Summer Design Project 2016
Week 4 & 5

Kartik Sayani : 201451030
Meghna Ayyar : 201451027
Mentor : Prof. Pratik Shah

5 June, 2016

1 Topics

Implementation of Lukas Kanade optical flow algorithm on a video:

We had successfully implemented Lukas Kanade algorithm for determining optical flow between two images, based on Lukas Kanade's own paper. Foe implementing it on a video, we had followed *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm by Jean-Yves Bouguet* which describes a faster approach to determine even large motions. However, the implementations could not be done as it required a lot of optimization on architecture as done in the built-in function *calcOptFlowLKPyr()* The implementation, given a set of points, was intended to return the position of those points in the next frame. Our implementation gives a very small displacement of the points, where as the built-in function gave a large value of displacement for the same set of points.

Papers on depicting motion in still images:

We have collected these following papers on Motion depiction by speed-lines, motion lines, replication of images, jagged contours, etc.

- *The synthesis of non-photorealistic motion effects for cartoon* by *Ying-Hua Lai*
  This is one of the most interesting and helpful paper. We have followed this paper and have partially implemented it too. We haven't posted the code on the drive because it is pretty crude so far. We are affirmative that we shall probably be able to implement this paper in code by the end of this week. The paper explains, firstly, what ar the various types of motion depiction techniques and then how to render them. To render these drawings, it is first necessary to determine which objects are moving, hence the optical flow comes in handy. Then, determine the rear edge of the moving object, because almost all the techniques of motion depiction are drawn behind the moving object. The edge detection firstly requires the image to be converted to gray-scale, then dilated. Dilation of an image is extrapolating the dominant features of the image over the non-dominant part. The dominant feature in this case is the pixel with a larger intensity value. that is, the value of output pixel will be the highest value in the window neighborhood of the input pixel. Hence, spreading the dominant feature over the other part. Then the image is converted to binary based on a threshold. All pixels having values below the threshold are black and the rest are white. After this the edge detection technique is applied.
  Canny Edge Detection Method:
  A brief understanding of how the algorithm works. Rigorous Mathematical formulation was not studied. This is similar to the Sobel Edge Detector in the initial steps. The image is converted to Grayscale and a Gaussian Filter is applied to supress the high frequency noise. Then convolution kernels are used to find the gradient in the x and y directions and thence Magnitude and angle of the gradient is found. From here Canny makes the edges better by dong hysteresis thresholding. i.e. instead of using a single threshold to decide the pixel that is exactly the edge it sets two thresholds. All the edge pixel values above the higher threshold are considered strong edges and retained, all below the low threshold are rejected. For the edge pixels lying in this range they are termed weak edges and retained only if they are connected to a strong edge and else removed. Doing this reduces finer edges of almost one-pixel width.
  References:
  Slide show www.intelligence.tuc.gr/ petrakis/courses/computervision/canny.pdf
  Video reference: https://www.youtube.com/watch?v=uihBwtPIBxM
  :https://www.youtube.com/watch?v=sRFM5IEqR2w
  Link to Original paper: http://cmp.felk.cvut.cz/ cernyad2/TextCaptchaPdf/A

Once we have the edges of the moving object, we determine the rear edge by a method described in the paper. Using these edges, we draw rectangles around the objects of interests and determine their centers. These centers are important and are used to determine the rear edge as well as the path that the object takes. We use this path along which we draw the

- *Nonphotorealistic Rendering of Speed-Lines* by *Vincenzo Caglioti and Alessandro Giusti*

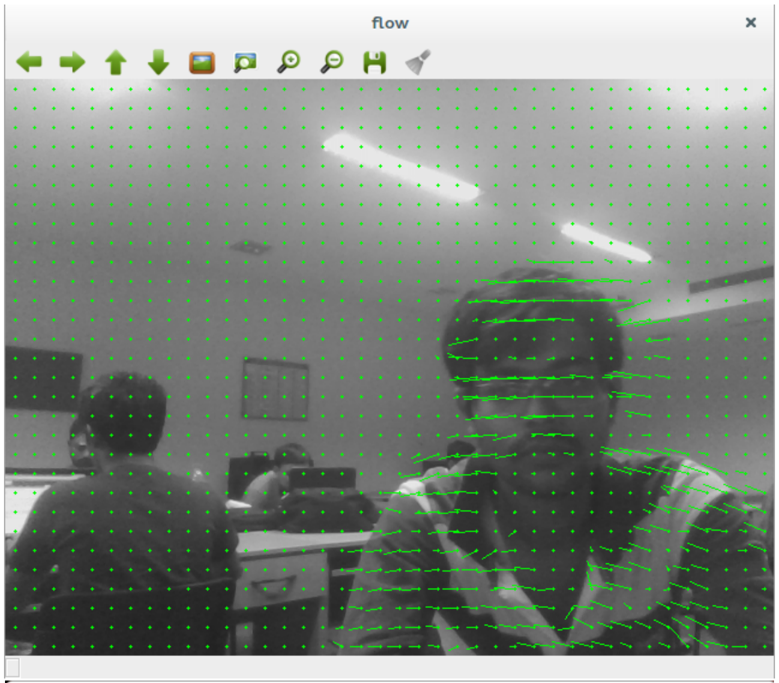
- *Drawing motion without understanding it* by *Vincenzo Caglioti et. al.*

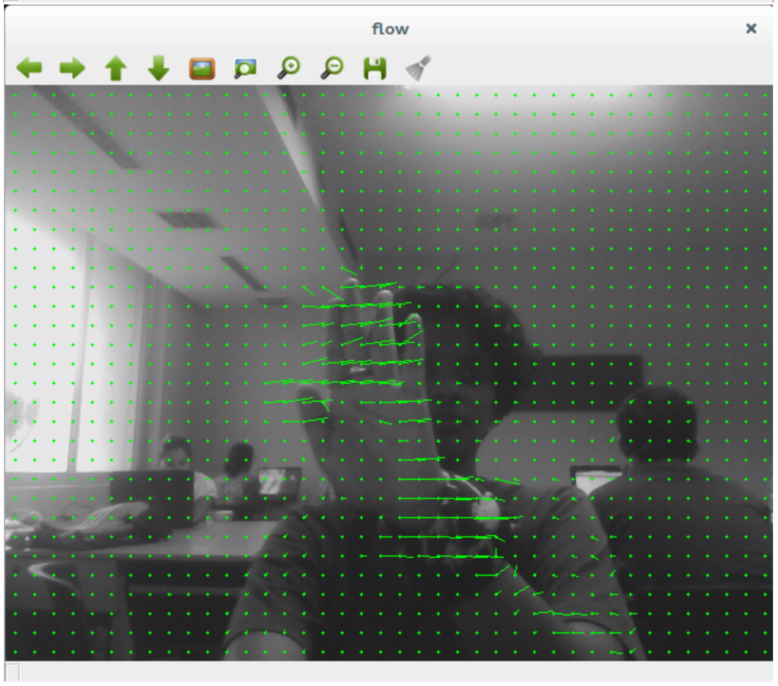
Feature tracking : Shi Tomasi Feature Tracker

Lucas Kanade Optical flow is a sparse optical flow method i.e. it uses only certain points to find the optical flow of the moving objects. In such a case finding the pixels that best captures the total motion in a frame is important. Shi and Tomasi propose a way to do this.
Good features to track:
In their thesis Shi and Tomasi suggest that total image motion can be modeled as two types *a.* Pure Translation and *b.* Affine motion model.
Pure translation can be assumed while computing motion to consecutive frames as the displacement will be small and hence the deformation matrix of the affine model will be small and can be assumed to be 0.
The affine model can be used when finding the dissimilarity between the features between frames set far apart.
In the paper they derive a mathematical formula to find the dissimilarity between the features and when this crosses a minimum threshold value t is discarded from the good features list.
They assume that a good window is the one that can be tracked well instead of using certain interest operators that other methods use, i.e. they construct an optimal solution. While tracking from frame to frame they derived an equation Zd = e, where d = displacement vector, Z is coefficients and e is the last two entries of the matrix a form the affine model. If this Z is well conditioned and less noisy then the feature window will be optical. Hence the Eigen values λ_1 and λ_2 must be large and do not differ by level order of magnitude, i.e. accept the window only if $min(\lambda_1, \lambda_2) > \lambda$ where λ is the predefined threshold. One problem with this is that a feature with high confidence can still be a bad feature to track. The feature might be computed in the image but may not exist in real world. Using the affine model to compare the features and their dissimilarity in two frames set a little apart in the video can help identify if the feature should be continued to be used or not. (feature monitoring)

Farneback Optical Flow

Farneback Optical Flow It is used to calculate dense optical flow. In this an approximation of the neighborhood of each pixel is done with a quadratic polynomial.
$f(x) = x'Ax + b'x + c$ The coefficients are calculated via least squares method and each neighboring pixel is given a weight with respect to their certainty and applicability. The main assumption while calculating is that the displacement field changes slowly from one frame to the other. Also instead of using a global displacement field the displacement is computed locally for each pixel. The local polynomials at the same coordinates in two consecutive frames are identical except for the displacement. Here are some screenshots obtained by using OpenCV's built-i function to calculate Farneback Optical Flow





Other

We are also looking into using Blender and asset files for 3D animation and modeling.