HW1

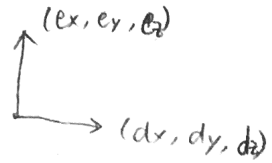
Yao Cai 25933599

02/04/2018

1. 1) We have proved in class that the vanishing point of a group of parallel line with direction (D_x, D_y, D_z) is

$$\left(\frac{fD_x}{D_z}, \frac{fD_y}{D_z}\right)$$

Now suppose (e_x, e_y, e_z) and (d_x, d_y, d_z) are the directions of two sets of parallel lines that are normal to each other, and they define a plane A



then the direction of any lines on plane A can be expressed as:

$$(ae_x + bd_x,\ ae_y + bd_y,\ ae_z + bd_z)$$

and the vanishing point for such direction is:

$$\left(\frac{f(ae_x + bd_x)}{ae_z + bd_z}, \frac{f(ae_y + bd_y)}{ae_z + bd_z}\right) \quad ①$$

$$= \left(\frac{ae_z}{ae_z + bd_z} \cdot \frac{fe_x}{e_z} + \frac{bd_z}{ae_z + bd_z} \cdot \frac{fd_x}{d_z},\ \frac{ae_z}{ae_z + bd_z} \cdot \frac{fe_y}{e_z} + \frac{bd_z}{ea_z + bd_z} \cdot \frac{fd_y}{d_z}\right)$$

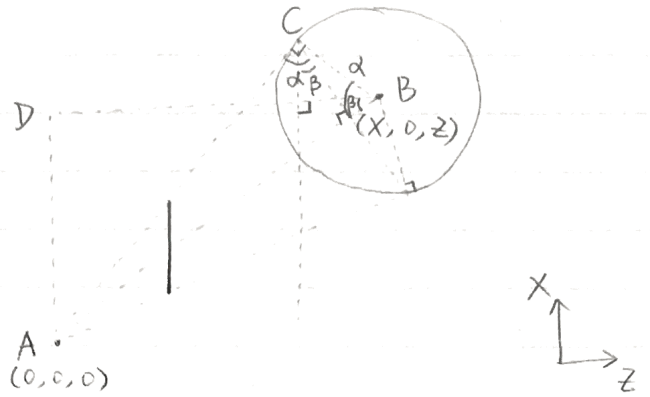
$$= \frac{ae_z}{ae_z + bd_z}\left(\frac{fe_x}{e_z}, \frac{fe_y}{e_z}\right) + \frac{bd_z}{ae_z + bd_z}\left(\frac{fd_x}{d_z}, \frac{fd_y}{d_z}\right)$$

$\left(\frac{fe_x}{e_z}, \frac{fe_y}{e_z}\right)$ ② and $\left(\frac{fd_x}{d_z}, \frac{fd_y}{d_z}\right)$ ③ are vanishing points of (e_x, e_y, e_z) and (d_x, d_y, d_z), respectively.

then point ① is just on the same line as point ② and ③

This proves that any lines on the same plane have vanishing points on the same line, which is the vanishing line of the plane

2)



The points on the sphere that relates to the edge of the silhouette form a cone with the origin (0, 0, 0), as shown above

we know that a proper cut on the cone can be an ellipse, with eccentricity calculated by

$$e = \frac{\sin\beta}{\sin\alpha}$$

α and β are defined above in the figure

A, B, C, D are points also defined above

$$\sin\alpha = \frac{AC}{AB} = \frac{\sqrt{x^2+z^2-r^2}}{\sqrt{x^2+z^2}}$$

$$\sin\beta = \frac{AD}{AB} = \frac{x}{\sqrt{x^2+z^2}}$$

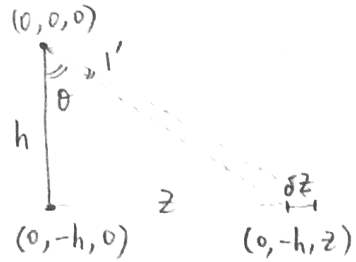
then $e = \frac{\sin\beta}{\sin\alpha} = \frac{x}{\sqrt{x^2+z^2-r^2}}$ when $z > r$, ellipse

when $z = r$ or $X >> Z, r$

$e \to 1$ then it will look like a parabola

when $z < r$, $e > 1$, then it will look like a hyperbola.

3)



we have the following relations:

$$\left.\begin{aligned} \tan\theta &= \frac{z}{h} \\ \tan(\theta + 1') &= \frac{z+\delta z}{h} \end{aligned}\right\} \Rightarrow \theta = \operatorname{atan}\left(\frac{z}{h}\right)$$

$$\Rightarrow \frac{z+\delta z}{h} = \tan\left(\operatorname{atan}\left(\frac{z}{h}\right) + 1'\right)$$

$$\Rightarrow \delta z = h\tan\left(\operatorname{atan}\left(\frac{z}{h}\right) + 1'\right) - z$$

smaller z, smaller δz, higher depth accuracy

2.(1) if $\hat{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}$ then $S = \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix}$

Proof: $\vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$

$$\begin{matrix} s_1 & s_2 & s_3 & s_1 & s_2 & s_3 \\ b_1 & b_2 & b_3 & b_1 & b_2 & b_3 \end{matrix}$$

$$\hat{s} \times \vec{b} = \begin{pmatrix} s_2 b_3 - b_2 s_3 \\ s_3 b_1 - b_3 s_1 \\ s_1 b_2 - b_1 s_2 \end{pmatrix} \qquad S\vec{b} = \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} -b_2 s_3 + s_2 b_3 \\ b_1 s_3 - b_3 s_1 \\ -b_1 s_2 + b_2 s_1 \end{pmatrix}$$

(2) $R = \exp(\phi S)$

$$= I + \phi S + \frac{1}{2!}(\phi S)^2 + \frac{1}{3!}(\phi S)^3 + \cdots$$

$$= I + \phi S + \frac{1}{2!}\phi^2 S^2 - \frac{1}{3!}\phi^3 S - \frac{1}{4!}\phi^4 S^2 + \frac{1}{5!}\phi^5 S +$$

$$= I + \left(\phi S - \frac{1}{3!}\phi^3 S + \frac{1}{5!}\phi^5 S\right) + \left(\frac{1}{2!}\phi^2 S^2 - \frac{1}{4}\phi^4 S^2 + \cdots\right)$$

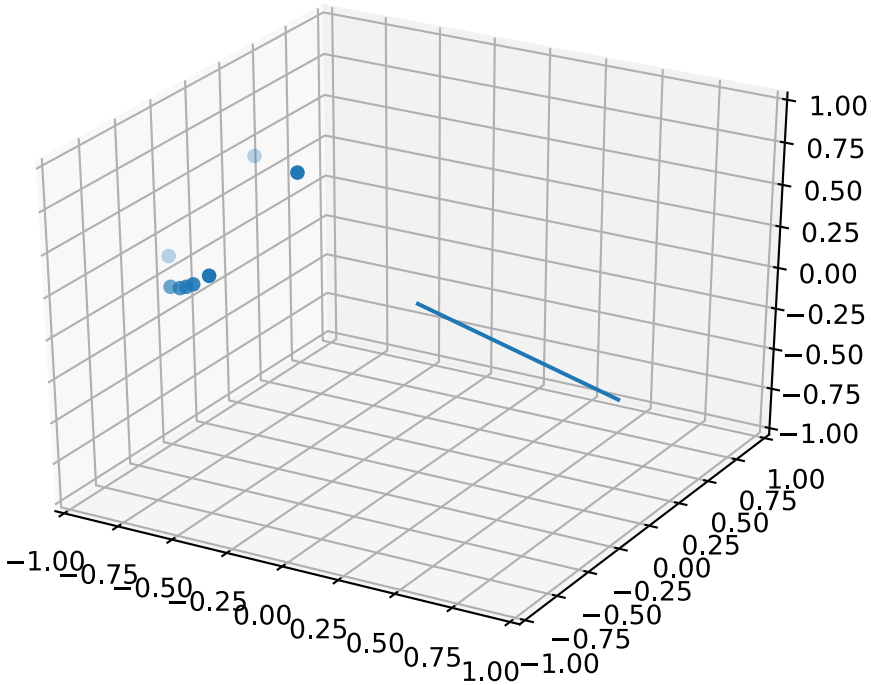
$$= I + \sin\phi S + (1 - \cos\phi) S^2$$

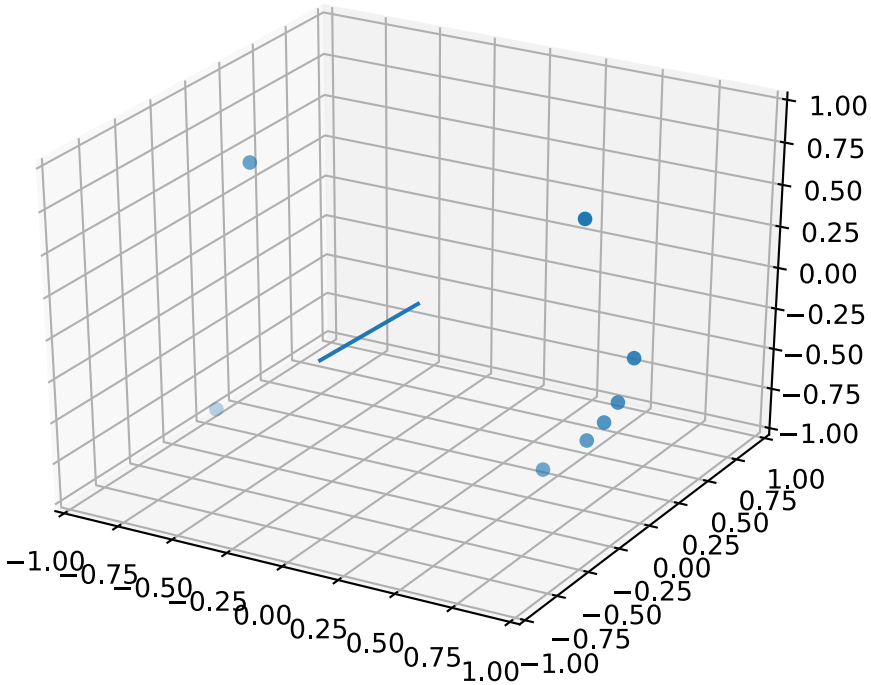
$$S^2 = \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix} = \begin{pmatrix} -s_2^2 - s_3^2 & s_1 s_2 & s_1 s_3 \\ s_1 s_2 & -s_3^2 - s_1^2 & s_2 s_3 \\ s_1 s_3 & s_2 s_3 & -s_1^2 - s_2^2 \end{pmatrix}$$

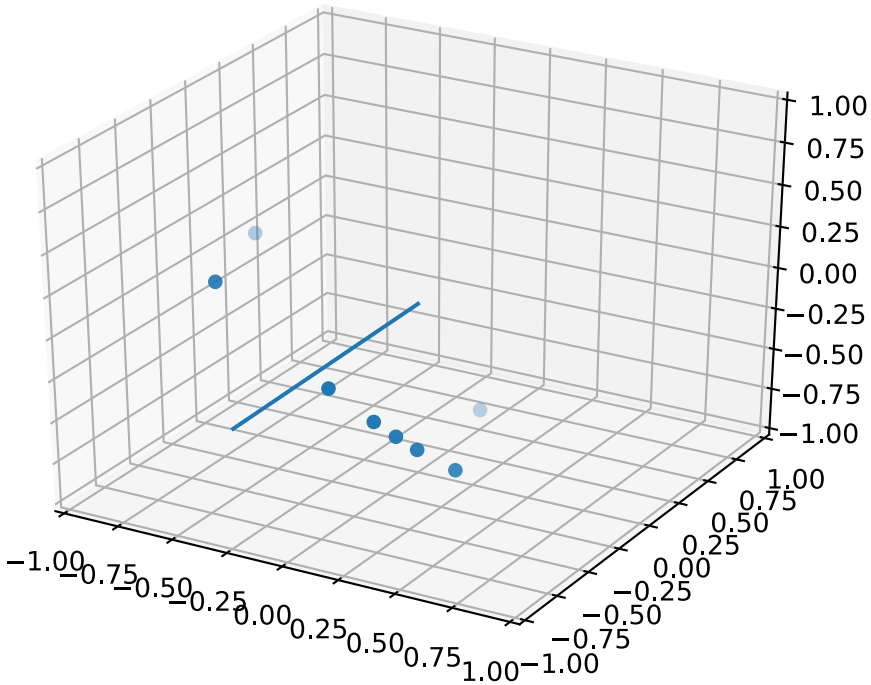
$$= \begin{pmatrix} s_1^2 - 1 & s_1 s_2 & s_1 s_3 \\ s_1 s_2 & s_2^2 - 1 & s_2 s_3 \\ s_1 s_3 & s_2 s_3 & s_3^2 - 1 \end{pmatrix}$$

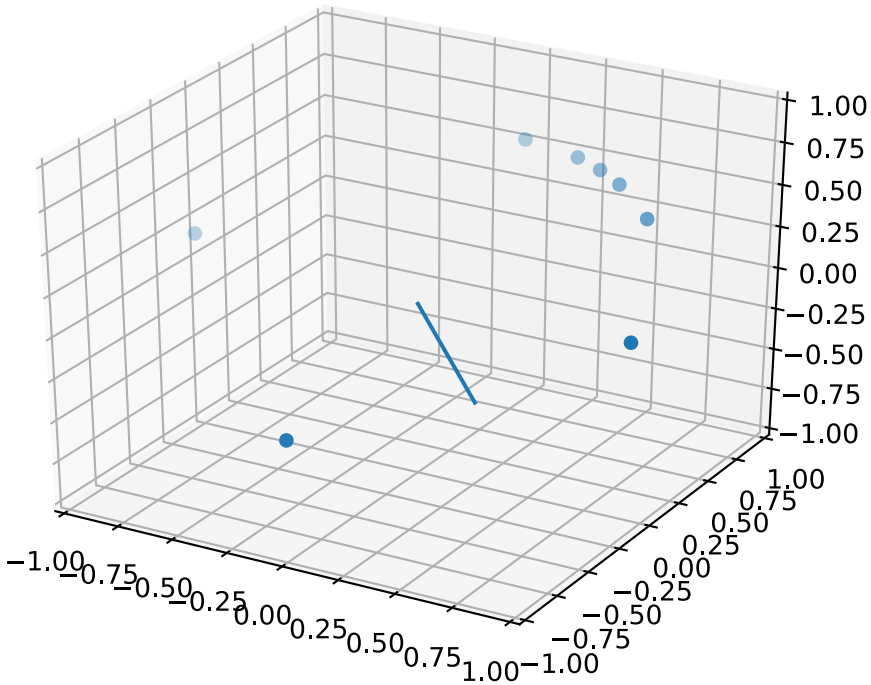
$$S^3 = \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix} \begin{pmatrix} s_1^2 - 1 & s_1 s_2 & s_1 s_3 \\ s_1 s_2 & s_2^2 - 1 & s_2 s_3 \\ s_1 s_3 & s_2 s_3 & s_3^2 - 1 \end{pmatrix} = \begin{pmatrix} 0 & s_3 & -s_2 \\ -s_3 & 0 & s_1 \\ s_2 & -s_1 & 0 \end{pmatrix} = -S$$

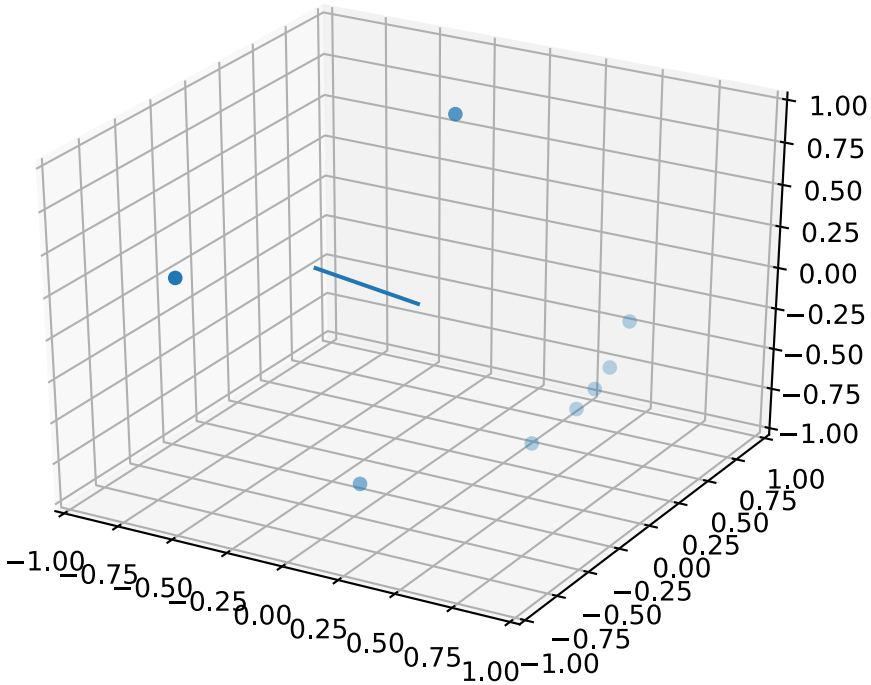
$S^4 = S^3 \cdot S = -S \cdot S = -S^2 \qquad S^5 = S^3 \cdot S^2 = -S \cdot S^2 = -S^3 = S$


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

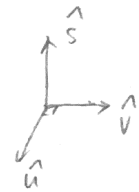

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

4.) the eigenvalues 1, $\cos\phi + i\sin\phi$, $\cos\phi - i\sin\phi$

related eigenvectors $\hat{s}$, $\frac{\sqrt{2}}{2}\hat{u} - \frac{\sqrt{2}}{2}i\hat{v}$, $\frac{\sqrt{2}}{2}\hat{u} + \frac{\sqrt{2}}{2}i\hat{v}$



Verify below:

$$(I + \sin\phi S + (1-\cos\phi)S^2)\hat{u} = \hat{u} + \sin\phi\hat{v} + (1-\cos\phi)(-\hat{u})$$
$$= \sin\phi\hat{v} + \cos\phi\hat{u}$$

$$(I + \sin\phi S + (1-\cos\phi)S^2)\hat{v} = \hat{v} + \sin\phi(-\hat{u}) + (1-\cos\phi)(-\hat{v})$$
$$= -\sin\phi\hat{u} + \cos\phi\hat{v}$$

① $(I + \sin\phi S + (1-\cos\phi)S^2)\hat{s} = I\hat{s} + 0 + 0 = \hat{s}$

② $(I + \sin\phi S + (1-\cos\phi)S^2)(\frac{\sqrt{2}}{2}\hat{u} - \frac{\sqrt{2}}{2}i\hat{v}) =$

$$\frac{\sqrt{2}}{2}(\sin\phi\hat{v} + \cos\phi\hat{u}) - \frac{\sqrt{2}}{2}i(-\sin\phi\hat{u} + \cos\phi\hat{v})$$
$$= (\cos\phi + i\sin\phi)(\frac{\sqrt{2}}{2}\hat{u} - \frac{\sqrt{2}}{2}i\hat{v})$$

③ $(I + \sin\phi S + (1-\cos\phi)S^2)(\frac{\sqrt{2}}{2}\hat{u} + \frac{\sqrt{2}}{2}i\hat{v}) =$

$$\frac{\sqrt{2}}{2}(\sin\phi\hat{v} + \cos\phi\hat{u}) + \frac{\sqrt{2}}{2}i(-\sin\phi\hat{u} + \cos\phi\hat{v})$$
$$= (\cos\phi - i\sin\phi)(\frac{\sqrt{2}}{2}\hat{u} + \frac{\sqrt{2}}{2}i\hat{v})$$

* when $\phi = 2\pi n$

eigenvalues: 1

eigenvectors: $\hat{s}$, $\hat{u}$, $\hat{v}$

* when $\phi = (2n+1)\pi$

eigenvalues: 1, -1

eigenvectors: $\hat{s}$, $\hat{u}$, $\hat{v}$

5.) $\text{trace}(R) = 1 + \cos\phi + i\sin\phi + \cos\phi - i\sin\phi$

$$= 1 + 2\cos\phi$$

$$\Rightarrow \cos\phi = \frac{1}{2}(\text{trace}(R) - 1)$$

In [154]:
```
from numpy.linalg import eig
from random import random
import numpy as np
from math import acos, pi, sin, cos

def get_unit_vector():
    s = np.array([2*random()-1, 2*random()-1, 2*random()-1])
    s = s/np.linalg.norm(s)
    return s

def get_s_matrix(s):
    return np.array([[0, -s[2], s[1]], [s[2], 0, -s[0]], [-s[1], s[0], 0]])

def get_rotation_matrix(s, phi):
    S = get_s_matrix(s)
    R = np.eye(3) + sin(phi) * S + (1-cos(phi)) * S @ S
    return R

def rot_to_ax_phi(R):
    if abs(np.trace(R)-3) < 1e-10:
        return np.array([0, 0, 0]), 2*pi

    if abs(np.trace(R)+1) < 1e-10:
        phi = pi
    else:
        phi = acos((np.trace(R)-1)/2)

    eigv, v = eig(R)
    for i in range(3):
        if is_one(eigv[i]):
            index0 = i
    index1 = (i+1)%3
    index2 = (i+2)%3
    s = [v[0, index0].real, v[1, index0].real, v[2, index0].real]
    return s, phi

def is_one(number):
    if abs(number-1) < 1e-10:
        return True

s = get_unit_vector()
phi = pi/3
R = get_rotation_matrix(s, phi)

print(s)
print(phi)
rot_to_ax_phi(R)
```

```
[ 0.16573982 -0.58044732  0.79725229]
1.0471975511965976
```

Out[154]:
```
([0.16573981989433331, -0.5804473225914154, 0.79725229243812856],
 1.0471975511965974)
```

In [ ]:

$\boxed{\text{homography}}$

3. (1)

$$\begin{pmatrix} V_x \\ V_y \\ 1 \end{pmatrix} \sim \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \begin{pmatrix} U_x \\ U_y \\ 1 \end{pmatrix}$$

$$V_x = \frac{h_{11}U_x + h_{12}U_y + h_{13}}{h_{31}U_x + h_{32}U_y + 1}$$

$$V_y = \frac{h_{21}U_x + h_{22}U_y + h_{23}}{h_{31}U_x + h_{32}U_y + 1}$$

$$\Rightarrow \left.\begin{aligned} &h_{11}U_x + h_{12}U_y + h_{13} - h_{31}U_xV_x - h_{32}U_yV_x = V_x \\ &h_{21}U_x + h_{22}U_y + h_{23} - h_{31}U_xV_y - h_{32}U_yV_y = V_y \end{aligned}\right\}$$

$$\begin{pmatrix} U_x & U_y & 1 & 0 & 0 & 0 & -U_xV_x & -U_yV_x \\ 0 & 0 & 0 & U_x & U_y & 1 & -U_xV_y & -U_yV_y \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} V_x \\ V_y \end{pmatrix}$$

four points:

$$\underbrace{\begin{pmatrix} U_{1x} & U_{1y} & 1 & 0 & 0 & 0 & -U_{1x}V_{1x} & -U_{1y}V_{1x} \\ 0 & 0 & 0 & U_{1x} & U_{1y} & 1 & -U_{1x}V_{1y} & -U_{1y}V_{1y} \\ U_{2x} & U_{2y} & 1 & 0 & 0 & 0 & -U_{2x}V_{2x} & -U_{2y}V_{2x} \\ 0 & 0 & 0 & U_{1x} & U_{1y} & 1 & -U_{2x}V_{2y} & -U_{2y}V_{2y} \\ U_{3x} & U_{3y} & 1 & 0 & 0 & 0 & -U_{3x}V_{3x} & -U_{3y}V_{3x} \\ 0 & 0 & 0 & U_{1x} & U_{1y} & 1 & -U_{3x}V_{3y} & -U_{3y}V_{3y} \\ U_{4x} & U_{4y} & 1 & 0 & 0 & 0 & -U_{4x}V_{4x} & -U_{4y}V_{4x} \\ 0 & 0 & 0 & U_{1x} & U_{1y} & 1 & -U_{4x}V_{4y} & -U_{4y}V_{4y} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix}}_{h} = \underbrace{\begin{pmatrix} V_{1x} \\ V_{1y} \\ V_{2x} \\ V_{2y} \\ V_{3x} \\ V_{3y} \\ V_{4x} \\ V_{4y} \end{pmatrix}}_{V}$$

$$Ah = V$$

solve $\underset{h}{\arg\min} \|Ah - V\|_2^2$

$$\Rightarrow h = (A^T A)^{-1} A^T V$$

$\boxed{\text{affine}}$

$h_{32} = h_{31} = 0 \qquad h_{33} = 1$

$$\underbrace{\begin{pmatrix} u_{1x} & u_{1y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{1x} & u_{1y} & 1 \\ u_{2x} & u_{2y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{2x} & u_{2y} & 1 \\ u_{3x} & u_{3y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{3x} & u_{3y} & 1 \\ u_{4x} & u_{4y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{4x} & u_{4y} & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \end{pmatrix}}_{h} = \underbrace{\begin{pmatrix} v_{1x} \\ v_{1y} \\ v_{2x} \\ v_{2y} \\ v_{3x} \\ v_{3y} \\ v_{4x} \\ v_{4y} \end{pmatrix}}_{V}$$

solve $\underset{h}{\arg\min} \|Ah - V\|_2^2$

$$\Rightarrow h = (A^T A)^{-1} A^T V$$

(2) for affine : $h_{31} = h_{32} = 0$, $h_{33} = 1$

for homography : $h_{33} = 1$

(3) Affine transformations keep parallel lines parallel, so if there are two parallel line in the first image, but the same two lines are not paralled in the second image, affine transformation cannot convert image 1 to image 2.

Homography keeps line straight, so if there are 3 points on a line in the first image, but not on a line in the second image, homography cannot convert image 1 to image 2.

3.4

Required functions attached.
Observation: Homography keeps line straight, and my pictures can fit into the planar surfaces by Homography; while affine transformation keeps parallel lines, so my pictures can't always fit into the planar surfaces, since some planar surfaces have non-parallel lines.

```
import numpy as np
from numpy.linalg import inv
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from scipy import ndimage
from PIL import Image

def affine_solve(U, V):
    b = np.ravel(V, order='F')
    b = b.reshape((b.shape[0], 1))
    A = get_A_affine(U[:, 0], V[:, 0])
    for i in range(1, U.shape[1]):
        A = np.vstack((A, get_A_affine(U[:, i], V[:, i])))
    h = inv(A.T @ A) @ A.T @ b
    H = np.vstack((h[:3, :].T, h[3:6, :].T, [[0, 0, 1]]))
    return H

def homography_solve(U, V):
    b = np.ravel(V, order='F')
    b = b.reshape((b.shape[0], 1))
    A = get_A_homo(U[:, 0], V[:, 0])
    for i in range(1, U.shape[1]):
        A = np.vstack((A, get_A_homo(U[:, i], V[:, i])))
    h = inv(A.T @ A) @ A.T @ b
    H = np.vstack((h[:3, :].T, h[3:6, :].T, [[h[6, 0], h[7, 0], 1]]))
    return H

def homography_transform(u, H):
    N = u.shape[1]
    u = np.vstack((u, np.ones((1, N))))
    V_target = H @ u
    V_target = from_3D_to_2D(V_target)
    V_target =  V_target.astype(int)
    return V_target


def get_A_homo(u, v):
    return np.array([[u[0], u[1], 1, 0, 0, 0, -u[0]*v[0], -u[1]*v[0]],  [0, 0, 0, u[0], u[1], 1, -u[0]*v[1], -
u[1]*v[1]]])
```

```
def get_A_affine(u, v):
    return np.array([[u[0], u[1], 1, 0, 0, 0],  [0, 0, 0, u[0], u[1], 1]])

def from_3D_to_2D(V):
    N = V.shape[1]
    V = V @ inv(np.diag(V[2, 0:N]))
    return V[0:2, :]
```

affine

homography

NBA 2K13
HYUNDAI
MAMMA MIA!
PALACE
Annie
ROXY

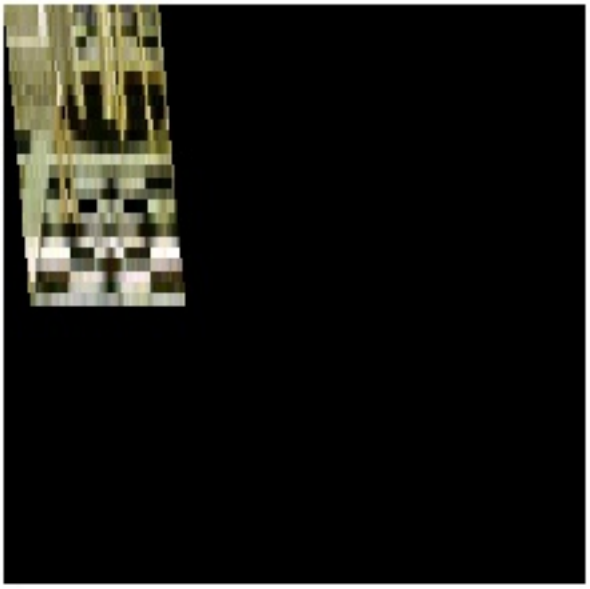
affine

NBA 2K13
HYUNDAI
MAMMA MIA!
PALACE
Annie
ROXY

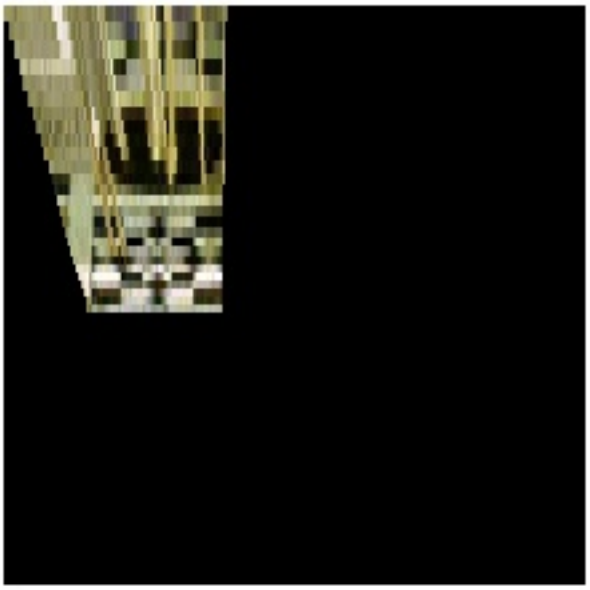
homography

affine

homography



affine



homography