Computer Vision Course

Computer Vision Assignment - 1

Team: Deepesh Bhageria(S20170010041), Thota Harshitha Chowdary(S20170010164), Aman Kumar(S20170010009)

February 03, 2020

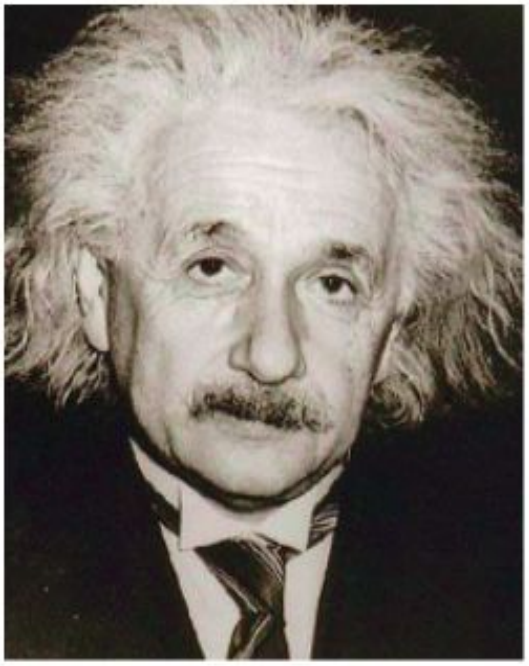
Que-1(Algorithm)

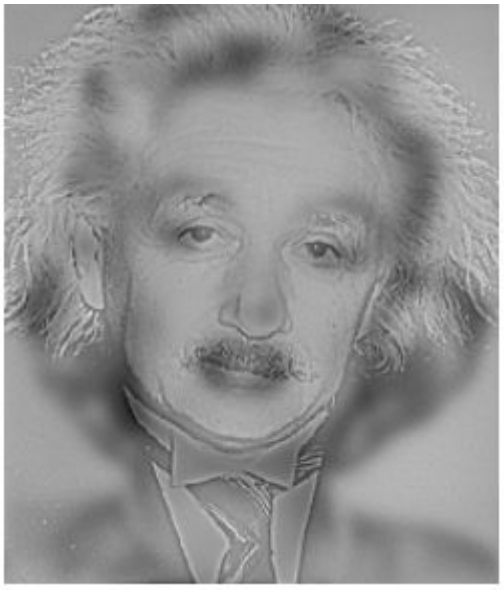
Problem 1 Detailed Approach :

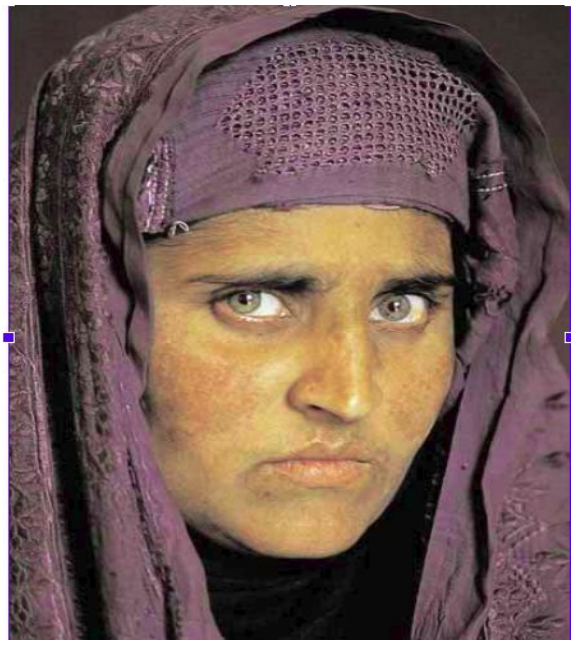
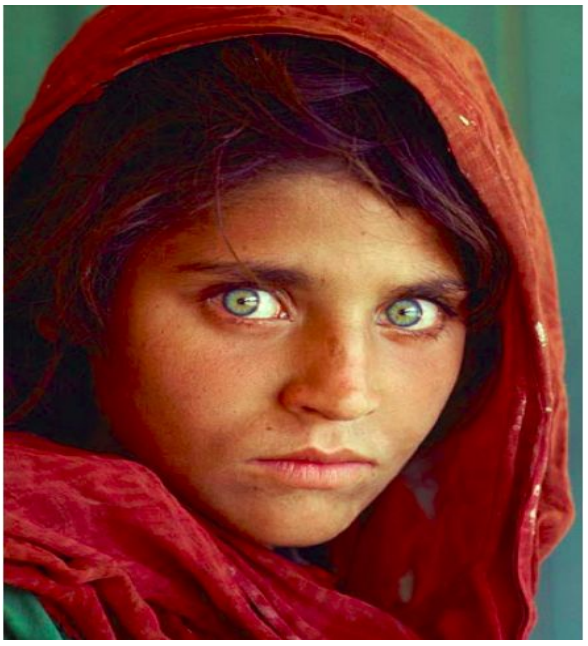
A hybrid image is the sum of a low pass filtered version of an image and a high-pass filtered version of the second image. There is a free parameter(standard deviation), which can be tuned for each image pair, which controls how much high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the "cutoff-frequency". In this problem, we were asked to use two cutoff frequencies alpha and beta (one tuned for each image). In the current approach, the cutoff frequency is controlled by changing the standard deviation (sigma) of the Gaussian filter used in constructing the hybrid images.

We first read the two images and pass the images and alpha, beta values as arguments to the function named ‘hybrid image’ where the images are sent to their corresponding filters i.e, high-pass and low-pass filters with their corresponding threshold values alpha and beta respectively. And in the filter functions with the help of the function ‘filterDFT’ we take the image matrix convert it into the frequency domain using fft2,fftshift from NumPy i.e., the Fourier transform and multiply it with the Gaussian kernel in its frequency domain with the standard deviation(threshold) passed to the function (Gaussian kernel is made in the function ‘makeGaussianFilter’) which is equivalent to the convolution between the image and Gaussian kernel in the spatial domain. And the inverse Fourier transform is returned using ifftshift, ifft2 and the spatial domain converted matrix is returned back to the ‘hybrid image’ function from the two filters and the sum of the two filter resultant matrices are returned and we save the hybrid image returned using ‘imsave’ from the misc module.

:





Que-2(Algorithm)

The Shi-Tomasi corner detector is based entirely on the Harris-Corner Detector. However, one slight variation in a "selection criteria " made this detector much better than the original. It works quite well where even the Harris corner detector fails.

The score for Harris corner detector was calculated like this (R is the score):

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

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

For **Shi-Tomasi**, it's calculated like this:

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

Procedure:

Compute the gradient at each point in the image.

Create the H matrix from the entries in the gradient.

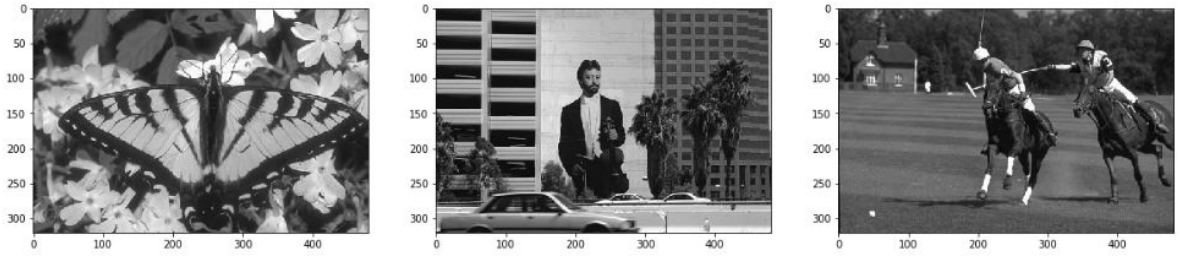
Compute the eigenvalues.

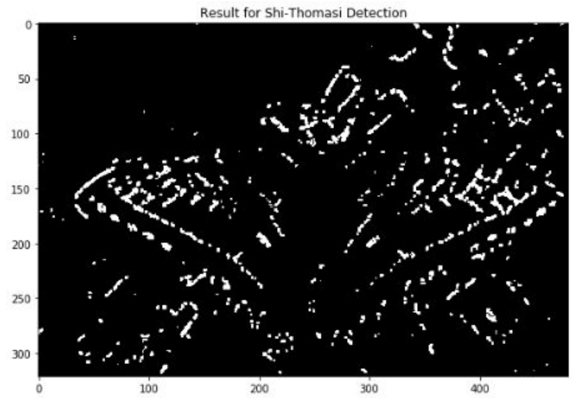
Find points with large responses (min > threshold).

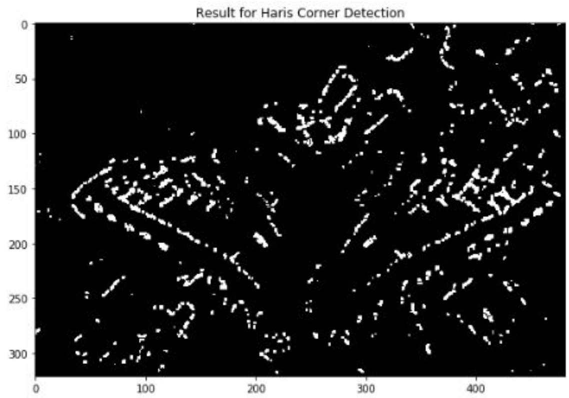
Choose those points where min is a local maximum as features.

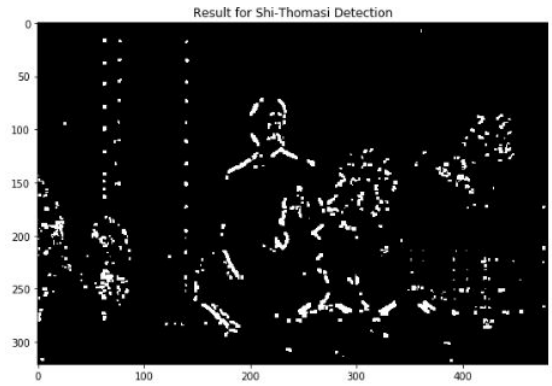
Result:

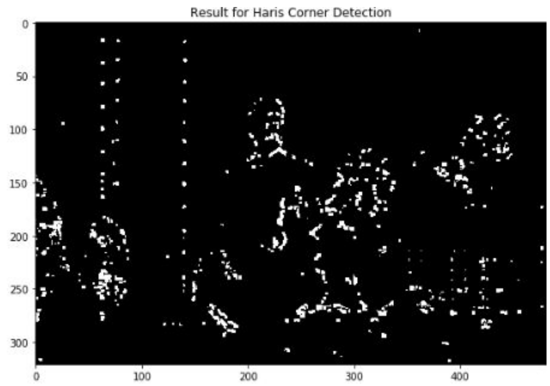
The Shi-Tomasi corner detector is better than the Harris corner detector, except for a minor change they did. However, it is much better than the original corner detector, so people use it a lot more. Also, OpenCV implements the Shi-Tomasi corner detection algorithm.

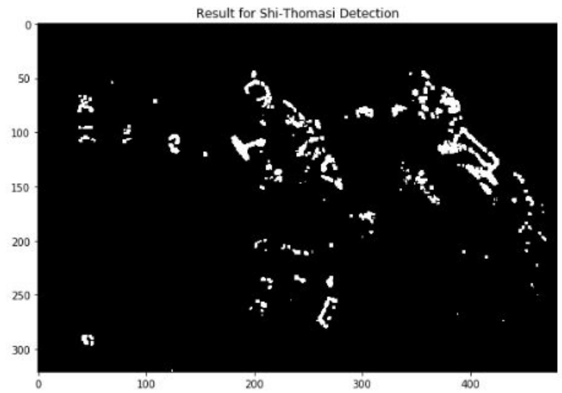


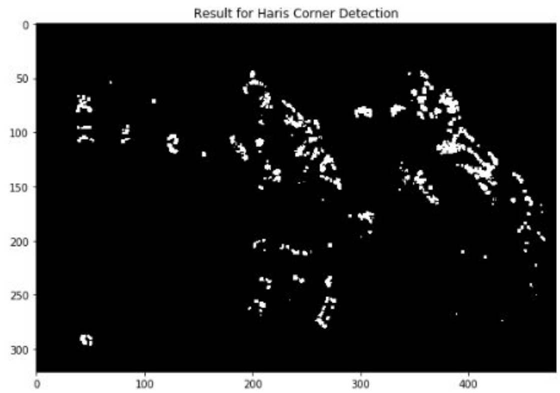












Que-3 (Algorithm)

- Generated a Laplacian of Gaussian filter.
- Build a Laplacian scale space, starting with some initial scale and going for n iterations:
    - ➢ Filtered image with scale-normalized Laplacian at the current scale.
    - ➢ Save square of Laplacian response for the current level of scale space.
    - ➢ Increase scale by a factor k.

- Performed non-max suppression for each response.
- Performed non-maximum suppression in scale space.
- Display resulting circles at their characteristic scales.