ANSWER TO TASK 1

Input image

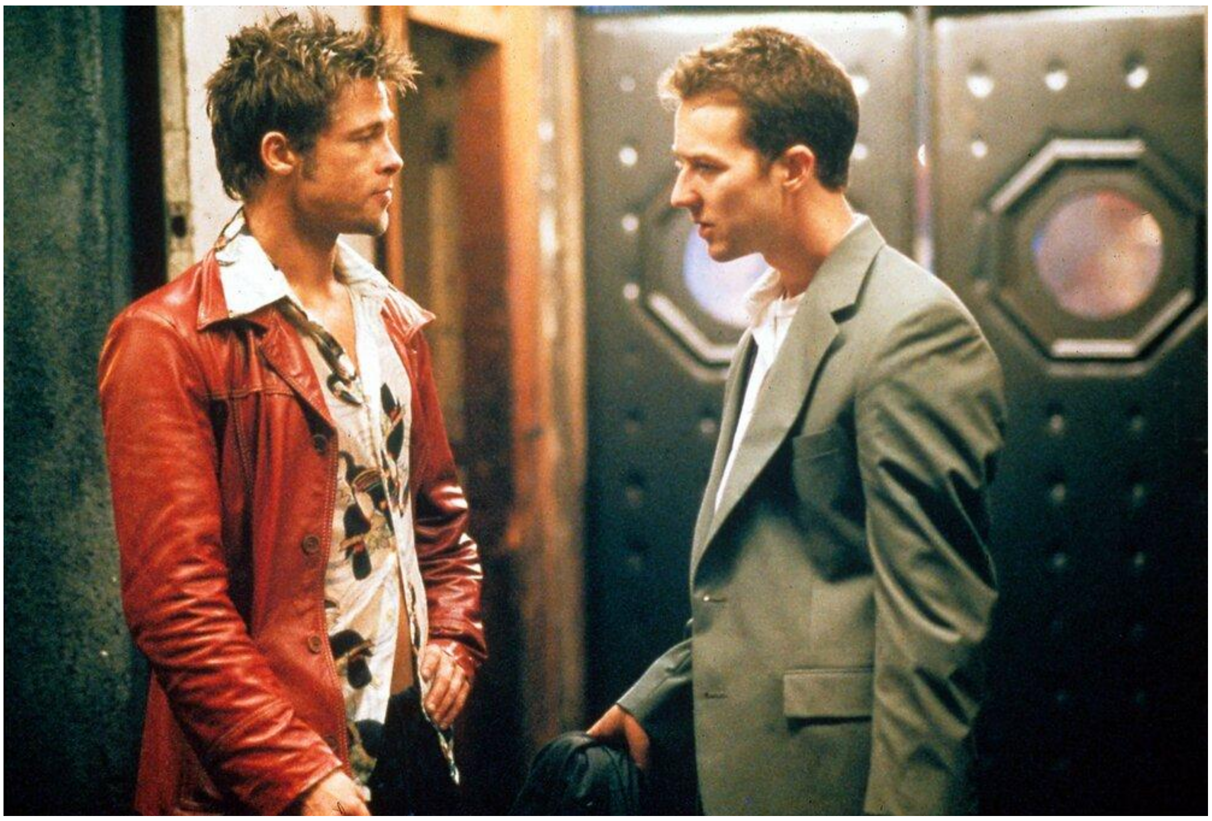

Figure: Input image

The steps of Canny Edge detection are shown below:

1. **Grayscale Image**: Firstly we have to convert the image into a grayscale image to apply edge detection

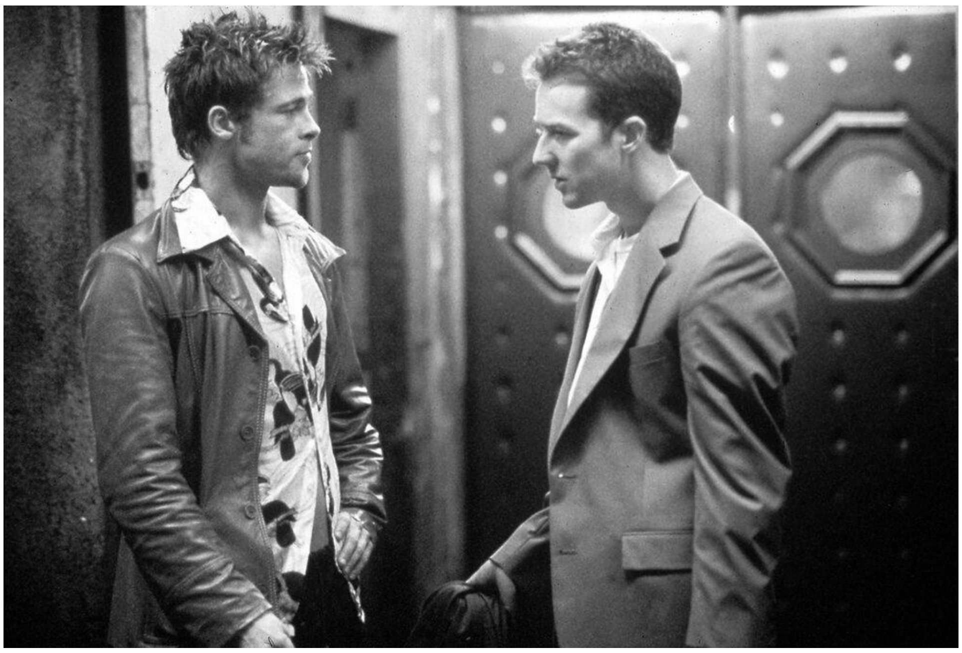

Figure: Grayscale Image

2. **Gaussian Smoothing (Noise Reduction):** Gaussian smoothing is applied to the image to reduce noise and eliminate small fluctuations in intensity.

Figure: Gaussian Smoothed Image

3. **Gradient Calculation:** Gradient magnitude and direction is calculated using Sobel edge detection operators.
4. **Non-Maximum Suppression:** This step involves suppressing all the edges that are not considered to be part of the final edge map. It is achieved by finding the local maximum in the gradient direction of each pixel.

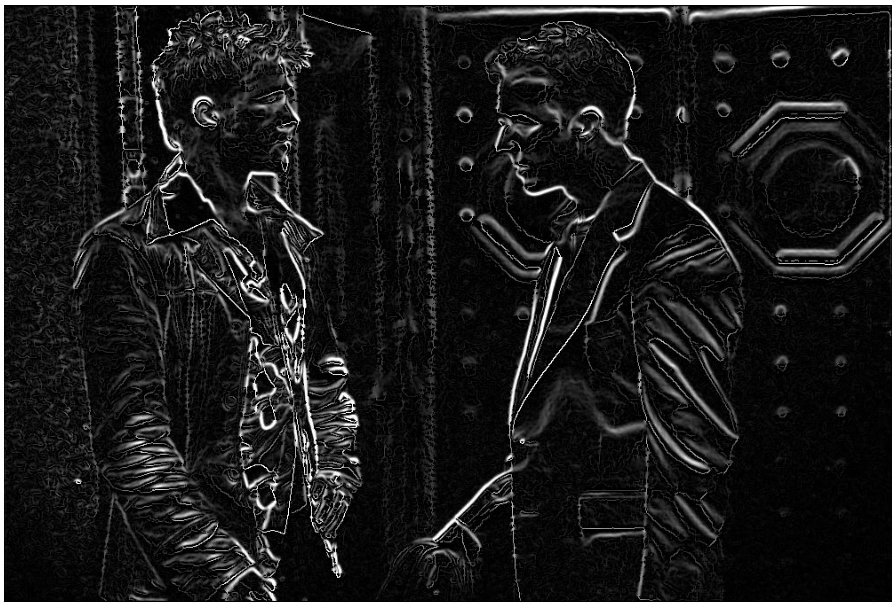

Figure: Non-maximum suppression

5. **Double Thresholding:** In this step, two thresholds are used to identify strong and weak edges. Strong edges are those with high gradient values, while weak edges are those with low gradient values. The threshold values are set based on a ratio between the maximum and minimum gradient values in the image.

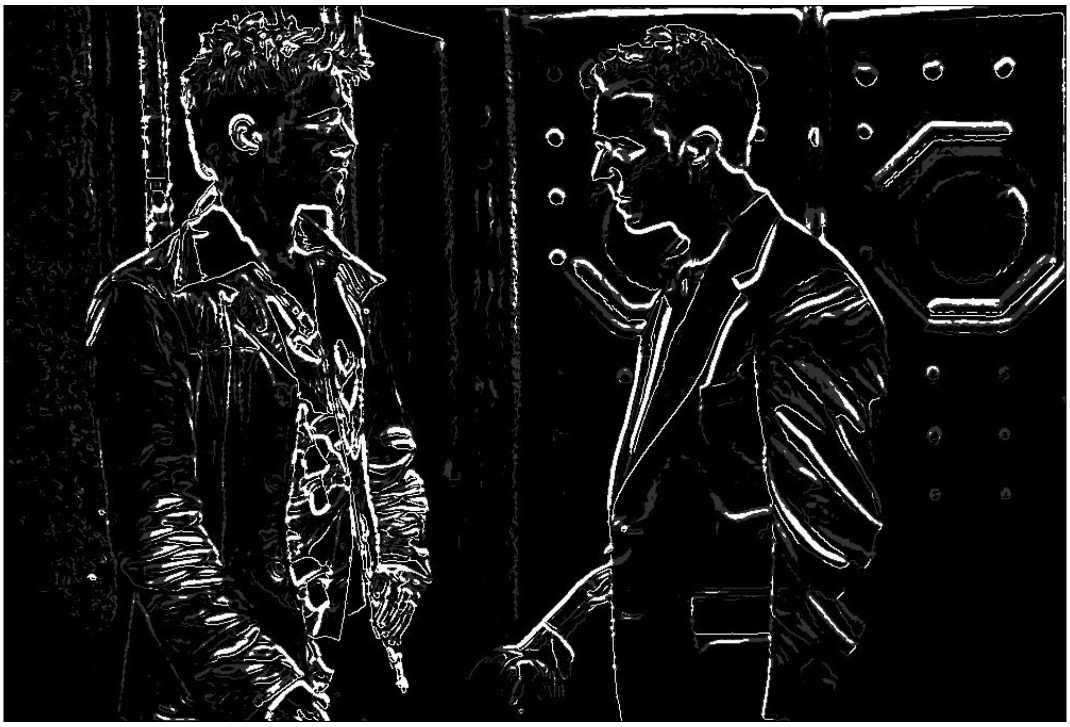

Figure: Double Thresholding

6. **Edge Tracking by Hysteresis:** The final step involves tracing edges along the strong edges identified in the previous step. Weak edges that are connected to strong edges are considered to be part of the final edge map

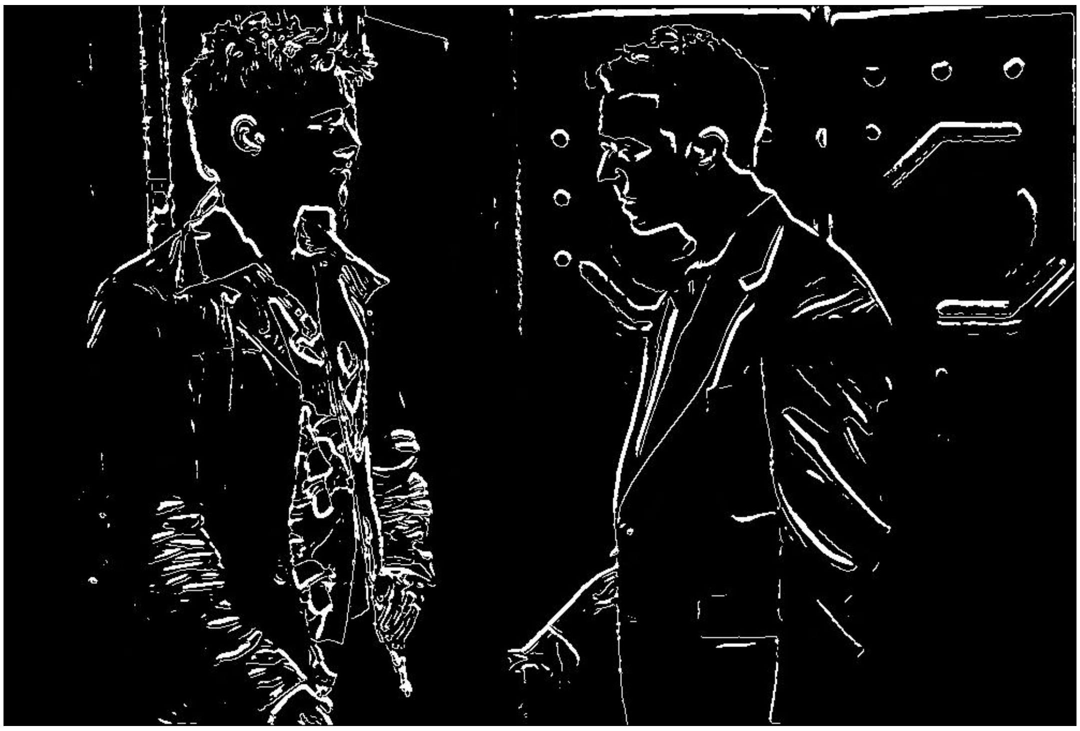

Figure: Final output (Canny Edge detected image)

ANSWER TO TASK 3

Canny edge detector speed can be increased in a number of ways:

1. **Reducing image size:** Downsampling an enormous input image can dramatically increase the Canny edge detector's throughput.

2. **Using faster Gaussian smoothing methods:** Gaussian smoothing is computationally expensive; therefore, using faster Gaussian smoothing algorithms, particularly when dealing with high kernel sizes can result in faster computation. Faster processing can be achieved by using a separable Gaussian filter or an approximation of the Gaussian filter, like the box filter.

3. **Implementing Sobel edge detection using integral images:** To use Sobel operators for computing gradients, many convolutions with large kernels are required, which can be a time-consuming process. Using integral pictures, which allow us to calculate the total of pixels in a rectangular portion of the image in constant time, can speed up this process.

4. **Use non-maximum suppression and thresholding in a single step:** Instead of conducting non-maximal suppression and double thresholding as distinct processes, using them together in a single operation can lessen the need for processing and free up more RAM.

5. **Using parallel processing:** The Canny edge detector can be modified to run in parallel on multi-core CPUs and GPUs. The algorithm's performance may be greatly boosted in this way.

Ultimately, the limitations of the program and the hardware being used will determine the optimization method selected. A balance between accuracy and speed should be considered to achieve optimal performance.

ANSWER TO TASK 4

The FAST (Features from Accelerated Segment Test) approach to corner identification consists of the following steps:

1. Choose a pixel P from the image.
2. Choose a threshold value for t.
3. Select a 16-pixel diameter circular circle surrounding P.
4. Examine whether there are n continuous pixels in the circle that are all brighter or darker than the intensity of P plus the threshold t.
5. If such a collection of pixels occurs, P is a corner.
6. Repeat steps 2-5 for each additional pixel in the image.
7. To delete duplicate corners, non-maximal suppression is applied.
8. Optionally, corner refinement techniques, such as sub-pixel accuracy, can be used to improve the localization precision of identified corners.

The FAST algorithm recognizes corners in the image by repeating these stages for each pixel in the image. The approach is efficient because it just requires a straightforward comparison of pixel intensities across neighbouring pixels, which can be efficiently performed using bitwise operations. Unfortunately, the system may produce false positives in textured or noisy regions.

Now the results:

Figure: Input image

OUTPUT

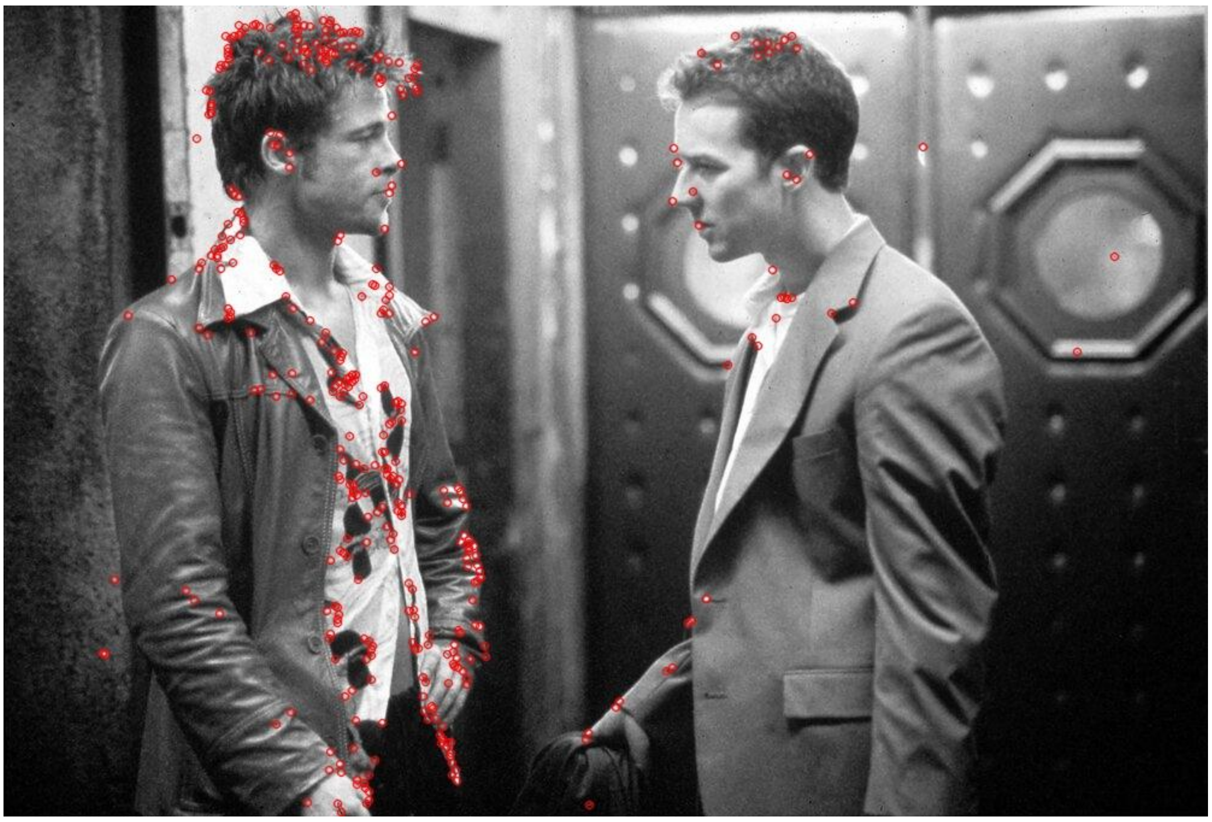

Figure: Output Image