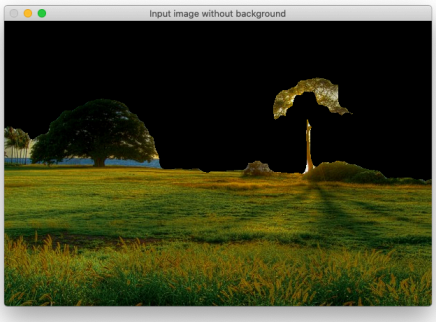
AUTOMATIC TREE DETECTION

APPROACH

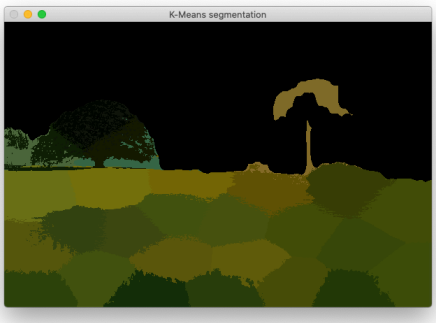
The automatic tree detection on images with several trees (at various scales) and with also grass meadows, I think is not a really easy task. This is caused by the presence of trees with different color, from dark green to a shade of yellow. For this reason, I based my detection algorithm on a first step of color thresholding, to highlight only green and dark yellow regions, and then on a features detection using a set of training images with trees on white background.

ALGORITHM

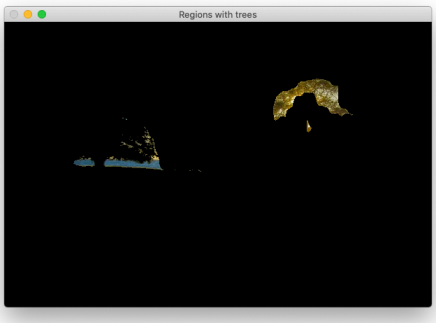
1. Segment the image with K-Means based only on HSV color space.



2. Filter out with a threshold the cluster with no green/ yellow color.

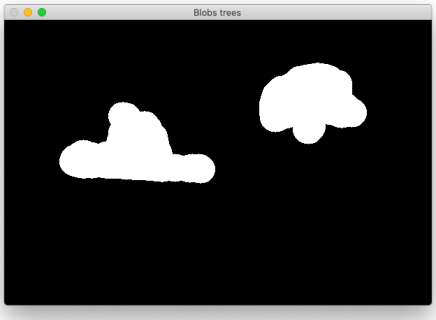


3. Segment the image without background using again K-Means, but emphasising also the distance between points.

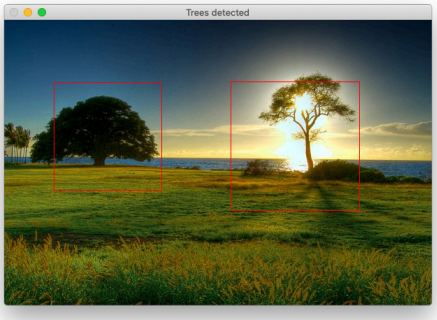


4. Filter out the regions with less key points matched w.r.t. the mean number of key points matched of the regions (there is also a threshold to filter out the regions of image without trees).

GIANMARCO BORTOLAMI - 1239293



5. Binarize the image and apply a dilatation operator to merge together neighbour regions.



6. Use Simple Blob Detector to detect the blobs and compute centers and areas. Apply a threshold to filter out blob with small area. Then draw a rectangle over the blobs remain.

PERFORMANCE LEVEL

The algorithm obtain quite good performance over all images detecting the most relevant trees and rejecting well the grass meadows, small bushes and brown grounds. Sometimes is tricked by stones, cut trunks or near trees. In this last case several times the algorithm is not able to distinguish them, like in Figure 10.

ANALYSIS OF CRITICAL CASES

First of all I want to highlight an issue, probably, arise with K-Means. If we run the program several times we can see that the rectangle are not always in the same position and sometimes some secondary trees are not well detect, instead other times they are well detected. In my opinion, based my experience on the Big Data Computing course, the K-Means algorithm implemented in OpenCV is an Approximation K-Means to achieve good performance. So, for this reason, the images are cluster in different ways each time we run the program, then we obtain different results. Another issue arise on Figure 3, here the initial thresholding is not able to filter out the regions with cut trunks and stone, due to their colors, and ORB detect a lot of keypoints on the stone. That keypoints will be matched with the training set detecting a tree. Also in Figure 12 we can see that the algorithm is tricked. In this case, the program matches the keypoints on the second tree on the right and also on the trees in the background. This leads to identify a single blob locates in the middle of the right trees and background trees, because all these trees are overlapped and the algorithm isn't able to distinguish them.

HOW TO RUN THE PROGRAM?

On the folder `data` there will be the folder `Benchmark_step_1` where there are (and we can add without any troubles) the set of images to detect the trees, while the folder `Training_images` where is stored the set of image to perform the feature of the trees. From the terminal: make the `build` directory, enter on this folder, launch `cmake ..` command, then launch `make` command. Now the executable file `main` will be created and ready to be executed on the `build` folder.

GIANMARCO BORTOLAMI - 1239293