ДОДАТОК А

Лістинги програми за різними методами

EM-метод

```
# input fllename >> output 3d array
def read_img(filename):
    img_3d = imageio.imread(filename)

    small = cv2.resize(img_3d, (0, 0), fx=0.1, fy=0.1)
    blur = cv2.blur(small, (4, 4))
    return blur


# input 3d array >> output 2d array
def flatten_img(img_3d):
    x, y, z = img_3d.shape
    img_2d = img_3d.reshape(x*y, z)
    img_2d = np.array(img_2d, dtype = np.float)
    return img_2d


# input 2d array >> output 3d array
def recover_img(img_2d, vis = False, X=80, Y=80, Z=3):
    #img_2d = cv2.resize(img_2d, (0, 0), fx=10, fy=10)
    img_2d = (img_2d * 255).astype(np.uint8)
    recover_img = img_2d.reshape(X, Y, Z)
    return recover_img


# input 2d array >> output estimated means, stds, pis
def kmeans_init(img, k):
    means, labels = kmeans2(img, k)
    try:
        means = np.array(means)
        cov = np.array([np.cov(img[labels == i].T) for i in range(k)])
        ids = set(labels)
        pis = np.array([np.sum([labels == i]) / len(labels) for i in ids])
    except Exception as ex:
        pass
    return means, cov, pis
```

```
# E-Step: Update Parameters
# update the conditional pdf - prob that pixel i given class j
def update_responsibility(img, means, cov, pis, k):
    # responsibilities: i th pixels, j th class
    # pis * gaussian.pdf
    responsibilities = np.array([pis[j] * scipy.stats.multivariate_normal.pdf(img, mean=means[j], cov=cov[j]) for j in range(k)]).T
    # normalize for each row
    norm = np.sum(responsibilities, axis = 1)
    # convert to column vector
    norm = np.reshape(norm, (len(norm), 1))
    responsibilities = responsibilities / norm
    return responsibilities


# update pi for each class of Gaussian model
def update_pis(responsibilities):
    pis = np.sum(responsibilities, axis = 0) / responsibilities.shape[0]
    return pis


# update means for each class of Gaussian model
def update_means(img, responsibilities):
    means = []
    class_n = responsibilities.shape[1]
    for j in range(class_n):
        weight = responsibilities[:, j] / np.sum(responsibilities[:, j])
        weight = np.reshape(weight, (1, len(weight)))
        means_j = weight.dot(img)
        means.append(means_j[0])
    means = np.array(means)
    return means
```

```
# update covariance matrix for each class of Gaussian model
def update_covariance(img, responsibilities, means):
    cov = []
    class_n = responsibilities.shape[1]
    for j in range(class_n):
        weight = responsibilities[:, j] / np.sum(responsibilities[:, j])
        weight = np.reshape(weight, (1, len(weight)))
        # Each pixels have a covariance matrice
        covs = [np.mat(i - means[j]).T * np.mat(i - means[j]) for i in img]
        # Weighted sum of covariance matrices
        cov_j = sum(weight[0][i] * covs[i] for i in range(len(weight[0])))
        cov.append(cov_j)
    cov = np.array(cov)
    return cov
```

```
# M-step: choose a label that maximise the likelihood
def update_labels(responsibilities):
    labels = np.argmax(responsibilities, axis = 1)
    return labels


def update_loglikelihood(img, means, cov, pis, k):
    pdf = np.array([pis[j] * scipy.stats.multivariate_normal.pdf(img, mean=means[j], cov=cov[j]) for j in range(k)])
    log_ll = np.log(np.sum(pdf, axis = 0))
    log_ll_sum = np.sum(log_ll)
    return log_ll_sum


def EM_cluster(img, k, error = 10e-4, iter_n = 9999):
    #  init setting
    cnt = 0
    likelihood_arr = []
    # Initialise E-Step by KMeans
    means, cov, pis = kmeans_init(img, k)
    likelihood = 0
    responsibilities = update_responsibility(img, means, cov, pis, k)
    while (abs(likelihood - new_likelihood) > error) and (cnt != iter_n):
        start_dt = datetime.datetime.now()
        cnt += 1
        likelihood = new_likelihood
        # M-Step
        labels = update_labels(responsibilities)
        # E-step
        responsibilities = update_responsibility(img, means, cov, pis, k)
        means = update_means(img, responsibilities)
        cov = update_covariance(img, responsibilities, means)
        pis = update_pis(responsibilities)
        new_likelihood = update_loglikelihood(img, means, cov, pis, k)
        likelihood_arr.append(new_likelihood)
        end_dt = datetime.datetime.now()
        diff = relativedelta(end_dt, start_dt)
        print("iter: %s, time interval: %s:%s:%s:%s" % (cnt, diff.hours, diff.minutes, diff.seconds, diff.microseconds))
    likelihood_arr = np.array(likelihood_arr)
    print('Converge at iteration {}'.format(cnt + 1))
    return labels, means, cov, pis, likelihood_arr

def main():
    FILENAME1 = r'D:\Study\PyApp\Hazelnuts.jpg'
    try:
        orig_img = read_img(FILENAME1)
        x, y, z = orig_img.shape
        plt.figure()
        plt.axis("off")
        plt.imshow(orig_img)
        plt.title('Original Image');
        plt.show()

        img = flatten_img(orig_img)
        labels, means, cov, pis, likelihood_arr = EM_cluster(img, 5)
        em_img = recover_img(means[labels], X=x, Y=y, Z=z)
        plt.figure()
        plt.axis("off")
        plt.imshow(em_img)
        plt.title('Image Segmented by EM Algorithm');
        plt.show()
    except Exception as ex:
        print(ex)

main()
```

Метод Канні

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from ipymol.compat import Image

path = r'D:\Study\PyApp\Hazelnuts.jpg'

img = cv2.imread(path,0)
edges = cv2.Canny(img,100,100)

plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

Оператори Робертса і Собеля

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

from skimage.data import camera
from skimage.filters import roberts, sobel, sobel_h, sobel_v, scharr, \
    scharr_h, scharr_v, prewitt, prewitt_v, prewitt_h, farid_v, farid_h

path = r'D:\Study\PyApp\Hazelnuts.jpg'

image = cv2.imread(path,0)

edge_roberts = roberts(image)
edge_sobel = sobel(image)

fig, ax = plt.subplots(ncols=2, sharex=True, sharey=True,
                       figsize=(8, 4))

ax[0].imshow(edge_roberts, cmap=plt.cm.gray)
ax[0].set_title('Roberts Edge Detection')

ax[1].imshow(edge_sobel, cmap=plt.cm.gray)
ax[1].set_title('Sobel Edge Detection')

for a in ax:
    a.axis('off')

plt.tight_layout()
plt.show()
```

Номер докумен-та	*Познака*	*Найменування*	*Прим.*
		Текстові документи	
1	*ГЮІК.122019.024ПЗ*	*Пояснювальна записка*	*110 с.*
2		*Рецензія*	*1 с.*
3		*Відзив керівника*	*1 с.*
4		*Науково-технічні публікації (апробація)*	*8 с.*
5		*Сертифікати про участь у науково-технічних заходах*	*1 с.*
		Графічні документи	
6		*Презентаційний матеріал*	*12 с.*
		Електронні матеріали	
7		*2019_ІНФ_Зеленський_М_О.docx*	
8		*2019_ІНФ_Зеленський_М_О.pdf*	
9		*2019_ІНФ_Зеленський_М_О.pptx*	
10		*Каталог з програмою – program*	
11		*2019_ІНФ_ Зеленський_М_О_readme.txt*	
12		*Оригінальність тексту _____ відсотків*	
		Керівник атестаційної роботи	
		доц. Творошенко І.С._____	

Зм	*Арк*	*№ докум.*	*Підпис*	*Дата*	*ГЮІК.122019.024Д4*		
Розроб.		*Зеленський М.О.*			***Дослідження гібридних методів для класифікації складноструктурованих зображень*** *Відомість атестаційної роботи магістра*	*Літ*	*Арк* / *Аркуш*
Перевір.		*Творошенко І.С*					1
Т.контр						*ХНУРЕ* *Кафедра Інформатики*	
Н.контр		*Творошенко І.С.*					
Затв.		*Путятін Є.П.*					