Moving Object Detection in Realtime

Sameem Taray

June 2017

Contents

1 Introduction

Detection and segmentation of moving objects is a fundamental problem in computer vision. As such there are extensive works that tackle the problem of segmentation of moving objects such as [JYA94]. These works, however, assume that the object of interest is already in motion when the video sequence begins and thus may not be applicable to applications such as surveillance or robotics. In these applications, an object can move or come into the field of the camera at an unknown instant of time. [LS16] proposes a framework for detection and segmentation that tries to tackle the this problem, where-in they design a system that takes in frames sequentially and makes a decision i.e., to wait for next frame or declare a detection. In this paper, we build on their methods while proposing changes that make it possible to detect and segment changes while running at interactive frame-rates. This makes our method more suitable for online and real-time applications such as video surveillance.

Our contributions are to propose an algorithm with simplifications to the methods of [LS16] for the problem of detection and segmentation to achieve interactive frame-rates. We also provide a public implementation of the algorithm available at [Tar].

2 Models for object detection

Let $\Omega \subset R^2$ be the domain of images and let $I_t : \Omega \rightarrow R^2, t \geq 1$ where t denotes the frame number be the sequence of images, $k = 3$ denotes the number of color channels. We denote n regions $R_t^i \subset \Omega$ for $i = 0, 1, 2, ...n-1$ and R_t^0 will correspond to the background, which form a segmentation of I_t. We denote O_t^i as the occlusion of the region R_t^i induced by change of viewpoint or camera or a self-occlusion between time t and $t+1$. See figure of schematic.

Region and Displacement Model The displacement between adjacent frames for region i is $v_t^i : R_t^i \, / O_t^i \rightarrow \mathbb{R}^2$. Although such displacements are not defined in the occluded parts of the domain, they will be smoothly extended into the occlusion and the entire domain Ω. We use the source code provided by [Ilg+16] to calculate v_t^i. It is a convolutional neural networks based methods which is the state of art in-terms of speed vs. accuracy.

Determining Residuals The residual from frame I_t to I_{t+1} is given by

$$Res = \rho(I_t(w_{t,t+1}(x)) - I_{t+1}(x)) \tag{1}$$

Here, $w_{t,t+1}(x)$ is the warp between consecutive frames and is given by

$$w_{t,t+1}(x) = x + v_t(x)$$

where $v_t(x)$ is the displacement at x which we get from the optical flow. Also $\rho(y)$ which is a truncated quadratic, which is a robust norm that eliminates the explicit estimation of the occlusion [SRB10]. It is given by

$$\rho(y) = min\{|y|, \beta\}(\beta > 0) \tag{2}$$

Probable Change time (t_c^*) calculation Determining the probable change time t_c^* from a window starting at $t-d$ (d denotes the size of the window) to t is based on two tests.

The first test is based on the spatial average of the optical flows (or residuals). For each frame in the window, we calculate the statistic defined by

$$F_{t_c,t} = (t - t_c + 1)(\hat{\mu}_{t-d:t_c-1} - \hat{\mu}_{t_c:t}) \tag{3}$$

Here, d denotes the size of the window and $\hat{\mu}_{t_1:t_2}$ is the average of the spatial average of flows from t_1 to t_2. Upon the acquisition of a new frame at time t, $F_{t_c,t}$ is calculated. The maximizer of $F_{t_c,t}$ over the entire window, gives us the probable change (t_c^*).

Next, we perform the standard deviation test. Now that we have (t_c^*), we calculate the standard deviation of the optical flow at (t_c^*). A large standard deviation indicates the presence of moving object. Typically if the standard deviation $\sigma_{t_c}(flow)$ exceeds a value of 5.0, the test is passed. Using optical flow for the standard deviation test instead of the residual is more robust and generalizes well for different settings.

Producing the segmentation mask To generate the segmentation mask, we convert the optical flow at time t_c^* into RGB and use Otsu's method to get the mask. It is a fast method based on adaptive thresholding. More accurate results at the expense of time can be obtained by using a simple K-Means clustering algorithm on combined motion and image intensity data.

TODO: Formulate KMeans for motion segmentation here

Likely-hood test The likely-hood test is based on calculating two Hypothesis. For each frame $flow_{t_i}$ starting from t_c^* to t, we calculate the the standard deviation of the background and foreground in the Residual and sum them up This gives us h_1. We calculate the standard deviation of the Residual as a whole and this gives us h_0. Next we propagate the segmentation using $flow_{t_i}$ and repeat the calculation over all the frames. The sum of h_0 and h_1 over all the frames gives $H0$ and $H1$ respectively. and $H1 - H0$ gives us the likelyhood ratio. If this ratio is greater than a threshold, we say that the test has passed and declare a detection. It is explained in Algorithm 2.

2.1 Algorithm 1

Algorithm 1: The main subroutine

```
   Result: Segmentation and Change Detection
1  Get frame_1 from the sequence.;
2  while TRUE do
3      Get frame_2 from the sequence.;
4      Determine optical flow (flow) from frame_1 to frame_2;
5      Determine residual (Res) from frame_1 to frame_2 using flow;
6      Determine the probable change time $t_c^*$;
7      if $t_c^*$ is updated then
8          Do the standard deviation test.;
9          if standard deviation test is passed then
10             Get segmentation from color coded flow at $t_c^*$;
11             Perform the likely-hood test using Algorithm 2;
12             if likelihood test is is passed then
13                 Declare a detection.;
14                 Print the segmentation to the screen.;
15             end
16         end
17     end
18     Set frame_1 to frame_2 and continue;
19 end
```

2.2 Algorithm 2

Algorithm 2: Algorithm for Likelihood Ratio test.

```
Input  : probable change time $t_c^*$ and $segmentation\_mask$
Output: $true$ if test is passed, $false$ otherwise
1  function LikelihoodTest ($t_c^*, segmentation\_mask$);
2  $H0, H1 = 0, 0$;
3  while  $t_i := t_c^*$ to $t$ do
4      Calculate the standard deviation of foreground in $Res_{t_i}$ using
         $segmentation\_mask$ and set it to $std_1$;
5      Calculate the standard deviation of background in $Res_{t_i}$ using
         $segmentation\_mask$ and set it to $std_2$;
6      $h_1 = std_1 + std_2$;
7      $H1 = H1 + h_1$;
8      Calculate the standard deviation of $Res_{t_i}$ as a whole and set $h_0$ to
         this standard deviation;
9      $H0 = H0 + h_0$;
10     Propagate $segmentation\_mask$ using $flow_{t_i}$;
11 end
12 if $H_1 - H_0 > LIKELYHOOD\_THRESHOLD$ then
13     return $true$
14 else
15     return $false$
16 end
```

3 Experiments

Mathematical Modelling: The problem at hand fits into the general framework developed by Veeravalli et al [VB12]. They provides generalized notations which we can adopt and modify if needed to frame our problem mathematically and then come up with solutions based on statistical properties to solve the problem.

Testing and Experimentation: [Jod] provides an extensive annotated baseline data-set for testing. They provide a collection of annotated video sequences set in different settings as well as comparisons of the state of the art method for change detection. The metrics for evaluating the effectiveness of quickest moving algorithms like Average False Positive Rate (FPR), Average False Negative Rate (FNR), Average F-measure, Average Precision defined by Veeravalli et al in [VB12].

References

[JYA94] E.H. Adelson J.Y.A. Wang. *Representing moving images with layers.* 1994. eprint: `ieee:10.1109/83.334981`.

[SRB10] D. Sun, S. Roth, and M. J. Black. "Secrets of optical flow estimation and their principles". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).* IEEE, June 2010, pp. 2432–2439.

[VB12] Venugopal V. Veeravalli and Taposh Banerjee. *Quickest Change Detection.* 2012. eprint: `arXiv:1210.5552`.

[Ilg+16] Eddy Ilg et al. *FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks.* 2016. eprint: `arXiv:1612.01925`.

[LS16] Dong Lao and Ganesh Sundaramoorthi. *Quickest Moving Object Detection.* 2016. eprint: `arXiv:1605.07369`.

[Jod] Pierre-Marc Jodoin. *Quickest Change Detection Dataset.* URL: `http://changedetection.net/`.

[Tar] Samim Taray. *Minimum Delay Moving Object Detection in Realtime.* URL: `https://github.com/zsameem/realtime-mdmod`.