Computer Vision

Homework 2: Structure from Motion (SfM)

Name: 陳博琳

Student ID: 112030504

Part A. Computing the 4 initial R,T transformations from Essential Matrix

Use SVD to find R and T from E. We can get totally get 4 RT matrices because Q,R both have two choice. Each RT matrix shape is 3X4, in estimate_initial_RT() function we will return a list combine with 4 RT matrices. This step can help us to estimate two photo's transformative matrix. This step can convert camera point to global point. Gaining four possible pose, use corresponding match to find which one is the best.

```
--------------------------------------------------------------------------------
Part A: Check your matrices against the example R,T
--------------------------------------------------------------------------------
Example RT:
 [[ 0.9736 -0.0988 -0.2056  0.9994]
 [ 0.1019  0.9948  0.0045 -0.0089]
 [ 0.2041 -0.0254  0.9786  0.0331]]

Estimated RT:
 [[[ 0.98305251 -0.11787055 -0.14040758  0.99941228]
  [-0.11925737 -0.99286228 -0.00147453 -0.00886961]
  [-0.13923158  0.01819418 -0.99009269  0.03311219]]

 [[ 0.98305251 -0.11787055 -0.14040758 -0.99941228]
  [-0.11925737 -0.99286228 -0.00147453  0.00886961]
  [-0.13923158  0.01819418 -0.99009269 -0.03311219]]

 [[ 0.97364135 -0.09878708 -0.20558119  0.99941228]
  [ 0.10189204  0.99478508  0.00454512 -0.00886961]
  [ 0.2040601  -0.02537241  0.97862951  0.03311219]]

 [[ 0.97364135 -0.09878708 -0.20558119 -0.99941228]
  [ 0.10189204  0.99478508  0.00454512  0.00886961]
  [ 0.2040601  -0.02537241  0.97862951 -0.03311219]]]
```

Part B: Determining the best linear estimate of a 3D point

Section B is counting the best 3D point by linear estimate, to match the point in each photo. Using P=MP' to get the answer. M is existing, 3D point is unknown. The third row formula is combine from first and second row, neglect the third one because this is a linear system. Construct two equation to each point. Divide matrixes by SVD, p is the least row in VT

```
--------------------------------------------------------------------------------
Part B: Check that the difference from expected point
is near zero
--------------------------------------------------------------------------------
Difference:  0.0029243053036863698
```

Part C: Calculating the reprojection error and its Jacobian

Section C calculating match point's reprojection error. Given a image i, camera matrix Mi and 3D point p to obtain reprojection error. One reprojection point can convert into two dimension, respectively estimate error.

Jacobian convert supply the same work like reprojection_error(). T(x)=Ax+b can reflection-rotation matrix.

```
------------------------------------------------------------------------------------
Part C: Check that the difference from expected error/Jacobian
is near zero
------------------------------------------------------------------------------------
Error Difference:  8.301299988565727e-07
Jacobian Difference:  1.817115702351657e-08
```

Part D: Determining the best nonlinear estimate of a 3D point

This section need former function to get linear estimate first, which estimate initial point. Ten times iteration to approximate reprojection error, sustain doing reprojection, Jacobian and Gauss-Newton optimization to make sure the final reprojection error is lower than the one of linearly estimated point.

```
------------------------------------------------------------------------------------
Part D: Check that the reprojection error from nonlinear method
is lower than linear method
------------------------------------------------------------------------------------
Linear method error: 98.73542356894183
Nonlinear method error: 95.59481784846035
```

Part E: Determining the correct R, T from Essential Matrix

Estimating the corresponding match R,T for each pair. In section A we get 4 probable RT matrix. In this section we choose the best one. First calculate 3D point's match point, correct RT estimate have positive Z value. Find the maximum amount of positive Z value.

```
------------------------------------------------------------------------------------
Part E: Check your matrix against the example R,T
------------------------------------------------------------------------------------
Example RT:
 [[ 0.9736 -0.0988 -0.2056  0.9994]
 [ 0.1019  0.9948  0.0045 -0.0089]
 [ 0.2041 -0.0254  0.9786  0.0331]]

Estimated RT:
 [[ 0.97364135 -0.09878708 -0.20558119  0.99941228]
 [ 0.10189204  0.99478508  0.00454512 -0.00886961]
 [ 0.2040601  -0.02537241  0.97862951  0.03311219]]
```

Part F: Run the entire SFM pipeline

```
------------------------------------------------------------------------------------
Part F: Run the entire SFM pipeline
------------------------------------------------------------------------------------
Save results to results.npy!
```

PtsVisualizer photo by code

All calculate with function

⊖ np. linalg. svd. (a, full_matrices=1, compute_uv=1)

input :

a - J (M, N) matrix

full_matrices value=1 u(M,M), v (N,N)

default = 1 value=0 K=min(M,N) u=(M,K) v=(K,N)

compute_uv value=1 count u,s,v

default=1 value=0 count s

output:

return u, s, v

u (M,M) s(M,N) v (N,N)

A = u * s * v

ex :

data = $\begin{bmatrix} 1, 2, 3 \\ 4, 5, 6 \end{bmatrix}$ u, s, v = np.linalg.svd(data)

data (2,3) u(2,2) v(3,3)

math count :

$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = U\Sigma V^T \quad AA^T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 14 & 32 \\ 32 & 77 \end{bmatrix}$

$\boxed{\text{Eigenvalue}}$

$\det\begin{pmatrix} 14-\lambda & 32 \\ 32 & 77-\lambda \end{pmatrix} = \begin{vmatrix} 14-\lambda & 32 \\ 32 & 77-\lambda \end{vmatrix} = (14-\lambda)(77-\lambda) - 32^2 = 1078 - 91\lambda + \lambda^2 - 1024$

$= \lambda^2 - 91\lambda + 54$

$\lambda = 90.4027$ and 0.5973

$\sqrt{\lambda} = \sqrt{90.4027} = 9.508$

$\sqrt{0.5973} = 0.772$

$\Sigma = \begin{pmatrix} 9.508 & 0 & 0 \\ 0 & 0.772 & 0 \end{pmatrix}$

$U = AA^T$ Eigenvector

$\Sigma = \sqrt{\lambda}$

$V = A^T A$ Eigenvector

Homogeneous system

Linear system 右邊的數字皆為0，必定有解，一組解的話，必定$(0,0,\ldots 0)$

SVD system

Image points $\overset{(17)}{2\times 2}$

Camera matrices $\overset{(17)}{2\times 3\times 4}$.

$$\underset{n}{m\,\boxed{A}} = \underset{m}{\boxed{U}}\ \underset{n}{\boxed{\Sigma}}\ \overset{n}{\boxed{V}}$$

$AA^T \qquad A^TA$

A_set 4×4.

$V_1 - m$

$m - U_1$

$V_2 - m$

$m - U_2$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha H \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

→ 單應矩陣有9個未知數.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{cases} x' = \alpha(h_1x + h_2y + h_3) \\ y' = \alpha(h_4x + h_5y + h_6) \\ 1 = \alpha(h_7x + h_8y + h_9) \end{cases}$$

$$\begin{cases} x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3) \\ y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6) \end{cases}$$

Homogeneous Linear problem Solve with SVD

$A = U\Sigma V^T$ V的每個 column 代表 $Ah=0$.

$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$

$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

$A_ih = 0$.

$$A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$h = [h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7\ h_8\ h_9]^T$$

為什麼 $Ax=0$ 的SVD解是V的最後一列.

1. 對A進行SVD分解，正交矩陣的列是 $AX=0$ 的解.

令 $\|x\|=1$ 範數

$$\hat{x} = \arg\min \|Ax\|^2, \text{ subject to } \|x\|^2 = 1 \quad (1)$$

$$L(x,\lambda) = \|Ax\|^2 + \lambda(1 - \|x\|^2)$$
$$= x^TA^TAx + \lambda(1 - x^Tx) \quad (2)$$

求極值，對x和λ導入令其等於0.

$$\frac{\partial L(x,\lambda)}{\partial x} = 2A^TAx - 2\lambda x = 0 \quad (3)$$

$$(A^TA - \lambda)x = 0 \quad (4)$$

$$A^TAX = \lambda X \quad (5)$$

$$\frac{\partial L(x,\lambda)}{\partial \lambda} = 1 - x^Tx = 0 \quad (6)$$

滿足式x為極值. x和λ為eigen value和eigen vector A^TA為SVD的右正交矩陣V

所以V對應1的解.

2. 最後一列為何最優，即目標函數最近0的解.

$$\|AX\|^2 = x^TA^TAx = x^T\lambda x = \lambda x^Tx = \lambda$$

目標讓Ax長度最小 x^T只改變方向 奇異值才會改變向量長度.

奇異值小↓
向量長度短↓

把A^TA分解 min eigen valuen對應的 eigen vector即解.

$A = UDV^T$

D→奇異值矩陣.

D對角線上的奇異值由大到小排列.

而V的最後一列為min eigen value即求解.

Jacobain 矩陣.

① $P \in D$ 區域 用 $T(x) = Ax+b$ 做鏡射变换. 近似 $F(x)$

符合條件應使誤差. $T(p) - F(p)$ 趨近 0

$$\lim_{x \to p} \frac{T(x) - F(p) - A(x-p)}{\|x-p\|} = 0$$

要求 $T(p) = F(p)$

$$F(p) = Ap + b$$

$$b = F(p) - Ap \quad ②$$

$$T(x) = Ax + F(p) - Ap \quad ① + ②$$

$$= A(x-p) + F(p)$$

$F: R^n \to R^m$ 在 p differentiable. 讓 x 可從任何方向逼近 p

另要求 p 是 D 內的点都屬於 D, F 在 P 点可導, 則 A 由 p 決定

R^n 基底 $\{e_1 \dots e_n\}$

$$\lim_{h \to 0} \frac{F(p+he_j) - F(p) - A(he_j)}{h} = 0 \quad j = 1, \dots, n$$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(p) & \cdots & \frac{\partial f_1}{\partial x_n}(p) \\ \frac{\partial f_2}{\partial x_1}(p) & \cdots & \frac{\partial f_2}{\partial x_n}(p) \\ \vdots & & \\ \frac{\partial f_m}{\partial x_1}(p) & \cdots & \frac{\partial f_m}{\partial x_n}(p) \end{bmatrix}$$

x_1 階導数 $\quad x_n$ 階導数.

P也可以換成

camera の $M \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$ $P \begin{bmatrix} A \\ B \\ C \end{bmatrix}$ $P \begin{bmatrix} A \\ B \\ C \\ 1 \end{bmatrix}$ 3D points

$y_1 = 1 \times A + 2 \times B + 3 \times C + 4$

從global轉到camera の y_1, y_2, y_3 的取得也可由 $M * P$

$$J_x = y_3[1\ 2\ 3] - y_1[9\ 10\ 11] \ / y_3^2 \qquad P \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$J_y = y_3[5\ 6\ 7] - y_1[9\ 10\ 11] \ / y_3^2$$

BA 優化. 重投影誤差.

將3維坐標投影到当前的像素平面上. (u, v)

幀上提取地图点的位置. (u_m, v_m)

誤差 $e = [u_m, v_m]^T - [u, v]^T$

1. 地图点在 global 坐標為 P 变换到相机坐標

$$P' = T_{wc} * P = [x'\ y'\ z']^T$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = RX + t = R\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}$$

2. 歸一化平面.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c / z_c \\ y_c / z_c \end{bmatrix}$$

鏈式法推 Jacobian 矩陣.

$$\frac{\partial u}{\partial f} = d(k_0, k_1, r^2)x \quad \frac{\partial v}{\partial f} = d(k_0, k_1, r^2)y.$$

$$\frac{\partial x}{\partial x_c} = \frac{1}{z_c}, \quad \frac{\partial x}{\partial y_c} = 0, \quad \frac{\partial x}{\partial z_c} = -\frac{x_c}{z_c^2}$$

$$\frac{\partial y}{\partial x_c} = 0, \quad \frac{\partial y}{\partial y_c} = \frac{1}{z_c}, \quad \frac{\partial y}{\partial z_c} = -\frac{y_c}{z_c^2}$$