Part 1: Segmentation of hands

In this part, you are given a sequence of video frames in which a person is playing the piano with both hands. Try to develop an algorithm to identify the pianist's hands. Portions of the hands are sometimes in deep shadow, which creates a challenging imaging situation.

In [292]:

```
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
from os import listdir, makedirs
from os.path import isfile, join, abspath, exists
```

In [293]:

```
# read data
data_path = abspath('./CS585-PianoImages')
data_list = [join(data_path, file) for file in listdir(data_path) if isfile(join(data_p
```

In [294]:

```
data_list
```

Out[294]:

```
['C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_14.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_15.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_16.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_17.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_18.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_19.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_22.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_23.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_24.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_25.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_26.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_27.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_33.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_34.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_35.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_36.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_37.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_38.png',
 'C:\\Users\\ljsPC\\Desktop\\CS585\\HW4\\HW4-CS585\\CS585-PianoImages\\p
iano_39.png']
```

In [295]:

```
data_frames = []

for file in data_list:
    img = cv2.imread(file)
    data_frames.append(img.copy())

    img_grayscale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

data_frames = np.array(data_frames)
```

compute the average value of all image frames

In [296]:

```
avg_img = np.sum(data_frames, axis=0)


avg_img = avg_img / data_frames.shape[0]
avg_img = avg_img.astype(np.uint8)
```

In [297]:

```
plt.imshow(avg_img)
plt.show()
```

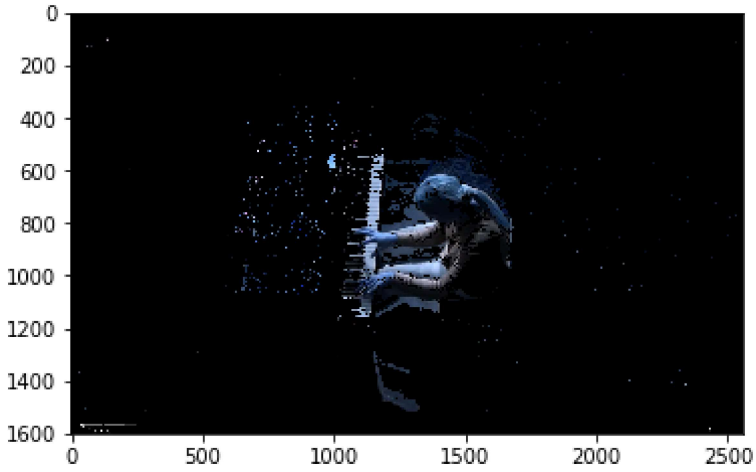
Difference the frames

In [298]:

```
motions = []

for frame in data_frames:
    diff = cv2.absdiff(frame, avg_img)
    diff_grayscale = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)

    boolean_filter = diff_grayscale > 5
    motion = np.zeros_like(frame)
    motion[boolean_filter] = frame[boolean_filter]
    motions.append(motion)

motions = np.array(motions)
```

In [299]:

```
plt.imshow(motions[0])
plt.show()
```



Skin Color Detection

In [300]:

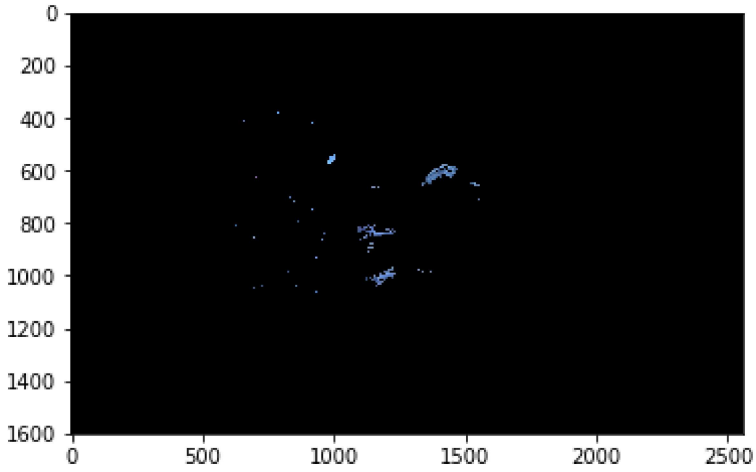
```
# returns the @param img with only skin color (the rest of the pixels are black)
def skinDetect(img):
    B = img[:, :, 0]
    G = img[:, :, 1]
    R = img[:, :, 2]
    skin = np.zeros_like(img)
    maxMat = imgMax(img)
    minMat = imgMin(img)

    cond = (R > 100) & (B > 50) & (B < 130) & (G > 70) & ((maxMat - minMat) > 50) & (np
    skin[cond] = img[cond]
    return skin

def imgMax(img):
    maxMat = np.max(img.reshape(img.shape[0]*img.shape[1], 3), axis=1).reshape(img.shap
    return maxMat

def imgMin(img):
    minMat = np.min(img.reshape(img.shape[0]*img.shape[1], 3), axis=1).reshape(img.shap
    return minMat
```

In [301]:

```
skin_imgs = []
for img in motions:
    skin_imgs.append(skinDetect(img))
```

In [302]:

```
plt.imshow(skin_imgs[6])
plt.show()
```



Find Hand Positions

In [303]:

```
# return the bounding boxes of the largest three blobs
def three_largest_blobs(grayscale_img):
    _, contours_opencv, hierarchy = cv2.findContours(grayscale_img, cv2.RETR_TREE, cv2.(

    # choose the largest size object
    largest_three_area = [0,0,0]
    largest_three_index = [0,0,0]
    for index in range(len(contours_opencv)):
        area = cv2.contourArea(contours_opencv[index])
        if area > np.min(largest_three_area):
            largest_three_index[np.argmin(largest_three_area)] = index
            largest_three_area[np.argmin(largest_three_area)] = area

    bounding_boxes = []
    for index in largest_three_index:
        x, y, w, h = cv2.boundingRect(contours_opencv[index])
        bounding_boxes.append([x, y, w, h])

    return bounding_boxes, largest_three_area
```

In [304]:

```
bounding_boxes = []
largest_three_areas = []
for skin_img in skin_imgs:
    img_skin_grayscale = cv2.cvtColor(skin_img, cv2.COLOR_BGR2GRAY)

    largest_three__boxes, largest_three_area = three_largest_blobs(img_skin_grayscale)
    bounding_boxes.append(largest_three__boxes)
    largest_three_areas.append(largest_three_area)

print(bounding_boxes, largest_three_areas)
```

```
[[[1118, 840, 129, 35], [1316, 624, 153, 86], [1110, 982, 127, 65]], [[1
120, 880, 113, 39], [1342, 579, 142, 86], [1110, 1000, 122, 34]], [[113
4, 985, 93, 48], [1120, 856, 104, 43], [1363, 577, 121, 76]], [[1109, 82
5, 131, 52], [1350, 579, 147, 78], [1109, 996, 127, 50]], [[1137, 988, 9
9, 53], [1316, 590, 163, 95], [1115, 871, 114, 40]], [[1134, 991, 98, 4
8], [1108, 870, 118, 40], [1316, 588, 163, 95]], [[1118, 990, 119, 52],
[1350, 581, 128, 76], [1102, 826, 134, 42]], [[1346, 583, 139, 78], [110
5, 872, 129, 43], [1110, 1012, 136, 42]], [[1117, 874, 114, 40], [1374,
570, 135, 79], [1114, 1001, 123, 34]], [[1099, 879, 139, 45], [1384, 55
9, 135, 77], [1111, 998, 125, 35]], [[1372, 553, 138, 77], [1145, 983, 9
4, 53], [1107, 845, 128, 45]], [[1108, 990, 121, 40], [1361, 562, 136, 8
7], [1064, 866, 162, 47]], [[1281, 442, 132, 69], [1199, 650, 44, 14],
[1123, 639, 124, 78]], [[1150, 548, 112, 43], [1294, 389, 125, 65], [113
8, 594, 110, 71]], [[1124, 736, 117, 40], [1121, 788, 126, 70], [1277, 5
59, 153, 70]], [[1098, 911, 126, 59], [1253, 649, 159, 85], [1117, 996,
134, 38]], [[1129, 893, 107, 45], [1107, 1050, 153, 37], [1271, 645, 15
4, 84]], [[1284, 613, 148, 82], [1119, 1050, 137, 41], [1105, 882, 141,
62]], [[1108, 672, 99, 40], [1461, 607, 27, 22], [1124, 804, 129, 53]]]
[[1056.0, 4222.0, 2237.5], [1493.0, 4826.0, 1600.5], [1463.5, 1236.0, 44
65.0], [1531.5, 4595.5, 2253.5], [1720.0, 4630.0, 1833.0], [1598.0, 184
9.5, 4795.0], [1932.0, 4369.5, 990.0], [4529.5, 2092.5, 1974.5], [1405.
5, 4346.0, 1724.5], [2530.5, 4262.5, 1584.0], [4179.5, 1845.0, 1960.0],
[1615.0, 4098.0, 2160.0], [4270.0, 325.0, 3912.0], [2777.5, 3988.0, 173
9.5], [2490.5, 2126.5, 4629.0], [2352.0, 4812.0, 1557.0], [1275.5, 1891.
5, 4207.0], [5204.0, 1867.5, 2444.5], [2241.0, 184.5, 2894.0]]
```

In [305]:

```
# two hands and the head are detected
# exclude head blob
for index in range(len(bounding_boxes)):
    x_max = 0
    max_index = 0
    for blob_index in range(len(bounding_boxes[index])):
        if bounding_boxes[index][blob_index][0] > x_max:
            x_max = bounding_boxes[index][blob_index][0]
            max_index = blob_index
    del bounding_boxes[index][max_index]
    del largest_three_areas[index][max_index]

bounding_boxes = np.array(bounding_boxes)
largest_three_areas = np.array(largest_three_areas)
bounding_boxes
```

Out[305]:

```
array([[[1118,  840,  129,   35],
        [1110,  982,  127,   65]],

       [[1120,  880,  113,   39],
        [1110, 1000,  122,   34]],

       [[1134,  985,   93,   48],
        [1120,  856,  104,   43]],

       [[1109,  825,  131,   52],
        [1109,  996,  127,   50]],

       [[1137,  988,   99,   53],
        [1115,  871,  114,   40]],

       [[1134,  991,   98,   48],
        [1108,  870,  118,   40]],

       [[1118,  990,  119,   52],
        [1102,  826,  134,   42]],

       [[1105,  872,  129,   43],
        [1110, 1012,  136,   42]],

       [[1117,  874,  114,   40],
        [1114, 1001,  123,   34]],

       [[1099,  879,  139,   45],
        [1111,  998,  125,   35]],

       [[1145,  983,   94,   53],
        [1107,  845,  128,   45]],

       [[1108,  990,  121,   40],
        [1064,  866,  162,   47]],

       [[1199,  650,   44,   14],
        [1123,  639,  124,   78]],

       [[1150,  548,  112,   43],
        [1138,  594,  110,   71]],
```

```
       [[1124,  736,  117,   40],
        [1121,  788,  126,   70]],

       [[1098,  911,  126,   59],
        [1117,  996,  134,   38]],

       [[1129,  893,  107,   45],
        [1107, 1050,  153,   37]],

       [[1119, 1050,  137,   41],
        [1105,  882,  141,   62]],

       [[1108,  672,   99,   40],
        [1124,  804,  129,   53]]])
```

In [306]:

```
# draw bounding box for each image
blob_area_threshold = 500

for img_index in range(data_frames.shape[0]):
    for bounding_box_index in range(bounding_boxes[img_index].shape[0]):
        x, y, w, h = bounding_boxes[img_index][bounding_box_index]

        # two hands are separated
        if np.min(largest_three_areas[img_index]) > blob_area_threshold:
            # left hand at the bottom
            if y == np.max(bounding_boxes[img_index][:,1]):
                cv2.rectangle(data_frames[img_index], (x, y), (x + w, y + h), (0, 255, 
                cv2.putText(data_frames[img_index], 'left hand', (x, y),cv2.FONT_HERSHE
            # right hand
            else:
                cv2.rectangle(data_frames[img_index], (x, y), (x + w, y + h), (0, 0, 25
                cv2.putText(data_frames[img_index], 'right hand', (x, y),cv2.FONT_HERSH
        # two hands are overlapped
        elif largest_three_areas[img_index][bounding_box_index] > blob_area_threshold:
            cv2.rectangle(data_frames[img_index], (x, y), (x + w, y + h), (0, 255, 0), 
            cv2.putText(data_frames[img_index], 'overlapped hand', (x, y),cv2.FONT_HERS
```

In [307]:

```
plt.imshow(data_frames[0])
plt.show()
```