Multiple-view Geometry 1

Feature Matching, Pinhole camera model, Camera modeling, Direct Linear Transform

Rahul Sajnani

RRC Summer Sessions 2020

Table of Contents I

Table of Contents II

Overview

Multiple-view geometry

The study of geometric relations between multiple views (images) of a 3D scene, that are related to the camera motion and calibration as well as the scene structure.

- Single-view geometry: Camera modeling, camera calibration
- Two-view geometry: Epipolar geometry, projective reconstruction, auto-calibration
- N-view geometry: Structure from motion

Overview

Applications

Figure: Multiple-view reconstruction

Figure: Stitching

Feature Matching and Extraction

Feature Matching and Extraction

Preliminaries

- **Detector algorithms:** Algorithms that find interest points in an image. The interest points are local maximum of some function. Eg. Harris corner detector
- **Descriptor algorithms:** Algorithms that compute vectors describing the interest points. Eg. SIFT descriptor.

Note: SIFT includes a detector as well as a descriptor.

- **Image gradients:** The gradient of an image is a vector of its partials:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

  To obtain gradients we convolve the image I with the following kernels:

$$I_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * I \qquad I_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * I$$

  I_x and I_y are gradients of image I along x and y directions respectively.

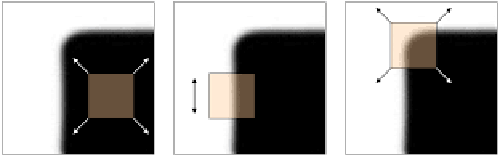
- **Gradient magnitude and direction:**

$$I_{mag} = \sqrt[2]{I_x^2 + I_y^2}$$

$$I_\theta = tan^{-1}(\frac{I_y}{I_x})$$

Feature Matching and Extraction

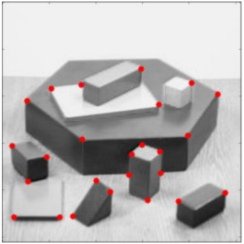
Harris corner detector



"flat" region: no change in all directions

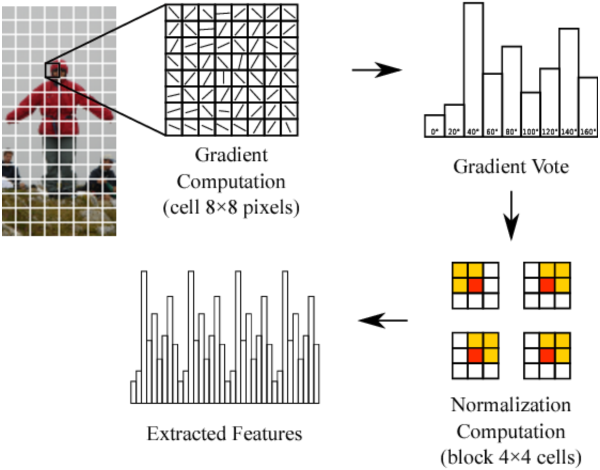
"edge": no change along the edge direction

"corner": significant change in all directions



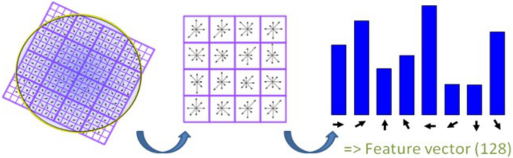
Feature Matching and Extraction

Example of a descriptor



Feature Matching and Extraction

Scale Invariant Feature Transform - SIFT



Pinhole camera model

Pinhole camera model

Introduction

Pinhole camera model mathematically relates 3D coordinates and its projection to the image coordinates of an **ideal** camera.

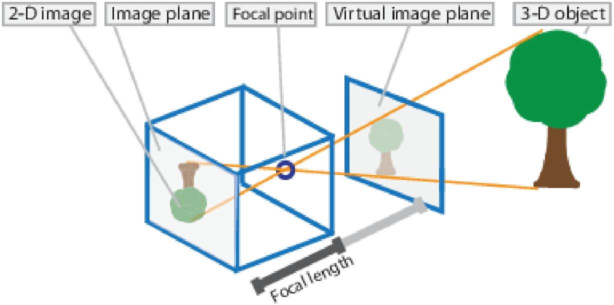


Figure: Pinhole camera

Pinhole camera model

Definitions

- **Camera centre (C):** The centre of projection is called the camera centre or the optical centre.
- **Image plane:** The plane Z=f (in camera's frame of reference) where the world is imaged.
- **Principal axis:** The line from the camera centre perpendicular to the image plane.
- **Principal point (p):** Centre of the image.

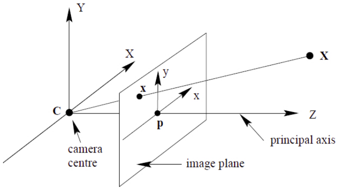


Figure: Pinhole camera model

In a pinhole camera model, a point in space with coordinates $(X, Y, Z)^T$ is mapped to the point $\left(\frac{fX}{Z}, \frac{fY}{Z}, f\right)^T$ on the image plane using similarity of triangles.

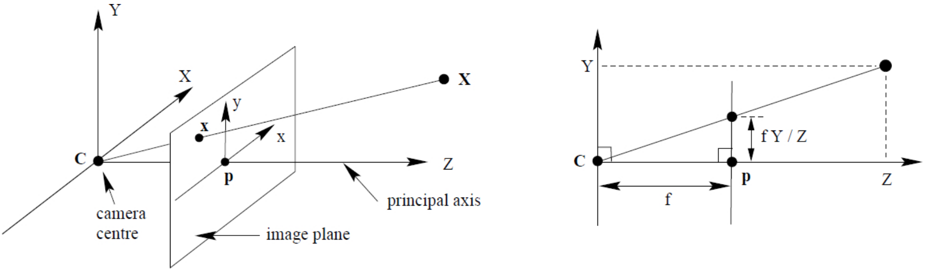


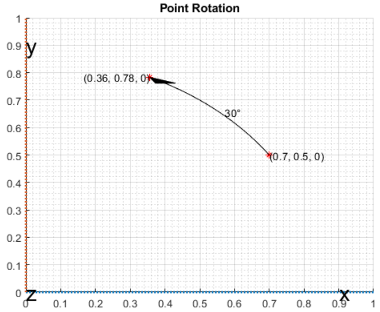
Figure: Pinhole camera model

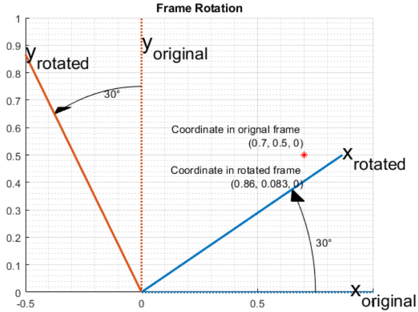
Camera modeling

- T^{World}_{Camera} transforms points from **camera to world**

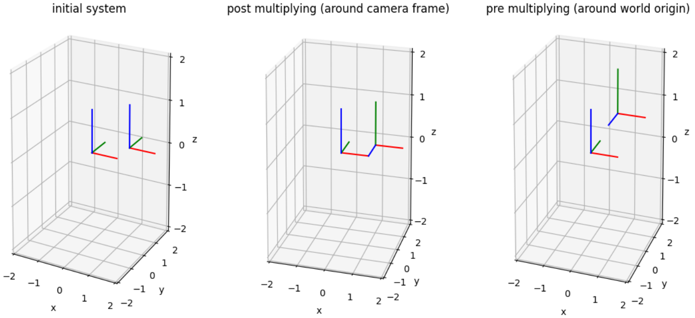
$$^{W}X = T^{World}_{Camera} \, {}^{C}X$$

- Frame and point transforms are **inverse** of each other.





- Post multiplying **frame transform** apply transformation around the **current frame**.
- Pre multiplying **frame transform** apply transformation around the **world frame**.



Q) The initial transformation matrix for a camera is T_i. We wish to transform the **frame** by applying the following transforms in the given order:

- T_1 from camera frame
- T_2 from camera frame
- T_3 from world frame
- T_4 from world frame

What is the final transformation matrix that transforms from **world frame to camera frame**?

Q) The initial transformation matrix for a camera is T_i. We wish to transform the **frame** by applying the following transforms in the given order:

- T_1 from camera frame
- T_2 from camera frame
- T_3 from world frame
- T_4 from world frame

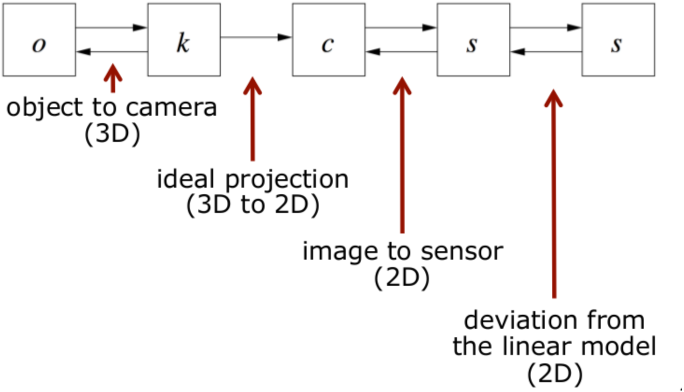
What is the final transformation matrix that transforms from **world frame to camera frame**?

Ans) $T_4 * T_3 * T_i * T_1 * T_2$

1. World/object coordinate system S_o
   written as: $[X, Y, Z]^\mathsf{T}$ ← **no index means object system**
2. Camera coordinate system S_k
   written as: $[\,^kX,\ ^kY,\ ^kZ]^\mathsf{T}$
3. Image (plane) coordinate system S_c
   written as: $[\,^cx,\ ^cy]^\mathsf{T}$
4. Sensor coordinate system S_s
   written as: $[\,^sx,\ ^sy]^\mathsf{T}$

Camera modeling

Modeling transform from world to sensor



Transforming **world frame to camera frame** we get the following:

$$\begin{bmatrix} I_{3x3} & X_O \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \hat{R} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \hat{R} & X_O \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Here, X_O is camera center in the world frame. We first translate with X_O and then apply rotation $\hat{R}$.

Inverting the above transformation matrix we obtain the transformation matrix that transforms **points from world to camera** T_W^k.

$$T_W^k = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} I_{3x3} & -X_O \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} R & -RX_O \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Note: R $= \hat{R}^{-1} = \hat{R}^T$

Camera modeling

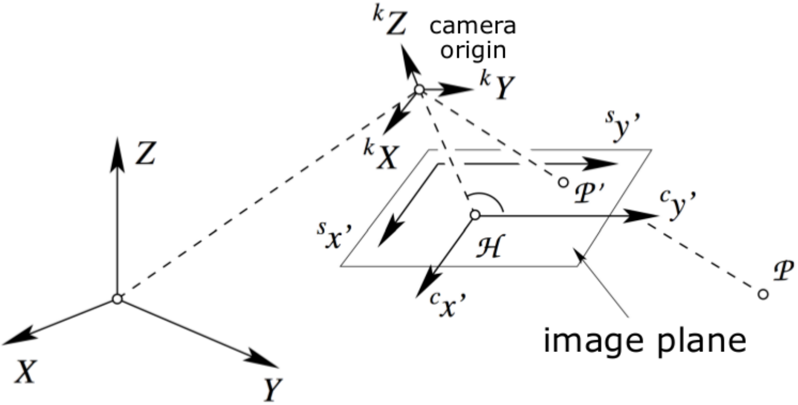
Modeling transform from world to sensor

$$\begin{bmatrix} {}^{k}X_p \\ {}^{k}Y_p \\ {}^{k}Z_p \end{bmatrix} = R \left[I_{3x3} | - X_O \right]_{3x4} \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix}$$

P is a point in world frame.

Camera modeling

Modeling transform from world to sensor



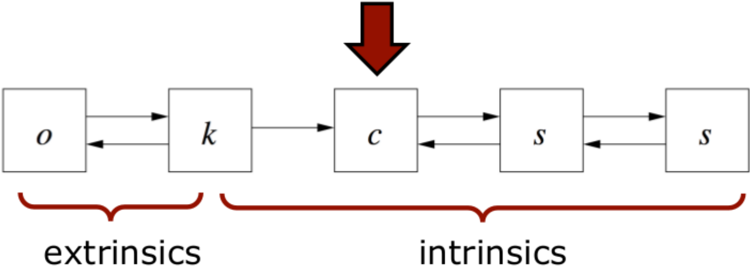
Camera modeling

Modeling transform from world to sensor

Considering an ideal perspective projection we get the following:

$$\begin{bmatrix} {}^c u_p \\ {}^c v_p \\ {}^c w_p \end{bmatrix} = \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^k X_p \\ {}^k Y_p \\ {}^k Z_p \end{bmatrix} = \begin{bmatrix} c\,{}^k X_p \\ c\,{}^k Y_p \\ {}^k Z_p \end{bmatrix}$$

c is the camera constant. **c** is the perpendicular distance along principal axis from the camera center to the center of the image plane.



After scaling the point we obtain:

$$\begin{bmatrix} {}^c x_p \\ {}^c y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{{}^c u_p}{{}^c w_p} \\ \frac{{}^c v_p}{{}^c w_p} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{c\,{}^k X_p}{{}^k Z_p} \\ \frac{c\,{}^k Y_p}{{}^k Z_p} \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^k X_p \\ {}^k Y_p \\ {}^k Z_p \end{bmatrix}$$

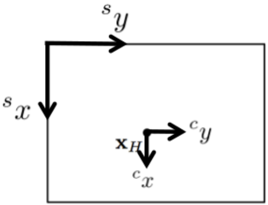
We define ${}^c K$ as:

$${}^c K = \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Next, we transform the coordinates from image plane to the sensor system.

- The origin of the sensor system is not at the principal point
- Compensation through a shift

$$ {}^{s}\mathrm{H}_{c} = \begin{bmatrix} 1 & 0 & x_H \\ 0 & 1 & y_H \\ 0 & 0 & 1 \end{bmatrix} $$



- Scale difference m in x and y
- Sheer compensation s (for digital cameras, we typically have $s \approx 0$)

$$ {}^s\mathsf{H}_c = \begin{bmatrix} 1 & s & x_H \\ 0 & 1+m & y_H \\ 0 & 0 & 1 \end{bmatrix} $$

- Finally, we obtain

$$ {}^s\mathbf{x} = {}^s\mathsf{H}_c \, {}^c\mathsf{K}R[I_3 | -\boldsymbol{X}_O]\mathbf{X} $$

Number of parameters?

Often, the transformation ${}^{s}\mathsf{H}_c$ is combined with the calibration matrix ${}^{c}\mathsf{K}$, i.e.

$$
\begin{aligned}
\mathsf{K} &\doteq {}^{s}\mathsf{H}_c \, {}^{c}\mathsf{K} \\
&= \begin{bmatrix} 1 & s & x_H \\ 0 & 1+m & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Camera modeling

Modeling transform from world to sensor

- The homogeneous projection matrix

  $$\mathsf{P} = \mathsf{K}R[I_3 | -\boldsymbol{X}_O]$$

- contains **11 parameters**
  - 6 extrinsic parameters: $R, \boldsymbol{X}_O$
  - 5 intrinsic parameters: c, x_H, y_H, m, s

Direct Linear Transform

Task: Estimate the 11 elements of P (11 parameters - 3 rotation, 3 translation, 5 intrinsics).
Given: 3D coordinates of object points in world frame X_i and their image pixel coordinates x_i

$$\mathrm{x}_i = PX_i$$

$$x_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \cdot X_i \tag{1}$$

Direct Linear Transform

Algorithm

$$\mathbf{x}_i = \underset{3\times 4}{\mathsf{P}}\,\mathbf{X}_i = \begin{bmatrix} \boxed{p_{11} \quad p_{12} \quad p_{13} \quad p_{14}} \\ \boxed{p_{21} \quad p_{22} \quad p_{23} \quad p_{24}} \\ \boxed{p_{31} \quad p_{32} \quad p_{33} \quad p_{34}} \end{bmatrix} \mathbf{X}_i$$

$$= \begin{bmatrix} \boxed{\mathbf{A}^\top} \\ \boxed{\mathbf{B}^\top} \\ \boxed{\mathbf{C}^\top} \end{bmatrix} \mathbf{X}_i$$

Direct Linear Transform

Algorithm

$$\mathbf{x}_i = \underset{3\times 4}{\mathsf{P}}\mathbf{X}_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \mathbf{X}_i$$

$$= \begin{bmatrix} \mathbf{A}^\top \\ \mathbf{B}^\top \\ \mathbf{C}^\top \end{bmatrix} \mathbf{X}_i$$

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} \mathbf{A}^\top \mathbf{X}_i \\ \mathbf{B}^\top \mathbf{X}_i \\ \mathbf{C}^\top \mathbf{X}_i \end{bmatrix}$$

Direct Linear Transform

Algorithm

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} \mathbf{A}^\top \mathbf{X}_i \\ \mathbf{B}^\top \mathbf{X}_i \\ \mathbf{C}^\top \mathbf{X}_i \end{bmatrix}$$

$$x_i = \frac{u_i}{w_i} = \frac{\mathbf{A}^\top \mathbf{X}_i}{\mathbf{C}^\top \mathbf{X}_i} \qquad y_i = \frac{v_i}{w_i} = \frac{\mathbf{B}^\top \mathbf{X}_i}{\mathbf{C}^\top \mathbf{X}_i}$$

$$x_i = \frac{u_i}{w_i} = \frac{\mathbf{A}^\top \mathbf{X}_i}{\mathbf{C}^\top \mathbf{X}_i} \quad \Rightarrow \quad x_i\,\mathbf{C}^\top \mathbf{X}_i - \mathbf{A}^\top \mathbf{X}_i = 0$$

$$y_i = \frac{v_i}{w_i} = \frac{\mathbf{B}^\top \mathbf{X}_i}{\mathbf{C}^\top \mathbf{X}_i} \quad \Rightarrow \quad y_i\,\mathbf{C}^\top \mathbf{X}_i - \mathbf{B}^\top \mathbf{X}_i = 0$$

Leads to an system of equation, which is **linear in the parameters A, B and C**

$$\begin{array}{lll} -\mathbf{X}_i^\top \mathbf{A} & & +x_i\,\mathbf{X}_i^\top \mathbf{C} = 0 \\ & -\mathbf{X}_i^\top \mathbf{B} & +y_i\,\mathbf{X}_i^\top \mathbf{C} = 0 \end{array}$$

- Collect the elements of P within a vector $\boldsymbol{p}$

$$\boldsymbol{p} = (p_k) = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix} = \mathrm{vec}(\mathsf{P}^\mathsf{T})$$

rows of P as column-vectors, one below the other (12x1)

Direct Linear Transform

Algorithm

- Rewrite $$\begin{aligned} -\mathbf{X}_i^\top \mathbf{A} \qquad\qquad & +x_i\,\mathbf{X}_i^\top \mathbf{C} = 0 \\ -\mathbf{X}_i^\top \mathbf{B} \quad & +y_i\,\mathbf{X}_i^\top \mathbf{C} = 0 \end{aligned}$$
- as $$\begin{aligned} \boldsymbol{a}_{x_i}^\top \boldsymbol{p} &= 0 \\ \boldsymbol{a}_{y_i}^\top \boldsymbol{p} &= 0 \end{aligned}$$
- with

$$\begin{aligned} \boldsymbol{p} &= (p_k) = \mathrm{vec}(\mathsf{P}^\top) \\ \boldsymbol{a}_{x_i}^\top &= (-\mathbf{X}_i^\top, \mathbf{0}^\top, x_i\mathbf{X}_i^\top) \\ &= (-X_i, -Y_i, -Z_i, -1, 0, 0, 0, 0, x_iX_i, x_iY_i, x_iZ_i, x_i) \\ \boldsymbol{a}_{y_i}^\top &= (\mathbf{0}^\top, -\mathbf{X}_i^\top, y_i\mathbf{X}_i^\top) \\ &= (0, 0, 0, 0, -X_i, -Y_i, -Z_i, -1, y_iX_i, y_iY_i, y_iZ_i, y_i) \end{aligned}$$

Direct Linear Transform
Algorithm

$$\boldsymbol{a}_{x_i}^{\top}\boldsymbol{p} = (-X_i,\ -Y_i,\ -Z_i,\ -1, 0,\ 0,\ 0,\ 0, x_iX_i,\ x_iY_i,\ x_iZ_i,\ x_i) \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix}$$

Direct Linear Transform

Algorithm

$$\boldsymbol{a}_{x_i}^\mathsf{T}\boldsymbol{p} = (-X_i,\ -Y_i,\ -Z_i,\ -1, 0,\ 0,\ 0,\ 0, x_iX_i,\ x_iY_i,\ x_iZ_i,\ x_i) \begin{array}{cc} \left[\begin{array}{c} \boxed{\begin{array}{c} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \end{array}} \\ \boxed{\begin{array}{c} p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \end{array}} \\ \boxed{\begin{array}{c} p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{array}} \end{array}\right] & \begin{array}{c} \boldsymbol{A} \\ \\ \\ \\ \boldsymbol{B} \\ \\ \\ \\ \boldsymbol{C} \end{array} \end{array}$$

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \\ \boldsymbol{C} \end{bmatrix}$$

Direct Linear Transform

Algorithm

$$\boldsymbol{a}_{x_i}^\top \boldsymbol{p} = \left(\boxed{-X_i,\ -Y_i,\ -Z_i,\ -1},\ \boxed{0,\ 0,\ 0,\ 0},\ \boxed{x_iX_i,\ x_iY_i,\ x_iZ_i,\ x_i} \right) \left[\begin{array}{c} p_{11}\\ p_{12}\\ p_{13}\\ p_{14}\\ p_{21}\\ p_{22}\\ p_{23}\\ p_{24}\\ p_{31}\\ p_{32}\\ p_{33}\\ p_{34} \end{array}\right] \begin{array}{l} \\ \boldsymbol{A}\\ \\ \\ \\ \boldsymbol{B}\\ \\ \\ \\ \boldsymbol{C}\\ \\ \\ \end{array}$$

$$= \left(-\mathbf{X}_i^\top,\ \mathbf{0},\ x_i\mathbf{X}_i^\top \right) \begin{bmatrix} \boldsymbol{A}\\ \boldsymbol{B}\\ \boldsymbol{C} \end{bmatrix}$$

Direct Linear Transform
Algorithm

$$
\begin{aligned}
\boldsymbol{a}_{y_i}^{\mathsf{T}}\boldsymbol{p} &= (0,\ 0,\ 0,\ 0,\ -X_i,\ -Y_i,\ -Z_i,\ -1,\ y_iX_i,\ y_iY_i,\ y_iZ_i,\ y_i) \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} \\
&= \left(\quad \mathbf{0}, \quad -\mathbf{X}_i^{\mathsf{T}}, \quad y_i\mathbf{X}_i^{\mathsf{T}} \quad \right) \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \\ \boldsymbol{C} \end{bmatrix} \\
&= -\mathbf{X}_i^{\mathsf{T}}\mathbf{B} + y_i\,\mathbf{X}_i^{\mathsf{T}}\mathbf{C}
\end{aligned}
$$

Direct Linear Transform

Algorithm

- For each point, we have

$$\boldsymbol{a}_{x_i}^\mathsf{T} \boldsymbol{p} = 0$$
$$\boldsymbol{a}_{y_i}^\mathsf{T} \boldsymbol{p} = 0$$

- Collecting everything together

$$\begin{bmatrix} \boldsymbol{a}_{x_1}^\mathsf{T} \\ \boldsymbol{a}_{y_1}^\mathsf{T} \\ \cdots \\ \boldsymbol{a}_{x_i}^\mathsf{T} \\ \boldsymbol{a}_{y_i}^\mathsf{T} \\ \cdots \\ \boldsymbol{a}_{x_I}^\mathsf{T} \\ \boldsymbol{a}_{y_I}^\mathsf{T} \end{bmatrix} \boldsymbol{p} = \underset{2I\times 12}{\mathsf{M}} \; \underset{12\times 1}{\boldsymbol{p}} \stackrel{!}{=} 0$$

Number of points to consider?

To solve for **p** we have the following optimization problem:

$$\min_{p} (M \cdot p)^T \cdot (M \cdot p) - \lambda \left(p^T \cdot p - 1 \right) \tag{2}$$

Differentiating w.r.t **p**

$$2 \cdot M^T \cdot M \cdot p - 2\lambda \cdot p = 0 \tag{3}$$

$$(M^T \cdot M) \cdot p = \lambda \cdot p \tag{4}$$

p is the last eigen vector of $M^T M$. As the last eigen vector has the lowest variance along its axis. This gives the least value of square error.

Direct Linear Transform

Questions

- When does DLT fail?
- How to decompose the projection matrix to obtain intrinsics and extrinsics?

Study material

References

- Multiple-View Geometry - Richard Hartley, Andrew Zisserman
- Invitation to 3D Vision - Yi Ma et al.
- https://hedivision.github.io/Pinhole.html
- https://in.mathworks.com/help/fusion/examples/rotations-orientation-and-quaternions.html

Study material

Courses

- Photogrammetry I & II - Cyril Stachniss
- Multiple-view Geometry - Daniel Cremers
- Vision algorithms for mobile robots - Davide Scaramuzza
- SIFT detection and matching - https://www.youtube.com/watch?v=NPcMS49V5hg
- Harris corner detector - https://www.youtube.com/watch?v=_qgKQGsuKeQ