン上の座標

x, y, z :実座標



(a)

透視投影

問題点

カメラのレンズ中心が座標の原点になければと成り立たない。

焦点距離は1に正規化されている

どうすればレンズ中心と焦点距離が任意の値でも平面に投影することが出来るのか

カメラ行列

$$\lambda x = PX$$

x 画像上の座標

λ 奥行の逆数

P カメラ行列

X 被写体の座標

もう少し
詳しく

$$\lambda x = K \left[R | t \right] X$$

K キャリブレーション行列

R カメラの向き

t カメラの位置

もっと詳しく

キャリブレーション行列の内容

$$K = \begin{pmatrix} \alpha f & s & C_x \\ 0 & f & C_y \\ 0 & 0 & 1 \end{pmatrix}$$

α アスペクト比

s スキュー

f 焦点距離

C_x, C_y 画像中心

アスペクト比は焦点距離の縦横の比である

アスペクト比が1でない場合、別々
焦点距離の値を使用する

$$f = f_y, \alpha f = f_x$$

$\alpha = 1$のとき、

縦横の焦点距離が同じ場合

$$f = f_y = f_x$$

焦点距離はカメラのレンズによって決まる

基本的にアスペクト比の値は1と仮定してよい

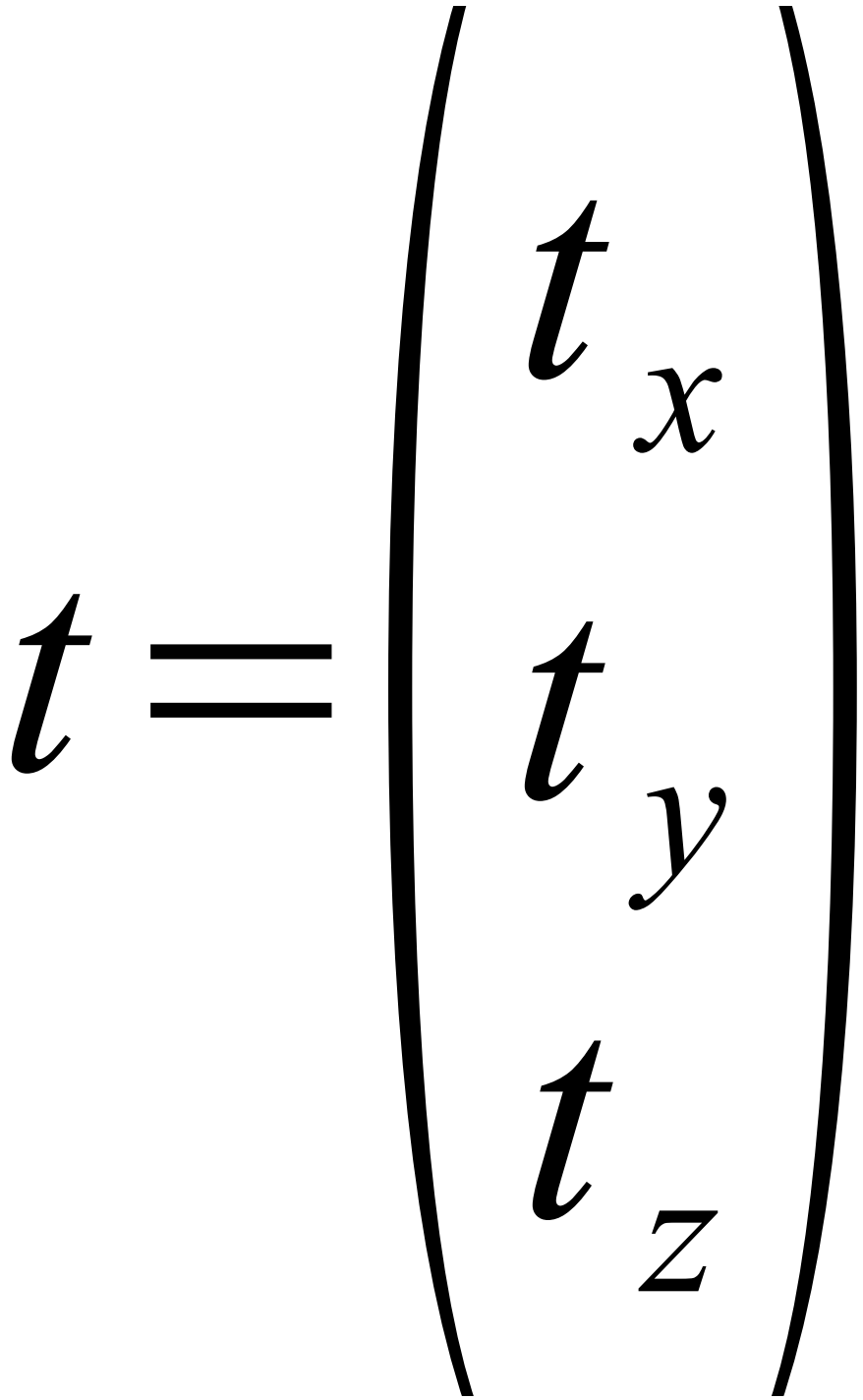
スキューの概要については3章参照。

撮像素子が少しずれているためスキューを考慮しなければならない場合があるが基本的に値は0と仮定してよい

カメラの向きRは回転行列

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

t はカメラ位置
のベクトル



$[R|t]$はカメラの位置と向きを表したの3×４の行列

$$[R|t] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix}$$

Xは被写体の座標
同次行列で表現する

xは画像平面の座標の
同次行列

全部まとめると

$$\lambda \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

4.1.2.camera.py

準備

モデルファイルをダウンロードする

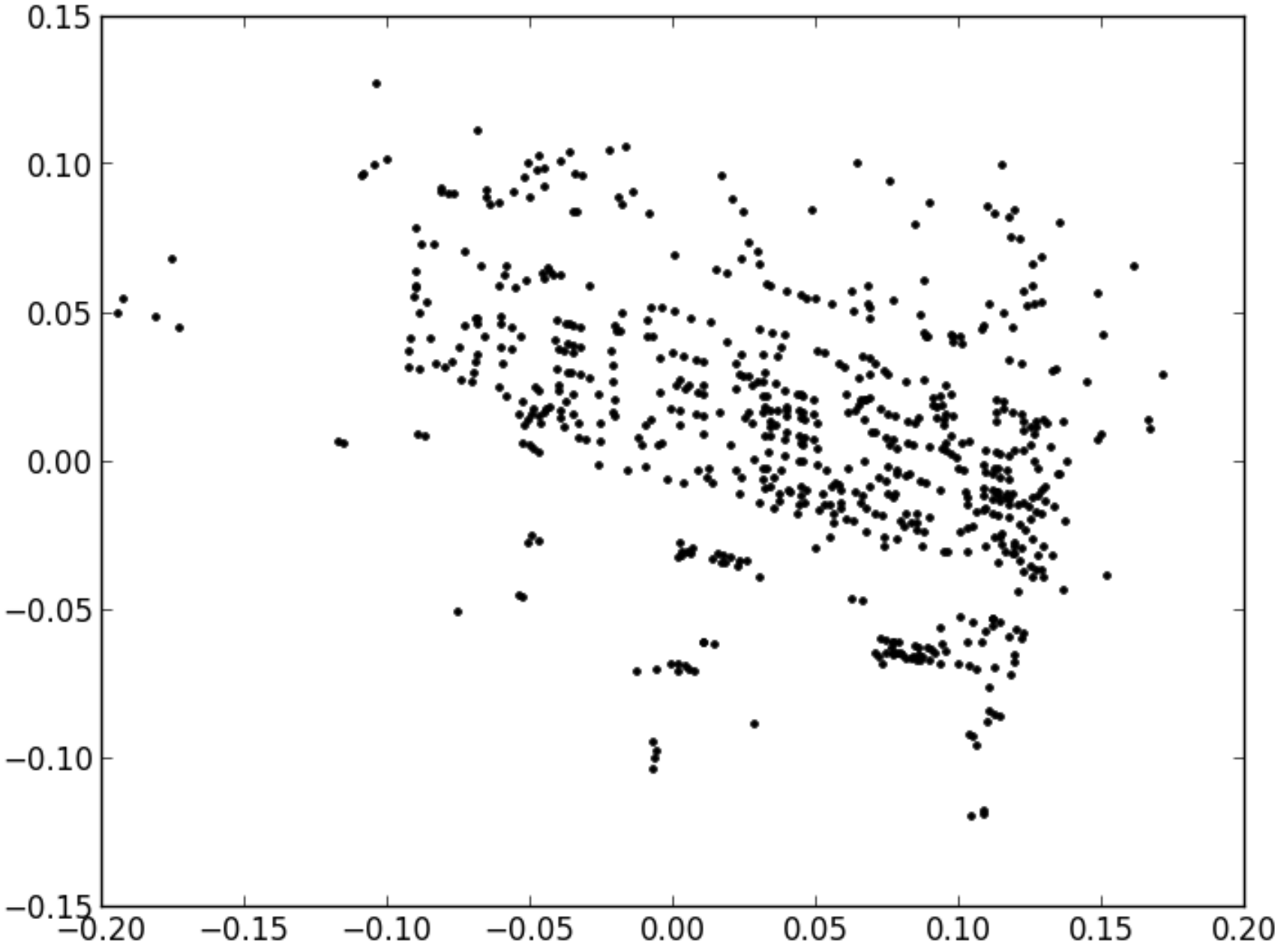
URL

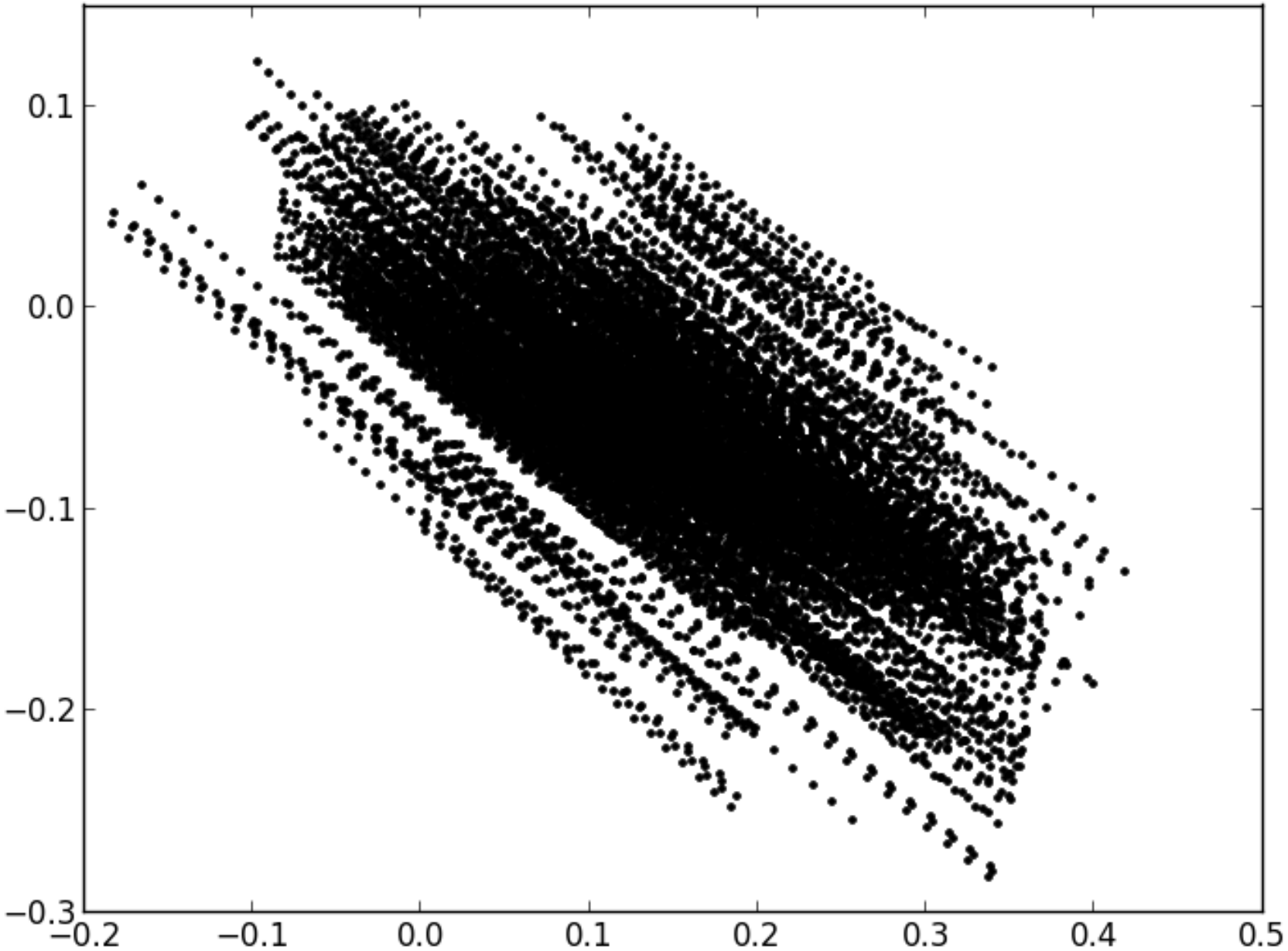
http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
http://www.robots.ox.ac.uk/~vgg/data/dunster/3D.tar.gz

からダウンロード、解凍しhouse.p3dファイルを取得する

4.1.2.camera.pyと同じフォルダにモデルファイルを置く

実行結果


0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20


0.1
0.0
−0.1
−0.2
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5

コードの説明

1つ目の画像は、カメラの向きを回転せず、に透視投影した画像である。カメラ行列の設定と射影は12行～14行で行っている

```
P = hstack((eye(3),array([[0],[0],[-10]])))
cam = camera.Camera(P)
x = cam.project(points)
```

カメラ行列Pの設定、内容は以下の通りである

$$P=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & -10\end{pmatrix}$$

２つ目の画像は、カメラの向きをランダムに回転軸と回転量を決め、20回透視投影した結果を重ねた画像である。カメラ行列の設定は23行～24行で行って、投影は28行～31行で行っている

```
r = 0.05*random.rand(3)
rot = camera.rotation_matrix(r)
figure()
for t in range(20):
  cam.P = dot(cam.P,rot)
  x = cam.project(points)
  plot(x[0],x[1],'k.')
show()
```

次の話

カメラ行列の分解

カメラ行列だけ分かっている場合、キャリブレーション行列とカメラの位置と向きを求めるにはどうすればいいのか。

RQ分解

RQ分解を行うとキャリブレーション行列と、カメラの向き(回転行列)を求めることが出来る

カメラの位置は、カメラ行列の4列目に対して、キャリブレーション行列の逆行列をかけることで求めることが出来る。

カメラ中心は以下の式で
求めることが出来る

$$C = -R^T t$$

次の話

カメラキャリブレーション（校正）

キャリブレーション行列Kを推定する作業

方法

長方形の物体の幅、高さを測定する

平らな台にカメラと物体を置く。物体はカメラ中央に置く

物体とカメラの間の距離を測定する。この結果をdzとする

写真を撮影する

画像上の物体の幅と高さをピクセル単位で測定する

結果

$$f_x = 2555$$

$$f_y = 2586$$

カメラキャリブレーションにおける注意点

撮像素子の画素数で結果が変わる

市販のカメラのほとんどが、焦点距離の可変(光学ズーム機能)を有しているので、キャリブレーションを行うさいはズームを行わない

次の話

平面とマーカを使った姿勢推定

概要

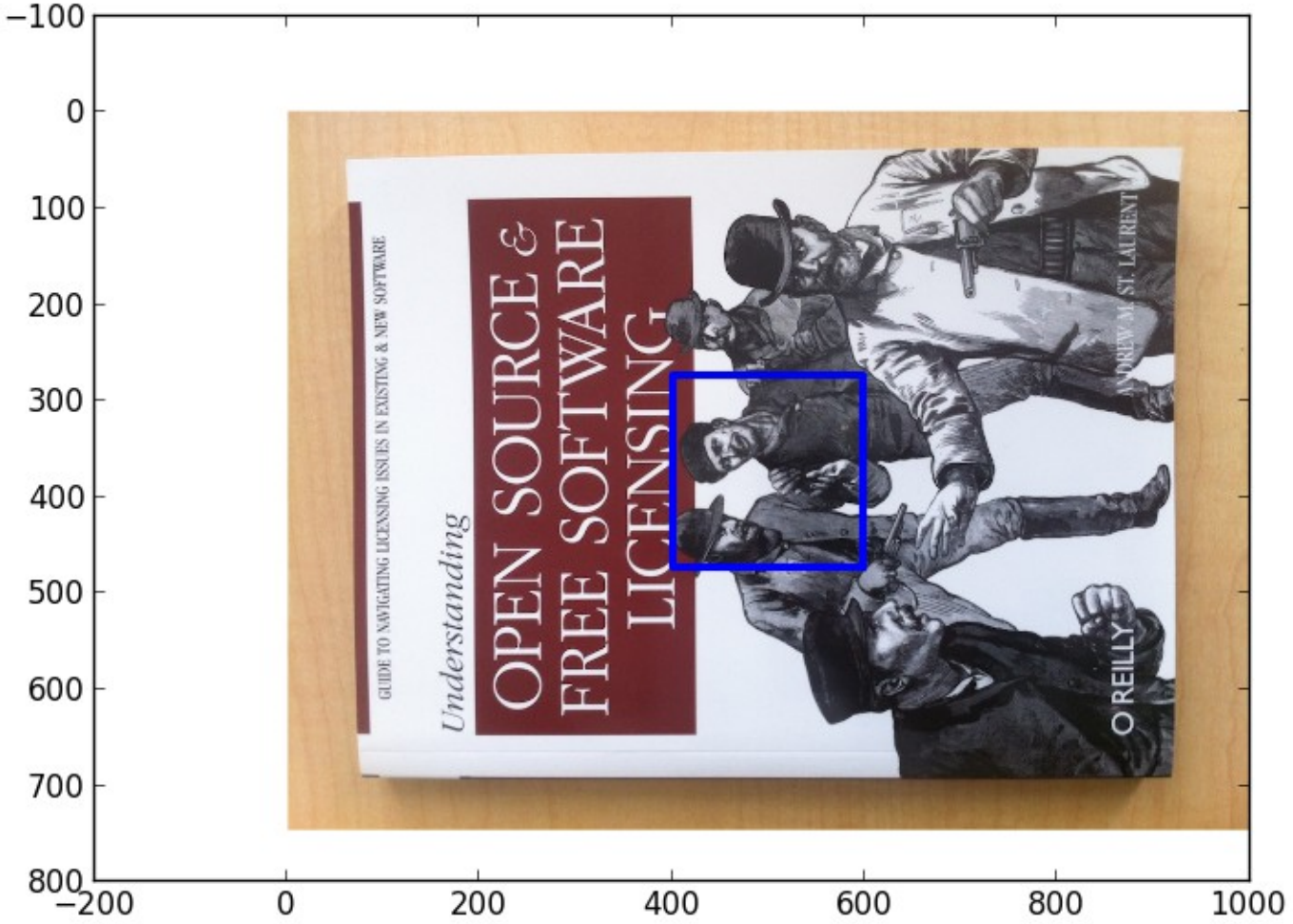
マーカ画像とマーカ画像が映っている画像のホモグラフィを計算する。ホモグラフィから、カメラ行列を求め、マーカ上に立方体を射影する

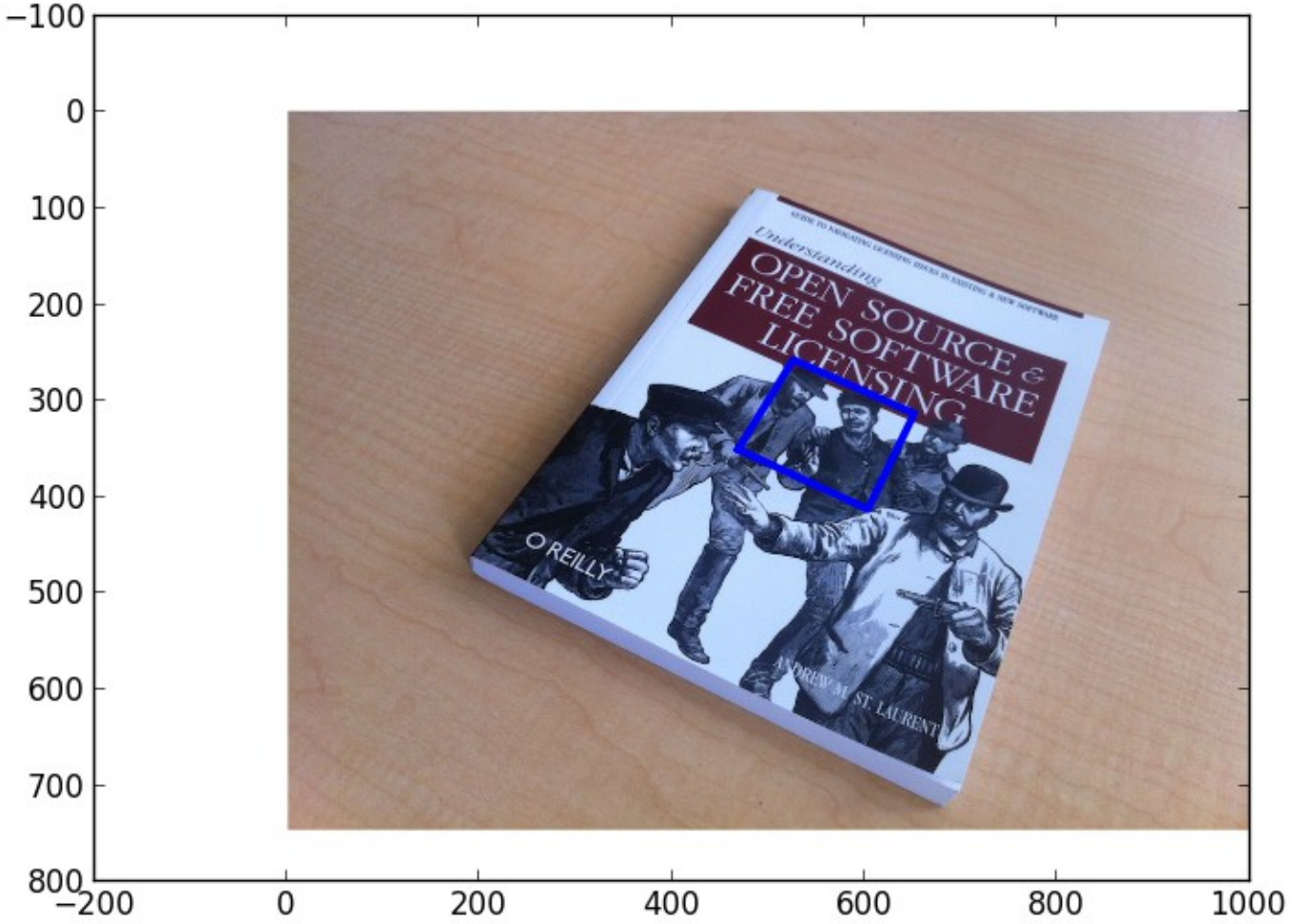
準備

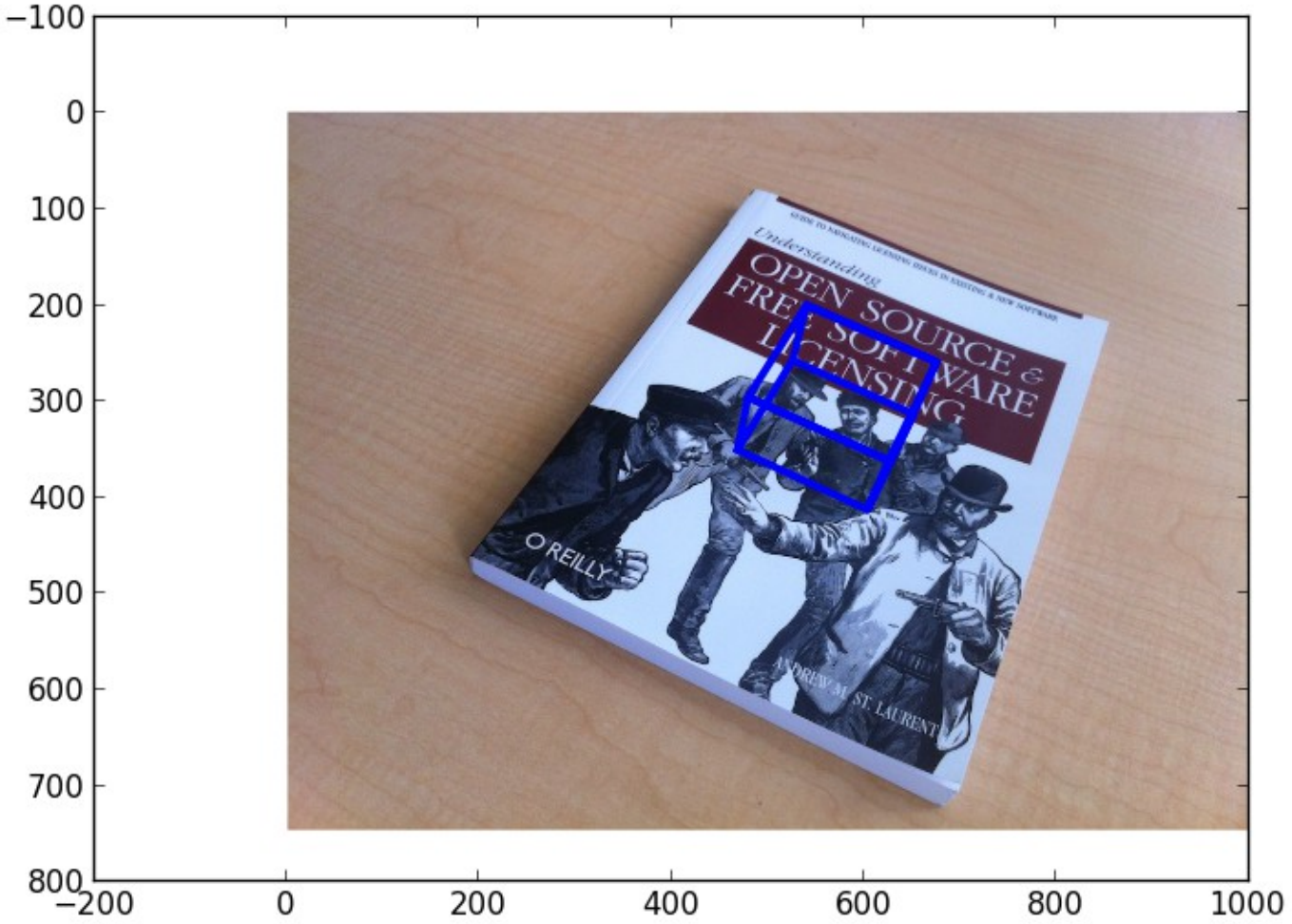
以下のファイルを
4.3.cube.pyと同じ
フォルダに置く。
book_frontal.JPG
book_perspective.JPG

実行

結果


−100
0
100
200
300
400
500
600
700
800
−200
0
200
400
600
800
1000
GUIDE TO NAVIGATING LICENSING ISSUES IN EXISTING & NEW SOFTWARE
Understanding
OPEN SOURCE &
FREE SOFTWARE
LICENSING
O'REILLY


−100
0
100
200
300
400
500
600
700
800
−200
0
200
400
600
800
1000
Understanding
OPEN SOURCE &
FREE SOFTWARE
LICENSING
O'REILLY
ST. LAURENT


−100
0
100
200
300
400
500
600
700
800
−200
0
200
400
600
800
1000
Understanding
OPEN SOURCE &
O'REILLY

コードの説明

SIFTによって2つの画像の特徴点を取得する。

(20行～24行)

```
sift.process_image('book_frontal.JPG','im0.sift')
l0,d0 = sift.read_features_from_file('im0.sift')

sift.process_image('book_perspective.JPG','im1.sift')
l1,d1 = sift.read_features_from_file('im1.sift')
```

特徴点からホモグラフィーを推定する。(27行～34行)

```
matches = sift.match_twosided(d0,d1)
ndx = matches.nonzero()[0]
fp = homography.make_homog(l0[ndx,:2].T)
ndx2 = [int(matches[i]) for i in ndx]
tp = homography.make_homog(l1[ndx2,:2].T)

model = homography.RansacModel()
H = homography.H_from_ransac(fp,tp,model)[0]
```

キャリブレーション行列の作成

(10行)

my_calibration関数

最初の画像(正面から撮った画像)
に変換をしていないカメラ行列を用いて立方体を射影(73行)

2枚目の画像に(横から撮った画像)
にホモグラフィを用いて立方体を射影
(80行～81行)

2枚目の画像に(横から撮った画像)
にホモグラフィによって変換したカメラ行列を用いて立方体を射影
(81行～82行)

ホモグラフィー推定後、
カメラ行列を取得する。
事前に用意した正方形と
立方体の座標をカメラ行
列を用いて射影する。

拡張現実感

概要

マーカ画像上に3DCGを描画する

準備

ライブラリのインストール

PyGame
PyOpenGL

4.4.4.teapot.pyの準備

book_perspective.bmpとar_camera.pklを4.4.4.teapot.pyと同じフォルダに置く

コードの説明

4.3で作成したarcamera.pklを読み込みカメラ行列を取得する。
(119行目～121行目)

```
with open('ar_camera.pkl','r') as f:
  K = pickle.load(f)
  Rt = pickle.load(f)
```

四角形のオブジェクトを用いて背景画像を描画する。

(65行　draw_background関数)

カメラ行列をOpenGL形式に変換する。(14行の

set_projection_from_camera関数)

焦点距離から、画角、アスペクト比を計算する。画像のサイズ、前方クリップ面（描画する最前面)、後方クリップ面(描画する最背面)を設定する

ティーポッドを描画する座標(位置)と姿勢(向き)を計算する。
(34行set_modelview_from_camera関数)
この際、ティーポッドをx軸周りで回転させる。

ティーポッドを描画する
(95行　draw_teapot関数)

4.4.5.toyplane.pyの準備

オブジェクトファイルの
ダウンロードし、toyplane.obj
を取得する

http://www.oyonale.com/modeles.php?page=56

mtlファイルの作成

toyplane.mtlを作成する

```
newmtl lightblue
Kd 0.5 0.75 1.0
illum 1
```

オブジェクトファイルのusemtl
タグを以下のように変更する

usemtl lightblue

objloader.pyの作成

http://www.pygame.org/wiki/OBJFileLoader

上側のソースをコピー

(MLT関数、OBJクラスが記述されている箇所)

4.4.5.toyplane.pyと同じフォルダに作成したファイルを保存する。

コードの説明

4.4.4.teapot.pyと処理内容はほぼ同じだが、違いはobj形式のファイルを読み込み、描画している。

以上