Cairo University
Faculty of Engineering
Department of Computer Engineering

Feature Extraction Module

Corner Detection

Report to explain the approach we took to make a corner detection

Introduction

We need to find matching points between different frames of an environment. Why? If we know how two images relate to each other, we can use both images to extract information from them.

When we say matching points, we are referring to characteristics in the scene that we can recognize easily. We call these characteristics features. so it must be uniquely recognizable.

So we will detect the most recognizable which are interest points : Corners and that is because it is the intersection of two edges which the directions of these two edges change. Hence, the gradient of the image (in both directions) have a high variation, which can be used to detect it.

Different Algorithms

We searched in the most known algorithms in finding corners which are the Harris Corner Detection and J. Shi and C. Tomasi which is a small modification to harris corner detection method but it showed better results compared to Harris Corner Detector the only change was in the scoring function which became the minimum of the two eigenvalues, so we selected : “J. Shi and C. Tomasi Algorithm”.

J. Shi and C. Tomasi Algorithm

To find a corner, find the difference in intensity for a displacement of (u,v) in all directions. This is expressed as below:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \left[\underbrace{I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}\right]^2$$

Window function is either a rectangular window or gaussian window which gives weights to pixels underneath.

We have to maximize this function E(u,v) for corner detection. That means, we have to maximize the second term. Applying Taylor Expansion to above equation and using some mathematical steps, we get the final equation as:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Where:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

Ix and Iy are image derivatives in x and y directions respectively, and then apply the score function:

$$R = min(\lambda_1, \lambda_2)$$

And that's it now you have all the corners!

Implementation

Here are the steps that we implemented in the program for feature extraction module "corner detection":-

1. Get the derivatives of the picture for the rows [yy] and cols [xx] and both [xy].
2. Average with a gaussian filter window the [xx*xx] , [yy*yy] and [xx*yy] so you can get the correlation matrix.
3. The corner measure is then defined as the smaller eigenvalue of correlation matrix Which we can get by solving an equation of 2nd order and get the minimum lamda.

Now, you have the corners, but you need the best n corners:

4. Apply maximum filter to extract the best in a region which we specified by: 2 * minimumDistance +1
5. Exclude the borders which could cause in a lot of unwanted corners.
6. Sort the remaining points to its highest intensities and select the n corners you want.
7. Select the n corners and return them.

Processing Time Takes

As we don't take the whole frame to process then the processing time differes but on averge it takes a [7 to 18] ms and we hope this speed don't affect on us when we attached later to the optical flow.

Comparison to Opencv Method

We compared our implementation to opencv method and the accuracy was very close you can't even determine which is better, but on the other side the speed of opencv method is much faster as results showed it's faster than our implementation - even after optimizing it - with [60 – 80]% faster than us and that is because their function is implemented by c/c++ which is already compiled.