COSC470 Cameras and Transforms

COSC 470: Special Topic
Computer Vision | 3D Reconstruction
Steven Mills

Vector and Matrix Review

- A vector is a 1D array of numbers
$$\boldsymbol{u} = [1 \quad 2 \quad 3]^T$$
- A matrix is a 2D array of numbers
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$
- Element-wise addition and subtraction, and scalar multiplication if same size
$$2\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} -3 & 2 \\ 1 & 0 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 5 & 1 \\ -2 & 3 \end{bmatrix}$$

Matrix Multiplication

- Multiplication is a little more complex
- If matrix size is (rows × columns) need inner dimensions to agree, outer dimensions give the size of the result
$$\frac{\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}\begin{bmatrix} g & h \\ i & j \\ k & l \end{bmatrix}}{(2 \times \mathbf{3}) \times (\mathbf{3} \times 2) = (2 \times 2)} \qquad \frac{\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}\begin{bmatrix} g & h \\ i & j \end{bmatrix}}{(2 \times \mathbf{3}) \times (\mathbf{2} \times 2)\text{Not OK}}$$
- Treat vectors as *n*×1 (column) matrices

Multiplying Vectors

- Inner (dot) and outer product as matrices
$$\boldsymbol{u} \cdot \boldsymbol{v} = [a \quad b \quad c]\begin{bmatrix} d \\ e \\ f \end{bmatrix} \qquad \boldsymbol{u} \otimes \boldsymbol{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}[d \quad e \quad f]$$
- Cross product of 3D vectors
$$\boldsymbol{u} \times \boldsymbol{v} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \times \begin{bmatrix} d \\ e \\ f \end{bmatrix} = \begin{bmatrix} bf - ce \\ cd - af \\ ae - bd \end{bmatrix}$$

Transformations in 2D

- Scaling
$$s\boldsymbol{v} = s\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} sx \\ sy \end{bmatrix}$$
- Translation
$$\boldsymbol{v} + \boldsymbol{t} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$
- Rotation
$$Rv = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Co-ordinates

- Using different methods makes combining transformations difficult
- Homogeneous co-ordinates make this easier by viewing a 2D point as a set of 3D points
$$\begin{bmatrix} x \\ y \end{bmatrix} \equiv k\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, k \neq 0$$
- All operations now become matrix multiply

Homogeneous Transformations

- Scaling:
  $S\boldsymbol{v} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Translation:
- $T\boldsymbol{v} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Rotation
  $R\boldsymbol{v} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
  where
  $c = \cos(\theta)$
  $s = \sin(\theta)$
- Can now combine operations easily

Hierarchy of Transforms – 2D

- Translation ($x' = x + t$), 2 degrees of freedom
  - Orientation, length, angles, parallelism, and straight lines preserved
- Rigid/Euclidean ($x' = Rx + t$), 3 DoF
  - Orientation no longer preserved
- Similarity ($x' = sRx + t$), 4 DoF
  - Length no lonver preserved

Hierarchy of Transforms – 2D

- Affine transform ($x' = Ax + t$), 6 DoF
  - Angles no longer preserved
- Perspective transform/Homography ($x' = Hx$)
  - Uses homogenous co-ordinates
  - 8 degrees of freedom
  - Only straight lines preserved

Transformations in 3D

- Scaling, Translation
  $S = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Rotation about each axis

$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_x = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3D Rotations

- The three axis-based matrices may be combined, giving a 3x3 rotation matrix
  - This has 9 elements, but only 3 degrees of freedom – not always ideal
- Quaternions are often used in graphics
- We'll use axis-angle representation
  - Rotation by some angle around some angle
  - If use normalised axis, can represent with 3 values

3D-2D Projection

- A camera takes 3D points to 2D points
- This is represented as a projection matrix

$$x = \begin{bmatrix} x \\ y \end{bmatrix} \equiv k \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = PX$$

- This represents a camera with unit focal length, at the origin, looking along +Z

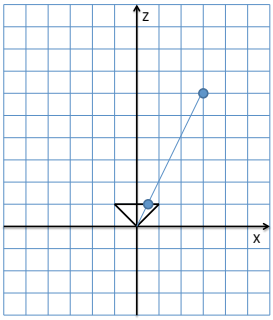
Pinhole Camera Model

- A more general model is
$$x = PX = K[R|t]X = KR[I|-C]X$$
- *R* and *t* give the orientation and translation from camera and world co-ordinates
- *K* is a calibration matrix, often of the form
$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$
where *f* is the focal length and (c_x, c_y) is the optical centre of the image

Camera Example

- We'll work in the X-Z plane, so y = 0 always
- Basic camera projecting point $[3, 0, 6]^T$

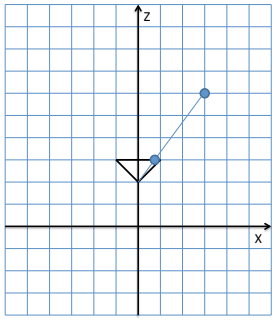
$$k\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



Camera Example

- Translate camera to C = $[0,0,2]^T$

$$k\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



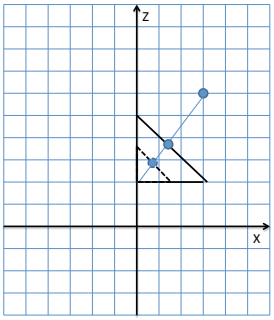
Camera Example

- Rotate camera about Y by 45 degrees

$$R_y(45) = \begin{bmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 0 & 1 & 0 \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix}$$

- Focal length of 2 units

$$K = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Camera Calibration

- Calibration is the process of estimating a camera's parameters
- Intrinisic parameters
  - Calibration matrix – focal length, optical centre
  - Lens distortion – radial model $k_1r^2 + k_2r^4 + k_3r^6 + \ldots$
- Extrinsic parameters
  - Location and orientation of the camera

The Eigen Matrix Library

- A C++ library for linear algebra
- Makes heavy use of templates:
`Eigen::Matrix<double, 3, 4> P;`
- Header-only library – nothing to link to
- Has various sub-libraries
  - Eigenvectors and matrix decompositions
  - Geometry (Rotations, etc.)