VM

Image segmentation based on k-means

Initially we have a set of images with people in some clothes. The images are located in the folder called **photos**. At first we apply the k-means for segmenting the image, then there is some manual processing.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import PIL
import pickle

from skimage import io
from PIL import Image

%matplotlib inline
```

In [2]:
```
def show_image(buf):
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    ax.axis('off')
    ax.imshow(buf)

def save_image(buf,name):
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    ax.axis('off')
    ax.imshow(buf)
    # dots per inches
    plt.savefig(name,dpi=600)
    plt.close(fig)
```

In [3]:
```
# images names from the folder 'photos'
img_names = os.listdir("photos/")

# initially listdir works in an awkward way
# let's sort the names in the natural order
import re
def sorted_aphanumeric(data):
    convert = lambda text: int(text) if text.isdigit() else text.lower()
    alphanum_key = lambda key: [ convert(c) for c in re.split('([0-9]+)', key) ]
    return sorted(data, key=alphanum_key)

img_names = sorted_aphanumeric(img_names)

# IMG list will contain all the images from 'photos' folder
IMG = []

for i in img_names:
    IMG.append(io.imread("photos/"+i))
```

In [5]:
```
# let's create a folder for the interim results
# there will be all the segments from out images
newpath = r'results'
if not os.path.exists(newpath):
    os.makedirs(newpath)
else:
    print('The folder is already existing')
```

In [6]:
```
# here will be the final images aka extracted clothes
newpath = r'final_results'
if not os.path.exists(newpath):
    os.makedirs(newpath)
else:
    print('The folder is already existing')
```

In [7]:
```
# K - number of segments
def kmeans_segmentation(image,K):
    # размываем чуток фотку
    image=cv2.GaussianBlur(image,(7,7),0)
    # преобразовываем в вектор матрицу
    flat=image.reshape(-1,3)
    # type(img[0][0][0]) = uint8, cv2 нужен float32
    flat=np.float32(flat)
    # критерий остановки
    crit=(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

    ret,label,centroids=cv2.kmeans(flat,K,None,crit,10,cv2.KMEANS_RANDOM_CENTERS)
    res = centroids[label.flatten()]
    segmented_image = res.reshape((image.shape))

    return label.reshape((image.shape[0],image.shape[1])),segmented_image.astype(np.uint8)

# label - интересущий нас класс
def extraction(image,labels,label):
    component=np.zeros(image.shape,np.uint8)+255
    component[labels==label]=image[labels==label]
    return component
```

In [8]:
```
K = 5
'''
# applying k-means with 5 segments
LABELS = []
for i in IMG:
    labels_i, segmented_image_i = kmeans_segmentation(i,K)
    LABELS.append(labels_i)

with open("labels.txt", "wb") as fp:   #Pickling
    pickle.dump(LABELS, fp)
'''
```

```
# Here we already launched the code, so the labels are stored in labels.txt
# and we should upload them
LABELS = []

with open("labels.txt", "rb") as fp:   # Unpickling
    LABELS = pickle.load(fp)
```

In [11]:
```
# saving all segments to 'results' folder
for i in range(len(IMG)):
    for j in range(K):
        save_image(extraction(IMG[i],LABELS[i],j),"results/"+str(i+1)+'_'+str(j+1)+'.png')
```

In [13]:
```
# manualy processing the data
# picking the necessary segements
# if one segment is not enough, adding more of them

fin_res = []

fin_res.append(io.imread('results/'+'1_2.png')+io.imread('results/'+'1_5.png'))
fin_res.append(io.imread('results/'+'2_1.png')+io.imread('results/'+'2_2.png'))
fin_res.append(io.imread('results/'+'3_5.png'))
fin_res.append(io.imread('results/'+'4_2.png'))
fin_res.append(io.imread('results/'+'5_4.png'))
fin_res.append(io.imread('results/'+'6_4.png')+io.imread('results/'+'6_5.png'))
fin_res.append(io.imread('results/'+'7_1.png'))
fin_res.append(io.imread('results/'+'8_4.png'))
fin_res.append(io.imread('results/'+'9_3.png'))
fin_res.append(io.imread('results/'+'10_3.png'))
fin_res.append(io.imread('results/'+'11_4.png'))
fin_res.append(io.imread('results/'+'12_5.png'))
fin_res.append(io.imread('results/'+'13_1.png'))
fin_res.append(io.imread('results/'+'14_2.png')+io.imread('results/'+'14_4.png'))
fin_res.append(io.imread('results/'+'15_4.png'))
fin_res.append(io.imread('results/'+'16_3.png')+io.imread('results/'+'16_4.png'))
fin_res.append(io.imread('results/'+'17_1.png'))
fin_res.append(io.imread('results/'+'18_4.png'))
fin_res.append(io.imread('results/'+'19_3.png'))
fin_res.append(io.imread('results/'+'20_3.png')+io.imread('results/'+'20_4.png'))

fin_res = np.array(fin_res)
```

In [14]:
```
# final touch: resizing

size = 512,512

for i in range(len(fin_res)):
    img = fin_res[i]
    # Convert to gray, and threshold
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    th, threshed = cv2.threshold(gray, 240, 255, cv2.THRESH_BINARY_INV)

    # Morph-op to remove noise
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))
    morphed = cv2.morphologyEx(threshed, cv2.MORPH_CLOSE, kernel)

    # Find the max-area contour
    cnts = cv2.findContours(morphed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
    cnt = sorted(cnts, key=cv2.contourArea)[-1]

    # Crop and save it
    x,y,w,h = cv2.boundingRect(cnt)
    dst = img[y:y+h, x:x+w]

    dst = PIL.Image.fromarray(dst)
    dst.thumbnail(size, Image.ANTIALIAS)

    save_image(np.array(dst),"final_results/"+str(i+1)+'.png')
```

Some sources

- https://dufferdev.wordpress.com/2014/12/21/image-segmentation-using-k-means/
- https://mubaris.com/posts/kmeans-clustering/
- https://stackoverflow.com/questions/48395434/how-to-crop-or-remove-white-background-from-an-image
- https://stackoverflow.com/questions/273946/how-do-i-resize-an-image-using-pil-and-maintain-its-aspect-ratio
- https://docs.opencv.org/3.1.0/d1/d5c/tutorial_py_kmeans_opencv.html
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html
- https://stackoverflow.com/questions/29313667/how-do-i-remove-the-background-from-this-kind-of-image
- https://stackoverflow.com/questions/4813061/non-alphanumeric-list-order-from-os-listdir