Introduction to Computer Vision S24 Assignment #1.

작성자 : 장원재 (2020-10290)

a) SVD of A : $A = UDV^T$ ($A_{m\times n}$, $U_{m\times m}$, $D_{m\times n}$, $V_{n\times n}$)

$A^TA = (VD^TU^T)(UDV^T) = VD^TDV^T$

Let's set $\left\{ \begin{array}{ll} v_1, v_2, \cdots, v_n & : \text{eigenvectors} \\ \lambda_1^2, \lambda_2^2, \cdots, \lambda_n^2 & : \text{eigenvalues} \end{array} \right\}$ of A^TA.

where $\lambda_i = D_{ii}$ & $v_i = V_i$ for $1 \le i \le n$.

We can express p as a linear combination of $v_1, \cdots, v_n$.

$\Rightarrow p = \sum_{i=1}^{n} a_i v_i$

$\operatorname{argmin}_p \|Ap\| = \operatorname{argmin}_p \|Ap\|^2 = \operatorname{argmin}_p p^TA^TAp.$

$p^T(A^TA)p = p^T \sum_{i=1}^{n} a_i (A^TA) P_i = \left(\sum_{i=1}^{n} a_i v_i^T\right)\left(\sum_{i=1}^{n} a_i \lambda_i^2 v_i\right)$

$= \sum_{i=1}^{n} \lambda_i^2 a_i^2 \|v_i\|^2$ ($\because v_i^T v_j = 0$ for $i \ne j$, $1 \le i, j \le n$. because V is orthogonal)

$\|p\| = 1 \rightarrow p^Tp = \left(\sum_{i=1}^{n} a_i v_i^T\right)\left(\sum_{i=1}^{n} a_i v_i\right)$

$= \sum_{i=1}^{n} a_i^2 \|v_i\|^2 = 1.$

$p^T(A^TA)p = \sum_{i=1}^{n-1} \lambda_i^2 a_i^2 \|v_i\|^2 + \lambda_n^2 \left(1 - \sum_{i=1}^{n-1} a_i^2 \|v_i\|^2\right)$

$= \sum_{i=1}^{n-1} (\lambda_i^2 - \lambda_n^2) a_i^2 \|v_i\|^2 + \lambda_n^2 \ge \lambda_n^2$

where λ_n is the smallest singular value. ($\lambda_i \ge 0$ for $1 \le i \le n$)

equal when $a_i = 0$ for $1 \le i \le n-1$. $\Rightarrow p = v_n / \|v_n\| = v_n$

$\therefore \operatorname{argmin}_p \|Ap\| = v$ where v is the singular vector corresponding to the smallest singular vector.

b) With a data of (u, v) and (X, Y, Z) pairs, let's determine the camera projection matrix P. Using the two equation that the pdf file gave, we can derive an equation below.

$$\underbrace{\begin{bmatrix} | & | & | & | & & -uX & -uY & -uZ & -u \\ X & Y & Z & 1 & 0 & | & | & | & | \\ | & | & | & | & & | & | & | & | \\ & & & & | \ | \ | \ | & | & | & | & | \\ & 0 & & & X \ Y \ Z \ 1 & -vX & -vY & -vZ & -v \\ & & & & | \ | \ | \ | & | & | & | & | \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{bmatrix}}_{=p} = 0$$

As shown in (a), use SVD to get p that minimizes $\|Ap\|$.

SVD Method

1 :

```
n, m = len(data), 12
A = np.zeros((2*n, m))
A[:n, 0:3] = data[['X', 'Y', 'Z']]
A[:n, 3] = 1.
A[:n, 8:11] = -data['u'].to_numpy().reshape(-1, 1) * data[['X', 'Y', 'Z']]
A[:n, 11] = -data['u']

A[n:, 4:7] = data[['X', 'Y', 'Z']]
A[n:, 7] = 1.
A[n:, 8:11] = -data['v'].to_numpy().reshape(-1, 1) * data[['X', 'Y', 'Z']]
A[n:, 11] = -data['v']
```

2 :

```
U, D, V = linalg.svd(A)
p_svd = V[-1]
P_svd = p_svd.reshape(3, 4)
```

1 : Build the matrix A.

2 : Do SVD and pick the singular vector corresponding to the smallest singular vector.

Then, we can get the projection metric P using SVD method.

```
- Matric P using SVD Method -
[[-3.09963996e-03 -1.46204548e-04  4.48497465e-04  9.78930678e-01]
 [-3.07018252e-04 -6.37193664e-04  2.77356178e-03  2.04144405e-01]
 [-1.67933533e-06 -2.74767684e-06  6.83964827e-07  1.32882928e-03]]
```

c) The goal is same as (b). But we use pseudo inverse method now.

By setting $m_{34} = 1$, the upper equation can be written as

$$\underbrace{\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -uX & -uY & -uZ \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -vX & -vY & -vZ \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{33} \end{bmatrix}}_{=q} = \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{=b}$$

$Aq = b \rightarrow q = A^{+}b$ where $A^{+} = (A^{T}A)^{-1}A^{T}$.

Vectorized form of P : $p = \begin{bmatrix} q \\ 1 \end{bmatrix}$

Pseudo Inverse Method

①

```
n, m = len(data), 11
A = np.zeros((2*n, m))
b = np.zeros(2*n)
A[:n, 0:3] = data[['X', 'Y', 'Z']]
A[:n, 3] = 1.
A[:n, 8:11] = -data['u'].to_numpy().reshape(-1, 1) * data[['X', 'Y', 'Z']]
b[:n] = data['u']

A[n:, 4:7] = data[['X', 'Y', 'Z']]
A[n:, 7] = 1.
A[n:, 8:11] = -data['v'].to_numpy().reshape(-1, 1) * data[['X', 'Y', 'Z']]
b[n:] = data['v']
```

②

```
A_plus = np.matmul(linalg.inv(np.matmul(A.T, A)), A.T)
q = np.matmul(A_plus, b)
```

③

```
p_pim = np.hstack([q, 1])
P_pim = p_pim.reshape(3, 4)
```

① : Build matrix A & b.

② : Get q using pseudo inverse of A

③ : Attach $m_{34} = 1$ to q and reshape it to get P.

So, now we get the projection matric P using Pseudo Inverse method.

```
- Matric P using Pseudo Inverse Method -
[[-2.33259099e+00 -1.09993074e-01  3.37413843e-01  7.36673912e+02]
 [-2.31050254e-01 -4.79506022e-01  2.08717636e+00  1.53627753e+02]
 [-1.26379607e-03 -2.06770916e-03  5.14635179e-04  1.00000000e+00]]
```

Evaluation)

The reconstructed u and v values using both projection matric are shown in the table below. The relative errors between the ground truth and the both reconstructed values are around 0.1%

The code is included in the zip file.

	svd_u	svd_v	pim_u	pim_v	real_u	real_v
0	879.435317	214.590586	879.432494	214.590473	880	214
1	43.289025	203.775431	43.289681	203.775025	43	203
2	269.694194	196.858653	269.694756	196.858345	270	197
3	885.640835	346.615978	885.643653	346.615045	886	347
4	745.196222	302.197516	745.192787	302.196688	745	302
5	943.305585	127.597382	943.307870	127.598864	943	128
6	476.274123	589.799211	476.281230	589.803546	476	590
7	419.075344	213.333377	419.074586	213.333068	419	214
8	316.518090	334.434324	316.517657	334.434389	317	335
9	783.113563	520.458968	783.124371	520.459984	783	521
10	236.091137	426.271621	236.089449	426.273137	235	427
11	665.619783	429.204462	665.629683	429.204800	665	429
12	655.755544	361.382945	655.755582	361.382394	655	362
13	426.882876	333.272990	426.882730	333.272812	427	333
14	410.181950	417.232853	410.183503	417.233638	412	415
15	746.291217	350.725799	746.288809	350.724932	746	351
16	433.960351	415.182804	433.963968	415.183568	434	415
17	524.690046	233.606901	524.689536	233.606534	525	234
18	715.799079	308.168242	715.794829	308.167403	716	308
19	602.398335	187.271044	602.396145	187.271025	602	187

```
- Relative error of SVD method -
u: 0.1267%
v: 0.1669%

- Relative error of Pseudo Inverse method -
u: 0.1269%
v: 0.1668%
```