CMSC-426 Homework 2

Prateek Arora[1]

Harris detection Implementation

The following steps were followed to implement Harris corner detection:

- Color to grayscale
  Converting the image scale will enhance our processing speed. This is done using rgb2gray function.
- derivative calculation
  The derivative for each pixel is calculated $I_x(x, y), I_y(x, y)$
- With $I_x(x, y)$, $I_y(x, y)$, we can construct the structure tensor M.
- Harris response calculation
  In this step, we will compute the smallest eigenvalue of the structure tensor with following approximation equation.

  $$R = det(M) - k(trace(M))^2$$

  where k is an empirically determined constant; $k \in [0.04, 0.06]$.
  The R is then thresholded by a value to get meaningful corners.

- Non-maximum suppression
  In order to pick up the optimal values to indicate corners, we find the local maxima as corners within the window which is a 3 by 3 filter.

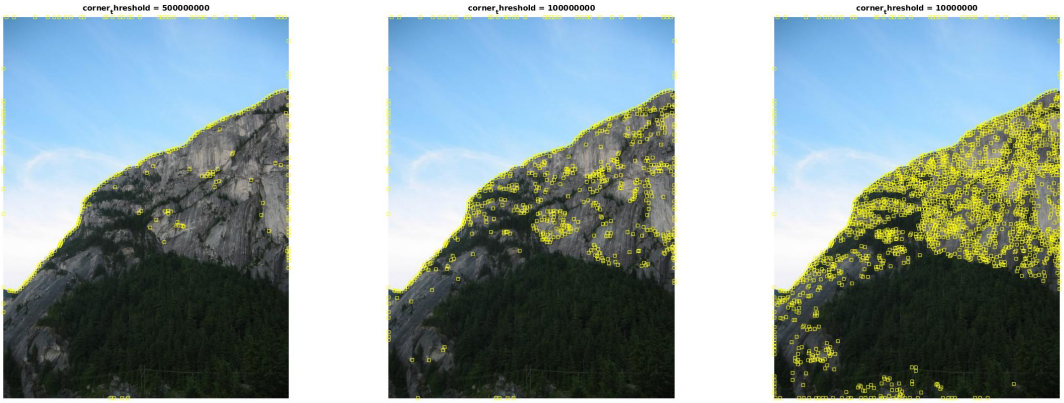


Figure 1: Response for Window length=3, Corner threshold = as displayed in image

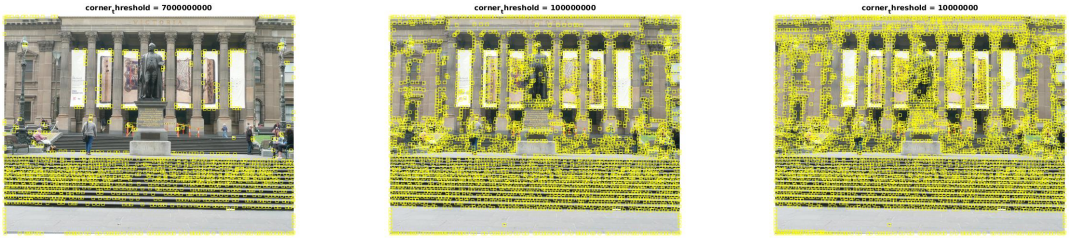


Figure 2: Response for Window length=4, Corner threshold = as displayed in image

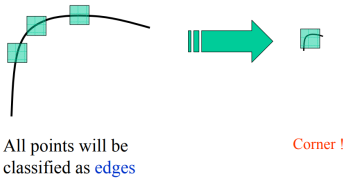
[1]Graduate student, Department of Robotics, University of Maryland, College Park, MD 20740, pratique@terpmail.umd.edu

Question 1: Harris Corner Detector properties

Effect of following Variations to the image are described in subsections:

1. Image Rotation:
   On rotating the image, the matrix **M**(computed from image derivative) will have different values that the original image, but the eigenvalues of the matrix **M** will remain the same and all the points will be enclosed by the same ellipse(this ellipse would be rotated as eigenvectors will change). Thus, the shape of ellipse and the eigenvalues are preserved. Therefore, Harris corner detection is **invariant** to image rotation.

2. Image scaling:
   On scaling the image, the corner in the original image enlarges to become a line. Thus, the magnitude of differential of intensities of the pixels in the scaled image has a lower value than the differential of intensities of the same pixels of the original image.Therefore, for the same value of threshold, the pixels don't qualify as corner. Harris corner detection is **not invariant** to multiplication of pixels by a constant value.



3. Increment all pixel values by a constant:
   Let's assume that the intensity of a pixel is given by $\mathbf{I_{x,y}}$ and the intensity of neighbouring pixel is given by $\mathbf{I_{x+dx,y+dy}}$. On incrementing all the pixel value by a constant **C** the derivative(difference in pixel intensities in case of images) of remains the same, i.e.

$$\delta = (I_{x+dx,y+dy} + C) - (I_{x,y} + C) = I_{x+dx,y+dy} - I_{x,y}$$

   This can also be seen in the illumination equation which is given by:

$$I'_{x,y} = \alpha * I_{(x,y)+\beta}$$

   where $I'_{x,y}$ is the new intensity $I_{x,y}$is the original intensity, α is the multiplication factor and β is the increment factor.
   Thus, Harris Corner detection is **invariant** to increment in pixel values a constant.

4. Multiply all pixel values by a constant:
   Multiplication of all pixels by a constant, on the other hand, changes the differential value as seen in the aforementioned differential formula. The α in the intensity equation mentioned above is the multiplication factor. Change in this factor causes the differential to vary. In other words, if a corner was detected in the original image, it may not be detected in the new image, wherein pixels are multiplied with a constant. Thus, Harris corner detection is **not invariant** to multiplication of pixels by a constant value.

Question 2: Image warping

Image warping is the process of converting an image to another image using transformation(generally homogeneous transformation). There are two commonly used process to perform image warping, which are as follows:

- **Forward Mapping**: In forward warping for each pixel(or location) $\tilde{p}$ in the original image, the corresponding pixel location is computed using the transformation matrix and the value of the new pixel $\tilde{p'}$ in the warped image is computed using the process called **'splatting'**. Through splatting the color of pixel $\tilde{p'}$ is computed. The value of color is distributed among neighbouring pixels if $\tilde{p}$ map to another decimal $\tilde{p'}$(i.e. $\tilde{p'}$ is a decimal and not an integer).

- **Reverse Mapping**: In Inverse mapping the process is reversed. Instead of finding the corresponding point in the warped image, we find a point in the original image that corresponds to a point in warped image.

$$\tilde{p} = H^{-1} * \tilde{p'}$$

The difference between the two approaches is that the forward mapping is much more computationally expensive. For each pixel in original image that falls between the pixels of warped image, color has to be distributed using weights. Also, for the same reason there is the possibility of 'holes' and 'overlaps' being generated. Since an image consists of discrete pixel the forward mapping is not accurate as we have to linearly interpolate the values to fill the 'holes'. Inverse mapping addresses the issues of holes and overlaps. Since we are straightaway picking up the color value of the corresponding pixel in the original image, the task becomes simpler. As in the forward map, the pixel coordinate had to be rounded, but in inverse mapping rounding takes place in the coordinate system of input image, not the output image. Thus, even with the discrete sampled image, it is easy to compute output image form the input image without the fear of holes and overlaps in the output image .Therefore, for the aforementioned reasons, inverse mapping is preferable.