Direct Linear Transform (DLT)

Tuesday, November 23, 2021 3:59 PM

1. What is DLT?

Direct linear transform or DLT is used to do camera calibration, and camera localization. It's a technique that help you find the cameras location, and at the same time calibrate it.

When referring to DLT, people may mean 2 things.

a. It can mean mapping of 3D points to 2D image plane using the affine camera model.
b. Approach for estimating the parameters for the affine camera model.

Requirements :

- 6, 3D points in the scene / (single image).

DLT approach has 11 DoF, 5 intrinsic parameters, and 6 extrinsic parameters. Since, each known point results in 2 equations, x and y, the minimum number of known points must be 11/2 = 5.5 ~= 6.

Note : By affine camera model, I mean there are no non linear error in the image (distortion coefficients). These need to be handles separately, or at the same time as solving the DLT in certain scenarios.

2. Why do we use DLT?

We may use DLT in mainly two scenarios. First, when we are trying to find the locations of the 2D points on the image plane, if the 3D world points, and camera projection matrix (intrinsics and extrinsics) are known for the affine camera model. Secondly, it may be used to find the extrinsic, and intrinsic parameters if the 3D, and 2D points are known for the affine camera model.

3. How is DLT used?

It is relatively easy to use the DLT as long as you meet the requirements. We can find the projection matrix parameters by doing SVD on the set of homogenous linear equations.

4. What are pros and cons of DLT?

- There are no solution for projection matrix, if the all world points X, are located on a plane. Meaning that the Z coordinates of the M matrix (refer to notes in the DLT derivation) are zero. Due to this, we are not able to estimate the extrinsic parameters of the camera. If you are familiar with Zhang's method, you may know that this is also a good thing, if we only want to find the intrinsic parameters (calibration matrix) of the camera.
  - Matrix M is of rank 11, hence if Z is zero we have rank deficiency, with rank at most 9.

5. How does DLT compare to other methods?

- Zhang and DLT methods : they are very similar, however Zhang method, is used only to find calibration matrix, while DLT is for both calibration and pose of the camera. Zhang's method, also requires an object of known size (chessboard), while DLT requires control points.

6. How is DLT derived?

Direct Linear Transform (DLT) Derivation

$\bar{x}_p = P X_\omega$

$\bar{x}_p$ and X_o are known.
P is unknown.

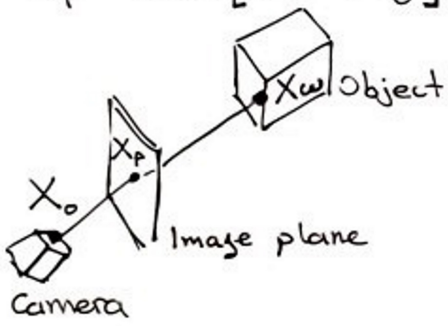
- $\bar{x}_p$: 2D coord. pixel
- P: projection matrix
- X_ω: 3D world coord.

Mapping can be further described by breaking P into intrinsic and extrinsic matrices.

$\bar{x}_p = KR[I_3 | -X_o] X_\omega$

$P = KR[I_3 | -X_o]$

- K: calibration matrix
- R: Rotation matrix
- I_3: Identity matrix
- X_o: camera origins



K is 3×3. X_o is 3×1.
R is 3×3. X_ω is 4×1.
I_3 is 3×3. $\bar{x}_p$ is 3×1.
$[I_3 | -X_o]$ is 3×4. P is 3×4.

As you can see, above is in homogeneous coordinates.

So P has 11 degrees of freedom. 5 in K, 6 in $R[I_3 | -X_o]$ (R: 3, $[I_3 | -X_o]$: 3).

*See notes on camera parameters to learn more...

To solve DLT we need at least 6 points

We need to estimate P knowing X_ω and $\bar{x}_p$.

$$P = \begin{vmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{vmatrix} = \begin{vmatrix} A^T \\ B^T \\ C^T \end{vmatrix} \rightarrow \bar{x}_{p_i} = \begin{bmatrix} A^T X_i \\ B^T X_i \\ C^T X_i \end{bmatrix}$$

i image/scene number.
Note, this is homogeneous.
X_i is 4×1, word coord.

$$A = \begin{vmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \end{vmatrix}, B = \begin{vmatrix} P_{21} \\ P_{22} \\ P_{23} \\ P_{24} \end{vmatrix} \quad \text{So, } \bar{x}_{p_i} = \begin{vmatrix} x_{p_i} \\ y_{p_i} \\ 1 \end{vmatrix} = \begin{vmatrix} u_i \\ v_i \\ \omega_i \end{vmatrix} = \begin{vmatrix} A^T X_i \\ B^T X_i \\ C^T X_i \end{vmatrix}$$

$x_{p_i} = u_i/\omega_i$

$y_{p_i} = v_i/\omega_i$

$$C = \begin{vmatrix} P_{31} \\ P_{32} \\ P_{33} \\ P_{34} \end{vmatrix}$$

$$\rightarrow x_{p_i} = \frac{A^T X_i}{C^T X_i} \text{ and } y_{p_i} = \frac{B^T X_i}{C^T X_i}$$

$$x_{p_i} = \frac{A^T X_i}{C^T X_i} \longrightarrow x_{p_i} C^T X_i - A^T X_i = 0$$

Same with $y_{p_i} \rightarrow \quad y_{p_i} C^T X_i - B^T X_i = 0$

This leads us to a system of linear equations, with parameters A, B and C.

$$-X_i^T A \quad + x_{p_i} X_i^T C^T = 0 \quad (1)$$
$$-X_i^T B \quad + y_{p_i} X_i^T C^T = 0$$

We can further vectorize P into $\rho = vec(P^T) = \begin{vmatrix} A \\ B \\ C \end{vmatrix}$ to estimate elements of ρ

Then we rewrite (1) to $a_{x_i}^T \rho = 0$ and $a_{y_i}^T \rho = 0$

$$\rho = \begin{vmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{vmatrix} \begin{matrix} \} A \\ \\ \\ \\ \} B \\ \\ \\ \\ \} C \end{matrix}$$

$$a_{x_i}^T = (-X_i^T, 0^T, x_{p_i} X_i^T) \qquad a_{y_i}^T = (0^T, -X_i^T, y_{p_i} X_i^T)$$

Note again, we are still in homogeneous coord.

$$a_{x_i}^T = (-X_i, -Y_i, -Z_i, -1, 0, 0, 0, 0, x_{p_i} X_i, x_{p_i} Y_i, x_{p_i} Z_i, x_{p_i}) \quad (2)$$

Note, i kept $-X_i$ the same variable in (2), but they mean different things. The X_i in (2) is a point in x-axis, and the X_i every where else is a matrix.

We can write the same for $a_{y_i}^T$.

$$a_{y_i}^T = (0, 0, 0, 0, -X_i, -Y_i, -Z_i, -1, y_{p_i} X_i, y_{p_i} Y_i, y_{p_i} Z_i, y_{p_i})$$

Now that we know how $a_{x_i}^T$, and ρ breakdown, we need to solve the equations.

$$\begin{bmatrix} a_{x_1}^T \\ a_{y_1}^T \\ \vdots \end{bmatrix} = M, \; M \text{ is of size } 2I \times 12. \; (I \text{ is number of points})$$

$M\rho = 0$, now in real world we will never get zero. Instead we try to get close to 0.

To solve $M\rho = 0$, we use singular value decomposition (SVD).

We choose the values from singular vector belonging to singular value of 0, as ρ They will bring the $M\rho$ solution close to zero.

$$M\rho = err \sim 0$$

Now that we found P, how do we obtain K, R, X_0?

$P = [KR | -KRX_0] = [H|h]$ where $H = KR$, and $h = -KRX_0$.

matrix 3×3 ↙ ↘ vector 3×1

$X_0 = -H^{-1}h$, and KR can be found using QR decomposition.

$qr\_decomp(H^{-1}) = q\,r = R^T K^{-1}$, $q = R^T$, and $r = K^{-1}$.

H is a homogenous matrix, hence, so is K.
So we must do the following to normalize it $K \leftarrow \frac{1}{K_{33}} K$ ← Homogenous

H^{-1} decomposition also leads to a K with positive diagonal elements.
To resolve it, we use rotation $R_{(z,\pi)} = \begin{vmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$, 180 degree flip.

$K \leftarrow KR_{(z,\pi)}$ and $R \leftarrow RR_{(z,\pi)}$

- This is not a statistically optimal solution. We donot account for errors in known points, nor it P approximation. To resolve it, we can use least squares approach to further refine our initial guess.

Note: to clarify H and h, $P = \left|\begin{matrix} H_{11} & H_{12} & H_{13} & h_{14} \\ H_{21} & H_{22} & H_{23} & h_{24} \\ H_{31} & H_{32} & H_{33} & h_{34} \end{matrix}\right| = [H|h]$.

Matrix (under $H_{11}\ H_{12}\ H_{13}\ H_{21}\ H_{22}\ H_{23}\ H_{31}\ H_{32}\ H_{33}$) vector (under $h_{14}\ h_{24}\ h_{34}$)

Not too difficult. Now try to implement it, yourself. ☺

7. How is DLT implemented?

https://github.com/ArtursPark/AP_3D_Reconstruction

8. How do you test DLT?

One simple way, is to compare true intrinsic and extrinsic parameters, to computed parameters.

Second way, is to use 3D object points, and apply the project matrix (intrinsic and extrinsic combined), to obtain the image points. Then we can compare the computed image points with true values. This way is specifically useful, if you already know image, and object points and are estimating the intrinsic, and extrinsic parameters. Then, you can just go back, and apply the computed project to obtain the image points.

9. Resources :

- Video Lectures :
  - https://www.youtube.com/watch?v=Fdwa0UEJ_F8&ab_channel=CyrillStachniss
  - https://www.youtube.com/watch?v=3NcQbZu6xt8&list=PLgnQpQtFTOGRYjqjdZxTEQPZuFHQa7O7Y&index=39&ab_channel=CyrillStachniss
- Notes :
  - http://www.kwon3d.com/theory/dlt/dlt.html
  - https://en.wikipedia.org/wiki/Direct_linear_transformation