Visual Learning and Recognition of 3-D Objects from Appearance

CS663 Fundamentals of Digital Image Processing

Rathour Param Jitendrakumar, 190070049
Tirthankar Mazumder, 20b090012
Divyansh Tiwari, 200020049

Indian Institute of Technology Bombay
https://github.com/wermos/CS-663-Project

Autumn 2023-24

Guide: Prof. Ajit Rajwade

Outline

Implementation

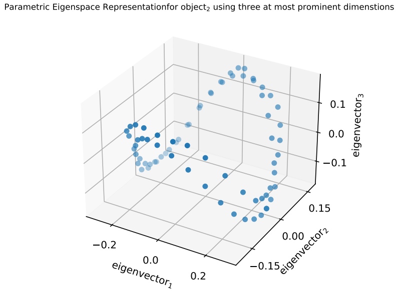
High-Level Overview

- Data-Loading
  - Generate Training, Testing sets (COIL-100 Dataset)
  - Image Pre-processing (RGB to Grayscale conversion)
- Train Model
  - Normalise Image Sets
  - Construct Universal Image Set and Object-specific Image Sets
  - Construct Universal Eigenspace and Object-specific Eigenspaces (Principal Component Analysis)
  - Construct Universal Manifolds and Object-specific Manifolds (Cubic Spline Interpolation)
- Test Model
  - Normalise Image
  - Construct Universal Eigencoefficients (Project to Universal Eigenspace)
  - Recognise Object (Closest universal manifold from projection of given image)
  - Construct Object-specific Eigencoefficients (Project to Eigenspace of recognised object)
  - Recognise Pose (Closest point on Object-specific Manifold from Eigencoefficients vector)

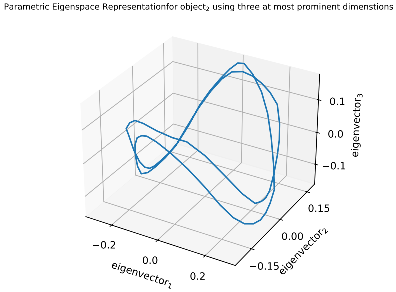
Manifold (Parametric Appearance Representation)

Benefits

- Provides a parameterised approach to approximate eigencoefficients for unknown poses
- Possible parameters
  - Rotation about $x-$axis, $y-$axis, $z-$axis
  - Illumination conditions of the environment
- Storage compression over PCA: cubic polynomials gives eigencoefficients of each eigenspace



(a) Available Poses of object



(b) Manifold (Cubic Spline interpolation)

Figure: Object 2

Dataset

Columbia University Image Library (COIL-100)

100 objects! Each object has 72 pose images with pose angle $\{0, 5, 10, \ldots, 345, 350, 355\}$

Figure: COIL-100 Dataset by Columbia Imaging and Vision Laboratory (CAVE)[1]

These objects can be categorised into two sets

- uniform reflectance but similar shape
- complex reflectance and geometric properties

[1] "Columbia Object Image Library," S. A. Nene, S. K. Nayar and H. Murase, CUCS-006-96, February 1996.

Dataset

Assumptions

COIL-100 dataset satisfies these assumptions

- Image Assumptions
  - Background region is assigned zero brightness value
  - Imaging sensor has a linear response (image brightness is proportional to scene radiance)
  - Object images
    - are not occluded by other objects
    - can be object segmented from the remaining scene
    - are invariant to image magnification and illumination intensity as segmented image region is

      normalized with respect to scale re-sampled to fit image size

      normalized with respect to brightness normalised by dividing euclidean norm
- Object Assumptions
  - Neither highly specular nor has a high-frequency texture

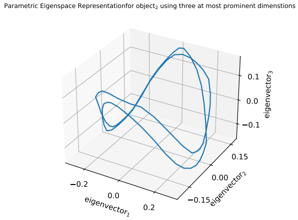
Manifold (Parametric Appearance Representation)

Examples

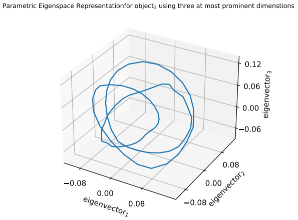
(a) Object 2

(b) Object 3

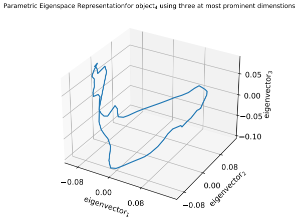
(c) Object 4



(d) Corresponding Manifold



(e) Corresponding Manifold



(f) Corresponding Manifold

Code Optimizations

Implementation

- Principal Component Analysis (PCA)
  - Eigenvector computation of $X^T X$ instead of the covariance matrix (XX^T)
  - `numpy.linalg.eigh` instead to `numpy.linalg.eig` to exploit the algorithms assuming symmetric input matrices
- Extensive usage of `numpy` objects and functions to facilitate multi-threading in `numba`
- `os.environ['OMP_NUM_THREADS'] = '16'` to increase number of parallel threads to 16
- Ability to store and load saved learnt variables using `pickle`

Optimal Values

PCA Threshold = 0.6 and Training Data Split = 0.7

Object Recognition accuracy 99.172%

Pose Estimation accuracy 76.172%

Mean Pose error 6.872°

In the next two slides, we see some examples of incorrect recognition.
Even in such cases we see that our model's outputs are reasonable.

- 180° pose errors are most frequent among pose recognition
- similar-looking objects get recognised incorrectly

These errors usually occur in bursts (consecutive poses), which implies that nearby points in manifolds might be too far to interpolate in between points accurately.
One way to solve this is by training on uniformly pose-separated object images.

Incorrect Object Detection (True and Estimated)

(a) Obect 7, Angle 75

(b) Obect 22, Angle 280

(c) Obect 97, Angle 195

(d) Object 22, Angle 80

(e) Object 75, Angle 280

(f) Object 83, Angle 25

Incorrect Pose Recognition (True and Estimated)

(a) Obect 33, Angle 75

(b) Obect 21, Angle 280

(c) Obect 20, Angle 100

(d) Object 33, Angle 80

(e) Object 21, Angle 95

(f) Object 20, Angle 280

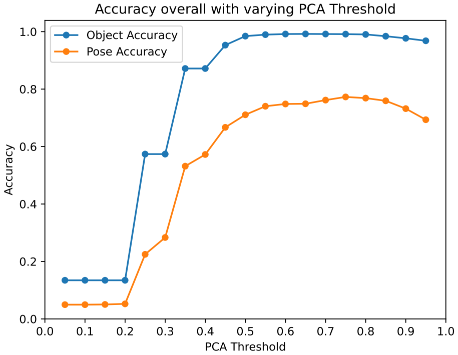
Variation with PCA Threshold and Training Data Split one at a time

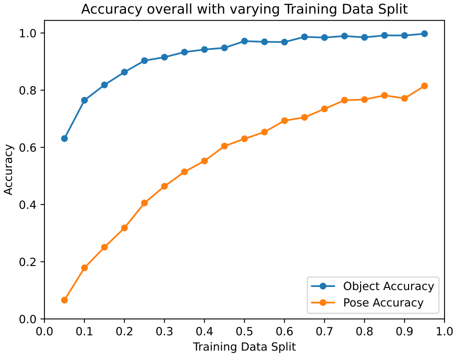
In the coming plots, we vary each parameter one by one and then vary them simultaneously

- Vary PCA Threshold as $\{0.05, 0.1, \ldots, 0.9, 0.95\}$, set Training Data Split to 0.6
  - higher PCA Threshold leads to overfitting
- Vary Training Data Split as $\{0.05, 0.1, \ldots, 0.9, 0.95\}$, set PCA Threshold to 0.7
  - more training data creates more accurate manifold which leads to better recognition
- Vary PCA Threshold as $\{0.05, 0.1, \ldots, 0.9, 0.95\}$ and vary Training Data Split as $\{0.05, 0.1, \ldots, 0.9, 0.95\}$

Variation with PCA Threshold and Training Data Split one at a time

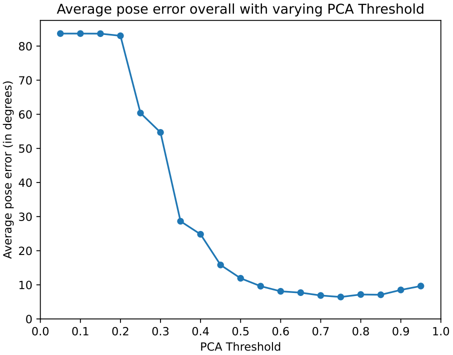
Accuracy

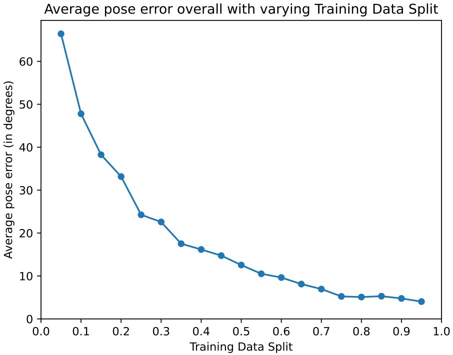




Variation with PCA Threshold and Training Data Split one at a time

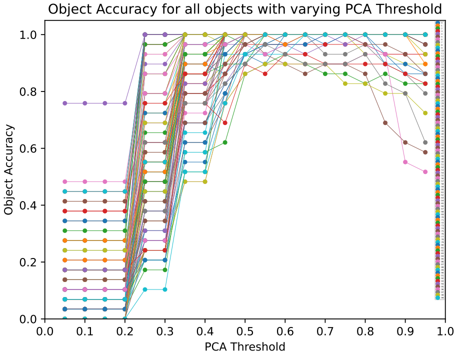
Mean Error

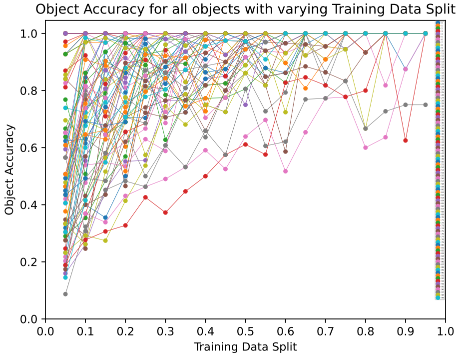




Variation with PCA Threshold and Training Data Split one at a time

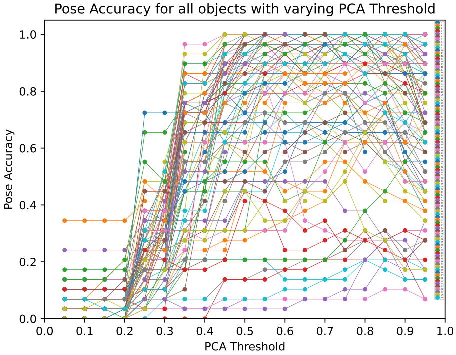
Object Accuracy

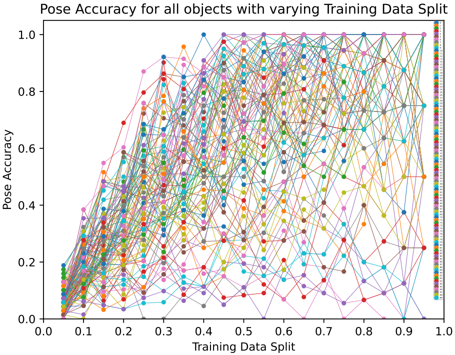




Variation with PCA Threshold and Training Data Split one at a time

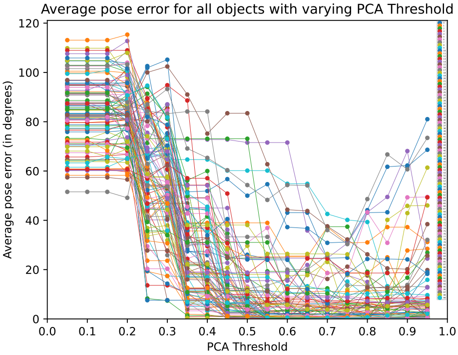
Pose Accuracy

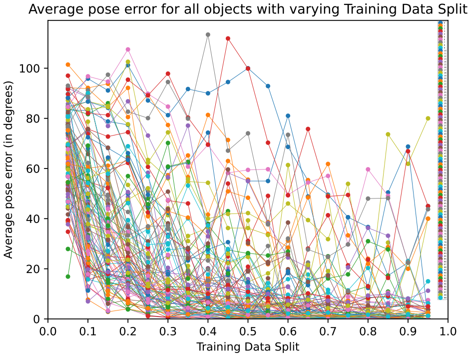




Variation with PCA Threshold and Training Data Split one at a time

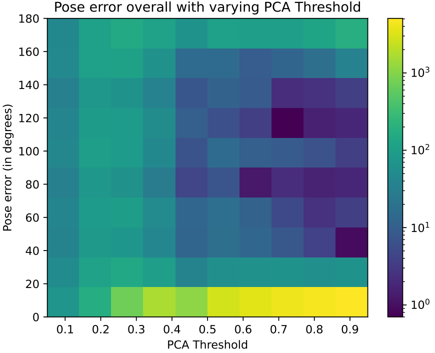
Mean Error

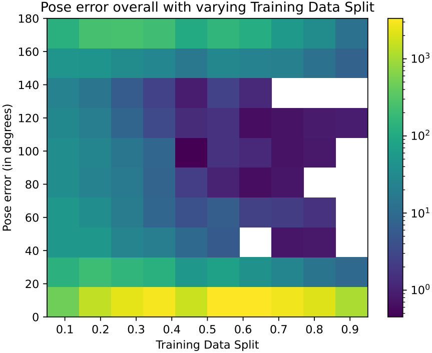




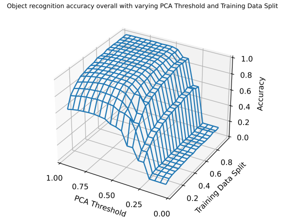
Variation with PCA Threshold and Training Data Split one at a time

Exact Error

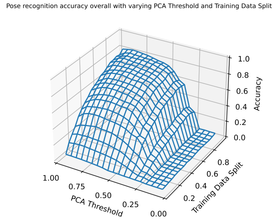




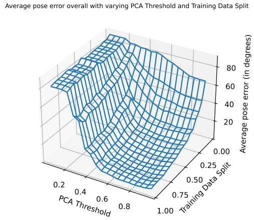
Simultaneous variation with PCA Threshold and Training Data Split



(a) Object Accuracy



(b) Pose Accuracy



(c) Mean Error

Variation with Object Complexity

Three types of objects

- Simple Objects, almost no variation in poses due to symmetrical nature and less details
- Simple Objects, almost no variation in poses due to symmetrical nature but highly detailed
- Complex Objects, different shapes in different poses

(a) Object 69

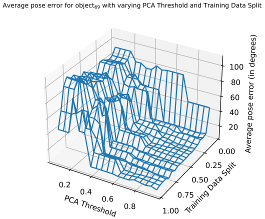
(b) Object 98

(c) Object 67

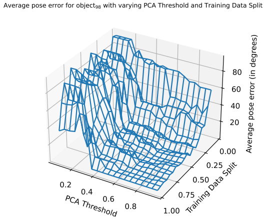
Observations

Types of Wireframe: Mean Accuracy

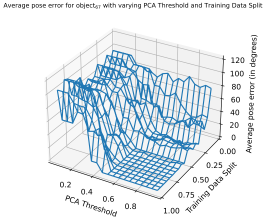
- as object complexity increases, more training data is needed for lower mean error
- only slightly higher pca threshold needed for complex objects



(a) Object 69



(b) Object 98

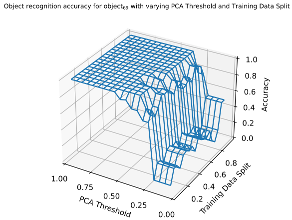


(c) Object 67

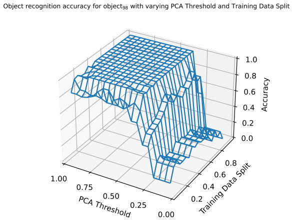
Observations

Types of Wireframe: Object Accuracy

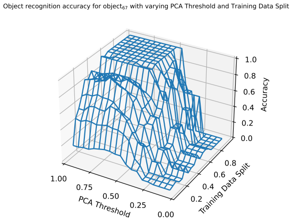
- as object complexity increases, more training data is needed for higher accuracy
  15-20 objects poses are enough for learning the simple objects, but 30-40 object poses are needed to learn complex dataset
- higher pca threshold only needed in complex objects



(a) Object 69



(b) Object 98

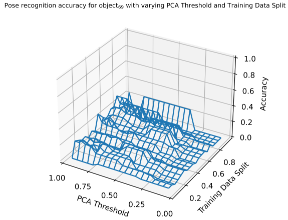


(c) Object 67

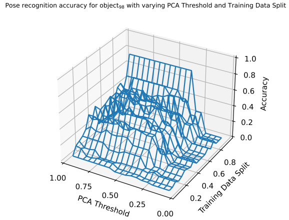
Observations

Types of Wireframe: Pose Accuracy

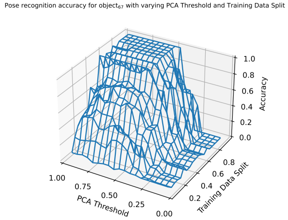
- accuracy increases as object complexity increases, because pose becomes easier to distinguish
- for simpler objects, more training data or pca threshold doesn't help much to estimate exactly



(a) Object 69



(b) Object 98



(c) Object 67

Theory I

Notation

Single image (normalised)

$$\hat{\mathbf{x}} = \left[\hat{x_1}, \quad \hat{x_2}, \quad \cdots, \quad \hat{x_N}\right]^T \tag{1}$$

Universal Image Set

$$\mathbf{X} \triangleq \{\mathbf{x}_1^{(1)} - \mathbf{c}, \mathbf{x}_2^{(1)} - \mathbf{c}, \ldots, \mathbf{x}_R^{(1)} - \mathbf{c}, \ldots, \mathbf{x}_R^{(p)} - \mathbf{c}\} \tag{2}$$

Object Image Sets

$$\mathbf{X}^{(p)} = \{\hat{\mathbf{x}}_1^{(p)} - \mathbf{c}^{(p)}, \hat{\mathbf{x}}_2^{(p)} - \mathbf{c}^{(p)}, \ldots, \hat{\mathbf{x}}_R^{(p)} - \mathbf{c}^{(p)}\} \tag{3}$$

Compute Universal Eigenspace

$$\hat{\mathbf{Q}} \triangleq \mathbf{X}^T\mathbf{X} \tag{4}$$

$$\lambda_i \hat{\mathbf{e}}_i = \hat{\mathbf{Q}}\mathbf{e}_i \tag{5}$$

Utilise PCA Threshold (T_i)

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i} \geq T_i \tag{6}$$

Theory II

Notation

$$\mathbf{Q} = \mathbf{X}\begin{bmatrix}\hat{\mathbf{e}}_1, & \hat{\mathbf{e}}_2, & \dots & \hat{\mathbf{e}}_k\end{bmatrix} \tag{7}$$

Similarly, Compute Object Eigenspaces

$$\hat{\mathbf{Q}}^{(p)} \triangleq {\mathbf{X}^{(p)}}^T \mathbf{X}^{(p)} \tag{8}$$

$$\lambda_i^{(p)} \hat{\mathbf{e}}_i^{(p)} = \hat{\mathbf{Q}}^{(p)} \hat{\mathbf{e}}_i^{(p)} \tag{9}$$

Utilise PCA Threshold (T_i)

$$\frac{\sum_{i=1}^{k^{(p)}} \lambda_i^{(p)}}{\sum_{i=1}^{N} \lambda_i^{(p)}} \geq T_i \tag{10}$$

$$\mathbf{Q}^{(p)} = \mathbf{X}^{(p)}\begin{bmatrix}\hat{\mathbf{e}}_1^{(p)}, & \hat{\mathbf{e}}_2^{(p)}, & \dots & \hat{\mathbf{e}}_{k^{(p)}}\end{bmatrix} \tag{11}$$

Compute manifolds

$$\mathbf{g}_i^{(p)} = \mathbf{Q}^T(\mathbf{x}_i^{(p)} - \mathbf{c}) \tag{12}$$

Theory III

Notation

$$\mathbf{f}_i^{(p)} = \mathbf{Q}^{(p)\,T}(\mathbf{x}_i^{(p)} - \mathbf{c}^{(p)}) \tag{13}$$

Interpolate manifolds

Universal Manifolds

$$\mathbf{g}^{(p)}(\theta) \tag{14}$$

Object-specific Manifolds

$$\mathbf{f}^{(p)}(\theta) \tag{15}$$

Closest universal manifold from projection of given image $\mathbf{y}$

$$\mathbf{z} = \mathbf{Q}^{T}(\mathbf{y} - \mathbf{c}) \tag{16}$$

$$d_1^{(p)} = \min_{\theta} \|\mathbf{z} - \mathbf{g}^{p}(\theta)\| \tag{17}$$

Point on object manifold closest from project of given image $\mathbf{y}$

$$\mathbf{z}^{(p)} = \mathbf{Q}^{(p)\,T}(\mathbf{y} - \mathbf{c}^{(p)}) \tag{18}$$

$$d_2^{(p)} = \min_{\theta} \|\mathbf{z}^{(p)} - \mathbf{f}^{p}(\theta)\| \tag{19}$$

References

Hiroshi Murase and Shree K. Nayar.
Visual learning and recognition of 3-d objects from appearance.
International Journal of Computer Vision, 14:5–24, 2005.
URL: https://api.semanticscholar.org/CorpusID:6611218.