<u>1. Lead Characters of the film:</u>

a. Face detection: Detecting all possible faces from the video using haar features. I've used only haarcascade_frontalface_default, which came along with opencv. Clustering will take a hit if I choose to detect tilted faces as well. So ignored the faces other than the frontal part of it.

b. Feature Extraction: I’ve used LBPP (http://en.wikipedia.org/wiki/Local_binary_patterns) features to define each of the face detected.

c. Clustering: Clustering of the above feature vectors is done using the k-means library available in opencv.

d. Selection of Lead Clusters: Top two clusters are selected based on the number of members allocated to that cluster.

e. Lead character selection: I’ve used Euclidean distance to find the most similar face with respect to the cluster center found by the k-means to find the lead character face.

<u>2. Binarization of hand written images:</u>

a. Smoothing: Smoothing of the input image is done using median_blur filter available in opencv.

b. OTSU threshold: Instead of manual selection of FG/BG, threshold from the OTSU algorithm is used for initial seeds. OTSU tries to find the threshold such that the deviation within the class is minimal.

c. Graph is built assigning the unary and pairwise costs.

d. Grabcut is used to binarize the pixel to either FG or BG.

<u>3. Fundamental Matrix between two images:</u>

a. Feature points from each of the images are extracted using SIFT descriptors.

b. Correspondence between the two image sets are found out scanning the entire set.

c. These image points are given to RANSAC algorithm to find out the best inliers from all possible combinations.

d. Normal fundamental matrix routine is used to find the F matrix.

4. FG-BG segmentation:

a. Effect of number of Gaussians: Increase in number of Gaussians has good impact on the results. Having more colour centers helps in discriminating between the FG and BG better .

b. Parameters affecting the computational complexity: Number of Gaussians directly increases the complexity as each pixel has to check the distribution over every added Gaussian and hence it polynomial increase. Even the video resolution has direct impact on the computation as expected.

c. Video where background subtraction fails: In the case where the foreground is covered with only one colour, since the distribution will fairly remain constant within the foreground, the program is not able to detect it as the foreground and only the boundary between the FG and BG is detected as FG. The example I've provided clearly shows this.

d. Why efficient: This algorithm instead of taking fixed Gaussian components for both FG and BG, it tries to adaptively decide the components based on the distribution within the FG and BG. This has direct impact on computational complexity as each pixel has to check its distribution over the Gaussians.