Hands-on AI based 3D Vision

Tutorial 1

Today’s Plan

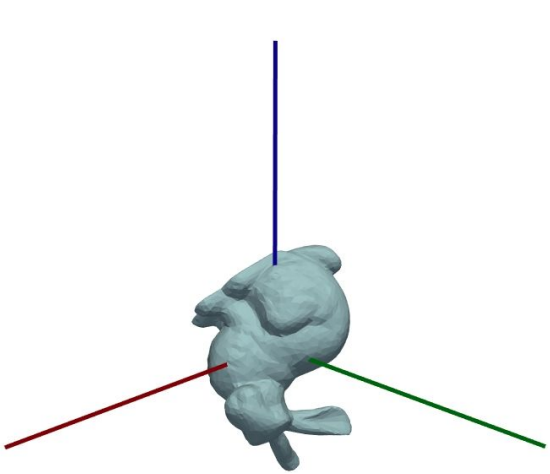
Recap:

- Rotation Matrices
- Homogeneous Coordinates

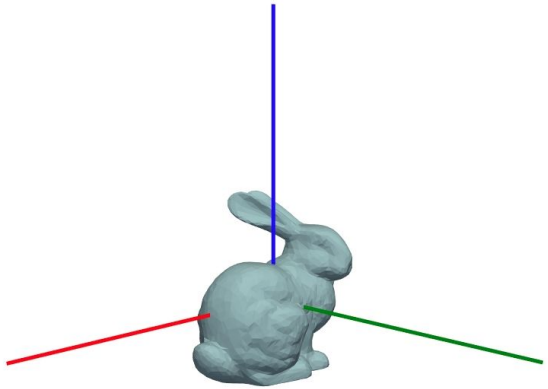
Assignment 1 Topics and Hints:

- Affine and Projective Transformations
- Perspective / Orthographic Projection
- Pose Estimation

Rotation Matrices



$$\mathbf{R}\mathbf{x}_i = \mathbf{x}'_i$$



Rotation matrices rotate points around the origin!

Understanding Rotation Matrices

- Group of rotation matrices SO(3):

$$RR^T = I = R^T R \text{ and } \det R = 1$$

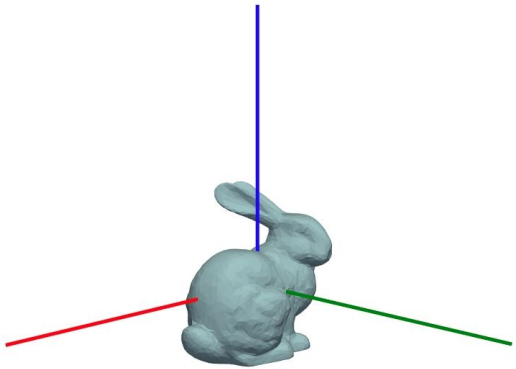
- Important properties of rotation matrices:
  - Rows (and columns) of R form orthonormal bases
  - Rotations have no effect on dot products: $\langle Rx, Ry \rangle = \langle x, y \rangle$
  - Rotations are angle- and length-preserving (theoretical exercise)

Understanding Rotation Matrices

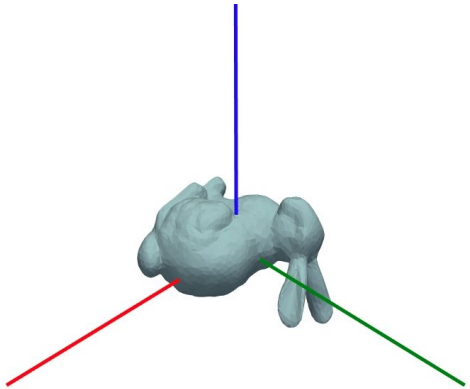
- Group of rotation matrices SO(3):

$$RR^T = I = R^T R \text{ and } \det R = 1$$

What about $\det R = -1$?



$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

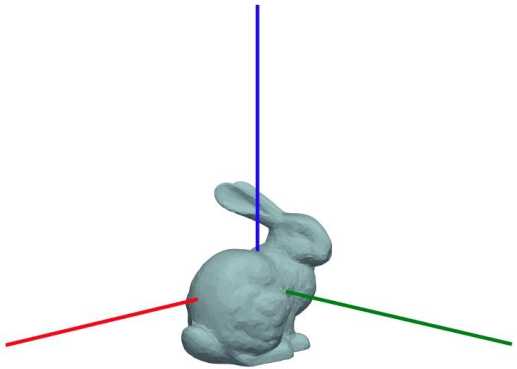


Understanding Rotation Matrices

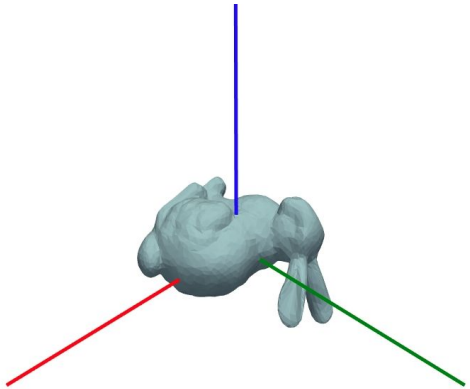
- Group of rotation matrices SO(3):

$$RR^T = I = R^T R \text{ and } \det R = 1$$

What about $\det R = -1$?



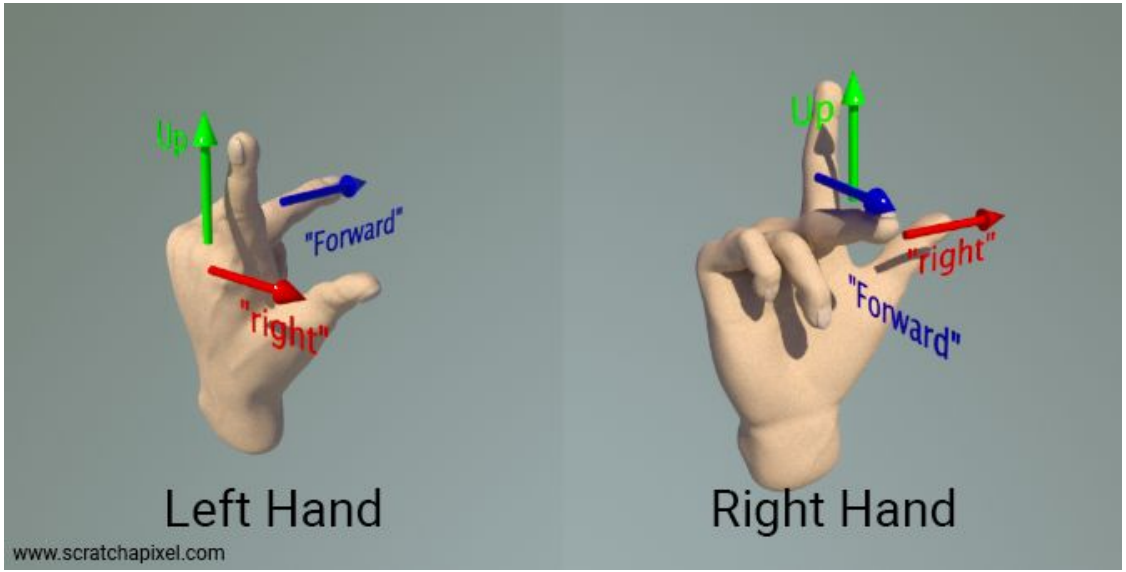
$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



Bunny is looking to the left now!

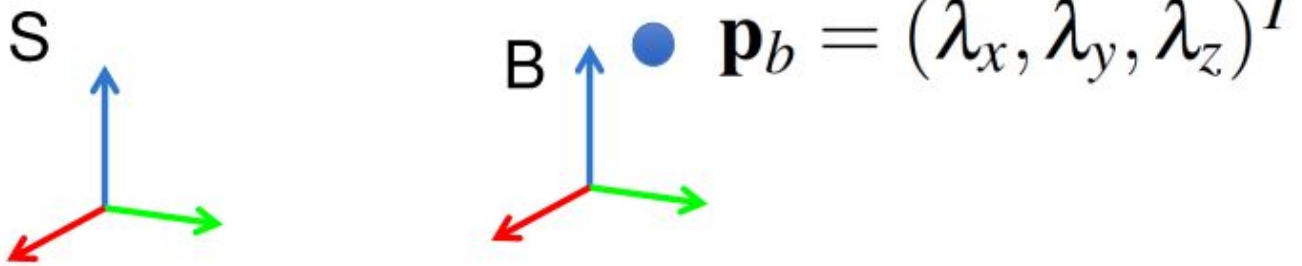
Coordinate System Conventions

Orthogonal matrices with det = -1 swap coordinate system orientations!



We will always assume right-hand coordinate systems!

Rotation Matrix = Change of Coordinates



$$\mathbf{p}_s = \lambda_x \mathbf{x}_s^B + \lambda_y \mathbf{y}_s^B + \lambda_z \mathbf{z}_s^B$$

$$\mathbf{p}_s = \mathbf{R}_{sb}\,\mathbf{p}_b \quad \Rightarrow \quad \mathbf{R}_{sb} = \begin{bmatrix} \mathbf{x}_s^B & \mathbf{y}_s^B & \mathbf{z}_s^B \end{bmatrix}$$

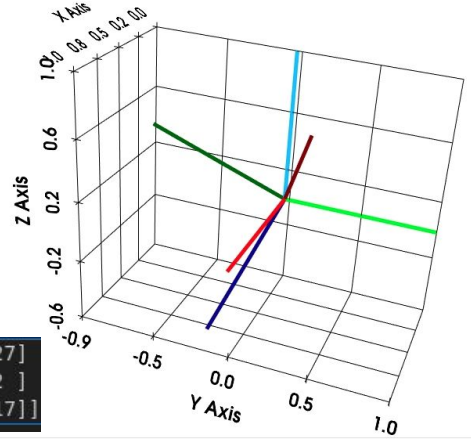
Rotation Matrix = Change of coordinates

Rotation matrix to rotate **old** frame to **new** frame

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

axes of old coordinate system expressed in new frame

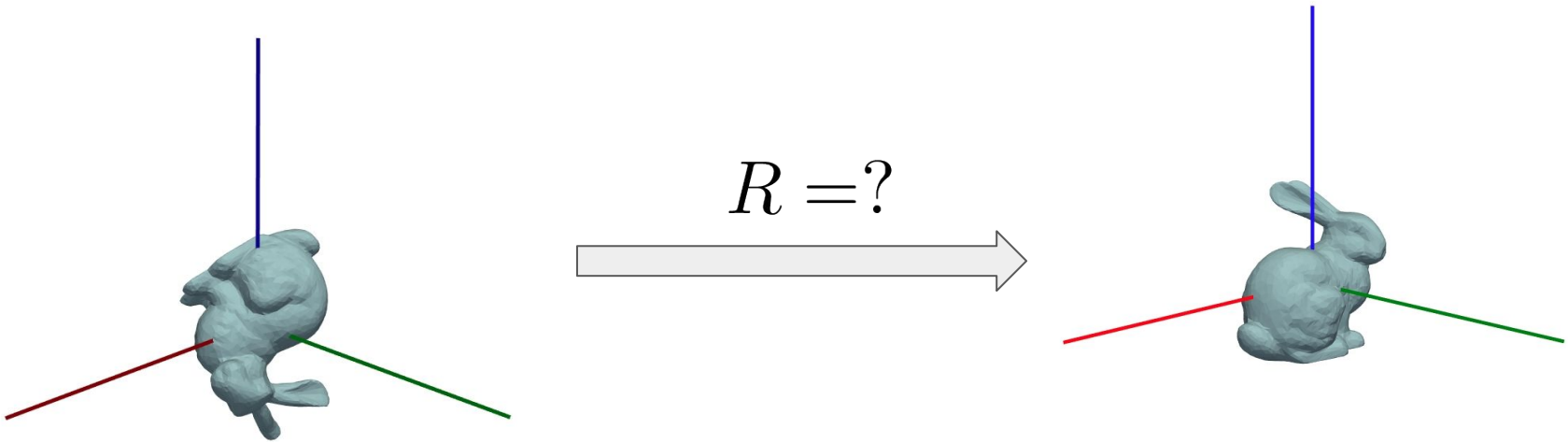
axes of new coordinate system expressed in old




```
[[ 0.60574529  0.03578994  0.79485327]
 [ 0.303717   -0.93374421 -0.1894142 ]
 [ 0.73541051  0.35614721 -0.57648117]]
```

Rotating Objects / Points

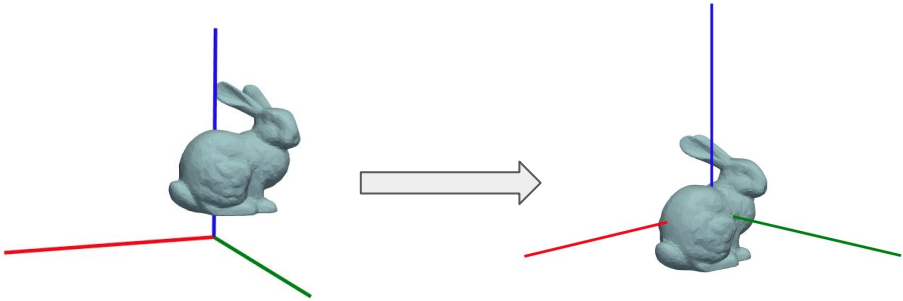
- How can we use rotation matrices to rotate an object into a desired pose?



- By using the **object-to-world** rotation matrix
- Express object axes in new frame!
- You will do this in the programming exercise on Transformations!

Translations and Homogeneous Coordinates

- Translation by t: $x' = x + t$
- Cannot be expressed as matrix matrix operation in 3D
- We need **homogeneous coordinates**
  - Add a fictitious 4th component 1



Homogenization	Dehomogenization
$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \tilde{\mathbf{x}}$	$\tilde{\mathbf{x}} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \mapsto \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix} = \mathbf{x}$

Translation as Product:

$$\tilde{\mathbf{x}}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{x}}$$

Rigid Transformations

- Rigid Transformation = Rotation + Translation
- Can be neatly expressed in homogeneous coordinates:

$$\tilde{\mathbf{x}}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{x}}$$

In which frame is t defined?

What is the matrix representation of the inverse rigid transformation?

Answer

$$\begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Projective Transformations

- Projective transformations are represented by invertible matrices

$$\mathbf{H} \in \mathbb{R}^{4\times 4}$$

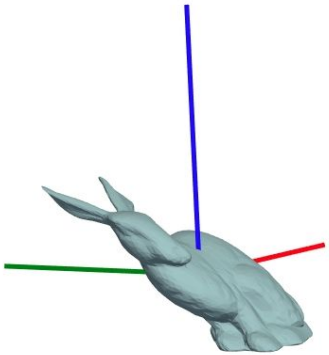
- Hierarchy of projective transformations:
- In theoretical exercise 1 you will show some properties of the different transformation types

Type	Matrix	Preserves
Rigid	$\begin{bmatrix}\mathbf{R} & \mathbf{t}\\ \mathbf{0} & 1\end{bmatrix}$	Lengths
Similarity	$\begin{bmatrix}s\mathbf{R} & \mathbf{t}\\ \mathbf{0} & 1\end{bmatrix}$	Angles
Affine	$\begin{bmatrix}\mathbf{A} & \mathbf{t}\\ \mathbf{0} & 1\end{bmatrix}$	Parallelism
Projective	$\mathbf{H}$	Straight lines

theoretical exercise 1

How to think about affine transformations

- Affine transformations are of the form: $\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^{\mathbf{T}} & 1 \end{bmatrix}$
- A can be any invertible 3x3 matrix
  - Rotations are special cases
  - Mirroring is also allowed
  - Non-uniform scaling
  - Anything else?

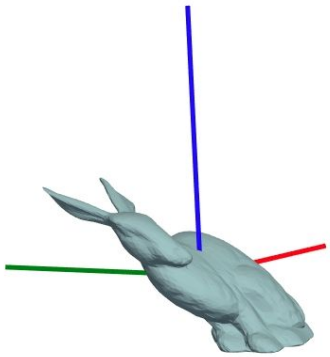


shearing is affine

How to think about affine transformations

- Affine transformations are of the form:
- A can be any invertible 3x3 matrix
  - Rotations are special cases
  - Mirroring is also allowed
  - Non-uniform scaling
  - Anything else?

$$\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix}$$



shearing is affine

SVD will be important for other exercises too!

Singular Value Decomposition:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$$

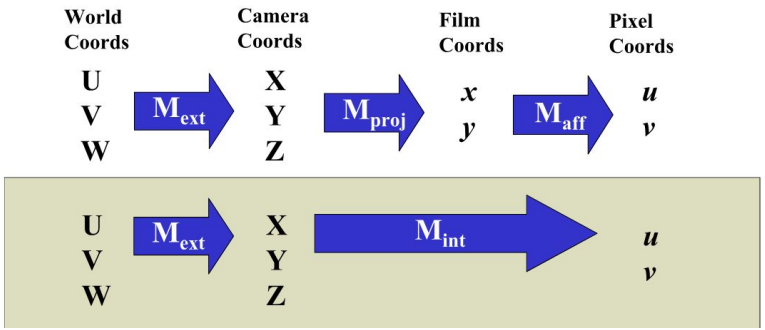
U, V are **orthogonal** (i.e. rotations + mirroring)
$\sum$ is a diagonal matrix

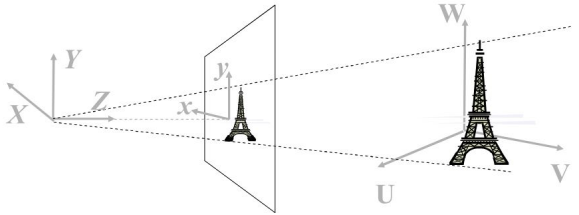
Affine transformations are combinations of

1. **rotation + mirroring**
2. **non-uniform scaling**
3. **rotation + mirroring**
4. **translation**

Image Formation (Programming Exercise)

- Perspective Projection



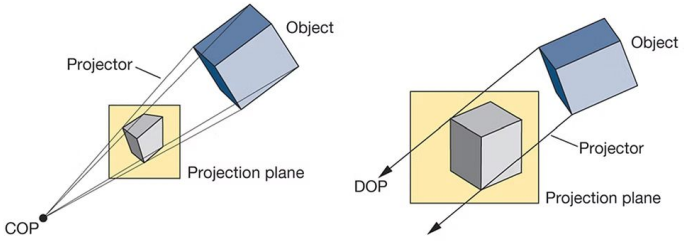


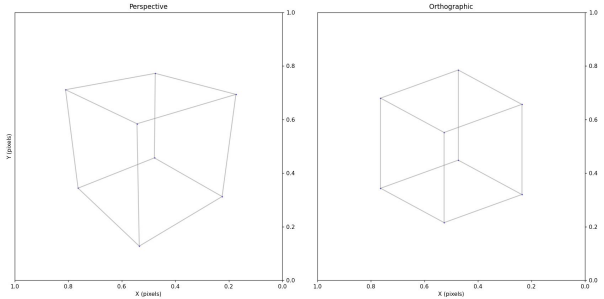
Forward Projection onto image plane.
3D (X,Y,Z) projected to 2D *(x,y)*

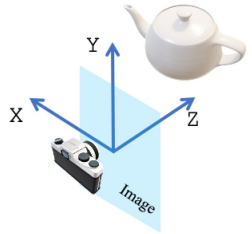
Conventions in this assignment:

$$\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} = \begin{bmatrix} f_x/w & & o_x/w \\ & f_y/h & o_y/h \\ & & 1 \end{bmatrix}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$$

- Orthographic Projection



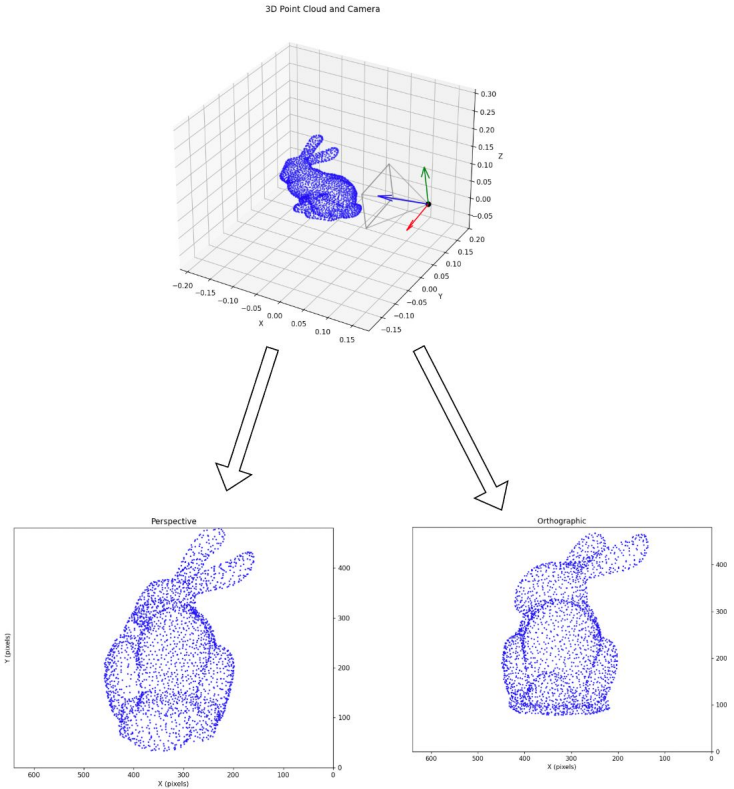




Camera View Coordinate System

What you will see in the exercise

- look_at method
  - Rotate the camera to look towards the object
- Perspective Projection
- Orthographic Projection

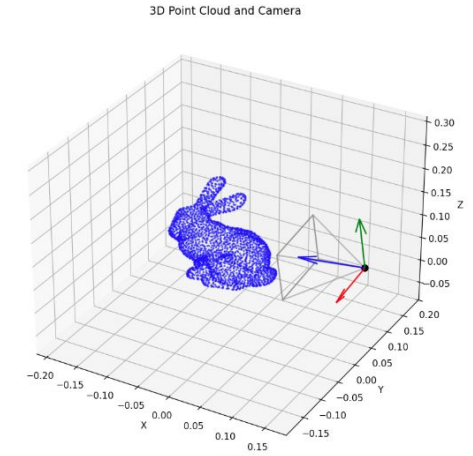


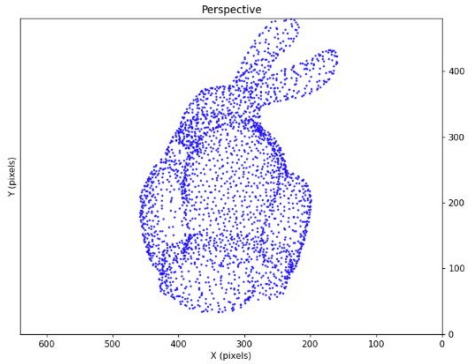
Camera Pose Estimation (Theory + Progr Exercise)

- Given 3D-2D point correspondences we want to know:

Where is the camera located in 3D space?

- You will derive and implement a simple algorithm that estimates R and t by solving a system of linear equations





Hints for Theoretical Exercise 2

- You will estimate the camera pose problem by solving a system of linear equations in the least squares sense:

$$\min_{\|\boldsymbol{\theta}\|=1} \|\mathbf{M}\boldsymbol{\theta}\|_2^2$$

- θ is a flattened vector containing R and t
- How do you solve such problems? (SVD, last column of V)
- Solving for θ yields candidates for R and t
  - Scale is wrong!
  - R is not necessarily a rotation matrix
- Hints:
  - To find “closest” orthogonal matrix, use SVD again, and drop diagonal matrix
  - How can you salvage the situation if the resulting matrix has det -1? (Note that $\|\mathbf{M}\theta\|_2^2 = \|\mathbf{M}(-\theta)\|_2^2$)
  - t also needs to be scaled. We accept different answers here.

Points

Theoretical Part (*29 Points*):

1) 3D Transformations (14 Points)
   a) i *2P*, ii *2P*, iii *2P* (+ iiii 2P ***bonus***)
   b) i *2P*, ii *2P*
   c) 4 x *1P*
2) Estimating Camera Pose from 3D-2D Correspondences (15 Points)
   a) 1P
   b) 6P
   c) 1P
   d) 1P
   e) 4P
   f) 2P

Coding Part (*50 Points*):

1) Part 0: Installation and Introduction (*0 Points*)
2) Part 0.1: Notebook 00 - PyTorch (**Not mandatory**) (*8* ***Bonus Points****)
3) Part 1: Notebook 01 - Transformations (*18 Points*)
4) Part 2: Point Cloud Projection (*16 Points*)
5) Part 3: Direct Linear Transform (DLT) algorithm for camera pose estimation (*16 Points*)

Questions

Write an email to us or ask in the forum!