Final Project

Video Processing/ 0512-4263

By Avi Epstein and

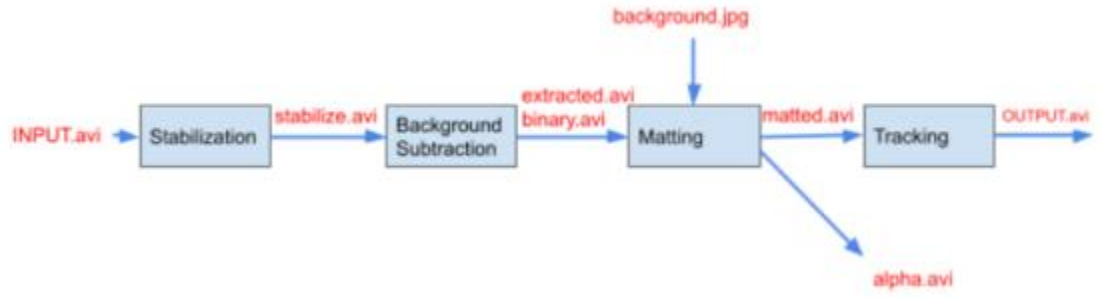
Tomer Shimshi

Tel Aviv University

June 2022

Abstract

In this project we will be implementing a chain of video processing algorithms the input to the system will be a shaky video of a man walking across a hallway. We assume inputs of unstable static backgrounds and the only object that is moving is the person. Our objective is to implement a video stabilizer, background subtractor, matting and tracking algorithms in a succession order. As can be seen in the follwing figure:



Contents

תוכן עניינים

Solutions:

Video stablizer :

In our algorithm there are 3 main steps:
1) motion estimation
2) motion smoothing
3) image composition
The transformation parameters between two consecutive frames are derived in the first stage. The second stage filters out unwanted motion and in the last stage the stabilized video is reconstructed.

In the algorithm we will iterate over all the frames, and find the motion between the current frame and the previous frame. This is done as follwos:
On every consecutive frames we apply orb detector retreving key points and their discribters . To understand how the orb detector works we need to understand how the fast key point descriptor works:

- **FAST**: (Features from Accelerated Segment Test)
  1. Select a pixel p in the image which is to be identified as an interest point or not. Let its intensity be Ip.
  2. Select appropriate threshold value t.
  3. Consider a circle of 16 pixels around the pixel under test.
  4. Now the pixel p is a corner if there exists a set of n contiguous pixels in the circle (of 16 pixels) which are all brighter than Ip+t, or all darker than Ip−t.. n was chosen to be 12.
  5. A high-speed test was proposed to exclude a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If p is a corner, then at least three of these must all be brighter than Ip+t or darker than Ip−t. If neither of these is the case, then p cannot be a corner. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle. This detector in itself exhibits high performance, but there are several weaknesses:

     a. It does not reject as many candidates for n < 12.
     b. The choice of pixels is not optimal because its efficiency depends on ordering of the questions and distribution of corner appearances.
     c. Results of high-speed tests are thrown away.
     d. Multiple features are detected adjacent to one another.

  These weaknesses can be addressed using machine learning corner detector , or nonmaximal suppression

Fast is several times faster than other existing corner detectors. But it is not robust to high levels of noise and It is dependent on a threshold.

- **ORB:** ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it uses FAST to find keypoints, then applys Harris corner measure to find top N points among them. It also uses pyramids to produce multiscale-features. The problem is that FAST doesn't compute the orientation so it computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. To improve the rotation invariance, moments are computed with x and y which should be in a circular region of radius r, where r is the size of the patch.

The descriptor:

- **BRIEF:** provides a shortcut to find the binary strings directly without finding descriptors. It takes smoothened image patch and selects a set of n_d(x,y) location pairs in an unique way. Then some pixel intensity comparisons are done on these location pairs. For eg, let first location pairs be p and q. If I(p)<I(q), then its result is 1, else it is 0. This is applied for all the n_d location pairs to get a n_d-dimensional bitstring. This n_d can be 128, 256 or 512.
- **ORB:** ORB uses BRIEF descriptors but because BRIEF performs poorly with rotation so orb steers brief according to the orientation of the keypoints. ORB discretize the angle to increments of 2π/30 (12 degrees), and construct a lookup table of precomputed BRIEF patterns. As long as the keypoint orientation θ is consistent across views, the correct set of points S_θ will be used to compute its descriptor. BRIEF has an important property that each bit feature has a large variance and a mean near 0.5. But once it is oriented along keypoint direction, it loses this property and become more distributed. High variance makes a feature more discriminative, since it responds differentially to inputs. Another desirable property is to have the tests uncorrelated, since then each test will contribute to the result. To resolve all these, ORB runs a greedy search among all possible binary tests to find the ones that have both high variance and means close to 0.5, as well as being uncorrelated. The result is called rBRIEF. For descriptor matching, multi-probe LSH which improves on the traditional LSH, is used. ORB is a good choice in low-power devices for panorama stitching etc.

The extracted key points are then matched between the frames using bfm matcher:

- **BF:** brute force matcher takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation(in our case Hamming norm) . And the closest one is returned.

The matches are then sorted and inputted to the cv2 estimate affine partial 2D which Computes an optimal limited affine transformation with 4 degrees of freedom between two 2D point sets

Partial 2D affine transform

The partial affine transform mentioned early has a reduced degree of freedom of 4 by excluding shearing leaving only rotation, uniform scaling and translation. How do we do this? We start with the matrices for the transforms we are interested in.

$$R(rotation) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$S(scaling) = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

$$t(translation) = \begin{bmatrix} tx \\ ty \end{bmatrix}$$

Our partial affine transform is:

$$A = [\, RS|t \,]$$

Expanding gives:

$$A = \begin{bmatrix} \cos(\theta)s & -\sin(\theta)s & tx \\ \sin(\theta)s & \cos(\theta)s & ty \end{bmatrix}$$

We can rewrite this matrix by defining:

$$a = \cos(\theta)s$$

$$b = \sin(\theta)s$$

$$c = tx$$

$$d = ty$$

$$A = \begin{bmatrix} a & -b & c \\ b & a & d \end{bmatrix}$$

Solving for [a, b, c, d]:

$$AX = Y$$

$$\begin{bmatrix} a & -b & c \\ b & a & d \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$ax - by + c = x'$$
$$bx + ay + d = y'$$

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{bmatrix}$$

Notice for the partial affine transform we only need 2 pair of points instead of 3.

This function is set with the ransac outlier removal.

outlier detection method. In the following figure we can see how to define ransac algorithm iterations:

- The number of iterations N that is necessary to guarantee that a correct solution is found can be computed by

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}$$

- s is the number of points from which the model can be instantiated
- ε is the percentage of outliers in the data
- p is the requested probability of success

Figure (): RANSAC number of iterations

We then extract the translation and rotation angles and store each transform.

We then smooth out each element of the transforms by using a moving average and reconstruct the transformation matrix according to the new values. We the use the wrap affine function to apply this transformation of the unstable images to retrieve a stabilized video.

Background subtraction:

In this section we start by using opencv BackgroundSubtractorKNN function.

KNN background subtractor:

Knn background subtractor is a Non-parametric Background Subtraction, The Normal distribution model of pixel intensities with no parameters is used in this technique. The pixel intensity probability is measured independently for each frame. This model accurately generates the background model non parametrically. This model is very sensitive to the changes and is more robust so it adapts to the background very easily. The idea is to capture the most recent information of the current frame and update it continuously to make it adaptable to the changes KNN algorithm stores all available cases and then it classifies the new samples based on the similarity measure. It is a widely used non parametric technique. In this algorithm, all the training samples are needed to be stored before the classification process. This could be a drawback if a very large data set is used. The training samples are described by n- dimensional numeric attributes. Each sample represents a point in an n- dimensional space. When any unknown sample needs to be classified, k nearest neighbour classifier searches the pattern space for the k training samples that are closest to the unknown sample. Euclidean distance is used to measure the closeness. Euclidean distance between two points, A = (a1, a2,....an) and B = (b1, b2,....bn) is given by the following equation:

$$d(A, B) = \sqrt{\sum_{n=1}^{\infty}(a_i - b_i)^2}$$

After learning the rough segmentation using the above algorithm and thresholding we post process the segmanted image using the following morphological operations

Morphological transformations:

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called **structuring element** or **kernel** which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play.

- **Erosion:** The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). So what it does? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). So what happens is that, all the pixels near boundary will be discarded depending upon the size of kernel. So, the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises (as we have seen in colorspace chapter), detach two connected objects etc.
- **Dilation:** It is just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of

foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.

- **Opening:** Opening is just another name of erosion followed by dilation. It is useful in removing noise, as we explained above.
- **Closing:** Closing is reverse of Opening, Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

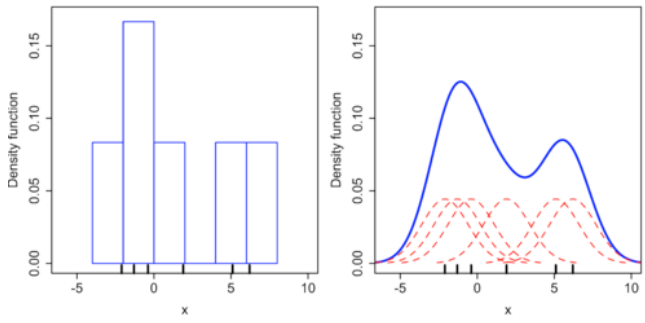
In the next step we find the contours of the mask using the cv2 find countours methode.

Contours: can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. there are three arguments in **cv2.findContours()** function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy. Contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

After finding the contours we draw the contours using the cv2.drawContours() methode
draw contours can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc

in **the next** steps we break up the masks into 3 parts: upper middle and lower segment's. And perform KDE filter on each element

KDE- Kernal density estimation:
Kernel density estimation is a way to estimate the probability density function (PDF) of a random variable in a non-parametric way. gaussian_kde works for both uni-variate and multi-variate data. It includes automatic bandwidth determination. The estimation works best for a unimodal distribution; bimodal or multi-modal distributions tend to be oversmoothed. KDE is closly related to histograms but enhabits the propertys of smothnes and continuety as can be seen in the following image:



Comparison of the histogram (left) and kernel density estimate (right) constructed using the same data. The six individual kernels are the red dashed curves, the kernel density estimate the blue curves. The data points are the rug plot on the horizontal axis.

This is done by taking a gaussian for every sample point and summing over all samples we retrieve the pdf:

$$f(c|F)\ \alpha \sum_{x \in Sample} (K\big(c, c(x)\big)\ \ where: K(c, c')\ \ \alpha\ exp\ (-\frac{\|c - c'\|_2^2}{2\sigma^2})$$

And the same is done for the background B.
By assuming P(F) = P(B) = 0.5 using Bayes theorem we get:

$$P(F|c) = \frac{f(c|F)}{f(c|F) + f(c|B)}$$

The same is done for the background.

Hence after breaking up the image into the three parts trying to capture the head the body and shoes separately. We select random indices from the previous rough background subtractor and input the to the KDE as sampled points. After bulding the KDE we go over each pixal and retrieve a full probability map of every pixel to be selected as a foreground/ background.
We then threshold the probability mask and combine the 3 parts and again post by using the morphological close and open, find and draw transformations. We the assume the biggest counter we receive as the segmented walking person.

Matting:

In this section we start by retrieving 3 different video frames the luma channel the blue channel and the previous mask acquired in the last section.

1) Creating a tri map:
   Our objective is to retrieve a tri map in order to do so we compute a distance map using: GeodisTK.geodesic2d_raster_scan().
   For that we looked at a small window around the max ,min pixels location in the binary mask that was computed in the previous part. We then used a eroded mask of the binary mask for the foregrounds , and the backgrounds is computed by an inverse mask of the dilated binary mask. the function does as follows:
   Computes the x and y derivative of the image and based on the 4 ball connections of each pixel (the gradient of the path between the pixels) computes the minimum distance to the mask that was inserted for each pixel this is done efficiently using raster scan optimization (allowing optimized access to cash). This is done for both background and forground
   We then transform this distance map into a probability map by dividing the FG distance map by their summation FG+BG distance map, giving us a foreground and background probability map (BG probability map is 1-FG probability map).
   We then decide on the alpha band as the area were the difference between the FG and BG distance is smaller then a small constant epsilon. And setting 0/1 to all areas not in this band that match FG and BG. This is done in a similar way to video colorization as we calc the geodesic distance an the luminance image (we could have run it also on the probability map acquired in the segmentation but our way gave better results)

2) Refinement: we now again use KDE for the outer BG and inear FG (ie th edges of our tri map) that are close to our segmented FG in order to calc a prior for the perimeter alpha we calculate the geodesic distance on the lumma(again could be done on the probability map) and get

$$\alpha(x) = \frac{w_f(x)}{w_f(x) + w_b(x)} \quad where: w_f(x) = D_F(x)^{-r} * P_F(x)$$

   This map completes the opacity map were alpha is 1 we paste the original image. Were New image is

$$J(x) = \alpha(x)c(x_F^*) + \big(1 - \alpha(x)\big)b(x)$$

3) After retrieving the opacity map we paste the image onto the new video

Bounding box:

We now take the segmented waking person and fit a bounding box it

Tracking:

For the best implementation in this section we found that by using the alpha video we obtained in the previous section if look at all the indices where the alpha is equal to one and by taking the min/ max of them we get the top, bottom, left, right pixels indices we get a tight bounding box around the person white a low running time. On top of that in order to deal with the vanishing head problem we set the top index to remain the lowest (which is to since the upper part of the frame starts at the top left corner) and by that we still see the person inside the bounding box even when he losses he’s head

Summary and results:

In this project we implemented video stabilization background subtraction matting we can see that we got good results on the video stabilization however at the end of the video the shaknes of the camera was to strong and we get a slight jump this jump possibly if we selected a more adaptable smoothing parameter we would get better results. The result of the background subtraction also came out pretty well however we could possibly improve it by incorporating technics from the matting stage (using the distance maps on probability maps and luminance) in addition because of the jump in stabilization we can see that the head gets cut off in the time of the jump .In the matting stage we get decent result however it could be improved if we were more able to remove the signs that sometimes appeared next to the head. More over a lot of the images were used on the blue scale ,we tried to do the same actions on all the channels but although it is more general, we got poor results using them. In conclusion the we received good results and future works could improve on this as well.