Blob Object Detection and Classification

Mohammad Mohaiminul Islam and Zahid Hassan Tushar

I. Introduction

Object detection and identification is a very challenging task in video-surveillance systems. Among multiple approaches and methods, this report is solely focused on the method based on foreground segmentation which can be divided into two sub tasks i.e., blob extraction and blob classification. In the blob extraction stage, OpenCV's floodfill module was used while in the classification stage a statistical classifier was implemented. Another challenge faced in video sequence analysis is to detect stationary foreground objects. For this reason, a routine was developed to detect stationary foreground blobs in the frame based on foreground history images. Further, a snippet of MATLAB code was generated to form an improved classification model.

II. Method

Since our routine is based on foreground segmentation, we needed a method to extract our foreground mask from each frame. To perform this, we leveraged OpenCV's background subtraction module called MOG2. It applies the Gaussian mixture model described in [1] and [2]. After that we implement a blob extraction routine to extract our points of interest. A statistical classifier was used to classify those blobs. Next we explored a stationary foreground detection algorithm using the method described in [3]. Finally we used some morphological operation to improve the extracted blob quality and developed a script to generate better classification model.

A. Blob Extraction

As the name suggests BLOB (Binary Large OBject) is a binary image containing a group of connected pixels of the current frame. The term "Large" indicates that only objects of a certain size are of interest and that "small" binary objects are usually noise. In this method, input is a foreground segmentation mask on which the extraction routine performs Sequential Grass-Fire algorithm [4] to extract the blobs based on the preferred connected component analysis. The algorithm scans the segmentation mask from top left to bottom right. It assigns an ID count which starts from zero. When an object pixel (white) is found, it increases the ID count and sets this value on the corresponding pixel of the output image. Then it burns the current pixel of the input image i.e., sets the value to zero (black) and checks the neighboring pixels (four or eight depending on the connectivity component) for object pixels. If there are any, then it places the current ID count in the output image and sets zero to the corresponding pixel in the input image. It also places those pixels in a list. The next step is to take the first pixel in the list and look for object pixels in its neighbors. If any of the neighbors is an object pixel, then the output image pixel is set to the current ID count, input image pixel is burned down and is put on the list. This keeps going on until all the pixels are examined. Although for the sake of programming convenience we have used OpenCV's floodFill [5], which labels one foreground pixel's neighbors by itself.

The extracted blobs are sorted based on their dimension. Usually small blobs are generated due to noise. Hence they are eliminated from the list of blobs. To facilitate this, a minimum threshold is set for the height and weight of the blobs. This helps in getting a meaningful list of blobs which will be classified in the next stage.

B. Blob Classification

Blob classification can be divided into two steps. Firstly getting the feature of the blob that best describes its characteristics and secondly based on those features using a classifier to put a label on each blob. Feature extraction means converting the blob into a representative number and ignoring the other information. There are many popular features such as area, bounding box, bounding circle, convex hull, compactness, circularity, perimeter etc. In our routine we have used the aspect ratio of the blob as the feature. The ratio is calculated using the formula in (1).

$$aspect\_ratio = \frac{blob\_width}{blob\_heigh} \tag{1}$$

This indicates the relative shape of the object of interest such as a person will have an aspect ratio less than one (since width is lower than height) while a car will have more than one (more width, less height). Since this work is constrained in classifying a limited type of blobs i.e. object, person and car, the feature aspect ratio performs quite well for these classes.

For classification, we have defined a prototype using mean and variance and developed a simple statistical classifier which measures the distance between the feature vector and the prototype. The smaller the distance, the more likely the blob resembles the prototype. The distance is calculated by weighted Euclidean distance measure given in (2).

$$WED\left(\vec{f_i}, prototype\right) = \sqrt{\frac{(f_i - mean)^2}{variance}} \tag{2}$$

$$ED\left(\vec{f_i}, prototype\right) = \sqrt{(f_i - mean)^2} \tag{3}$$

Since the performance of weighted Euclidean distance measure was not up to the mark, we used only Euclidean distance instead, which is given in (3). Feature normalization was not implemented as we used only one feature.

C. Stationary Foreground Detection

Sometimes a situation arises where an object of interest suddenly stops and remains stationary for a certain period and then starts moving again. We want to keep those objects in our foreground model. Since our background model is updating for each frame, it incorporates our stationary object of interest in the background. To overcome this issue, we have implemented a stationary foreground detection algorithm with help of the method detailed in [3]. We get the foreground mask $FG_t(x,y)$ from our background subtraction model. Then we calculate the Foreground History Images $FHI_t(x,y)$ using the temporal variation of the foreground based on (4) and (5).

$$FHI_t(x,y) = FHI_{t-1}(x,y) + w_{pos}^f \cdot FG_t(x,y) \quad (4)$$

$$FHI_t(x,y) = FHI_{t-1}(x,y) - w_{neg}^f \cdot \left(\sim FG_t(x,y)\right) \quad (5)$$

Where ~ is the logical NOT operation; w_{pos}^f and w_{neg}^f are two weights to manage the contribution of the foreground and background detection. Primarily, in the paper, it is suggested to increase the value of the history image pixels when they belong to the foreground and reset to zero when they belong to the background. However, resetting to zero for background pixels may cause temporarily sparse errors in foreground detection. A common scenario for this kind of problem is a crowded scene where fast moving objects cause camouflage errors in static regions. Hence, w_{neg}^f should be higher than w_{pos}^f for increasing the robustness against foreground errors (e.g., $w_{neg}^f = 5$ in our implementation). In short, we get a FHI score which increases if the pixel belongs to foreground and decreases if the pixel belongs to the background. This score is then normalized to the range [0, 1] considering the video frame rate (*fps*) and the stationary detection time (t_{static}) as shown in (6). Finally we use a threshold (η) to extract our Stationary Foreground ($SFG_t(x,y)$) model using (7).

$$\overline{FHI_t}(x,y) = min\{1, FHI_t(x,y)/(fps \cdot t_{static})\} \quad (6)$$

$$SFG_t(x,y) = \begin{cases} 1 & if \ \overline{FHI_t}(x,y) \geq \eta \\ 0 & otherwise \end{cases} \quad (7)$$

D. Foreground Mask Refinement with Morphological Operation

This section involves improving the foreground mask using some morphological operation. This is due to the fact that the quality of blobs extracted by the blob extraction routine depends on the quality of the foreground mask. Hence, a better foreground mask will result in an extraction of meaningful blobs. Also this will directly impact the outcome of the blob classification routine as the blob classification works with the output of blob extraction method.

While working with the basic background subtraction algorithm, we found that the overall output was very noisy. Hence, we implemented a post-processing method to remove noises as much as possible. To do this, a function is included in Lab2.0AVSA2020.cpp which does an opening operation followed by a closing operation. For this purpose we used OpenCV's open and close method [6] with a 4x4 kernel.

E. Improved Classification Model

Performance of classification depends mostly on the underlying model it employs. Hence we implement a snippet of MATLAB code which uses the dataset provided by the lab to generate a more accurate model. We found that the new model parameter demonstrates comparatively better results.

III. IMPLEMENTATION

This project has been mainly written on C++ with the OpenCV Library package. Following are the system specification for the development of this project
OS: Ubuntu 18.04.4
IDE: Eclipse
Language: C++
Library: OpenCV

A framework was provided to us which included 5 files: blob.cpp, blob.hpp, Lab2.0AVSA2020.cpp, ShowManyImages.cpp and ShowManyImages.hpp. We mainly worked with the blob.cpp, blob.hpp, Lab2.0AVSA2020.cpp files and kept unmodified ShowManyImages.cpp & ShowManyImages.hpp since their only functions was to display results in an external window. All of our methods were implemented in the blob.cpp class and blob.hpp contains the class definition of blob.cpp. To use the program, one must specify the dataset path first along with the category. Then the total number of category sequences and their names need to be mentioned.

First of all to implement blob detection we have used OpenCV's floodfill algorithm. In the OpenCV implementation of the algorithm it take foreground mask, seed point, new value that will be replaced in the connected region , address of a rectangular object to mark the connected region and an integer value which refers to the connectivity as the inputs of the algorithm. This method yields the rectangle where appropriate parameters are set for the connected region found by the algorithm. We chose to set the number of neighbors checked during the connected component analysis to 4: we have experimented both with 4 and 8. The results were quite similar so we decided to use 4 as in this case it checks less pixels thus less computationally demanding. Finally, the Rect class was transformed into cvBlob objects – a struct defined in the code that was given to us – and stored into the blob list. After that we have implemented a method to filter out the small blob (width > 20 and height > 20) which were created out of noise.

For classification, we were given models for 3 different classes: PERSON, CAR and OBJECT. We computed the Euclidean distance between the blob and the model for each blob and gave to the blob the label corresponding to the smallest value (so the closest to the model).

TABLE I. GENERATED CLASSIFICATION MODEL

Categories	Calculated		Tuned	
	Mean	Standard Deviation	Mean	Standard Deviation
Person01	0.5329	0.3032	0.5329	0.3032
Person02	0.3950	0.0863	0.4050	0.0303
Car	1	0	1.5	0.2000
Object	1.0367	0.2029	1.0888	0.2028

Next we have painted all the blob bounding boxes in different colors according to their labels, with a blob painting method which was provided to us. For example this method paints a bounding box of PERSON blob into blue color, a bounding box of CAR blob into green color etc.

In the next stage we have implemented a method for stationary foreground extraction based on the simplified version of the paper [3] by Ortego, D.and SanMiguel, J. C. First we had to deal with the fact that the background subtraction does not return a binary foreground mask but a grayscale one which does not only contain pixels with 0 or 255 as a value: some pixels with value 127 can be seen. Yet, our blob extraction algorithm only cares about true white pixels with a value of 255.We solved this problem by applying a threshold to the obtained foreground mask so that it becomes an actual binary mask. .Then we have computed a logical foreground mask and background mask from our foreground. Then we have created the FHI (foreground history image) by multiplying the foreground logical masks with a certain weight and adding them for each frame .On the other hand, we have multiplied the background logical mask with another weight and subtracted it from FHI for each frame. After that we have normalized and threshold the FHI to create the final stationary foreground image.

Further, we have developed a MATLAB script to load the dataset [7] and calculate the statistical mean and standard deviation for a simple Gaussian model. There are mainly three categories of objects namely CARS, OBJECT and PERSON. Some categories contained positive and negative folders while others have only positive folders. For uniformity, we considered only the positive folder instances for computation. Although the mean and standard deviation for different categories we have got from the dataset are not quite suitable in practise as we employ them in our experiment. So in order to make them more appropriate we had to hand tune them once again. Table. I show both original and tuned statistical mean and standard deviation value for different categories.

IV. DATA

In order to have an evaluation range as wide as it can be, we have tried to use videos with diverse contents and different angles of view but more importantly, with an acceptable background subtraction quality. To evaluate our blob extraction and classification – we were given some video sequences. There were 3 categories consisting of multiple video sequences. The category ETRI video sequences contains only PERSONS with fairly empty scenes. By evaluating our model on this sequence we can understand how our model is performing in detection of PERSON on a non crowded simple scene. On the PETS2006,there are 3 video sequences in total, in these sequences the viewing angle is different and scenes are mostly crowded with people and also some stationary foreground (PERSON , OBJECT etc.) . Through evaluating PETS2006 we can estimate how our model is performing in crowded scenes and temporary stationary objects. Next VISOR category focuses on vehicles mainly CAR and temporary foreground objects in the scene. One the main reason to evaluate these video sequences is to check how well our stationary foreground extraction is performing. However we are also expected to evaluate our model on the video sequence that is related to people and cars provided during LAB 1.All the video sequences are kept in minimal resolution due to the computational cost.

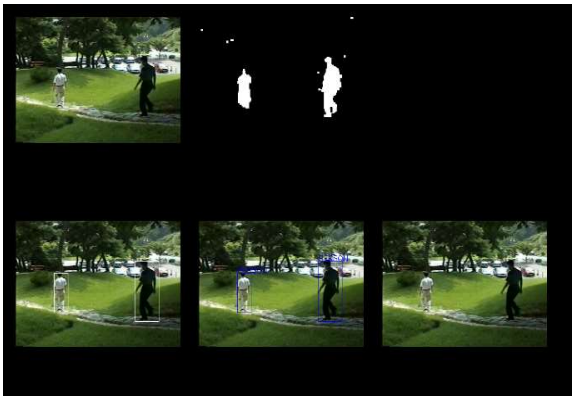
Regardless of the dataset the results' quality of foreground segmentation based approach like ours, is highly dependent on how well we are able to extract foreground.

V. RESULTS AND ANALYSIS

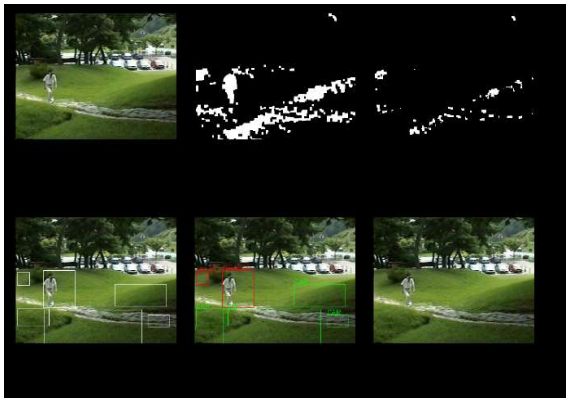
In our test video sequences most of the videos are about persons. So , we started evaluating our blob analysis with videos about persons. The figure 1 below shows blob analysis for two different frames of the same video (ETRI_od_A) and illustrates quite well the results we can get for a video with a satisfying background subtraction and only persons. In the bottom middle section of Fig. 1 (a), the extracted blobs (blue) are right and the blobs resulting from noise are filtered out . However, on Fig. 1(b), we can see a lot of noise in our foreground mask hence there are a lot of unwanted blobs detected. This is because of the learning rate of the Gaussian Mixture-based Background/ Foreground Segmentation that we are using for this experiment. At first we were using an adaptive learning rate which did not create the above mentioned problem.

Now provided the results seen on Fig. 1, we can say that we have a working model for blob extraction and a correct classification to an acceptable level when the foreground segmentation is neat, simple and a proper learning rate. The next result in Fig. 2 has been obtained from PETS2006_S1_C3 video sequence which is about persons with some object with them and crowded scenario with many people. In Fig. 2 (a) it can be seen that two persons are detected which are correct. However in Fig. 2 (b) we can see that a person is dragging a trolley-like object along with him which is detected as the whole object in the foreground segmentation mask hence also detected as a whole big blob. As a result in the classification it is detected as a car instead of a person and an object individually. This can be improved by more advanced segmentation technique or post foreground segmentation processing. In Fig. 2(c) some people are passing together in groups. Our model has falsely detected it as people although that blob should have been detected as an UNKNOWN since we do not have a model for the group in our experiment .The blob extraction is unstable, depending on the background subtraction.

Although we are not really caring about the classifications of the parasitic blobs, it is interesting to notice that depending on their shapes, the blobs' attributed classes correspond to the ones we would have given them by only looking at the aspect ratio (PERSON for blobs with a height superior to their width, CAR for those with an aspect ratio closer to 1 and slightly superior to it, etc.).

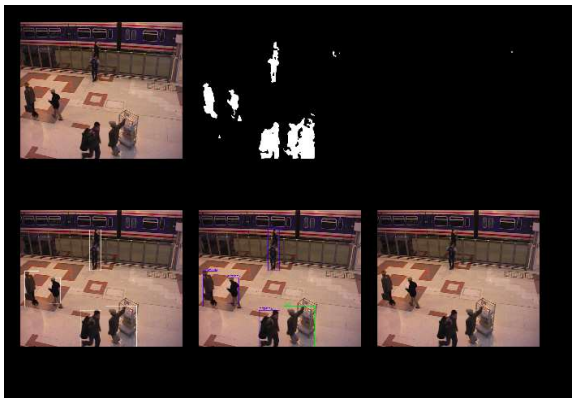


(a) Snapshot 01 from ETRI

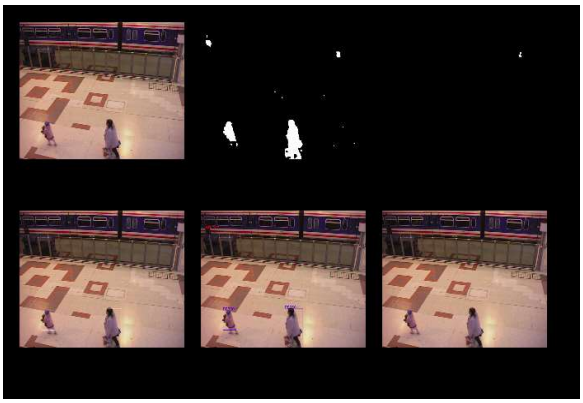


(b) Snapshot 02 from ETRI

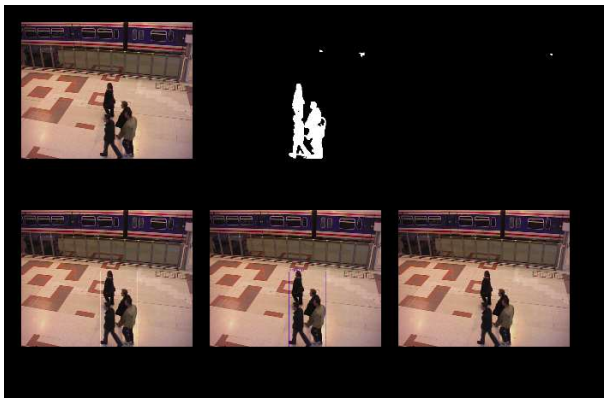
Fig. 1 Results of the blob analysis. Top-left: original frame; top-middle: foreground mask obtained with background subtraction; top-left: stationary foreground mask; bottom-right: detected blobs; bottom-middle: Classified blob bottom-right: classified blob based on stationary foreground.



(a) Snapshot 01 from PETS2006_S1_C3

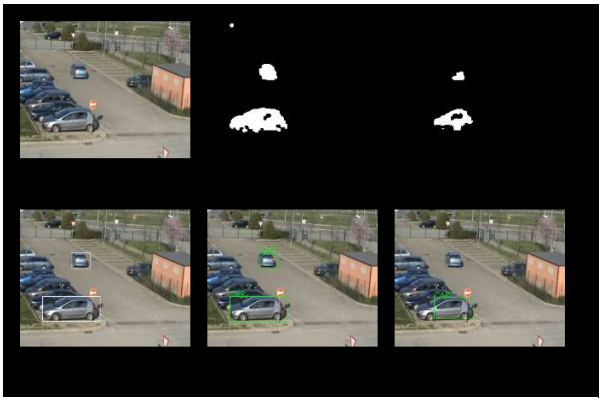


(b) Snapshot 02 from PETS2006_S1_C3

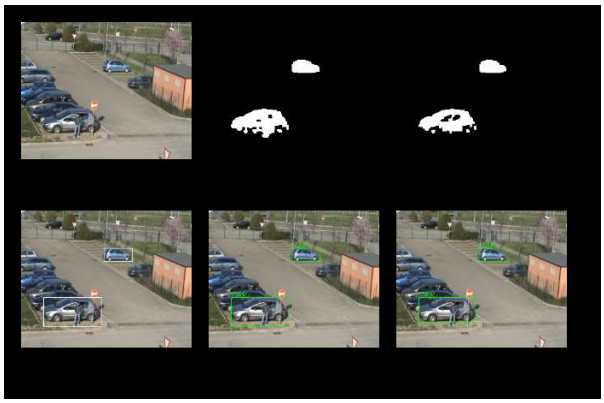


(c) Snapshot 03 from PETS2006_S4_C3

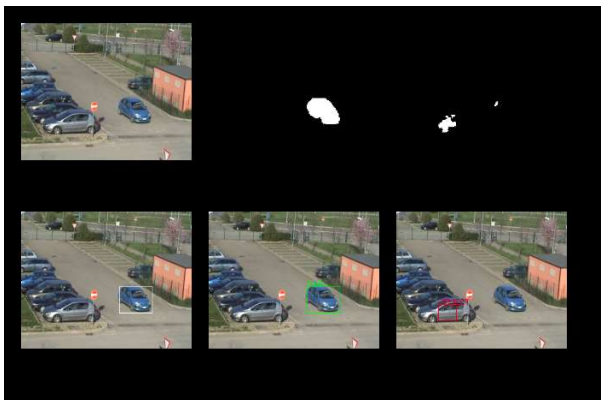
Fig. 2 Blob analysis of video sequence PETS2006_S1_C3 & PETS2006_S4_C3. Top-left: original frame; top-middle: foreground mask obtained with background subtraction; top-left: stationary foreground mask; bottom-right: detected blobs; bottom-middle: Classified blob; bottom-right: classified blob based on stationary foreground.



(a) Snapshot 01 from PETS2006_S1_C3



(b) Snapshot 02 from PETS2006_S1_C3



(c) Snapshot 03 from PETS2006_S1_C3

Fig. 3 Blob analysis of video sequence *visor_Video00*. Top-left: original frame; top-middle: foreground mask obtained with background subtraction; top-left: stationary foreground mask; bottom-right: detected blobs; bottom-middle: Classified blob; bottom-right: classified blob based on stationary foreground.

In Fig. 3 we can see the result of our stationary foreground extraction.as soon as both of the cars appears in the scene and stays in a certain position for a certain time , the object in this case, cars are start to incorporate into the Stationary foreground mask (In Figure 3(a): Top-middle portion we can see the mask and bottom-middle portion original scene with detected blob bounding boxes). After a given time all the temporary and permanent stationary objects in the scene are fully incorporated into the stationary foreground mask. (In Fig. 3(b): Top-right portion we can see the stationary foreground mask of both the cars in the scene). Although at the rate the stationary foreground mask adapts to our current scene depends on a few parameters such as two weights that are associated with the foreground and inverse foreground of our current scene and with the thresholding parameter (η). Next one of the cars which is at near end of the scene, stays still permanently in the scene but other one at far end of the scene starts to move again after a certain time. Hence as the time progresses the car at near end disappears from our foreground mask as our background subtraction model updates and our stationary foreground mask also updates.

(We can see at Fig. 3(c): Top-right portion we can see both the cars are disappearing from the stationary foreground mask). So we can conclude that our implementation of stationary foreground is working and we can keep track the objects that are not mobile for a short period of time in our scene by this approach. From the results and analysis of those results we roughly say that our method is working but there is a lot of scope to improve the current limitation such as eliminating unwanted blobs aroused due to learning rate, using improved background subtraction approach and using more improved models for blob classification. We can also improve our stationary foreground extraction method by implementing the Motion analysis mentioned in the paper [3].

VI. Conclusions

In conclusion, performance of blob extraction and blob classification are interlinked and dependent on the foreground segmentation scheme. Applying morphological operations and removal of insignificant blobs also improve the quality of the extracted blobs which in turn provide enhanced

classification. Detection of stationary foreground objects is highly sensitive to the proper parameterization. Further, more classification models can be incorporated in the routine to extend the work. Future work can also include exploring different classifiers and comparing their responses on several surveillance conditions.

VII. Time Log

- **Blob Extraction** 4 hours: Understanding the framework provided by the lab took 1 hour, 30 minutes to design the algorithm, 30 minutes to implement in the code and another 2 hours to solve type errors in OpenCV and finding the optimal parameters.
- **Blob Classification** 2 hours: 1 hour to understand and implement the routine and 1 hour in debugging the method.
- **Stationary Foreground Detection** 5 hours: 2 hours understanding the research paper, 30 minutes to design the program flow, 30 minutes to write the algorithm and 2 hours for debugging and finding the appropriate thresholds.

- **Foreground mask refinement** 0.5 hours: To understand the problem type and implement the methods to refine the foreground mask.
- **Improved Classification Model** 2 hours: 45 minutes to figure out the dataset provided by the lab, 30 minutes to design the program flow and 45 minutes to implement in MATLAB.
- **Evaluation** 5 hours: Experimenting with different combinations of parameters to find the optimal set of parameters best suited for the overall dataset.
- **Report** 8 hours: writing, taking screenshots, proof-reading and editing.

References

[1] Zoran Zivkovic and Ferdinand van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction". *Pattern recognition letters*, 27(7):773–780, 2006.

[2] Zoran Zivkovic, "Improved adaptive gaussian mixture model for background subtraction". In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[3] D. Ortego and J. C. SanMiguel, "Stationary foreground detection for video-surveillance based on foreground and motion history images," *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, Krakow, 2013, pp. 75-80.

[4] Thomas B. Moeslund, "Introduction to Image and Video Processing", *Article-7.1.2: The Sequential Grass-Fire Algorithm*

[5] https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=floodfill accessed at 3-Apr-20

[6] https://docs.opencv.org/3.4/d3/dbe/tutorial_opening_closing_hats.html accessed at 3-Apr-20

[7] https://posgrado.uam.es/course/view.php?id=39208 accessed at 3-Apr-20