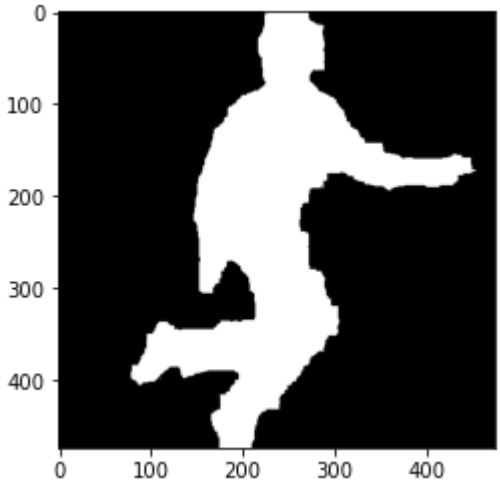
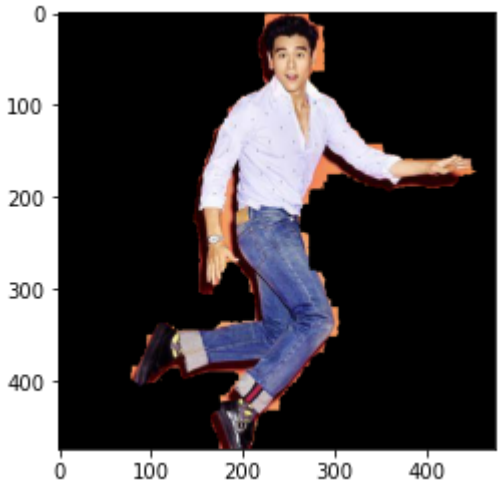
In [1]:

```
from cv2 import cv2
import numpy as np
import maxflow
import matplotlib.pyplot as plt
MAXVAL=1000000
def get_node_num(i, j, M):
    return j*M+i
def get_xy(nodenum, M):
    return nodenum % M, int(nodenum/M)

def colorplatte():
    shirtc = np.zeros_like(I)
    shirtc[:,:,0].fill(62)
    shirtc[:,:,1].fill(151)
    shirtc[:,:,2].fill(255)
    personc = np.zeros_like(I)
    personc[:,:,0].fill(218)
    personc[:,:,1].fill(114)
    personc[:,:,2].fill(99)
    jeansc = np.zeros_like(I)
    jeansc[:,:,0].fill(231)
    jeansc[:,:,1].fill(205)
    jeansc[:,:,2].fill(137)
    bgc = np.zeros_like(I)
    bgc[:,:,0].fill(93)
    bgc[:,:,1].fill(141)
    bgc[:,:,2].fill(125)
    ac = np.zeros_like(I)
    ac[:,:,2].fill(255)
    bc = np.zeros_like(I)
    bc[:,:,0].fill(255)
    cc = np.zeros_like(I)
    cc[:,:,0].fill(255)
    cc[:,:,1].fill(255)
    dc = np.zeros_like(I)
    dc[:,:,1].fill(255)
    dc[:,:,2].fill(255)
    platte = [shirtc,personc,jeansc,bgc,ac,bc,cc,dc]
    return platte

def parse_data(data):
    datafile=open(data)
    dataArr=[]
    for line in datafile.readlines():
        curpair=line.split("\t")
        curpair=list(map(int,curpair))
        dataArr.append((curpair[0]-1,curpair[1]-1))
    return dataArr

def const_G(I,M,N,bg,fg):
    G=np.array([0.5]*(M*N))
    G=np.reshape(G,(M,N))
    # print(G.shape)
    # print(bg[0])
    for i in range(len(bg)):
        G[bg[i][0]][bg[i][1]] = 0
    for i in range(len(fg)):
        G[fg[i][0]][fg[i][1]] = 1
    return G
```

```
def VertexEdge(G,M,I):
    V=[]
    E=[]
    for i in range(M):
        for j in range(N):
            if G[i,j]==0:
                V.append((get_node_num(j,i,M), MAXVAL, 0))
            elif G[i,j]==1:
                V.append((get_node_num(j,i,M), 0, MAXVAL))
            else:
                V.append((get_node_num(j,i,M), 0, 0))
    for i in range(M-1):
        for j in range(N-1):
            state=get_node_num(j,i,M)
            nstate=get_node_num(j+1,i,M)
            tmp=np.sum((I[i,j]-I[i,j+1])**2)
            w=1/(1+tmp)
            E.append((state,nstate,w))

            nstate=get_node_num(j,i+1,M)
            tmp=np.sum((I[i,j]-I[i+1,j])**2)
            w=1/(1+tmp)
            E.append((state,nstate,w))
    return V,E

def find_maxflow(I,V,E,M):
    mask = np.zeros_like(I, dtype=bool)
    g = maxflow.Graph[float](len(V), len(E))
    vArr = g.add_nodes(len(V))
    for v in V:
        # add edges between a non-terminal node and terminal nodes
        # Add an edge 'SOURCE->i' with capacity cap_source and another edge 'i->SINK' with capac
        # This method can be called multiple times for each node. Capacities can be negative.
        g.add_tedge(vArr[v[0]], v[1], v[2])
    for e in E:
        # add_edge(self, int i, int j, long capacity, long rcapacity)
        # Adds a bidirectional edge between nodes i and j with the weights cap and rev_cap.
        g.add_edge(e[0], e[1], e[2], e[2])
    g.maxflow()
    for idx in range(len(V)):
        # Returns which segment the given node belongs to.
        if g.get_segment(idx) == 1:
            i,j=get_xy(idx, M)
            mask[j,i]=(1,1,1)
    return mask
def visulize(mask):
    res = np.zeros_like(I)
    np.copyto(res, I, where=mask)
    plt.imshow(mask.astype(np.float))
    plt.show()
    plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
    plt.show()

def runGC(I,M,N,bg,fg):
    G=const_G(I,M,N,bg,fg)
    V,E=VertexEdge(G,M,I)
    mask=find_maxflow(I,V,E,M)
    return mask

def MultiGC(I,bg,shirt,person,jeans):
    groups=[shirt,person,jeans,bg]
```

```
    res = np.zeros_like(I)
    thickc = np.zeros_like(I)
    platte=colorplatte()
    for i in range(len(groups)):
        fg = groups[i]
        bg = []
        for j in range(len(groups)):
            if j != i:
                bg += groups[j]
        part=runGC(I,M,N,bg,fg)
        np.copyto(res, platte[i], where=part)
        np.copyto(thickc, platte[i+4], where=part)
    plt.imshow(res)
    plt.show()
    multi = cv2.addWeighted(cv2.cvtColor(I, cv2.COLOR_BGR2RGB), 0.8, thickc, 0.4, 0)
    plt.imshow(multi)
    plt.show()


I=cv2.imread("Pyy.jpg")
bg=parse_data('points/bg_points.txt')
# bgv2=parse_data('bgv2_points.txt')
fg=parse_data('points/fg_points.txt')
fg_comb=parse_data('points/fg_comb.txt')

shirt=parse_data('points/shirt_points.txt')
person=parse_data('points/person_points.txt')
jeans=parse_data('points/jeans_points.txt')
M,N,P=I.shape


mask=runGC(I,M,N,bg,fg_comb)
visulize(mask)
```






In [2]:

```
MultiGC(I,bg,shirt,person,jeans)
```