UE19CS333 – Assignment 5: Video frame processing/colour image processing

Team No: 5

Name and SRN of Members of the Team:

1. **Abhishek Aditya BS - PES1UG19CS019**
2. **T Vijay Prashant - PES1UG19CS536**
3. **Vishal R - PES1UG19CS571**
4. **Yashas KS - PES1UG19CS589**

Assignment 5

1. Read a video into Matlab

```
vidObj = VideoReader('FakeSmash.mp4');
numFrames = vidObj.NumFrames
allFrames = read(vidObj);
whos allFrames

%Store frames as images
i = 1;
while i <= numFrames
currentFrame = read(vidObj, i);
combinedString = strcat(int2str(i-1), '.jpg');
imwrite(currentFrame, combinedString);
i = i + 1;
end
```

2. How do we distinguish a ‘game in play’ from a zoomed in version of a player or the audience or an advertisement?

a. Use ‘key frame’ detection or ‘scene detection’

A keyframe is a frame where a change occurs in the timeline in media production, it is the location on a timeline which marks the beginning or end of the transition
A keyframe is a frame where a change occurs in the timeline

Algorithm:
- Extract frames one by one.
- Calculate difference between 2 consecutive frames.
- Set a threshold value
- Compare the difference with T if it is greater than T. Select it as key frame.
- Continue till the end.

Code:

```python
import cv2
import os
import numpy as np

def keyframeDetection(video_path, threshold):

    cap = cv2.VideoCapture(video_path)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    print("Total frames in the video: ", total_frames)
    fps = int(cap.get(5))
    if(fps==0):
        print("Not available")

    keyframePath = 'keyframes'
    if not os.path.exists(keyframePath):
        os.makedirs(keyframePath)
    ret, prev_frame = cap.read()

    i = 0
    count = 0
    cv2.imwrite('keyframes/'+str(i)+'.jpg',prev_frame)
    while ret:
        ret, curr_frame = cap.read()
        if ret:
            diff = cv2.absdiff(curr_frame, prev_frame)
            non_zero_count = np.count_nonzero(diff)
            if non_zero_count > threshold:
                print("Saving Frame number: {}".format(i), end='\r')
                cv2.imwrite('keyframes/'+str(i)+'.jpg',curr_frame)
                count += 1
            prev_frame = curr_frame
            i += 1
    print("Total Number of frames saved: {}".format(count))

    cap.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    video_path = 'FakeSmash.mp4'
    threshold = 100000
    keyframeDetection(video_path, threshold)

```

Output:

```
-zsh
(MI-Lab-ENV) IPCV Assignment 5 % python3 KeyFrame.py
Total frames in the video:  707
Total Number of frames saved: 281
(MI-Lab-ENV) IPCV Assignment 5 %
```

```
-zsh
(MI-Lab-ENV) keyframes % ls
0.jpg   225.jpg 260.jpg 355.jpg 402.jpg 531.jpg 606.jpg 634.jpg 661.jpg 73.jpg
1.jpg   226.jpg 261.jpg 376.jpg 403.jpg 532.jpg 607.jpg 635.jpg 662.jpg 75.jpg
10.jpg  227.jpg 262.jpg 377.jpg 404.jpg 533.jpg 608.jpg 636.jpg 663.jpg 78.jpg
100.jpg 228.jpg 263.jpg 378.jpg 405.jpg 534.jpg 609.jpg 637.jpg 664.jpg 79.jpg
101.jpg 229.jpg 264.jpg 379.jpg 406.jpg 535.jpg 610.jpg 638.jpg 665.jpg 8.jpg
102.jpg 23.jpg  265.jpg 38.jpg  407.jpg 536.jpg 611.jpg 639.jpg 667.jpg 80.jpg
103.jpg 230.jpg 266.jpg 380.jpg 41.jpg  54.jpg  612.jpg 64.jpg  669.jpg 83.jpg
104.jpg 231.jpg 267.jpg 381.jpg 414.jpg 540.jpg 613.jpg 640.jpg 671.jpg 84.jpg
11.jpg  232.jpg 268.jpg 382.jpg 42.jpg  542.jpg 614.jpg 641.jpg 673.jpg 85.jpg
12.jpg  233.jpg 269.jpg 383.jpg 43.jpg  545.jpg 615.jpg 642.jpg 675.jpg 88.jpg
13.jpg  234.jpg 270.jpg 384.jpg 45.jpg  547.jpg 616.jpg 643.jpg 677.jpg 90.jpg
136.jpg 235.jpg 271.jpg 385.jpg 461.jpg 549.jpg 617.jpg 644.jpg 679.jpg 91.jpg
148.jpg 236.jpg 272.jpg 386.jpg 464.jpg 55.jpg  618.jpg 645.jpg 68.jpg  92.jpg
15.jpg  237.jpg 273.jpg 387.jpg 479.jpg 551.jpg 619.jpg 646.jpg 681.jpg 93.jpg
159.jpg 238.jpg 28.jpg  388.jpg 48.jpg  553.jpg 620.jpg 647.jpg 683.jpg 94.jpg
160.jpg 239.jpg 283.jpg 389.jpg 480.jpg 58.jpg  621.jpg 648.jpg 685.jpg 95.jpg
165.jpg 24.jpg  284.jpg 390.jpg 482.jpg 59.jpg  622.jpg 649.jpg 687.jpg 96.jpg
18.jpg  240.jpg 29.jpg  391.jpg 485.jpg 596.jpg 623.jpg 65.jpg  689.jpg 97.jpg
184.jpg 241.jpg 298.jpg 392.jpg 486.jpg 597.jpg 624.jpg 650.jpg 69.jpg  98.jpg
19.jpg  245.jpg 3.jpg   393.jpg 49.jpg  598.jpg 625.jpg 651.jpg 691.jpg 99.jpg
2.jpg   246.jpg 30.jpg  394.jpg 491.jpg 599.jpg 626.jpg 652.jpg 693.jpg
20.jpg  25.jpg  313.jpg 395.jpg 492.jpg 6.jpg   627.jpg 653.jpg 695.jpg
217.jpg 251.jpg 319.jpg 396.jpg 5.jpg   60.jpg  628.jpg 654.jpg 697.jpg
218.jpg 252.jpg 320.jpg 397.jpg 50.jpg  600.jpg 629.jpg 655.jpg 699.jpg
```

Figure: Key Frames Extracted

b. Use colour dominance

Code

```
import numpy as np
from matplotlib import pyplot as plt
import cv2
from sklearn.cluster import KMeans
```

```
image = cv2.imread("./347.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
r, c = image.shape[:2]
out_r = 640
image = cv2.resize(image, (int(out_r*float(c)/r), out_r))
pixels = image.reshape((-1, 3))
plt.axis("off")
plt.imshow(image)
```

Input Image used for finding Colour Dominance


```
km = KMeans(n_clusters=8)
km.fit(pixels)

colors = np.asarray(km.cluster_centers_, dtype='uint8')
print(colors)
```

```
>>>  Output :
     [[ 97 130  83]
      [ 18  33  77]
      [214 222 221]
      [ 20  23  29]
      [ 46  74 136]
      [ 70  72  69]
      [160 166 175]
      [118 122 129]]
```

```
plt.figure(0)
for ix in range(colors.shape[0]):
   patch = np.ones((20, 20, 3)).astype(np.uint8)
   patch[:, :, :] = 255 - colors[ix]
   plt.subplot(1, colors.shape[0], ix+1)
   plt.axis('off')
   plt.imshow(patch)
plt.show()
```

Dominant Colours in image

```
dom = [[percentage[ix], colors[ix]] for ix in range(km.n_clusters)]
dominance = sorted(dom, key=lambda x:x[0], reverse=True)

plt.figure(0)
plt.axis('off')
patch = np.zeros((50, 500, 3)).astype(np.uint8)
start = 0
for cx in range(km.n_clusters):
   width = int(dominance[cx][0] * patch.shape[1])
   end = start + width
   patch[:, start:end, :] = 255 - dominance[cx][1]
   start = end
plt.imshow(patch)
plt.show()
```

Contribution of Individual Colours

```
for px in range(pixels.shape[0]):
   for ix in range(colors.shape[0]):
       pixels[px] = colors[km.labels_[px]]
img = pixels.reshape(out_r, -1, 3)
print(img.shape)
plt.imshow(img)
```

Image Reconstruction using the above colour palette