Title:	Intrusion Detection (Exam Report)
Name/Surname:	Burhan Mohayaddin
Course:	Image Processing and Computer Vision
Date:	05/25/2021

Table of Contents

Introduction

This report illustrates details of the development of a software system that, based on automatic video analysis, can detect objects (intruders) that do not belong to a static reference scene (background) and establish which of such objects are person, true and false objects.

Explanation of Algorithm

In this section the general explanation of the method that has been used, will be illustrated.

Analysis of the environment

The environment in the video has static camera. Moreover, the background is also static, except the change of lightning conditions which necessitates the handling of the background more carefully.

The algorithm

There are several algorithms to detect the motion in the scene like two frame difference, three frame difference and background subtraction. However, as in this case it is not required to detect the motion, but the change in the scene, we will utilize background subtraction method. Namely, the background frame which is static will be subtracted from the current frame in order to get the related changes in the scene. In order to realize this method below steps have to be followed:

- Get the model of the background image from some initial frames. The number of initial frames for modeling must be chosen as small as possible, in order not to miss any important detail in the video sequence.
- The absolute difference between the current frame and the reference background frame has to be calculated in order to create suitable mask for the detection of the changes in the scene.
- If the lightening conditions would be stable, then the above steps should be sufficient. However, in this case the lightning condition is not stable, which cause additional false positives to be detected due to the change in the pixel values of the frames in the video. This necessitates additional step which is updating background image at each frame. In this project, alpha-blending technique has been utilized to update the background image.

Modeling Background

In the method of background subtraction, the model of background has to be available. This can be taken from the video when there is nothing but background. However, this is not always possible. Sometimes it is required to model the background by looking at first few frames. In this case this technique has been applied. First 100 frames has been taken and their median has been found. Based on this, the model of the background has been obtained. The more the frame number, the better the model is. However, it should be taken into account that choosing the frame numbers as high can cause missing of some important details in the video, like missing an important change that has happened in the video.

```
def generate_background(self):
   self.bg_interpolated = np.stack(self.background_buffer,
                                   axis=0)
   self.bg_interpolated = np.median(self.bg_interpolated, axis=0)
```

Background Subtraction

As mentioned before, in this project background subtraction method has been utilized to detect the changes in the scene.

```
def l1_distance(self, img1, img2):
     img1 = cv2.GaussianBlur(img1, (5, 5), 0)
     img2 = cv2.GaussianBlur(img2, (5, 5), 0)

     diff = np.abs(img1 - img2)
     if img1.shape[-1] == 3 and len(img1.shape) == 3:
     diff = np.sum(diff, axis=-1)
     return diff
```

The function accepts two images, namely the current frame and the background image. Then, both images have been blurred using Gaussian filter in order to get rid of some unnecessary small details in the images. After blurring, absolute difference between two images has been calculated pixel by pixel and stored in the ‘diff’ numpy array. In the next step it is checked whether the image is RGB (3 channel) or gray scale (single channel). In the former case, the three-dimensional matrix has been converted to two dimensional by finding the sum between individual two-dimensional matrices.

```
mask = ( self.l1_distance( frame, self.bg_interpolated) >
self.threshold )
```

After the difference between two images has been found, predefined threshold has been applied in order to differentiate between the desired changes and noisy ones. Choosing the threshold value too low will cause to detect very small changes that are either noise, or very small changes that are not interesting for this application. However, choosing the threshold value too high will cause to discard almost any changes including useful ones. In this case the threshold has been tuned for the best result ad it has been chosen as 50. After threshold has been applied, the mask has been created which will enable to us to get only area of interest. Difference between the original video and masked video can be seen from the figure1.

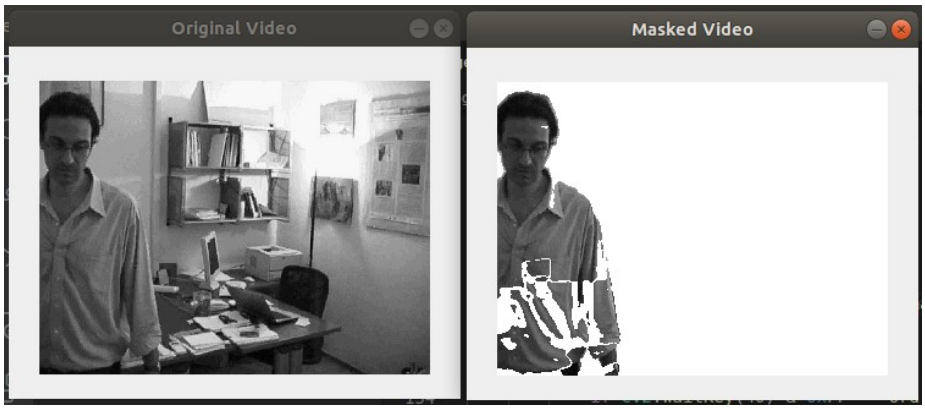

Figure 1: Original and masked video

Overall procedure can be summarized in equation1:

$$C_t(i,j)=\begin{cases}255 & D_t(i,j)=\sqrt[p]{|\Delta R_t(i,j)|^p+|\Delta G_t(i,j)|^p+|\Delta B_t(i,j)|^p}>T \\ 0 & otherwise\end{cases}$$

Equation 1: Background subtraction

Background Update

In terms of the background update procedure, selective “alpha-blending” technique has been applied. Overall procedure can be formulated as below:

$$\vec{B}_{t+1}(i,j)=\begin{cases}\alpha*\vec{F}_t(i,j)+(1-\alpha)*\vec{B}_t(i,j) & if \quad C_t(i,j)=0 \\ \vec{B}_t(i,j) & otherwise\end{cases}$$

Equation 2: Selective "alpha-blending" method

In the above equation, the range of α is [0,1] and it is called “adaptation rate”. The purpose of this parameter is to adjust the speed of adaptation of the background model to changes occurring in the monitored scene. If it is chosen as 1, then it will be the same as two-frame difference algorithm and will not be suitable for changing background scenes. As α goes to 1, the adaptation becomes slower and slower, so amount of the noise increases. In this case, this parameter is tuned as 0.15 empirically. Additionally, the matrix C in the equation represents mask which is output of background subtraction algorithm and additionally morphological operations have been applied for improving the quality of mask. This will be discussed in the next section.

```
self.bg_interpolated[new_mask == 0] = (
     self.alpha * frame[new_mask == 0]
     + (1 - self.alpha) * self.bg_interpolated[new_mask == 0] )
```

Mask Improvement and Object Classification

This section will describe the methods that have been utilized to improve the mask which has been obtained by background subtraction, and classifying objects as person, false or true objects.

Improving mask

Mask that is the output of background subtraction step is very noisy even if application of Gaussian filter and updating the background image. So, it contains small-sized blobs that have to be removed in order to get better results during the application of morphological operations. In area-opening step, the contours have been found in the mask and their areas have been calculated. Contours which have are that is smaller than predetermined threshold, have been drawn in a temporary mask. So, temporary mask contains only the blobs that have small areas.

```
contours = cv2.findContours(np.uint8(mask), cv2.RETR_LIST,
               cv2.CHAIN_APPROX_SIMPLE)[0]
               temp_mask = np.ones(mask.shape[:2],dtype=np.uint8)

for cnt in contours:
   if cv2.contourArea(cnt) < 350:
   cv2.drawContours(temp_mask, [cnt], 0, 0, -1)
       new_mask = cv2.bitwise_and(np.uint8(mask), np.uint8(mask),
                      mask=temp_mask)
```

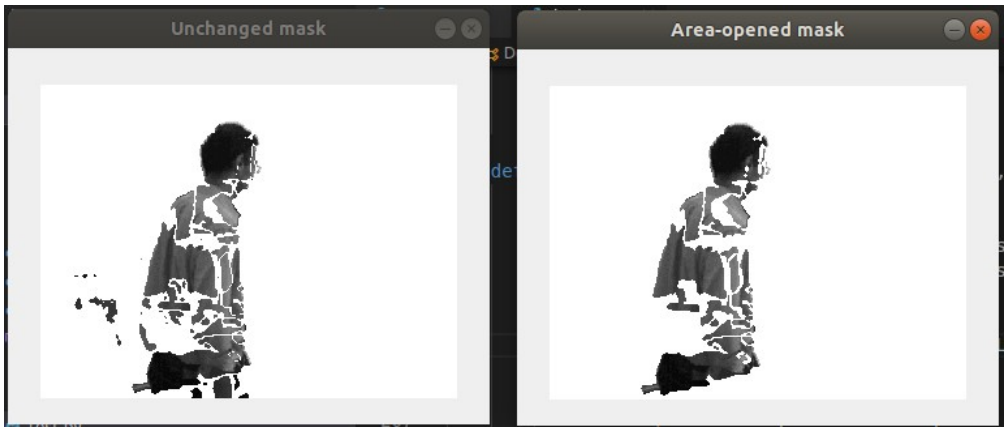

Figure 2: Area-opened mask

Then, “bitwise-and” operation has been utilized between the old mask and the temporary mask in order to get new mask which is free from small blobs. In the next step, morphological operations of closing and opening have been applied. By closing, small holes have been closed inside the foreground objects and by opening, the noise in the mask has been removed, because of erosion step. After opening step, additionally dilation has been applied in order to remove any holes in the foreground if there exists. This step will solve the problem of getting the object of interest into small several parts, but will additionally cause some uninteresting blobs to appear. However, they will be eliminated in further filtering process by area of blobs.

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
new_mask = cv2.morphologyEx(np.uint8(new_mask), cv2.MORPH_CLOSE,
               kernel, iterations=1)
new_mask = cv2.morphologyEx(np.uint8(mask), cv2.MORPH_OPEN,
               kernel, iterations=1)
new_mask = cv2.dilate(np.uint8(new_mask), (15, 15), iterations=2)
```

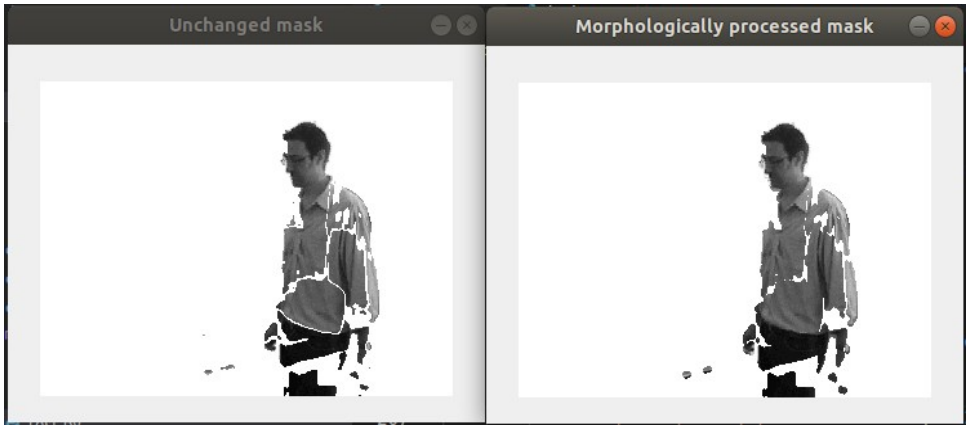

Figure 3: Mask after application of morphological operations

Finally, one more time the contours in the mask have been calculated and filtered again by area. However, additionally in this step, the contour details have also been calculated and stored which will be utilized during classification of objects. The details of contours include:

- Area of contour
- Perimeter of contour
- Compactness of contour
- Moments and barycenter of contour

```
num_detected_objects = 0
contours = cv2.findContours(
               np.uint8(new_mask), cv2.RETR_LIST,
               cv2.CHAIN_APPROX_SIMPLE)[0]
contours_data = []
for cnt in contours:
   area = cv2.contourArea(cnt)
   if area > 500:
      perimeter = cv2.arcLength(cnt, True)
      compactness = perimeter * perimeter / area
      M = cv2.moments(cnt)
      cx = int(M['m10']/M['m00'])
      cy = int(M['m01']/M['m00'])
      data = (area, cnt, perimeter, compactness, cx, cy)
      contours_data.append(data)
```

Classifying objects

The essential part of the task is to classify objects as person or object. This can be done easily by analyzing the areas of the contours. The contour that has the biggest area has to belong to human. So, the threshold area for human has been taken as 3000. If the contour area is less than this

```
contours_data = sorted(contours_data, key=lambda x: x[0],
                       reverse=True)
flags = np.ones(len(contours_data))
obj_type = ""
obj_types = []
for i in range(len(contours_data)):
   if flags[i] != 0:
      num_detected_objects += 1
      color = None
      obj_type = None
      cmpts = contours_data[i][3]
      if 15.0 <= cmpts <= 17.0:
         color = self.cnt_colors[1]
         obj_type = "false object"
      elif len(contours_data) == 1 or contours_data[i][0] > 3000:
         color = self.cnt_colors[0]
         obj_type = "person"
      else:
         color = self.cnt_colors[2]
         obj_type = "true object"

      obj_types.append(obj_type)
      cv2.drawContours(frame, [contours_data[i][1]], -1,
                       color, 2)
      bRi = cv2.boundingRect(contours_data[i][1])
      for j in range(i+1, len(contours_data)):
         bRj = cv2.boundingRect(contours_data[j][1])
         if self.does_intersect(bRi, bRj):
            flags[j] = 0
```

threshold it will be classified as object. The object can be further specified as whether it is true object or false which has appeared because of change of frame when object is relocated by human. The false object has rectangular shape contour, so it is enough just calculate compactness of the object and check whether it is between 15 and 17. Because for rectangle, compactness has to be 16, but due to noise the contour is not a perfect rectangle. While the results of this procedure are good, they can be improved further. The issue is that, the contour of objects of interest can contain unnecessary additional contours which can pass filtering process. Moreover, sometimes the head of human can be detected as a separate object from his body, especially when he bends. So, in order to solve this issue, a custom method has been utilized. Namely, as a first step the filtered contours have been sorted from biggest to smallest which will decrease the analyzing steps. Then for each contour it;s bounding rectangle has been calculated. Then, the intersection of this rectangle with the bounding rectangles of other contours has been checked. During checking the intersections, the

dimensions of bounding boxes have been increased by parameter of alpha in order to get better results. If there is intersection, then this means that these contours belong to the same object. After checking intersection, two approaches can be taken, namely either the intersected contours can be merged or the smaller contour can be removed. The former approach has greater complexity, so the latter approach has been taken in this case which also gives satisfactory results. Checking the intersection between the contour's bounding rectangles don't take too much time, because there have been left few contours after previous filtering processes. Below code snippet performs the step of analyzing contours:

Conclusion

In conclusion, this report demonstrated the detection of changes in the scene by background subtraction method. Additionally, details of "alpha-blending" technique for background updating and classification of objects have been discussed.