컴퓨터비전 개론 (공) (CSE404): Lecture 12

홍재성

DGIST 로봇공학전공

Template (correlation) Matching

$$c(x, y) = \sum_s \sum_t w(s, t) f(x + s, y + t)$$

- It is possible to normalize correlation for changes in intensity values of the functions being processed
- Normalizing for size and rotation is a more complicated problem
- The size to which an image should be rescaled must be known. Similarly, the angle to which images should be rotated must be known
- In unconstrained situations, normalizing for size and orientation can become a challenging task

Template (correlation) Matching

It is sensitive to scale changes in *f* and *w*, so we use normalized correlation coefficient.

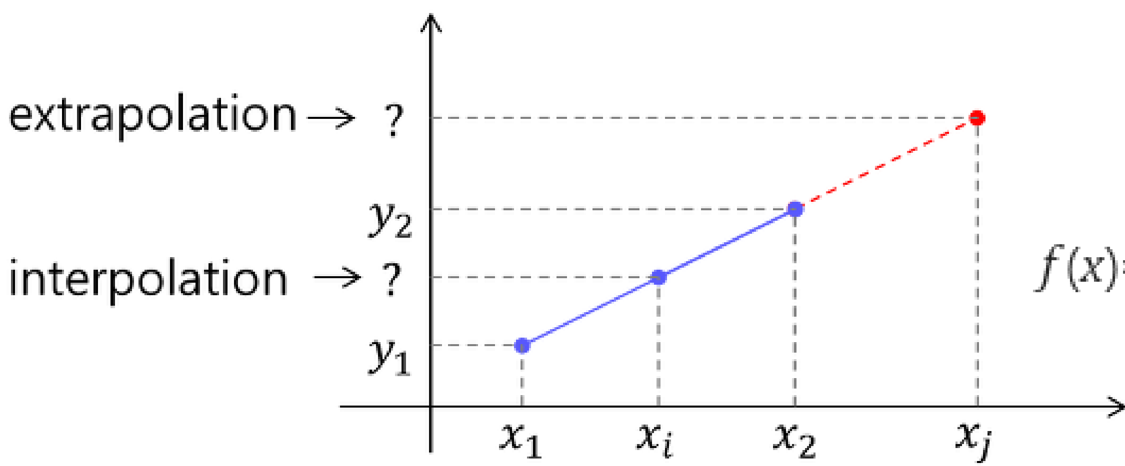
$$\gamma(x, y) = \frac{\sum_s \sum_t \left[w(s, t) - \overline{w}\right] \sum_s \sum_t \left[f(x + s, y + t) - \overline{f}(x + s, y + t)\right]}{\left\{ \sum_s \sum_t \left[w(s, t) - \overline{w}\right]^2 \sum_s \sum_t \left[f(x + s, y + t) - \overline{f}(x + s, y + t)\right]^2 \right\}^{\frac{1}{2}}} \tag{12.2-8}$$

where the limits of summation are taken over the region shared by w and f, $\overline{w}$ is the average value of the mask (computed only once), and $\overline{f}(x + s, y + t)$ is the average value of f in the region coincident with w. Often, w is referred to as a *template* and correlation is referred to as *template matching*.

- Rotation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

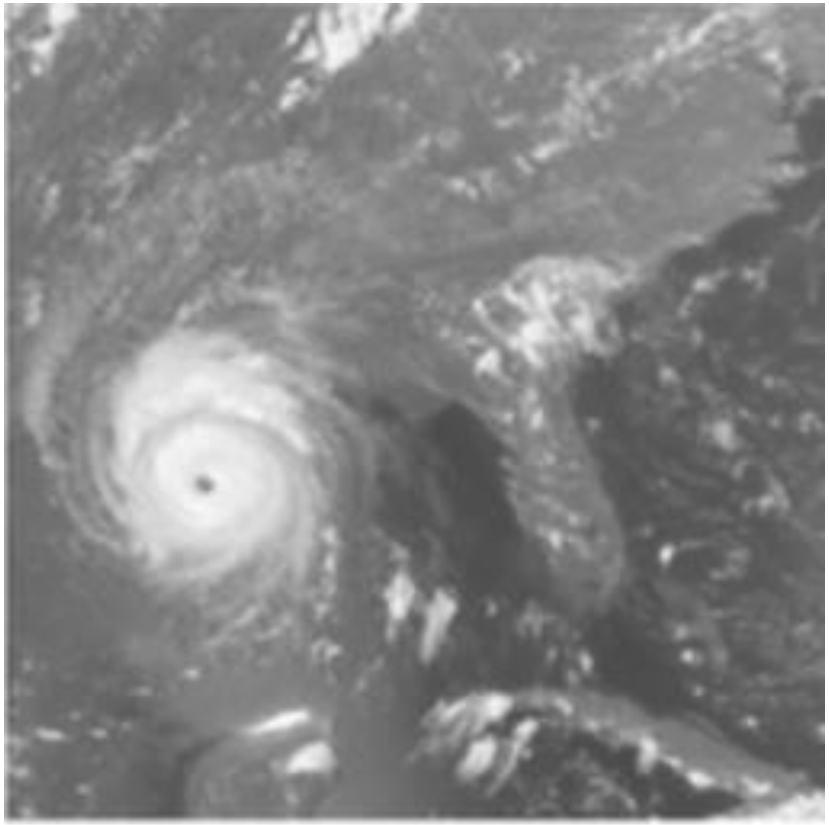
- Interpolation



$$f(x) = \frac{d_2}{d_1+d_2} f(x_1) + \frac{d_1}{d_1+d_2} f(x_2)$$

Practice

- Write a program to find the hurricane by template matching



Object (Pattern) Recognition

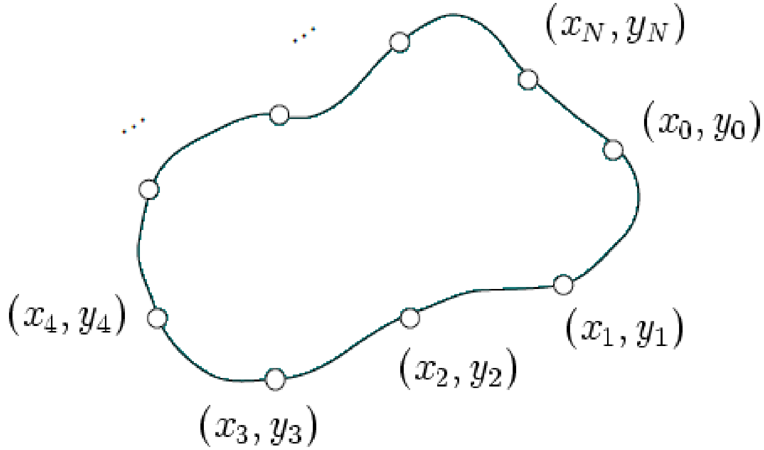
- Three common pattern arrangements are vectors (for quantitative descriptions), strings, and trees (for structural descriptions)
- Pattern vectors are represented by

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

where each component, x_i, represents the i th descriptor and n is the total number of such descriptors associated with the pattern

Fourier Descriptor

- Object boundary can be expressed by



$$s(k) = x(k) + jy(k),\ \ k = 0, 1, 2, \dots N-1$$

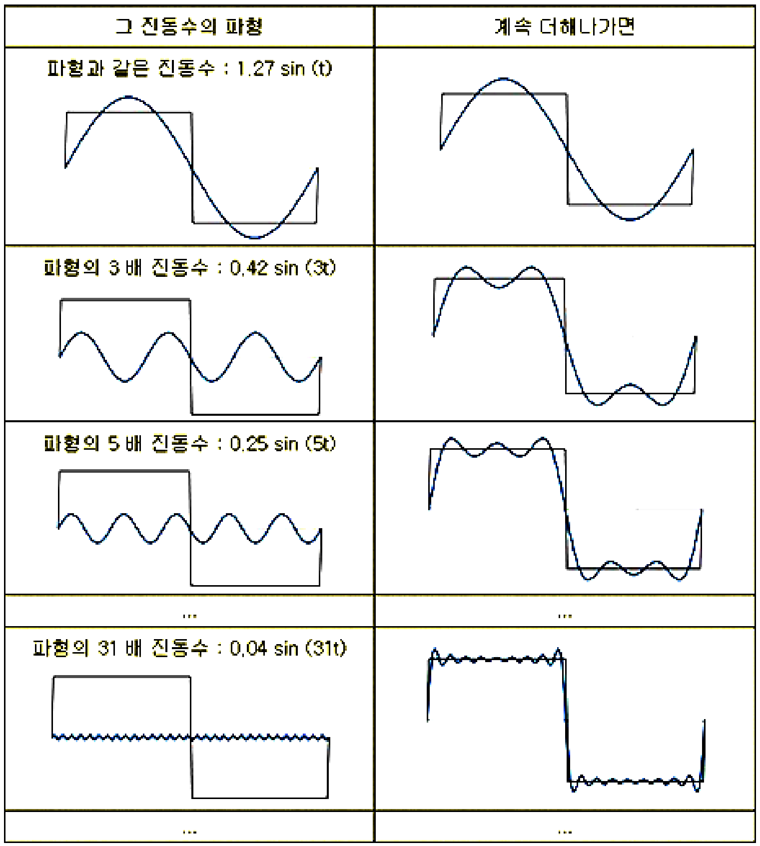
$$a(u) = \sum_{k=0}^{N-1} s(k)\, e^{-j2\pi u \frac{k}{N}} \quad \mathrm{k} = 0, \dots, \mathrm{N}-1$$

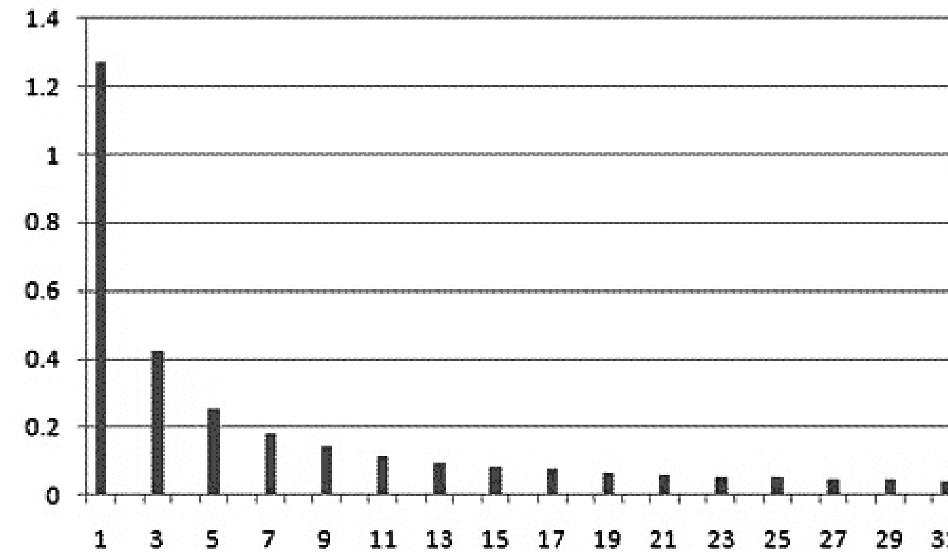
- a(u) is called Fourier Descriptors (FD)

- The descriptors a(u) describe the frequency contents of the curve: a value of u close to zero will describe **low frequency information, an approximated shape, and the higher frequencies will describe details**.
- For u = 0, a(u) represents the **position of the center of gravity** of the shape. This term is not interesting for the shape description. Without this term, the description won't be affected by a translation of the shape.
- The first frequency component, a(u) for u = 1, describes the **size of the shape**, if all the other components are set to zero. The shape becomes a circle (in fact, a N-sided polygon). One can use this component to normalize the set of Fourier descriptors, so they remain constant after an homothety on the shape.

- The other frequency components will **make alterations** on the circle described by u(1). The descriptors a(u) have opposite effects for positive and negative values of u.
- The phase of a(u) is a **rotation of this perturbation**, which means it describes the place on the circle where the action is performed.
- Appropriate pre-processing steps make Fourier Descriptors invariant to common transformations like translation, changes in scale, rotation
- The contour of a known object can therefore be recognized **irrespectively of its position, size and orientation**

Time (space) and Frequency





$$1.27\sin(t) + 0.42\sin(3t) + 0.25\sin(5t) + \cdots$$

Any periodic function can be expressed as a weighted sum (infinite) of sine and cosine functions of varying frequencies.

Translation

- If the 2D shape is translated by a distance $s_0 = x_0 + jy_0$

  $s'(k) = s(k) + s_0 \quad (k = 0, \dots N-1)$

- Its FD becomes

$$a'(u) = \sum_{k=0}^{N-1} (s(k) + s_0) e^{-j2\pi u \frac{k}{N}}$$

$$= \sum_{k=0}^{N-1} s(k)\, e^{-j2\pi u \frac{k}{N}} + \sum_{k=0}^{N-1} s_0\, e^{-j2\pi u \frac{k}{N}} = a(u) + s_0 \delta(u)$$

- Integral or summation of translation over a period (k is 0 to N-1) is 0, except u = 0
- Translation only affects the DC component a(0) of the FD

- Scaling

If the 2D shape is scaled by a factor M:

$$s'(k) = M\ s(k)$$

its FD is scaled by the same factor

$$a'(u) = M\ a(u)$$

- Rotation

If the 2D shape is rotated about the origin by an angle ϕ:

$$s'(k) = \ s(k)\ ej^{\phi}$$

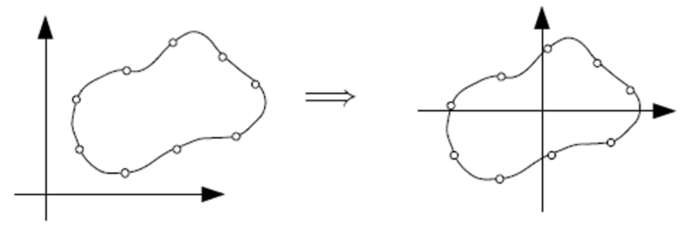
its FD is multiplied by the same factor

$$a'(u) = a(u)\ ej^{\phi}$$

Normalization

- Translation invariance: center the contour at the origin

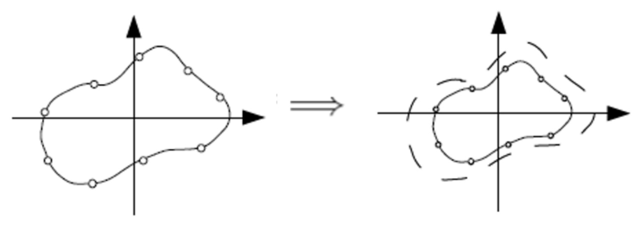
$$a(0) = 0$$



0 frequency contains all and only the information related to translation

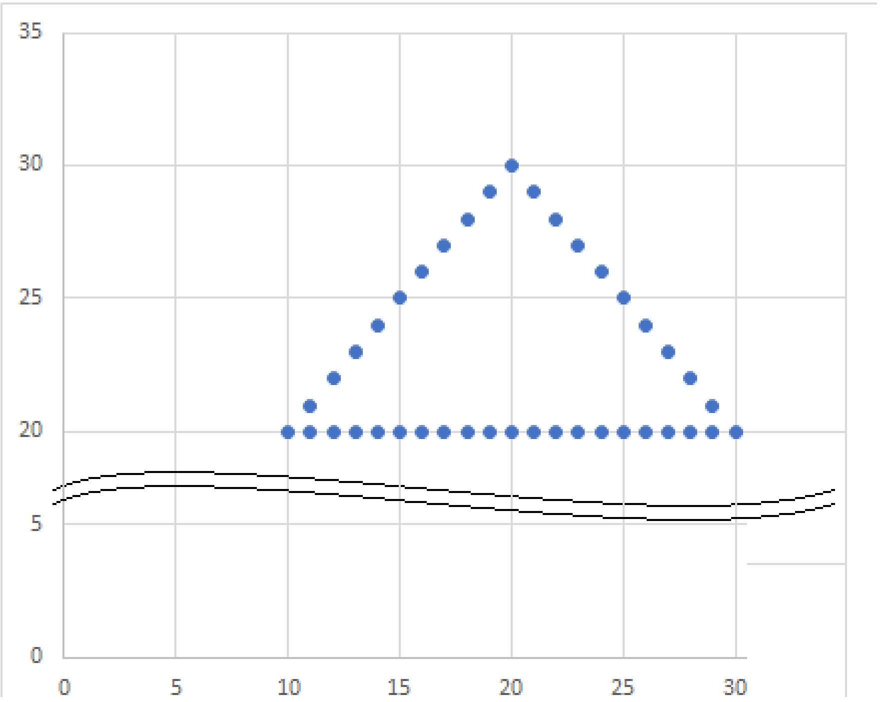
- Scale invariance: standardize the size of the contour

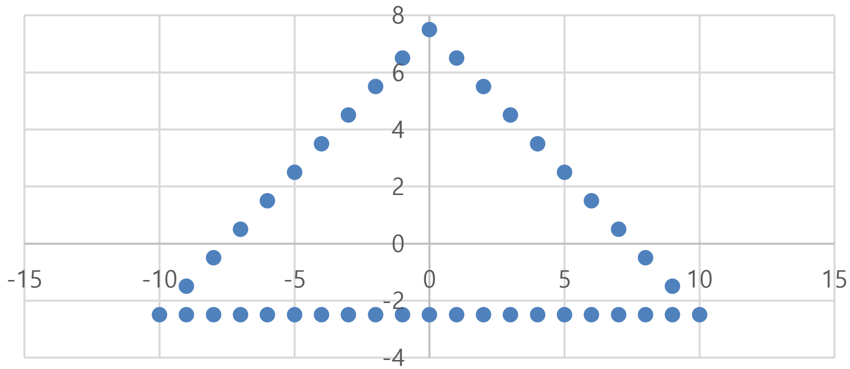
$$a'(u) = a(u)/|a(1)|$$



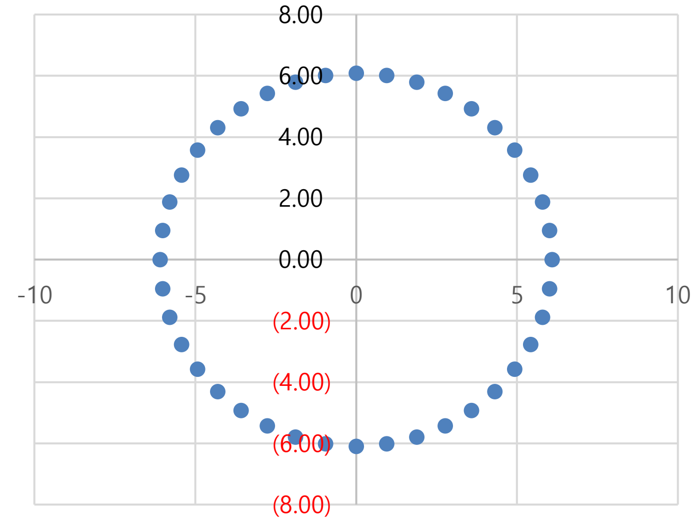
Practice

- Find descriptors invariant to the change in translation, scale and rotation

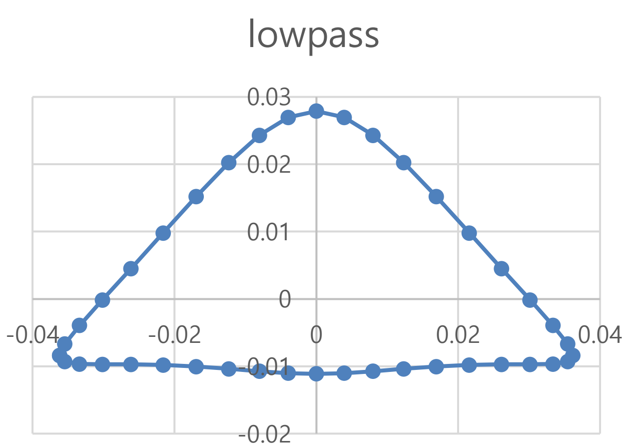




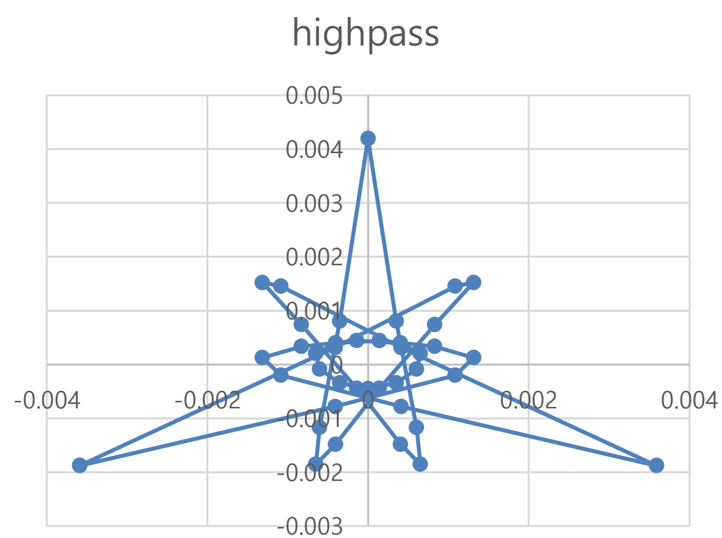
IDFT after removing a(0)



IDFT after leaving only a(1)

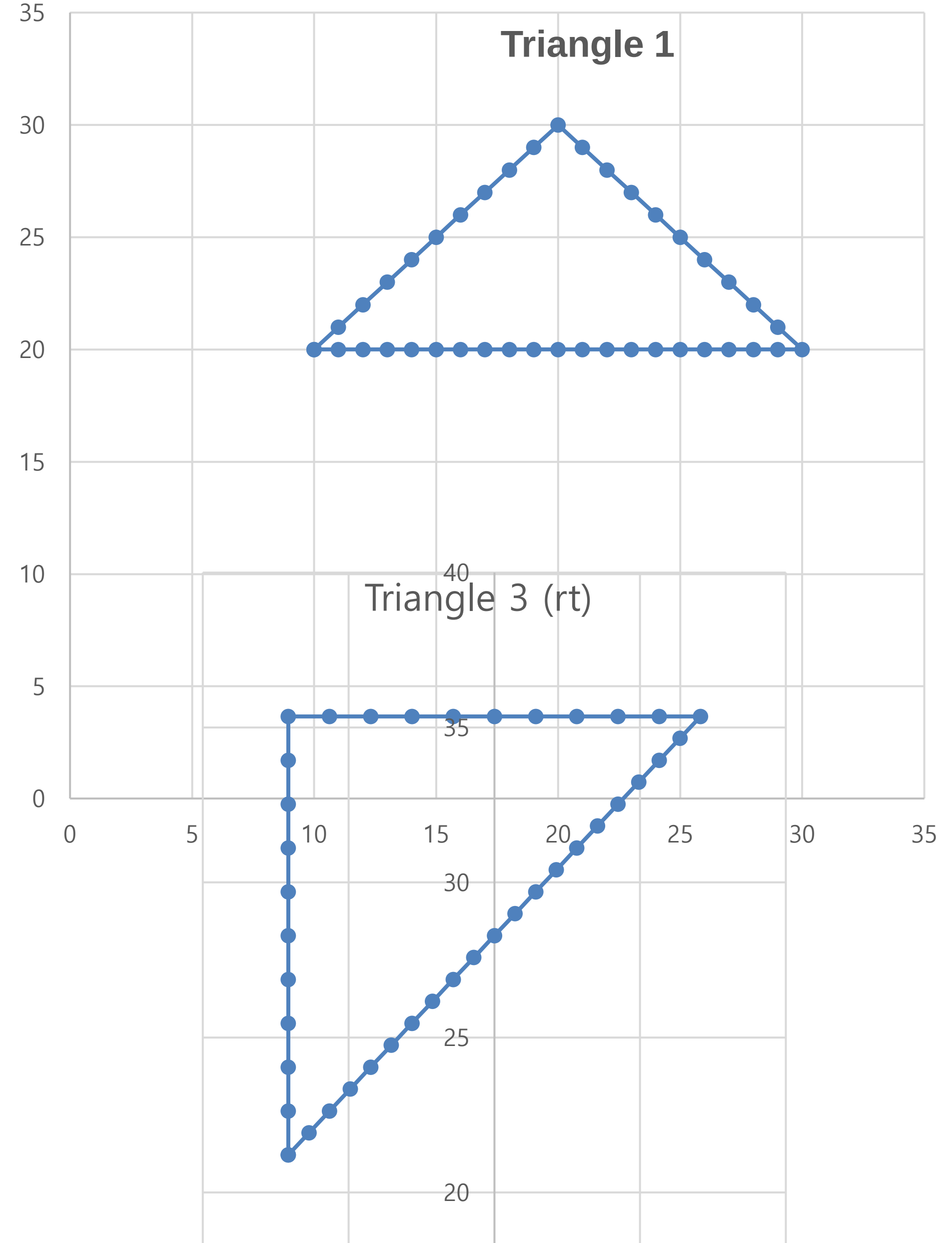


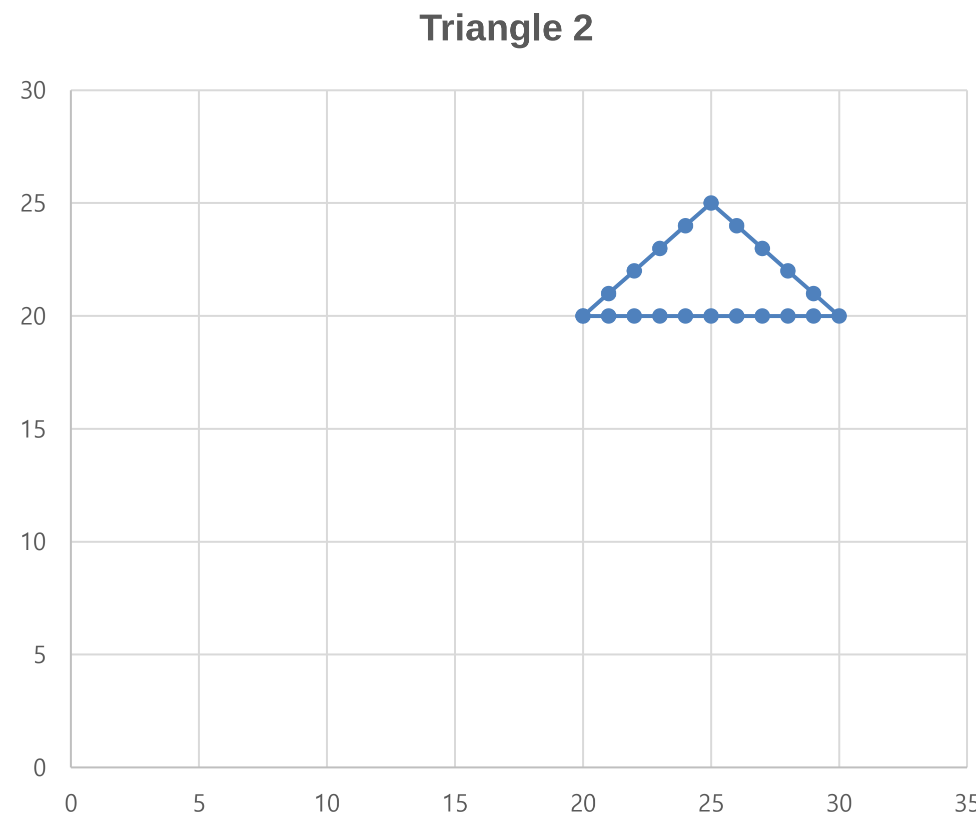
Lowpass filtering

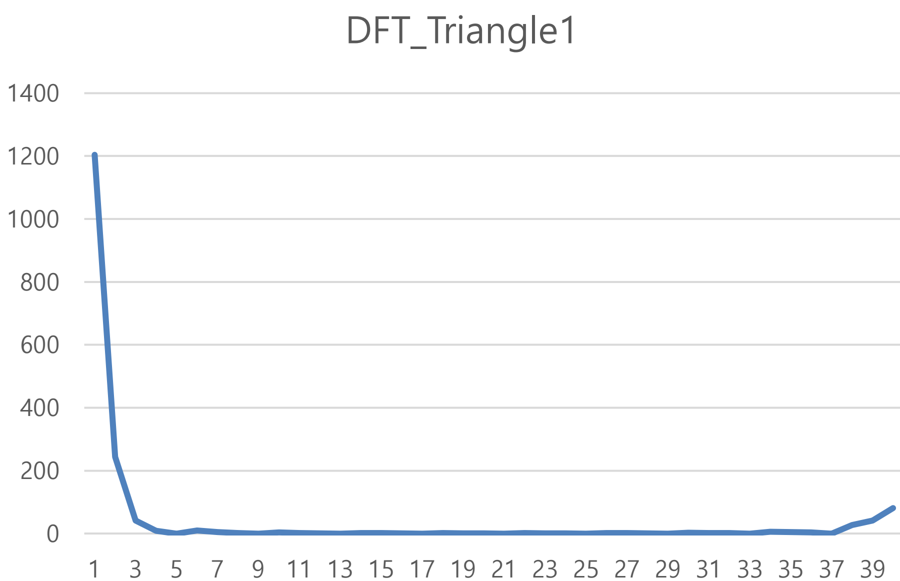


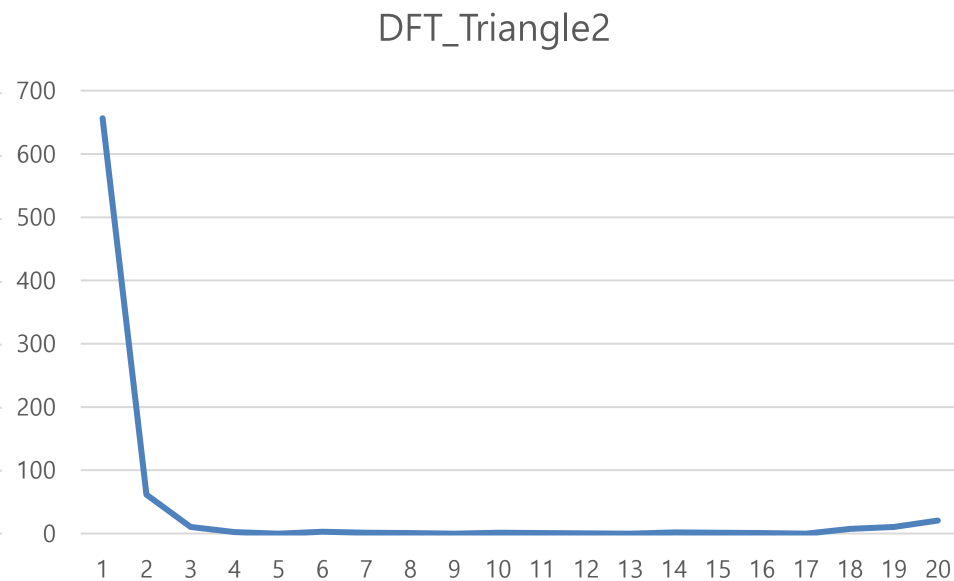
Highpass filtering

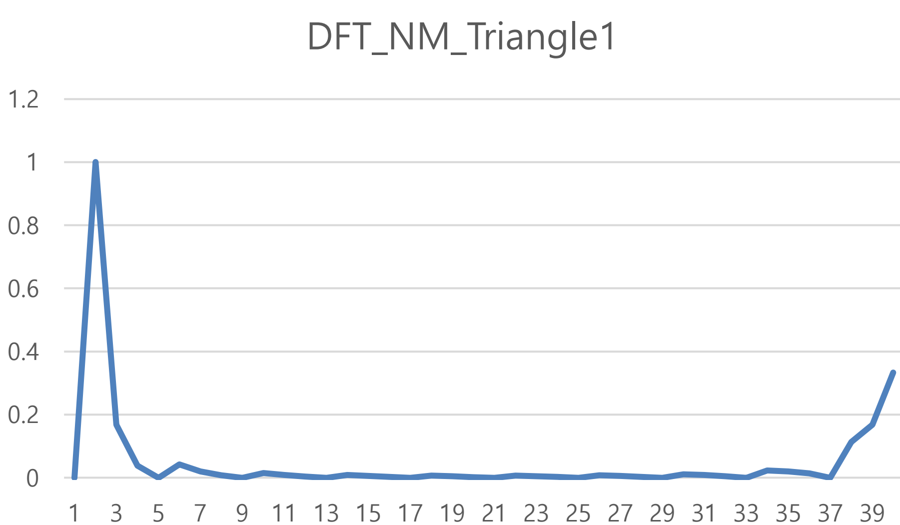
Translation and Scaling

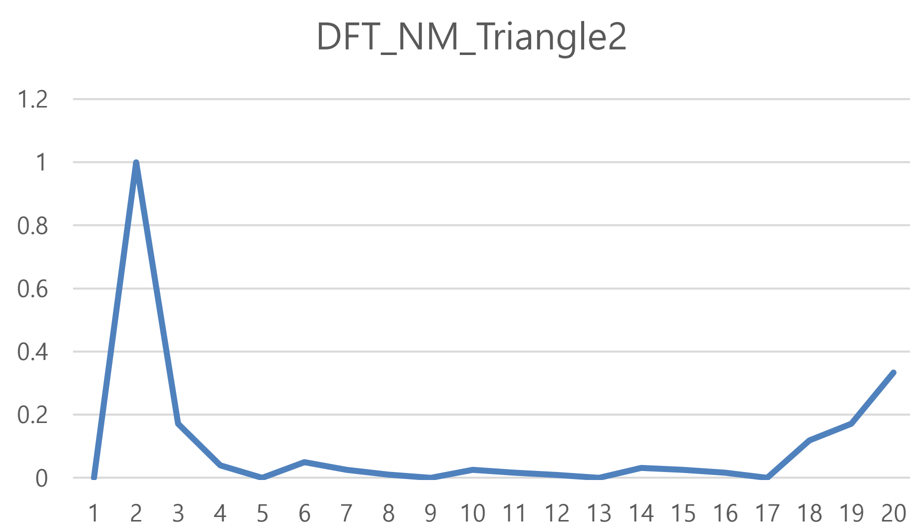


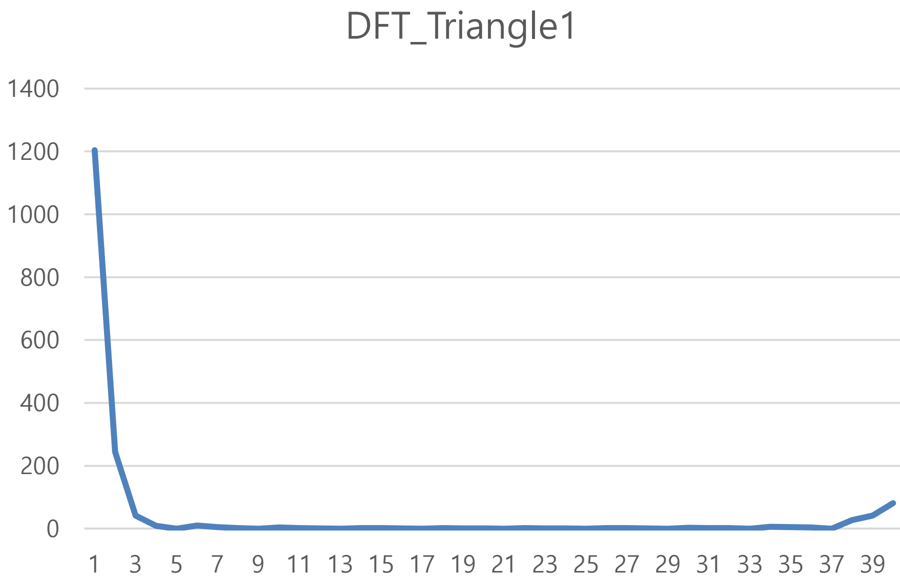


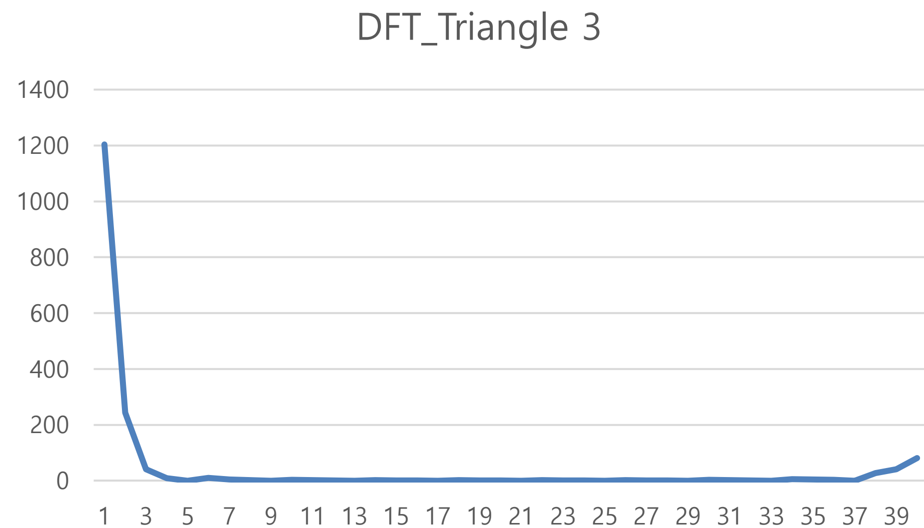

DFT_Triangle1
1400
1200
1000
800
600
400
200
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39

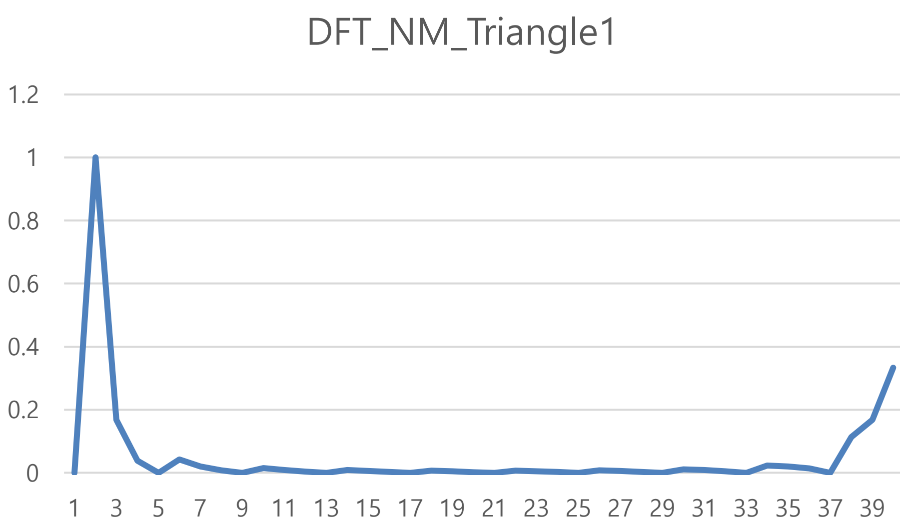

DFT_Triangle2
700
600
500
400
300
200
100
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

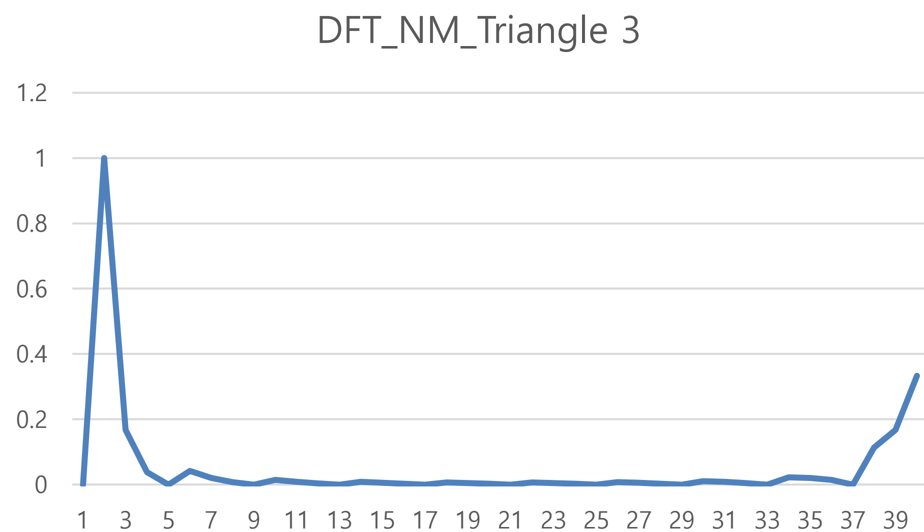

DFT_NM_Triangle1
1.2
1
0.8
0.6
0.4
0.2
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39


DFT_NM_Triangle2
1.2
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


DFT_Triangle1
1400
1200
1000
800
600
400
200
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39


DFT_Triangle 3
1400
1200
1000
800
600
400
200
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39


DFT_NM_Triangle1
1.2
1
0.8
0.6
0.4
0.2
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39


DFT_NM_Triangle 3
1.2
1
0.8
0.6
0.4
0.2
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39

Homework #2

- Write a program to recognize the floor number of the display in an elevator from 1 to 3.
- You can use feature vectors, template matching, frequency profile, histogram, etc. If possible, compare between different two methods.
- If possible, try different-brightness and noised input images
- If possible, try different scales and orientations of input images
- Make use of instructed techniques first, and utilize others if needed next.
- Submit in zip to to Subin Lee (TA) (spiritas@dgist.ac.kr) by the due date (5/31), 23:59:59
- Source codes + input images and result files (or screen capture) + max. 3 page brief explanation including why to use a series of those techniques in a PDF

Minimum Distance Classifier

```
include <iostream>
#include <fstream>
using namespace std;

void averagepattern(double ap[], double p[][3])
{
        int i,j;
        for(j=0;j<3;j++)  {
           ap[j]=0.0;
           for(i=0;i<5;i++)  ap[j] += p[i][j]/5.0;
  }
}

double getdistance(double a[], double b[])
{
        return sqrt((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1])+(a[2]-b[2])*(a[2]-b[2]));
}


int main()
{
        int i;
        double dn,dab;
        double normalpattern[5][3],abnormalpattern[5][3],casepattern[3][3];
        double averagenormalpattern[3],averageabnormalpattern[3];
        double decision[3];

        ifstream ifile("pattern.txt");
        ofstream ofile("out.txt");

        if(!ifile.is_open() || !ofile.is_open() ) { cout << " File Open Error "; exit(1); }
```

```
        for(i=0;!ifile.eof();i++) {
          for(i=0;i<5;i++) {
            ifile >>  normalpattern[i][0] >> normalpattern[i][1]  >> normalpattern[i][2];
        }

        for(i=0;i<5;i++) {
          ifile >>  abnormalpattern[i][0] >> abnormalpattern[i][1]  >> abnormalpattern[i][2];
        }

        for(i=0;i<3;i++) {
          ifile >>  casepattern[i][0] >> casepattern[i][1]  >> casepattern[i][2];
        }

        averagepattern(averagenormalpattern,normalpattern);
        averagepattern(averageabnormalpattern,abnormalpattern);

        for(i=0;i<3;i++) {
            dn = getdistance(averagenormalpattern,casepattern[i]);
            dab = getdistance(averageabnormalpattern,casepattern[i]);

            if(dn>dab) decision[i] = 1;
            else decision[i] = 0;

            ofile <<"Decision " << decision[i] <<  endl;
        }

        ifile.close();
        ofile.close();
        return 1;
}
```

Template Matching

```
#include <iostream>
using namespace std;

#define ROW 256
#define COL 256

unsigned char readimage[ROW][COL], writeimage[ROW][COL],

void tmatching(unsigned char in[][COL],int *maxi, int *maxj)
{
 int i,j,k,m;
 double sum=0,max=255*81;
 int tp[9][9] =   {{230,230,230,230,230,230,230,230,230},
                    {230,230,230,230,230,230,230,230,230},
                    {230,230,230,150,150,150,230,230,230},
                    {230,230,150,150,150,150,150,230,230},
                    {230,230,150,150,150,150,150,230,230},
                    {230,230,150,150,150,150,150,230,230},
                    {230,230,230,150,150,150,230,230,230},
                    {230,230,230,230,230,230,230,230,230},
                    {230,230,230,230,230,230,230,230,230}};

 for(i=4;i<ROW-4;i++)
   for(j=4;j<COL-4;j++) {
     sum =0;
     for(k=-4;k<=4;k++)
       for(m=-4;m<=4;m++)

                  sum += in[i+k][j+m] * tp[k+4][m+4];

                  if(sum>max) {
                             max = sum;
                             *maxi = i;
                             *maxj = j;
                  }
  }
}
```

Various Input Images