Computer Vision Coursework Semester 1, Part 1

Week 5

Jacob Lourie 210867165

**Questions**

*a) Write a function that performs pixel-by-pixel frame differencing using, as reference frame, the first frame of an image sequence. Apply a classification threshold and save the results.*

*b) Repeat the exercise using the previous frame as reference frame (use frame It-1 as reference frame for frame It, for each t). Comment the results in the report.*

*c) Write a function that generates a reference frame (background) for the sequence using for example frame differencing and a weighted temporal averaging algorithm.*

*d) Write a function that counts the number of moving objects in each frame of a sequence. Generate a bar plot that visualizes the number of objects for each frame of the whole sequence. Discuss in the report the implemented solution, including advantages and disadvantages.*

**a)** **Pixel-by-pixel Frame differencing with varying thresholding**

Process for pixel-by-pixel frame differencing

I followed the following process in order to implement the frame differencing:

1. For each frame, calculate the numerical difference between that frame and the reference frame (element-wise matrix subtraction)
2. Return the absolute value of that difference to capture intensity changes in both directions
3. Convert to grayscale
4. Gaussian blur the grayscale to remove unwanted effects of rain.

I saved the frames in RGB, gray and gray with gaussian blur. The frame differencing for frames 1 and 80 are shown for these three channels in figure 1:

*Figure 1. Frame differencing with frame 0 for RGB, gray and blurred-gray images for frames 1 and 80*

Graphical user interface, application, calendar

Description automatically generated

Thresholding

The frames were converted to the 0<x<1 range and I used 4 different thresholds for the images – [0.05, 0.1, 0.2, 0.4]. For each of the thresholds, I applied a transformation on the pixels:

Figure 2 shows the frame differencing results with thresholds 0.05 and 0.2, and each for the same frames (1 and 80) with respect to frame 0.

*Figure 2. comparison thresholds for pixel by pixel frame differencing*

Graphical user interface, application, calendar

Description automatically generatedGraphical user interface, calendar

Description automatically generated

**b) Perform frame differencing with different thresholds for consecutive frames**

For this exercise, I used roughly the same algorithm as for part a, but with 2 changes:

* I used frames 15 and 80; I didn’t use frame 1 because the result would have been exactly the same.
* I used thresholds of [0, 0.02, 0.06 and 0.15] as I expected the differences to be less pronounced given the close temporal proximity of the frames.

However, having no threshold produced results that were incoherent and of no use, due to the level of noise in between only two frames, as shown by figure 3:

*Figure 3. Impact of using a threshold of 0 on frame differencing for consecutive frames*

Diagram

Description automatically generated with low confidence

A higher threshold, around 0.15, produced results that were comparable with our semantic interpretation of the moving objects in the video. Figure 4 shows the results for this threshold for frames 15 and 80:

*Figure 4. consecutive frame differencing with a threshold of 0.15 for frames 15 and 80*

Graphical user interface, calendar

Description automatically generated

I also experimented with upsampling the image to increase the extent to which objects in the “distance” were recognised, but the results were almost identical (see notebook)

**c) Creating a background image using frame differencing**

In order to generate the background image, I followed the following process:

1. Calculate the frame difference between that frame and the previous frame.
2. Calculate the mean of frames, excluding pixels in frames where the frame differencing value was above a certain threshold.

Using 0.02 and 0.05 as a threshold return a good semantic definition of the background,

Figure 5 below shows the background generated using 0.05 as a threshold, and only using the *t-1* frame differencing:

*Figure 5. background generated using frame differencing and a 0.05 threshold*

A picture containing text, scene, way, road

Description automatically generated

Note: I later switch to using the median for generating the background, instead of the mean, explained in the *counting images* section.

**d) Counting number of moving objects in a video**

Two approaches to counting moving objects

In order to count objects, I experimented firstly with two methods to extract moving objects:

Using the background image and frame differencing with the background

Using the background found in part *c*, I used the frame differencing method to extract objects. I experimented with using a gaussian filter on the background and the frame in question to remove camera-induced or rain-induced noise. I also convert to grayscale and take the absolute values of the difference. Figure 6 shows a result from applying this to frame 70 of the video:

*Figure 6. application of frame differencing and a mean background for counting objects*

A close-up of some train tracks

Description automatically generated with low confidence

It appears that using the mean, even after thresholding out the objects when creating the background, there is still some noise in the background image, which results in spurious objects appearing as if they are moving (such as the side of the road).

If we use the median in the creation of the background, this effect diminishes significantly. Image below shows the same frame but using the median for the background creation:

*Figure 7. application of frame differencing and a median background for counting objects*

A picture containing text, dark, night, night sky

Description automatically generated

Using a temporal convolution to extract moving images

I secondly experimented with using a temporal convolution to extract moving images. I used a temporal filter with dimension [1, 1, 7] (*rows, columns, frames*). The following values were used, based on an edge detection filter

However, this resulted in a significant amount of noise. It’s possible that this would be reducible through a gaussian filter, but the ‘edges’ found that were spurious were significantly higher than the values for the actual objects, as shown in figure 8:

*Figure 8. Using a temporal convolution to extract moving images*

A picture containing text, track, train, outdoor

Description automatically generated

As a result, I progressed with using the background and frame differencing.

Removing shadows

I noted that when using frame differencing, object shadows were being returned. For correctly placing bounding boxes around objects, this is not ideal. In order to remove these shadows, I used the following pipeline

1. Calculate the frame referencing in RGB format.
2. Convert the background and the image to HSV format (manual formula included in notebook).
3. Calculate the frame referencing in HSV format.
4. If the change in brightness is negative, and the absolute value of this change is significantly higher (5x) than the change in saturation, identify that pixel as a likely shadow.
5. Convert image to grayscale
6. Apply gaussian filter to image to remove noise
7. Use a threshold that converts all values above threshold to 1 and those below threshold to 0.

The effect on this can be figure 9 below, with the first image showing frame referencing without shadow reduction, and the second image showing the post shadow reduction frame:

*Figure 9. Removing shadows in frame differencing from background*

A picture containing text

Description automatically generated A picture containing graphical user interface

Description automatically generated

However, the shadow reduction also led to the loss of some correct pixels.

Counting objects using bounding boxes

In order to count the objects, I proposed a method to find images and place them in a bounding box. I use various sized boxes, ranging from (8,8) to (64,64) in size, and place a priority on the larger boxes.

After scanning a frame for objects that fit inside a bounding box, I then use a threshold to discard boxes that overlap.

Finally, these boxes are counted and drawn onto the image. Figure 10 shows the results of a successful and a less successful response from the algorithm:

*Figure 10. Successful and unsuccessful bounding box application on frames*

A picture containing text, way, scene, road

Description automatically generated

A high angle view of cars on a road

Description automatically generated with medium confidence

Finally, I plotted a line chart of the number of objects found per frame, which is directionally correct and shown in figure 11:

*Figure 11. Number of objects per frame in video*

Chart, histogram

Description automatically generated