Tutorial
Applications involving Linear Algebra

SIDDHARTH TANWAR

ENGR108 – Introduction to Matrix Methods
Overview

• Pose of an object
• Image Formation
• Neural Networks
Where is an object?

**Example:** The quadrotor is flying through an urban area. It has a mounted camera which can see a known building.

**GOAL:** Obtain the pose of the quadrotor.
What is Pose?
What is Pose?

We define the **pose of a body** with respect to a **reference frame**

Pose

- Position \( t \)
- Orientation \( R \)
Position

\[ \mathbf{p} = (x_0, y_0) \]

\[ \mathbf{p} = \mathbf{p}_0 + \mathbf{p} \]

(Slide adopted from AA173 Spring’20)
Orientation

2 forms:

1. Angles - yaw, pitch & roll

2. Rotation Matrices
Rotation Matrix

\[ \hat{\mathbf{x}}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \]

\[ \hat{\mathbf{y}}_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \]

\[ \hat{\mathbf{z}}_1 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \]

\[ R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \]
Rotation Matrix Transformation

\[
\begin{align*}
\mathbf{P}_1 &= a \mathbf{\hat{x}}_1 + b \mathbf{\hat{y}}_1 + c \mathbf{\hat{z}}_1 \\
R \mathbf{P}_1 &= a \mathbf{R}\hat{x}_1 + b \mathbf{R}\hat{y}_1 + c \mathbf{R}\hat{z}_1 \\
&= a \mathbf{\hat{x}}_0 + b \mathbf{\hat{z}}_0 - c \mathbf{\hat{y}}_0 \\
\mathbf{\hat{R}} \mathbf{P}_1 &= \mathbf{\hat{P}}_0
\end{align*}
\]
Rotation Matrix Chaining

\[ R_2 \rho^2 = \rho' \]
\[ \rho' = R_x \rho^2 \]
\[ R_1 \rho' = \rho_0^0 \]
\[ R_Y \rho_0^0 = \rho' \]

\[ \rho_0^0 = R_Y R_1 \rho_2^2 = R \rho^2 \]

(Slide adopted from AA173 Spring’20)
Transformation

\[ \rho^1 = R_2 \rho^2 \]

\[ \rho^0 = \rho^1 + t^0 \]

\[ \rho^0 = R_2 \rho^2 + t^2 \]

(Slide adopted from AA173 Spring’20)
Pose of a body

We define the pose of a body using the tuple \((R, t)\) which is the transformation needed to take a point in space from the body frame to a reference frame (e.g. local ENU frame).
Overview

• Pose of an object
• Image Formation
• Neural Networks
How does camera form an image?

**Example:** The quadrotor is flying through an urban area. It has a mounted camera which can see a known building.

**GOAL:** Obtain image pixel locations corresponding to 3D points.
Camera Matrix – Pinhole Camera

\[
P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \rightarrow \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}
\]

\[
\begin{align*}
x' &= f \frac{x}{z} \\
y' &= f \frac{y}{z}
\end{align*}
\]

[Eq. 1]

Derived using similar triangles

(Slide adopted from CS231A Spring’22)
Camera Matrix – Pinhole Camera

Retina plane

 Pixels, bottom-left coordinate systems

Digital image

\((1080, 1920)\)
Camera Matrix – Pinhole Camera

\[
\begin{pmatrix}
 f & 0 & c_x \\
 0 & f & c_y \\
 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
 x \\
 y \\
 z
\end{pmatrix}
= \begin{pmatrix}
 fx + cz^2 \\
 fy + cyz \\
 z
\end{pmatrix}
\]

1. Offset

\[(x, y, z) \rightarrow \left( \frac{x}{z} + c_x, \frac{y}{z} + c_y \right)\]

[Eq. 5]

Is this affine, linear or non-linear?
Camera Matrix – Pinhole Camera

1. Offset

\[ (x, y, z) \rightarrow (f \frac{x}{z} + c_x, f \frac{y}{z} + c_y) \]  

[Eq. 5]

\[
\begin{pmatrix}
0 & f & c_x \\
0 & f & c_y \\
1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
= 
\begin{pmatrix}
x + cxz \\
y + cyz \\
z
\end{pmatrix}
\]
Putting it together

$p = (x, y, z)$

$p' = Kp$

what it looks like to the camera.

$p_{robot} = (R_{robot \to world} p_{world} + t_{robot \to world})$

$p_{camera} = (R_{camera \to robot} p_{robot} + t_{camera \to robot})$

$(p')_{camera} = Kp_{camera}$
What we have learnt

We can transform a 3D point into 2D camera pixel space. So, if we know where something is in the world, we can know with high certainty which pixel it will be visible in.
Overview

• Pose of an object
• Image Formation
• Neural Networks
Single Neuron

weight: \( w = (w_1, \ldots, w_n) \in \mathbb{R}^n \)

bias: \( b \in \mathbb{R} \)

\[
y = w^T x + b
\]

\[
y' = f(w^T x + b)
\]
Neural Network

\[ w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}, w_{24} \in \mathbb{R}^4 \]
\[ b_{10}, b_{12}, b_{31}, b_{22}, b_{23}, b_{24} \in \mathbb{R} \]

\[ \sum_{j=1}^{4} \sum_{i=1}^{4} \sum_{k=1}^{4} \sum_{l=1}^{4} x^{(i)} y^{(j)} z^{(k)} + b_{31} \approx 39 \]
Convolutional Network

(Slide adopted from CS231N Spring’22)
Convolutional Network
## Linear Regression

<table>
<thead>
<tr>
<th>Living area (feet$^2$)</th>
<th>Price (1000$'s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>400</td>
</tr>
<tr>
<td>1600</td>
<td>330</td>
</tr>
<tr>
<td>2400</td>
<td>369</td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
</tr>
<tr>
<td>3000</td>
<td>540</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

We can plot this data:

\[
y = \mathbf{\theta}^T \mathbf{x}
\]

\[
\begin{align*}
\min_{\theta} & \sum_{i=1}^{N} \left( \hat{y}^{(i)} - y^{(i)} \right)^2 \\
& = \sum_{i=1}^{N} \left( \mathbf{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right)^2 \\
\text{solution: } & \mathbf{x} = \mathbf{A}^{+} \mathbf{b}
\end{align*}
\]
Linear Regression

<table>
<thead>
<tr>
<th>Living area (feet²)</th>
<th>Price (1000$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>400</td>
</tr>
<tr>
<td>1600</td>
<td>330</td>
</tr>
<tr>
<td>2400</td>
<td>369</td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
</tr>
<tr>
<td>3000</td>
<td>540</td>
</tr>
</tbody>
</table>

We can plot this data:

\[
y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad x = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}
\]

\[
x^T \theta = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} \theta = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}
\]

\[
\| x^T \theta - y \|^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2
\]
\[
\min \| \mathbf{x} \theta - \mathbf{y} \|^2 \\
\theta = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y}
\]

Fin.