Introduction to Computer Vision S24 Assignment #1

2019-12172 이재원, SNU

March 26, 2024

1 Problems

Camera Calibration - A 3 × 4 camera projection matrix $\mathbf{P}$ maps a world point $\mathbf{X_i} = [X_i Y_i Z_i 1]$ to its corresponding image point $\mathbf{x_i} = \omega[u_i v_i 1]$ such that $x_i = \mathbf{PX_i}$, which gives two equations w.r.t. the unknown parameters of $\mathbf{P} = [m_{ij}]$:

Given n ≥ 6 pairs of corresponding points, we have an over-determined system of equations, $\mathbf{AP} = 0$, where p is the vectorized form of P. Consider the SVD of A as $A = U\Sigma V^T$. Then the least square solution $\mathbf{p} = \text{argmin}_p ||\mathbf{AP}||$, $||\mathbf{p}|| = 1$ is the last column of $\mathbf{V}$, which is the singular vector corresponding to the smallest singular value. Then, you can construct the camera projection matrix P using p up to scale.

2.1Show that the solution p is the last column of $\mathbf{V}$ corresponding to the smallest singular value of A.

2.2 Given the following corresponding points, determine the camera projection matrix P using the SVD method.

2.3Determine P for the data in 2. using the pseudo inverse method.

2 Solution to Problems

2.1 Proof of last column of V

$$\begin{aligned}
\mathbf{p} &= \underset{p}{\text{argmin}}\, ||\mathbf{Ap}||^2 \\
&= \underset{p}{\text{argmin}}\, ||\mathbf{UDV^Tp}||^2 \\
&= \underset{p}{\text{argmin}}\, ||\mathbf{DV^Tp}||^2 \text{ (using orthonormality}\mathbf{U^U} = I)
\end{aligned}$$

$$\begin{aligned}
\text{substitute } y &= \mathbf{V^Tx} \\
\hat{y} &= \underset{y}{\text{argmin}}\, ||\mathbf{D}y||^2 \\
\text{if diagonals are sorted in decreasing order, } y &= [0, 0, \cdots, 1]^T \\
\mathbf{p} &= \mathbf{V}y
\end{aligned}$$

where y is the last column of $\mathbf{V}$

2.2 SVD

2.2.1 Code in python

```
U, S, Vh = np.linalg.svd(A, full_matrices=True)
Vh[-1, :].reshape(3, 4)
```

Here, the last column of $\mathbf{V}$ is same as the last row of $\mathbf{V^T}$

2.2.2 Results

$$\begin{bmatrix} 3.09963996e-03 & 1.46204548e-04 & -4.48497465e-04 & -9.78930678e-01 \\ 3.07018252e-04 & 6.37193664e-04 & -2.77356178e-03 & -2.04144405e-01 \\ 1.67933533e-06 & 2.74767684e-06 & -6.83964827e-07 & -1.32882928e-03 \end{bmatrix}$$

2.3 Pseudo-Inverse

2.3.1 Code in python

```
A_tild = A[:, :-1]

b = np.array(u_v)
b = b.reshape(-1)

A_dagger = np.linalg.pinv(A_tild)

p_tild = A_dagger @ b

p = np.append(p_tild, 1)
P = p.reshape(3, 4)
```

2.3.2 Results

$$\begin{bmatrix} -2.33259098e+00 & -1.09993113e & 01, 3.37413916e-01 & 7.36673920e+02 \\ -2.31050254e-01 & -4.79506029e & 01, 2.08717636e+00 & 1.53627756e+02 \\ -1.26379606e-03 & -2.06770917e & 03, 5.14635233e-04 & 1.00000000e+00 \end{bmatrix}$$

3 Discussion

To compare the result from 2.2.2 and 2.3.2, the result from SVD was changed in a form of the result of pseudo-inverse result ($i.e. m_{34} = 1$). The following python code changes SVD Result, setting $m_{34} = 1$.

```
constant = 1 / Vh[-1, -1]
p = Vh[-1, :] * constant
p.reshape(3, 4)
```

The SVD result after this conversion is

$$\begin{bmatrix} 2.33260962e+00 & -1.10025080e-01 & 3.37513233e-01 & 7.36686567e+02 \\ -2.31044166e-01 & -4.79515070e-01 & 2.08722206e+00 & 1.53627263e+02 \\ -1.26377057e-03 & -2.06774255e-03 & 5.14712341e-04 & 1.00000000e+00 \end{bmatrix}$$

which is almost identical to the result 2.3.2 using pseudo-inverse