Virtual Shape Recognition using Leap Motion

EC 520: Digital Image Processing and Communication

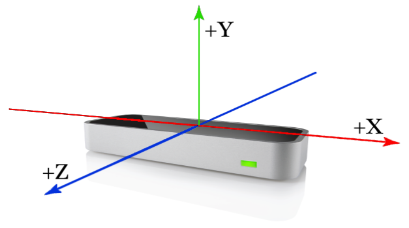
Aditya Chechani and Harshil Prajapati

Department of Electrical and Computer Engineering: Boston University

Problem Statement

To develop a robust algorithm to extend Leap Motion capabilities to recognize hand drawn gestures of various 2D shapes (e.g. circle, ellipse, square, triangle, etc.)

(a) Leap Motion [2]



(b) Coordinates axes of Leap Motion

Data Acquisition

- Leap Motion SDK - Python
- x,y,z coordinates of the tip of the index finger
- Store it in .json file

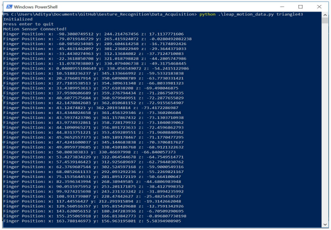


Figure 2: Capturing Leap data

Shape	Data
Circle	50
Rectangle	50
Triangle	50

Table 1: Generated Data

Data Preprocessing: *Planner Fitting of 3D points*

Shapes will not be on a plane || to XY plane of Leap Motion so we find a hypothetical plane z which has the least squared error with the data points provided [4].

$$z = Ax + By + D \tag{1}$$

The coefficients of plane are given by

$$c_x = -\frac{A}{D}; \qquad c_y = -\frac{B}{D}; \qquad c_z = \frac{1}{D}$$

The normal vector of plane is

$$n_p = [-\frac{A}{D} \quad -\frac{B}{D} \quad \frac{1}{D}] \tag{2}$$

Normal vector to X-Y plane

$$n_{XY} = [0 \quad 0 \quad 1]$$

Data Preprocessing: *Planner Fitting of 3D points*

Rotation Vector (u):

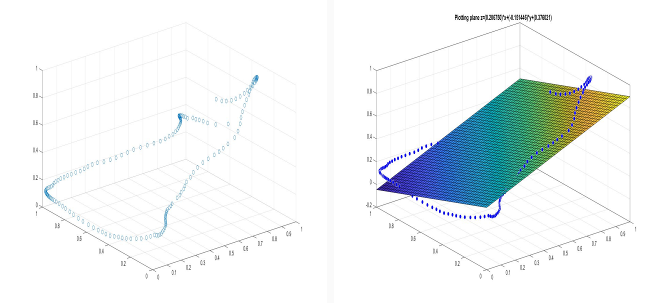
$$u = \frac{n_p}{\mid n_p \mid} \times n_{XY} \tag{3}$$

Rotation angle (θ):

$$\theta = sin^{-1}(\mid n_p \mid) \tag{4}$$

Using the rotation vector and the angle we calculate rotation matrix "R" we get the transformed data

$$[x_{new}, y_{new}, z_{new}]^T = R[x, y, z]^T \tag{5}$$

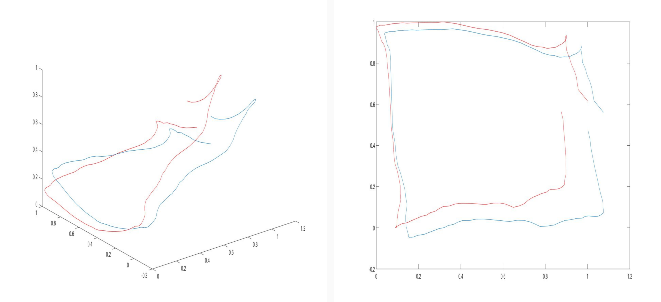


(a) Data points in 3D Space

(b) Best Fitted Plane

Figure 3: Plane Fitting of 3D points

Data Preprocessing: *Plane Fitting of 3D points*



(a) Rotated coordinate in 3D space

(b) Coodrinates on X-Y plane post processing

Figure 4: Plane Fitting of 3D points

Feature Extraction: *Fourier Descriptor*

Fourier Descriptor: We define

$$s[n] = x_n + iy_n$$

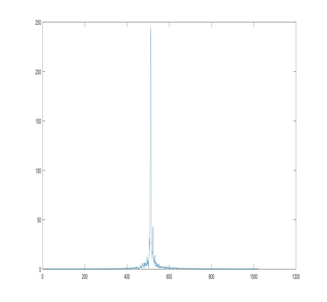
Taking the DFT of s[n]

$$a[k] = \frac{1}{N}\sum_{n=0}^{N-1} s[n] e^{\frac{-j2\pi nk}{N}} \tag{6}$$

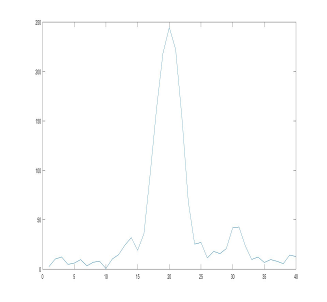
and coefficients can be recovered by

$$s[n] = \sum_{k=0}^{N-1} a[k] e^{\frac{j2\pi nk}{N}} \tag{7}$$

Feature Extraction: *Fourier Descriptor*



(a) Fourier Descriptor of data points.



(b) Considering only 40 high energy points

Classification: k-NN

For a Data set $\mathcal{D} = (\mathbf{x}_1, y_1), \cdots (\mathbf{x}_n, y_n)$ with $\mathbf{x}$ as 20 point feature vector from feature extraction and test point $\mathbf{x}$ Let $(\mathbf{x}_1, y_1), \cdots (\mathbf{x}_n, y_n)$ be reordered such that

$$d(x, x_1) \leq d(x, x_2), \cdots d(x, x_n)$$

where $d(x, x_i)$ can be l_p distance

$$|| \, x - x_{test} \, ||_p := (\sum_{i=1}^{d} | \, x_i - x_{test} \, |^p)^{\frac{1}{p}}$$

For our project we are considering the Euclidean distance (l_2)

$$h_{k-NN} = \underset{y=1 \cdot n}{\operatorname{argmax}} \sum_{j=1}^{k} 1(y_{(j)} = y) \tag{8}$$

where $1(y_{(j)} = y)$ is the number of k-NNs of $\mathbf{x}$ with label =y [5]

Classification: k-NN

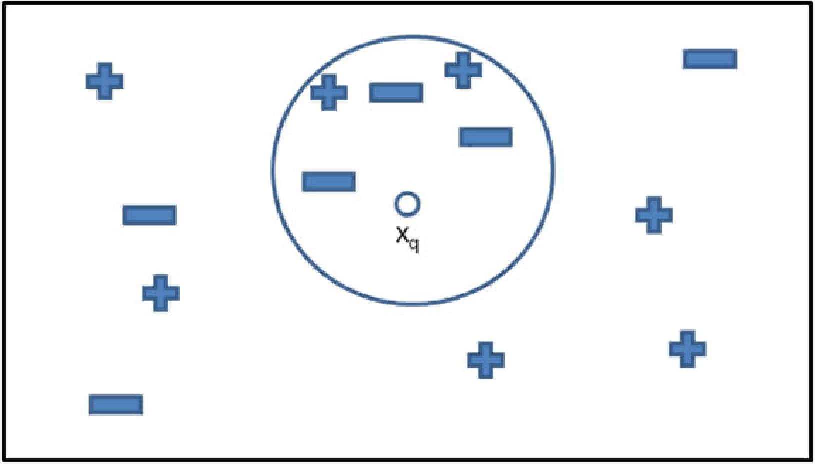


Figure 6: k-NN Example [1]

Classification: k-NN

	Ground Truth		
Predict	10	0	0
	0	9	3
	0	1	7

Table 2: 80:20 Split (NN)

CCR: 86.67%

	Ground Truth		
Predict	19	1	1
	1	18	3
	0	1	16

Table 3: 60:40 Split (NN)

CCR: 88.33%

	Ground Truth		
Predict	10	1	1
	0	8	4
	0	1	5

Table 4: 80:20 Split (3-NN)

CCR: 76.67%

	Ground Truth		
Predict	10	1	1
	0	7	6
	0	2	3

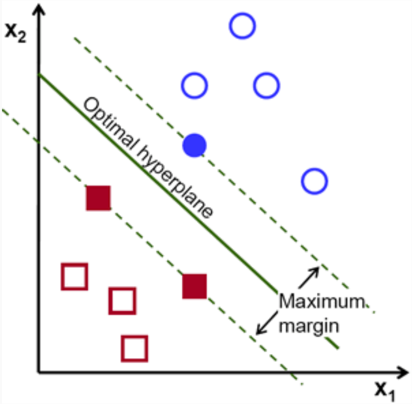
Table 5: 80:20 Split (5-NN)

CCR: 66.67%

Classification: SVM

For multiclass classification we extend binary SVM, using One v/s All (OVA) or One v/s One (OVO) approach.

Figure 7: SVM Example [3]



Results: SVM Classification

One v/s One:

	Ground Truth		
Predict	8	0	0
	2	10	6
	0	0	4

Table 6: C=16, σ 16

CCR: 73.33%

	Ground Truth		
Predict	10	0	0
	0	9	2
	0	1	8

Table 7: C=16, σ 32

CCR: 90%

Results: SVM Classification

One v/s All:

- Mean CCR = 71.11% (C = 2, σ = 16)
- Mean CCR = 88.9% (C = 2, σ = 32)

C1	Ground Truth		C2	Ground Truth		C3	Ground Truth	
Predict	8	0	Predict	7	1	Predict	6	0
	2	20		3	19		4	20

Table 8: OVO (C = 2, σ = 32)

Thank You

- k-NN Example.
  `http://www.d.umn.edu/~deoka001/knn.html`.
  Accessed: 2018-04-18.
- Leap Motion Development SDK.
  `https://developer.leapmotion.com`.
  Accessed: 2018-04-09.
- Open CV Tutorials:Introduction to Support Vector Machines.
  `https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`.
  Accessed: 2018-04-18.
- D. Eberly.
  Least squares fitting of data.
  Chapel Hill, NC: Magic Software, 2000.

References ii

P. Ishwar.
EC 503 - Learning from Data , January - May 2018.