Day 5

Teemu Sarapisto

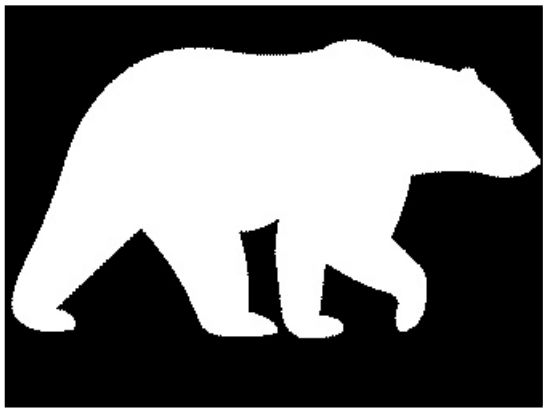
May 22, 2018

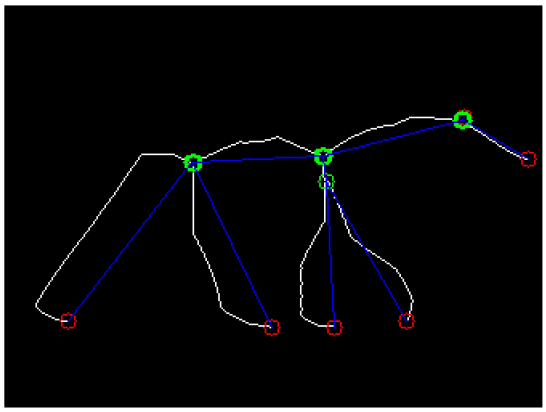
1 Hands-on

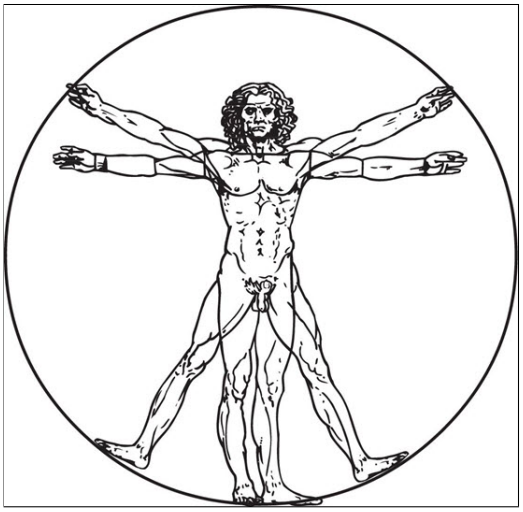
1.1 Create the object skeleton with OpenCV's ximgproc.thinning() function from the original image and try if different values for the thinningType argument would make any difference

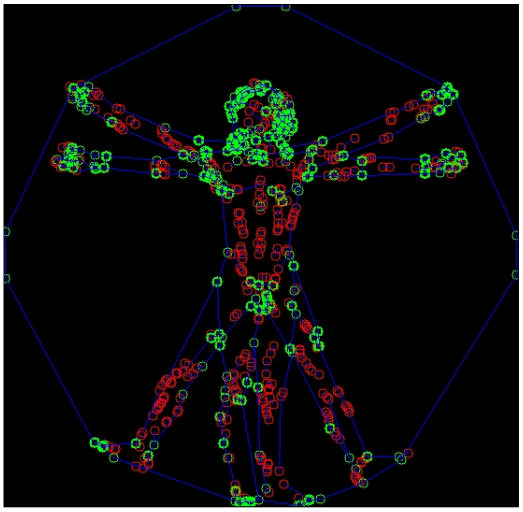
Main difference was that some of the thinners didn't create 1px wide lines.

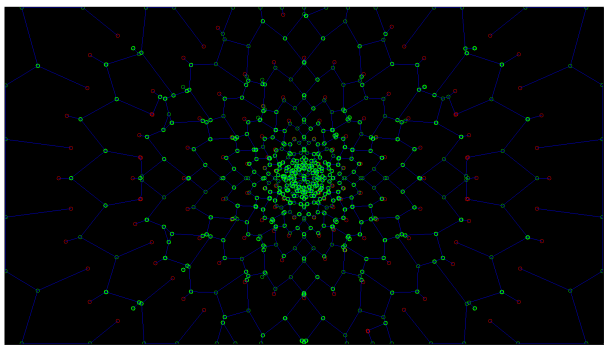
1.2 Graphing



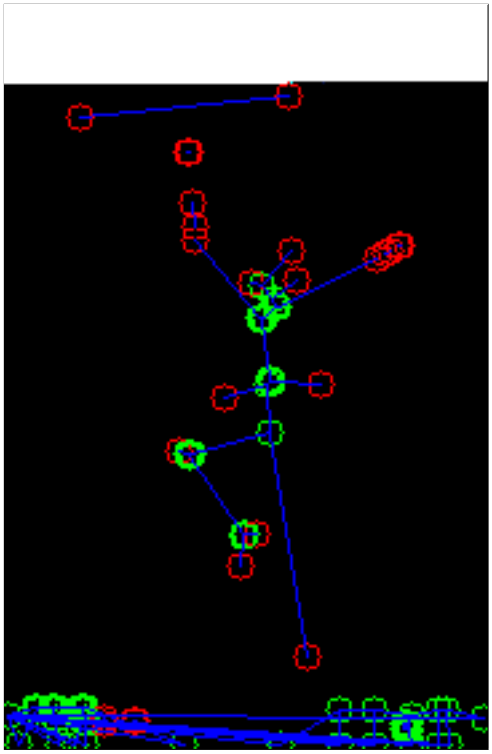


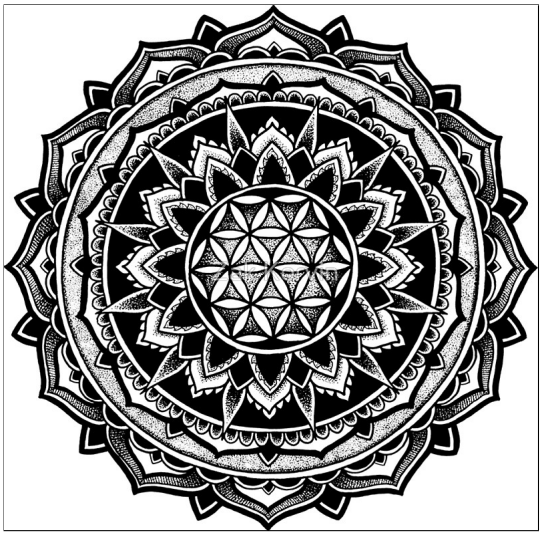


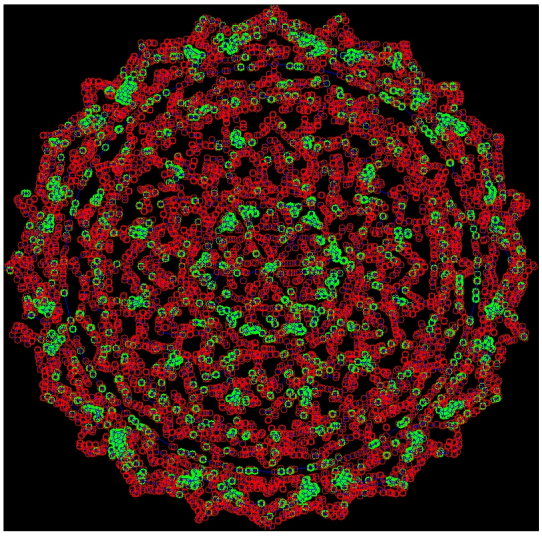


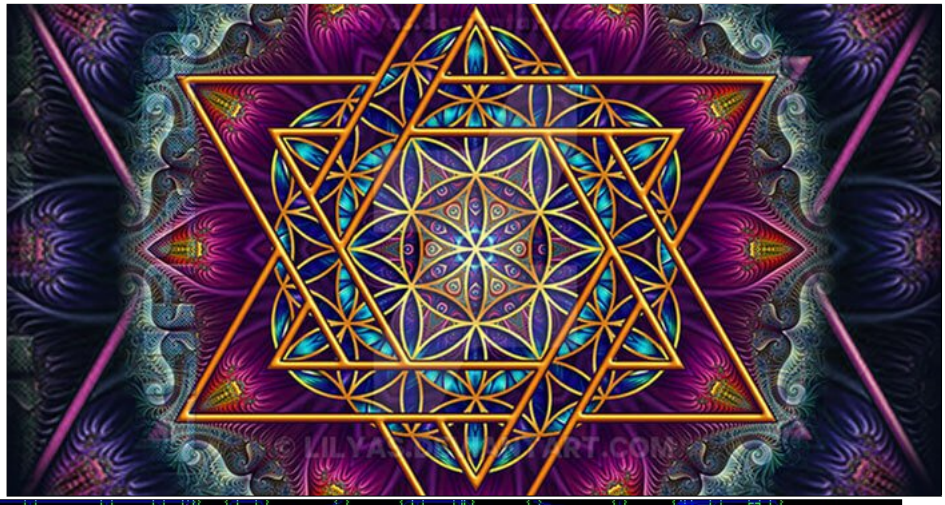


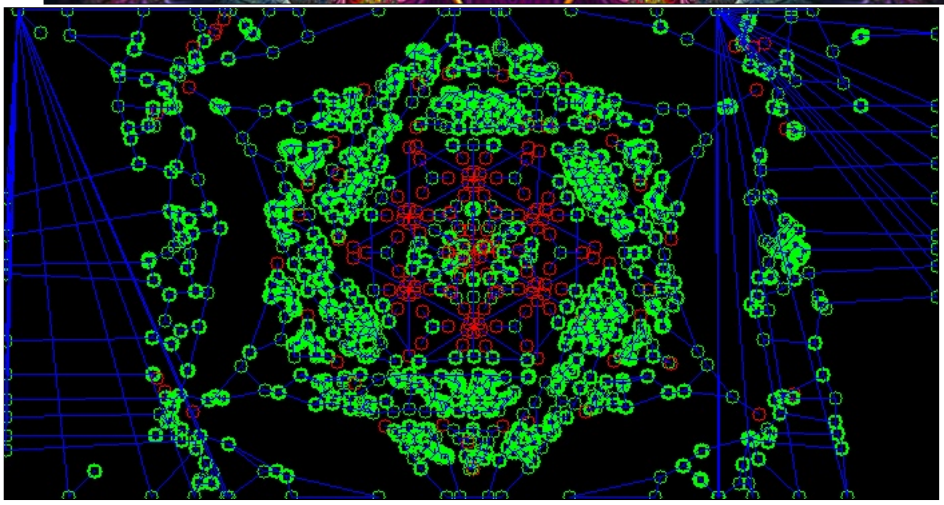

VectorStock®
VectorStock.com/656015











1.3 How long it took

around 4 hours

2 Homework 3

42% classification error with 10k samples. 52.8% classification error with 1k samples.

2.1 Code

```
#! /usr/bin/env python3
```

```
import pickle
import gzip
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import numpy as np
import cv2

with gzip.open('mnist.pkl.gz', 'rb') as fs:
    train_set, valid_set, test_set = pickle.load(fs, encoding='latin1')

train_x, train_y = train_set
valid_x, valid_y = valid_set
test_x,  test_y  = test_set

def get_pixel_lbp(image, pixel, x, y):
    sum = 0
    if image[x-1, y-1] > pixel: sum += 1
    if image[x-1, y] > pixel: sum += 2
    if image[x-1, y+1] > pixel: sum += 4

    if image[x, y-1] > pixel: sum += 8
    if image[x, y+1] > pixel: sum += 16

    if image[x+1, y-1] > pixel: sum += 32
    if image[x+1, y] > pixel: sum += 64
    if image[x+1, y+1] > pixel: sum += 128
    return sum

def lbp_histogram(image):
    histogram = np.zeros(256)
    for y, row in enumerate(image):
        if y == 0 or y == image.shape[1] - 1: continue
        for x, cell in enumerate(row):
            if x == 0 or x == image.shape[0] - 1: continue
            histogram[get_pixel_lbp(image, cell, x, y)] += 1
    return histogram

def train(images):
    results = []
    for image in images:
        hist = lbp_histogram(image.reshape((28, 28)))
        results.append(hist)
    return results

count = 1000
```

```
results = train(train_x[:count])
labels = np.asarray(np.arange(0, count), dtype=np.float32).reshape(count, 1)
for idx, n in enumerate(results):
    labels[idx] = train_y[idx]
knn = cv2.ml.KNearest_create()

knn.train(np.asarray(results, dtype=np.float32), cv2.ml.ROW_SAMPLE, labels)

test_results = train(test_x[:count])
test_results = np.asarray(test_results, dtype=np.float32)

ret, predictions, neighbours, dist = knn.findNearest(test_results, k=1)

success = 0
for idx, res in enumerate(predictions):
    if int(res[0]) == test_y[idx]:
        success += 1

print(success / len(predictions))
```

2.2 How long it took

Around 4 hours. Though I was doing something really stupid with creating the labels for the KNN at first, had I not done that I could've easily done it in two hours...