A Vector Gradient Approach for Detecting Boundaries in Images and Video Sequences

Sreenjoy Chatterjee

Department of Electrical Engineering, Rochester Institute of Technology,
Rochester, NY 14623, USA.
sc8678@rit.edu

ABSTRACT

In this paper, we propose an algorithm which effectively detects boundaries in vector fields. An edge map is first created for a single image, be it grayscale or color. Then the same algorithm is extended to video, by reading the frames one at a time and processing them in real-time. A vector gradient edge detector is employed, wherein the gradient magnitude is calculated by using the partial derivative analysis of the pixel intensities.

Keywords – Edge detection, vector gradient, real-time processing, Image Processing.

1. INTRODUCTION

Edges in an image correspond to discontinuities in the 2D domain, or an abrupt change in the values of the pixel intensities. Thereby they come under the domain of features in an image, since their distribution also contributes to features in the image. Feature detection is one of the primary aspects of Image Processing, and most algorithms rely on an effective feature detection technique, as a pre-processing tool. While there are many segments of feature detection, one of the basic points has always been in terms of edge detection, corner detection and blob detection. Although edge detection for a single image has been relatively dealt with quite majorly in the past, however, the extension to video has not received that amount of attention. The motivation for this paper thereby lies in the fact that we aim to implement an algorithm which detects boundaries in the vector field in real-time, thereby simultaneously providing us with the edge map of the vector field.

There are many existing algorithms which efficiently detect boundaries in vector fields. In [1] they use a two-stage technique for detailed feature extraction. That coupled with the Plessy Corner detector provides effective feature extraction analysis using UNIVAR/OMNIVAR windows. At the same time [2] aims at reviewing the two aspects of edge detection, localization of corners and trihedral vertices, and the influence of noise. [3] solves the edge-detection problem by formulating a set of n^2 luminance samples from an image sub-area and then determines whether the sub-area contains a boundary element or not. Also [4] makes use of the three attributes of a Canny edge detector; being, good detection, good localization and low-responses multiplicity. The concept of

confidence map generation for boundary detection is employed in [5]. [6] shows the algorithm to compute the spatial gradient estimation, by calculating the spatial gradient matrix and looking at the eigenvectors and eigenvalues.

2. PROPOSED ALGORITHM

Edge detection is a simple technique where we concentrate on the pixel intensity values to understand the presence or absence of an edge in that region. Thereby the first criteria comes down to selecting a region, or a window to operate on. It is generally better to use an odd-sized window, since then we would have a central pixel. The idea is that if, in the region, there is a disparity in the pixel value, it is more likely that the region contains an edge than a region where the pixels are rather closely related in terms of intensities. Then comes the edge direction. Broadly classified, edges come down to two types – horizontal edges and vertical edges. It is very easy to understand that when there is a difference in pixels in a vertical direction, there shall be a horizontal edge and vice-versa. So we detect each of them individually and then integrate them in one single edge map. The difference in pixel values is obtained using partial derivatives. The co-ordinate system used in this approximation is that abscissa denotes the row and the ordinate denotes the column of the image matrix.

Horizontal Edge Detection:

$$\partial x(x, y) = f(x + 1, y) - f(x, y)$$

Vertical Edge Detection:

$$\partial y(x, y) = f(x, y + 1) - f(x, y)$$

2.1 Color Vector Gradient Approach

The above approach is the basic idea, which is appropriate for grayscale images, wherein there is a single channel and thereby computations with pixel intensities are restricted to one single matrix of values. This becomes complicated when there are three channels for say, a color image, namely red, green and blue channels. Now we can convert the RGB image to a grayscale format and then perform edge detection. This works too since the features don't change when we convert formats but in comparison, it has been seen that performing color vector gradient on a color image is much more accurate in terms of edge detection, rather than simple RGB to gray conversion and edge detection.

[6] gives an effective approach for implementing the color vector gradient algorithm on color images. If a color image is represented by a vector field $y = [y_1, y_2, y_3]^T$ where y_1, y_2, y_3 represent the three color channels, say, red, green and blue or LAB, if we use the LAB color space.

The gradient matrix is then defined as –

$$\boldsymbol{D}(\boldsymbol{i}, \boldsymbol{j}) = \begin{bmatrix} \frac{\partial y1}{\partial u} & \frac{\partial y1}{\partial v} \\ \frac{\partial y2}{\partial u} & \frac{\partial y2}{\partial v} \\ \frac{\partial y3}{\partial u} & \frac{\partial y3}{\partial v} \end{bmatrix}$$

where, *u* and *v* are the spatial co-ordinates for the pixel (*i*,*j*). The expressions thereby denote the partial derivative either in the x-direction or the y-direction.

The square root of the largest eigenvalue of

$$\mathbf{D^T D}\ (i,j) = \begin{bmatrix} p(i,j) & t(i,j) \\ t(i,j) & q(i,j) \end{bmatrix}$$

The largest eigenvalue λ of DTD can be expressed as –

$$\lambda(i,j) = \frac{1}{2}(p(i,j) + q(i,j) + \sqrt{\left(p(i,j) + q(i,j)\right)^2} - 4(p(i,j)q(i,j) - t(i,j)^2)\ )$$

Where,

$$p(i,j) = \left(\frac{\partial y1}{\partial u}\right)^2 + \left(\frac{\partial y2}{\partial u}\right)^2 + \left(\frac{\partial y3}{\partial u}\right)^2$$

$$t(i,j) = \left(\frac{\partial y1}{\partial u}\right)\left(\frac{\partial y1}{\partial v}\right) + \left(\frac{\partial y2}{\partial u}\right)\left(\frac{\partial y2}{\partial v}\right) + \left(\frac{\partial y3}{\partial u}\right)\left(\frac{\partial y3}{\partial v}\right)$$

$$q(i,j) = \left(\frac{\partial y1}{\partial v}\right)^2 + \left(\frac{\partial y2}{\partial v}\right)^2 + \left(\frac{\partial y3}{\partial v}\right)^2$$

Finally, the gradient magnitude is given by,

$$G(i,j) = \sqrt{\lambda(i,j)}$$

2.2 Edge-Detection for video sequences

The approach in 2.1 can be extended to video, since video may simply be considered to be a collection of many images or frames, at a certain rate, termed frames per second (FPS). The concept remains the same, with the only difference being that we view these images at 30 frames per second now. So it appears to us as a live video, but essentially the vector gradient algorithm functions at each frame individually.

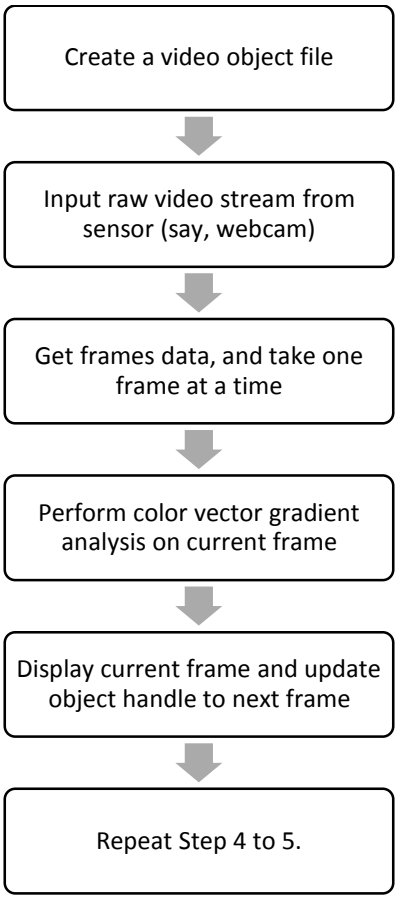

Fig 1: Block Diagram for boundary detection in video

3. EXPERIMENTAL RESULTS

The proposed algorithm shows good results for a single image in terms of edge detection. However, when applied to video, the point of FPS comes into play, and thus the FPS of the

output video does not exactly match the real-time video feed's FPS. This is because of the fact that when we process each frame and thereby compute its vector gradient, a certain amount of computational time is required, which in turn reduces the frames-per-second of the output video.

In a nutshell, while seeing the real-life video feed, if we move our hands, correspondingly the video feed changes almost simultaneously. However, when it comes to the edge detected video feed, there is a certain lag to the video before it updates to the present frame.

Fig 2: Original Image

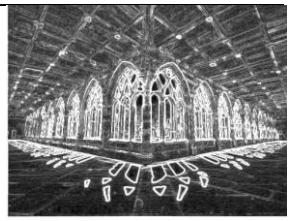

Fig 3: Edge-detection on Image

It can be observed that for a two-perspective image such as this one, the edge-map comes out to be quite appropriate.

Fig 4: Original Frame

Fig 5: Edge-detection on Frame

The lag in the video occurs because we can only acquire a certain FPS while operating via the CPU. If we were to, say, run this very algorithm on the GPU, we would be entitled to a higher FPS, maybe even close to real-time FPS.

4. CONCLUSIONS

The paper aims at making an efficient algorithm to perform one of the basic operations in Image and Video Processing. The algorithm runs on a few simple lines of code, however, this is a very significant project in video processing since it educates the programmer on how to access and deal with video especially that of a live webcam feed.

5. ACKNOWLEDGMENTS

The project was part of the Digital Video Processing class, as one of the Matlab projects. The class is under the Department of Electrical Engineering, Rochester Institute of Technology.

6. REFERENCES

[1]. Shigeru Ando, "Image Field Categorization and Edge/Corner Detection from Gradient Covariance", IEEE Transactions on Pattern Analysis and Machine Intelligence, February 2000.
[2]. E. de. Mitchelli, B. Caprile et al, "Localization and Noise in Edge Detection".
[3]. Werner Frei, "Fast Boundary Detection, A Genearalization & New Algorithm".
[4]. Didier Demigny, "A Discrete Expression of Canny's Criteria for Step Edge Detector".
[5]. Hsien-Che Lee, "Detecting Boundaries in a Vector Field".
[6]. Eli Saber, A. Murat Tekalp, "Fusion of color and edge information for improved segmentation and edge linking.