Moving Object Detection

Gyan Prakash Singh, *Student IIITG*

***Abstract*- Moving object detection has been point of interest from a long time in computer vision and image processing[6]. The task has been completed in many ways , that means many methods has been proposed to detect moving object. Some important techniques among them are background subtraction, frame differencing, temporal differencing etc. In this paper, markov random field model has been to detect moving object. It can detect even multiple object inside a window. A low level markov random field model namely optical flow has been used to detect moving object from background. The detection is based on concepts derived from lucas kanade method. Method used for object detection comes under domain of video tracking. Markov random field (MRF) models are robust to noise, that is main advantage of using MRF models over other methods as well they don't need much dependency of frames in context of implementation. Camera as well as videos has been used for moving object detection. Tracking percentage has been used for as performance index.**

***Index Terms*- Segmentation , Tracking , Optical Flow , Markov Random Field[1][6] , Temporal , Shi Tomashi Corner detection , Lucas kanade .**

I. Introduction

Optical flow is the velocity field in the image plane caused by the motion of he observer and objects in the scene. It contains important information about cues for region and boundary segmentation, shape recovery etc . It is Low level Markov random field model. Concepts from lucas canade optical flow algorithm is used to implement as it fulfills all requirements that a Markov model possess for movingobject detection , Since optical flow belongs to low level markov field models[2].

Suppose a pixel I(x,y,t). here i have added time field to represent snapshots of video as images. It has been called a frame. It moves by distance (dx,dy) in next frame taken after dt time. So since those pixels are the same and intensity does not change, we get,

$$I(x,y,t)=I(x+dx,y+dy,t+dt) \qquad (1)$$

Then take taylor series approximation of right-hand side, removeing common term and divide by dt to get the following equation[5]:

$$fxu+fyv+ft=0 \qquad (2)$$

where::

$$fx=d(I)/dx, \qquad (3)$$

$$fy=d(I)/dy, \qquad (4)$$

$$u=dx/dt, \qquad (5)$$

$$v=dy/dt; \qquad (6)$$

Above equation is called Optical Flow equation[7]. In it, we can find fx and fy, they are image gradients. Similarly ft is the gradient along time. But (u,v) is unknown. Flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighbourhood, by the least squares criterion. So ,i have used least square criterion for this.

II. Implementation Scheme

Moving object detection has been done in following Steps:

A. Parameters for lucas kanade optical flow

criteria has two interesting parameters here - the max number (10 above) of iterations and epsilon (0.03 above). More iterations means a more exhaustive search, and a smaller epsilon finishes earlier. These are primarily useful in exchanging speed vs accuracy, but mainly stay the same.

When maxLevel is 0, it is the same algorithm without using pyramids (ie, calcOpticalFlowLK). Pyramids allow finding optical flow at various resolutions of the image. Pyramids

B. Drawing Stuff on the Image

In this step, writing instructions for tracking object in video has been implemented i.e what button to use for initialization as well rectangular window initialization for keeping object to put in it. Inializing its location on tracking window which includes its alignment, thickness etc.

(i)Finding Region of interest(ROI) and extracting Corners as well as converting to complete image coordinates(new_corners)

Region of interest is found using left-top and right-bottom vertex of rectangle used already to draw window. Then, i extract corner point using goodFeaturesToTrack function from opencv library. Thus , I got a estimate of image coordinate of object in that image frame of video.Here , cv2.goodFeaturesToTrack() finds N strongest corners in the image by Shi Tomashi method. As usual, image should be a grayscale image.

Then number of corners to be found is specified. Then we specify the quality level, which is a value between 0-1, which denotes the minimum quality of corner below which everyone is rejected. Then we provide the minimum euclidean distance between corners detected.With all these informations, the function finds corners in the image. All corners below quality level are rejected. Then it sorts the remaining corners based on quality in the descending order. Then function takes first strongest corner, throws away all the nearby corners in the range of minimum distance and returns N strongest corners.

(ii)Using B(i) , Drawing the corners in the original image and old corners and oldFrame is updated

C. Actual Tracking has been implemented in following steps:

(i)Reading new frame and Converting to gray

It is required since corner detection requires gray scale images.

(ii)Finding the new tracked points

we give some points to track and receive optical flow vectors of these points . For small motion it is fine but for large motion it may fail . So , I have used pyramidal concept according to which if we go up in pyramid small motion are removed and large motions becomes small motions . Then applying lucas kanade algorithm there we get optical flow along with the scale.

(iii)Pruning far away points

(iv)First finding centroid of object to be detected

(v)Drawing centroid

Summarizing above steps , We take the first frame, detect some Shi-Tomasi[4] corner points in it, then we iteratively track those points using Lucas Kanade optical flow. For the function calcOpticalFlowPyrLK() we pass the previous frame, previous points and next frame. It returns next points along with some status numbers which has a value of 1 if next point is found, else zero. We iteratively pass these next points as previous points in next step.

D. Updation (Inclusion & Exclusion of outlayer points) as well as Final Tracking showing

Adding only those corners to new_corners_updated which are at a distance of <=30 acoording to my implementation. It can be found by experiment. Now based on new corner points we draw circle of min radius enclosing all points. Below step by step process is shown:

(i)Drawing the new points

(ii)Finding the min enclosing circle

After D(ii) we try to see if number of updated corner points fall below a certain threshold, it simply tells that object is lost. Then a message is printed telling object is lost , Reinitialize it.

(iii)Updating old_corners and oldFrameGray

(iv)Showing stuff on video

In this step , tracking percentage is shown which stands for performance measure.

Result and Analysis

Accuracy is shown in terms of tracking integrity or percentage which is basically based on movement and its detection closeness with respect to time which varies randomly wirh new frames inialy and increases as frame lasts longer.

Dataset used for testing accuracy was either web cam or video with object motion inside window to be detected.

However , This approach may fail where assumption that intensity is constant in very small time interval with respect to displacement of pixels is not valid. In general intensity doesn't changes much , so it works fine in general case.

References

[1] B. N. Subudhi, S. Ghosh, P. K. Nanda, A. Ghosh, "Moving object detection using spatio-temporal multilayer compound Markov Random Field and histogram thresholding based change detection", Multimedia Tools and Applications, vol. 76 ,pp. 13511 -13543 ,2017.

[2] S.Z. Li, Markov Random Field Modeling in Image Analysis, Springer,London, 2001.

[3] B. N. Subudhi , S. Ghosh, S. B. Cho , A. Ghosh, "Integration of fuzzy Markov random field and local information for separation of moving objects and shadows" , Information Sciences, vol. 331, Pages 15-31,2016.

[4] https://en.wikipedia.org/wiki/Lucas-Kanade_method.

[5] https://docs.opencv.org/3.4.0/d7/d8b/tutorial_py_lucas_kanade.html

[6] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Addison Wesley Longman Publishing Co. Inc., Boston, MA, 2001.

[7] https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node47.html