Lab 1-2

Young Min Kim

2022.07.13.

3D LAB VISION

Table of Contents

1. Estimate camera matrix P

- In this example, we will implement what we have learned on class to estimate camera matrix given **2D points $\boldsymbol{x}$** on image and their corresponding **3D points $\boldsymbol{X}$**.

- In other words, given a set of matched points $\{\boldsymbol{X_i}, \boldsymbol{x_i}\}$ in homogeneous coordinate system and camera model

$$\begin{bmatrix} x \\ w \end{bmatrix} = f(X; p) = P \begin{bmatrix} X \\ 1 \end{bmatrix}$$

- We want to find the estimate of the camera matrix $P \in \mathbb{R}^{3\times4}$.

1. Estimate camera matrix P

- Write a function `estimate_pose` in `lab1-2/python/utils.py` file.
- Function `estimate_pose` estimates the camera matrix P given 2D and 3D points x and X.
- x is $2\times N$ matrix denoting the (x, y) coordinates of the N points on the image plane
- X is $3\times N$ matrix denoting the (x, y, z) coordinates of the corresponding points in the 3D world.

- Once you finish implementing this function, you can run the provided script `test_pose.py` to test your implementation.

1. Estimate camera matrix P

- Let's choose one of the 3D point $(X, Y, Z)^T$ and its projected 2D point $(x, y)^T$.
- Then, we have to find matrix $\mathbf{P}$ that satisfies

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{pmatrix} sx \\ sy \\ s \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- Rearranging the coefficients, we can represent x as

$$x = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

- and get

$$p_{11}X + p_{12}Y + p_{13}Z + p_{14} - p_{31}xX - p_{32}xY - p_{33}xZ - p_{34}x = 0$$

1. Estimate camera matrix P

- Similarly,

$$y = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

- and

$$p_{21}X + p_{22}Y + p_{23}Z + p_{24} - p_{31}yX - p_{32}yY - p_{33}yZ - p_{34}y = 0$$

- By this arrangement, we're able to set up linear regression to find elements of the matrix $\mathbf{P}$. Because $\mathbf{P}$ has an unknown scale, this equation have many possible solutions.

- We can fix the scale by setting the last element to 1 and find remaining coefficients with linear regression.

1. Estimate camera matrix P

- Let $\mathbf{P}_{\text{flat}} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{33} \end{pmatrix}$

$$A = \begin{pmatrix} X_1 & X_2 & \cdots & X_N & 0 & 0 & \cdots & 0 \\ Y_1 & Y_2 & \cdots & Y_N & 0 & 0 & \cdots & 0 \\ Z_1 & Z_2 & \cdots & Z_N & 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & X_1 & X_2 & \cdots & X_N \\ 0 & 0 & \cdots & 0 & Y_1 & Y_2 & \cdots & Y_N \\ 0 & 0 & \cdots & 0 & Z_1 & Z_2 & \cdots & Z_N \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ -x_1X_1 & -x_2X_2 & \cdots & -x_NX_N & -y_1X_1 & -y_2X_2 & \cdots & -y_NX_N \\ -x_1Y_1 & -x_2Y_2 & \cdots & -x_NY_N & -y_1Y_1 & -y_2Y_2 & \cdots & -y_NY_N \\ -x_1Z_1 & -x_2Z_2 & \cdots & -x_NZ_N & -y_1Z_1 & -y_2Z_2 & \cdots & -y_NZ_N \end{pmatrix}$$

$$B = \begin{pmatrix} x_1 & x_2 & \cdots & x_N & y_1 & y_2 & \cdots & y_N \end{pmatrix}$$

- What we want to minimize： $|\mathbf{P}_{\text{flat}}A - B|$

1. Estimate camera matrix P

$$
\begin{aligned}
|\mathbf{P}_{\text{flat}}A - B|^2 &= (\mathbf{P}_{\text{flat}}A - B)(\mathbf{P}_{\text{flat}}A - B)^T \\
&= \mathbf{P}_{\text{flat}}AA^T\mathbf{P}_{\text{flat}}^T - \mathbf{P}_{\text{flat}}AB^T - BA^T\mathbf{P}_{\text{flat}}^T + BB^T
\end{aligned}
$$

$$
\frac{\partial L}{\partial P} = -2BA^T + 2AA^T\mathbf{P}_{\text{flat}} = 0
$$

$$
\mathbf{P}_{\text{flat}} = BA^T(AA^T)^{-1}
$$

1. Estimate camera matrix P - example code

```python
def estimate_pose(x, X):
    """
    Estimate_pose computes the camera pose matrix P given 2D and 3D points.
    Args:
        x: 2D points with shape [2, N]
        X: 3D points with shape [3, N]
    Output:
        P: Camera matrix with shape [3, 4]
    """

    # Use linear regression for elements of P (assuming that P34 = 1)
    N = x.shape[1]
    X_homogeneous = np.concatenate((X, np.ones((1, N))), axis=0)
    X1 = X * x[0, :]
    X2 = X * x[1, :]
    A1 = np.concatenate((X_homogeneous, np.zeros((4, N)), -X1), axis=0)
    A2 = np.concatenate((np.zeros((4, N)), X_homogeneous, -X2), axis=0)
    A = np.concatenate((A1, A2), axis=1)
    B = np.concatenate((x[0, :], x[1, :]), axis=0).reshape(1, 2 * N)
    P_tmp = np.matmul(B, np.transpose(A))
    P_flat = np.matmul(P_tmp, np.linalg.inv(np.matmul(A, np.transpose(A)))).reshape(11,)
    P = np.zeros((3, 4))
    P[0, :] = P_flat[:4]
    P[1, :] = P_flat[4:8]
    P[2, :3] = P_flat[8:]
    P[2, 3] = 1
    return P
```

2. Estimate intrinsic/extrinsic parameters

- Write a function `estimate_params` in `lab1-2/python/utils.py` file.
- Function `estimate_params` estimates both intrinsic and extrinsic parameters from camera matrix.
- Once you finish your implementation, you can run the provided script `testKRt.py` to test your implementation.

- Hint)
  - Compute the camera center c.
  - Compute the intrinsic K and rotation R by using QR decomposition. K is a right upper triangle matrix while R is an orthonormal matrix.
  - Compute the translation by t = $-Rc$

2. Estimate intrinsic/extrinsic parameters

1. Obtain camera center c
   - Note that $\mathbf{P} = \mathbf{K}[\mathbf{R}|-\mathbf{Rc}] = [\mathbf{M}|\mathbf{Mc}]$
   - You can directly get $\boldsymbol{c}$ by multiplying $\mathbf{M}^{-1}$ to the last column of $\mathbf{P}$.

2. Obtain intrinsic K and rotation R
   - Note that $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$
   - Because $\mathbf{K}$ is upper triangular matrix and $\mathbf{R}$ is orthogonal matrix, we may use RQ decomposition to the left 3x3 matrix of $\mathbf{P}$.
   - (RQ Decomposition)
     1) Let $\mathrm{E} = \begin{pmatrix} 0 & \cdots & 1 \\ 0 & \iddots & 0 \\ 1 & \cdots & 0 \end{pmatrix}$ (flipped identity matrix, note $EE = I$, $E^{-1} = E$)
     2) Compute $\tilde{A} \coloneqq EA$
     3) Compute QR decomposition of $\tilde{A}^T = \tilde{Q}\tilde{R}$
     4) Set $Q \coloneqq E\tilde{Q}^T = \widetilde{\mathbf{R}}$
     5) Set $R \coloneqq E\tilde{R}^T E = \widetilde{\mathbf{K}}$
     - (Altogether, $RQ = E\tilde{R}^T EE\tilde{Q}^T = E\tilde{R}^T\tilde{Q}^T = E\left(\tilde{Q}\tilde{R}\right)^T = E\left(\tilde{A}^T\right)^T = E\tilde{A} = EEA = A$)

2. Estimate intrinsic/extrinsic parameters

- You have to check that determinant of rotation vector $\mathbf{R}$
- If $\mathbf{R}$ is a proper rotation matrix, det $\mathbf{R}$ should be 1
- Thus, what you have to do in final is to negate $\mathbf{R}$ if its determinant is negative.

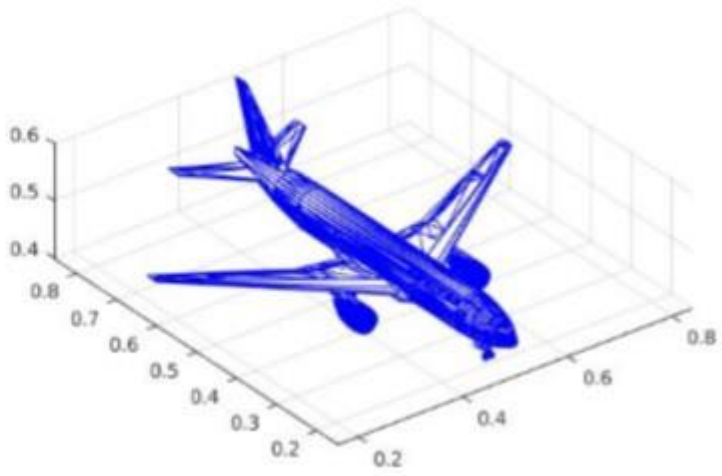
3. Compute translation
   - $t = -\mathbf{Rc}$

2. Estimate intrinsic/extrinsic parameters example code

```python
def estimate_params(P):
    """
    Estimate intrinsic matrix K, rotation matrix R, and translation t from given camera
    matrix P.
    Args:
        P: Camera matrix with shape [3,4]
    Output:
        (K, R, t) : intrinsic matrix K with shape [3,3],
                    rotation matrix R with shape[3,3],
                    translation t with shape [3,1]
    """
    A = P[:3, :3]
    c = np.matmul(-np.linalg.inv(A), P[:, 3])
    E = np.fliplr(np.eye(3))
    A0 = np.matmul(E, A)
    Q0, R0 = np.linalg.qr(np.transpose(A0))
    K_tmp = np.matmul(E, np.transpose(R0))
    K = np.matmul(K_tmp, E)
    R = np.matmul(E, np.transpose(Q0))
    if np.linalg.det(R) < 0:
        K = -K
        R = -R
    t = -np.matmul(R, c)
    return (K, R, t)
```

3. Project CAD model into image

- Now we will utilize what we have implemented to estimate the camera matrix from a real image, shown in below(left) and project the 3D object (CAD model), shown in bellow (right), back on to the image plane.



3. Project CAD model into image

- `lab1-1/python/projectCAD.py` script does the following:
  1. Load an image, a CAD model, 2D points x and 3D points X from `PnP.mat`
  2. Run `estimate_pose` and `estimate_params` to estimate camera matrix P, intrinsic matrix K, rotation matirx R, and translation t.
  3. Use estimated camera matrix P to project the given 3D points X onto the image.
  4. Plot the given 2D points x and the projected 3D points on screen.
  5. Draw the CAD model
  6. Project the CAD model's all vertices onto the image

3. Project CAD model into image