# Computer Vision

*Assignment 4 Report*
*By Chris Jimenez*

---

## *I - Boosted Object Detector*

For this part of the assignment, the boosting-based object recognition system was walked through using the provided file `hw4_qu1.zip`. After the the path in parameters.m and `initpath.m` were modified, a demonstration of the system was performed using the demoGentleBoost script. The script ran for 20 rounds and the results can be seen below.

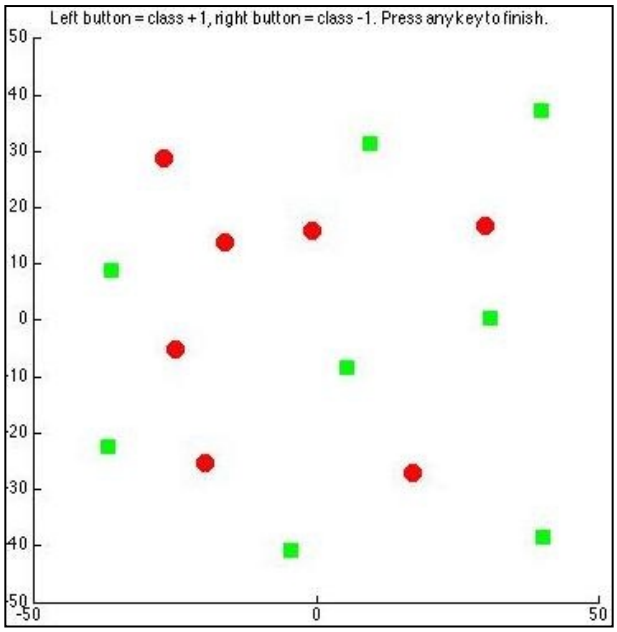


*Figure 1: Points picked when running demoGentleBoost.m;*

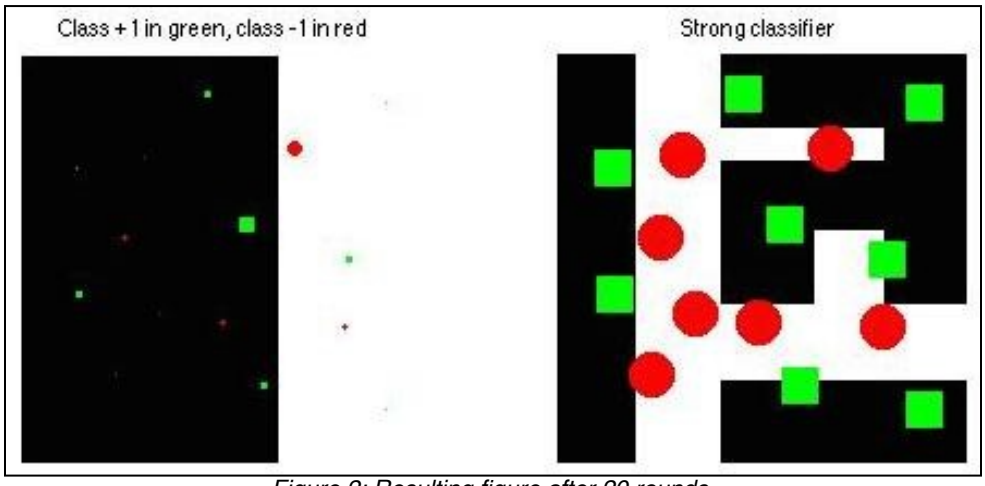


*Figure 2: Resulting figure after 20 rounds.*

Using the same boosting framework, an object detector is trained. Firstly, a dictionary of features is created. Then the features are computed. The detector is then trained and tested. This was done with 5, 30 and 100 weak classifiers. The resulting plots can be seen below in Figure 3 and Figure 4.

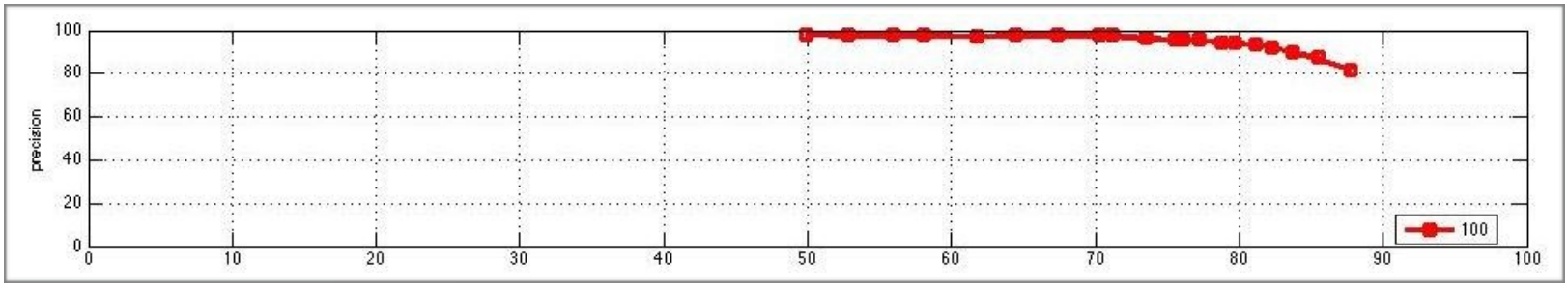


*Figure 3: Resulting plot using 100 weak classifiers.*

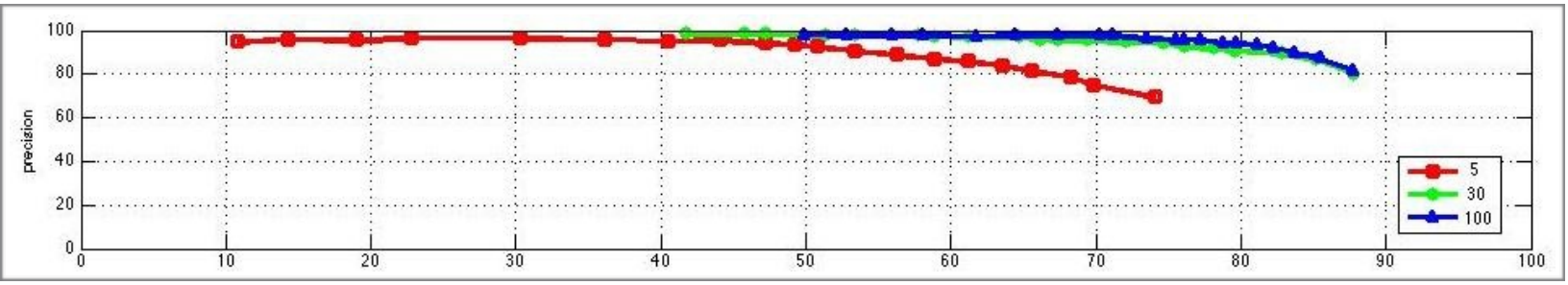


*Figure 4: Resulting plot using 5, 30 and 100 weak classifiers.*

The weights for the first four boosting rounds are shown below in Figure 5. As the number of rounds(iterations) increases, the overall weight per iteration decreases. But the points with the largest weight change. The matrix of the weak classifier predictions can be seen below in Figure 6. The gentleBoost.m function was then modified to use a random value of k. Under the “Written Answers” of this report, problem 6 covers the reason behind the use of random k. The weights for the first four boosting rounds for this random k can be seen below in Figure 7, and the weak classifier predictions matrix can be seen in Figure 8.

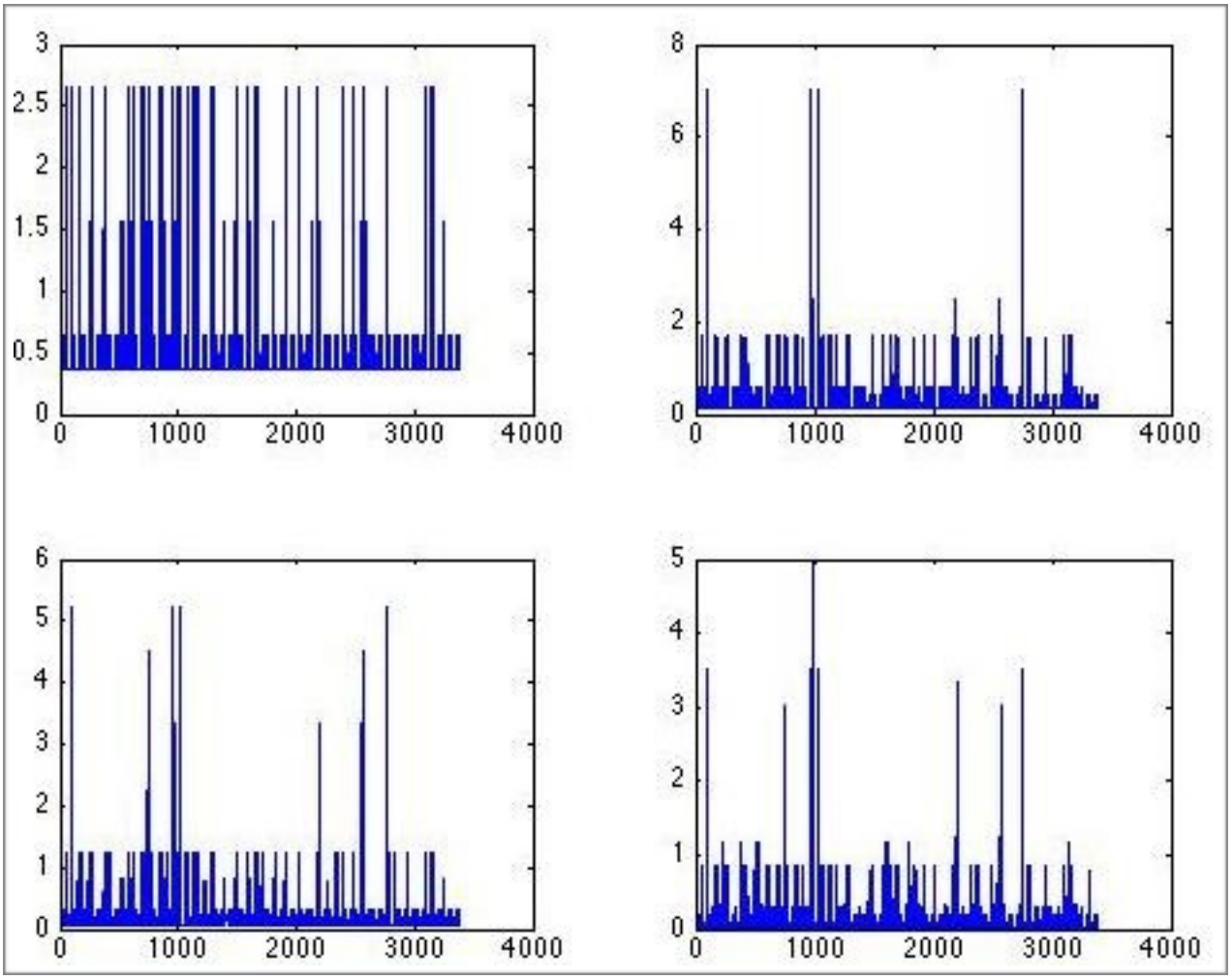


*Figure 5: The plots of the weights of the first four boosting rounds.*

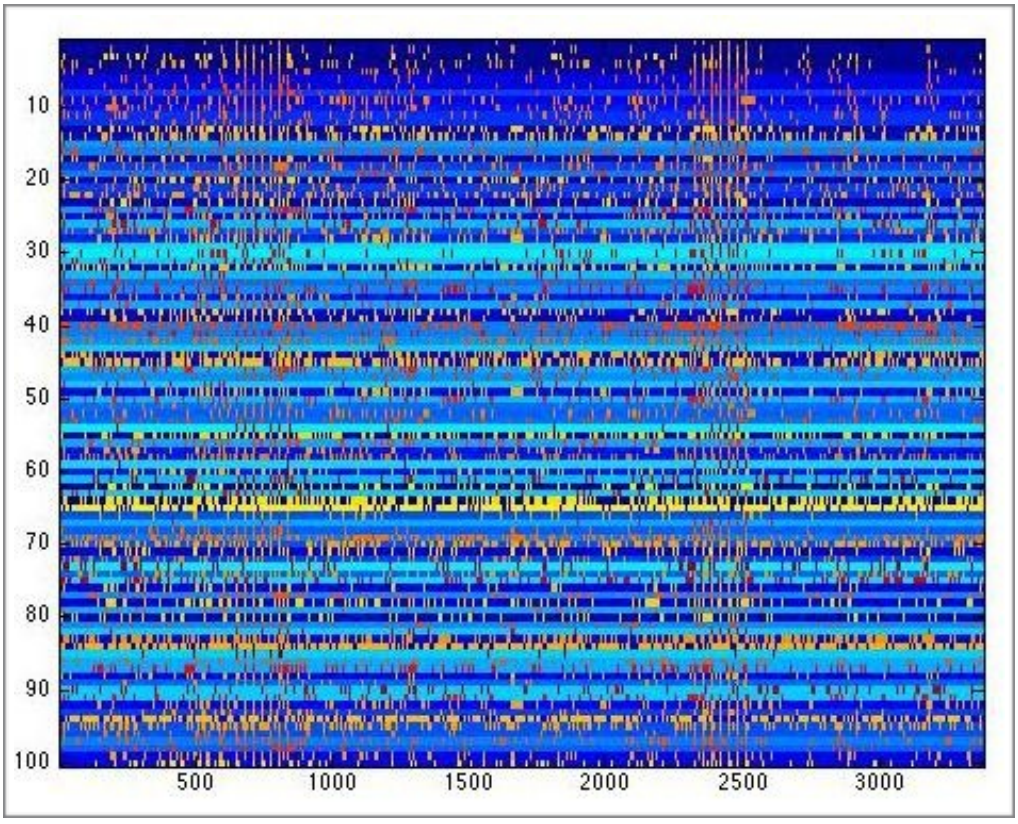


*Figure 6 : The matrix of weak classifier predictions.*

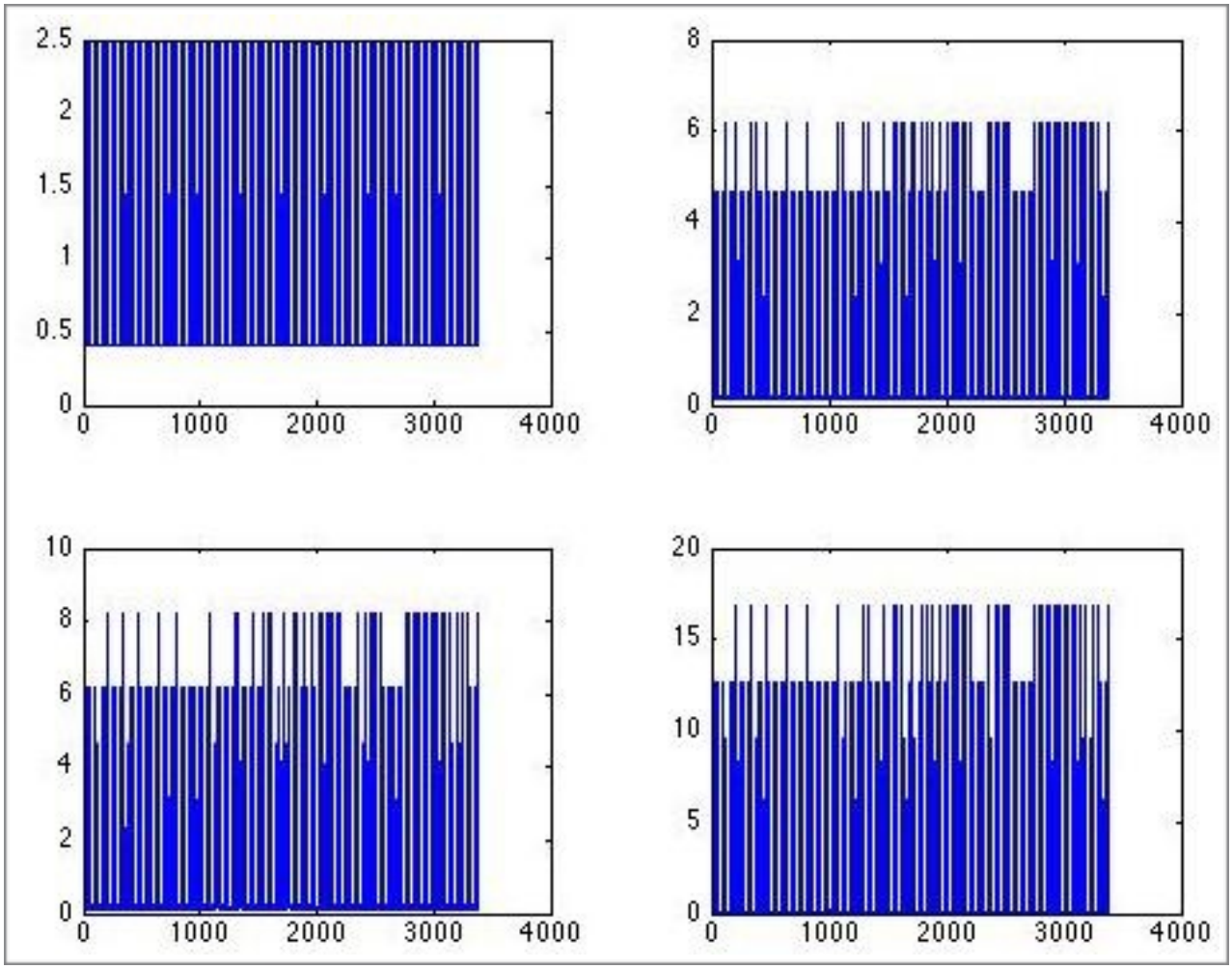


*Figure 7: The plots of the weights of the first four boosting rounds with random k.*

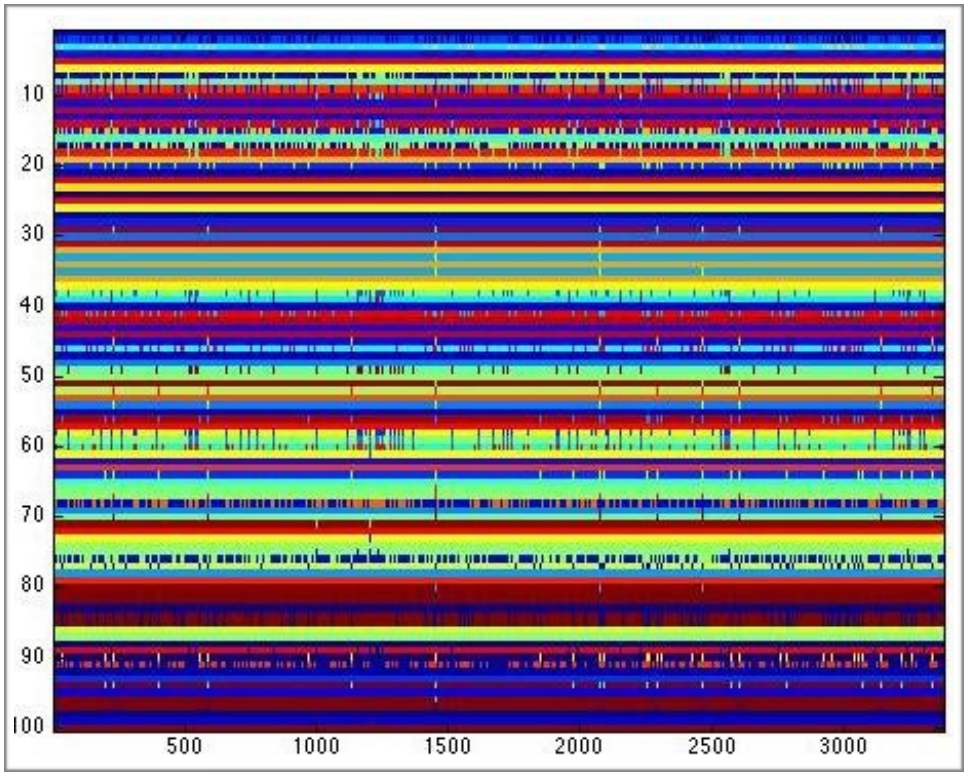


*Figure 8: The matrix of weak classifier predictions with random k.*

## Written Answers

6. In Figure 5, k was calculated using the `selectBestRegressionStump` sub-routine. In the next part of the assignment, k was randomly chosen to be 6(which can be seen in the sub-routine). This was done to show that the classifier does better than if k was calculated by chance, but still not as well if k was optimally chosen. This is because changing k, thus changing the stump parameters, doesn't show wide variations between the different features, but instead they are roughly close across all dimensions. In other words, when the parameters change, the classifier it doesn't compromise the classifier.

7. In the `selectBestRegressionStump` sub-routine, the most optimal k and stump parameters are calculated. The strong classifier may use multiple weak classifiers that utilize the same k, each having different stump parameters. This is because what may be optimal for one dimension for one round of boosting might not be optimal for another round because the weights change. This is why the strong classifier may need to go back to the same dimension.

8.

Ultimately, the strong classifier is a combination of weak classifiers. The final weighting $w_i$ for each point i is equal to $e^{-y_i F(x_i)}$ because when boosting, we minimize the exponential loss :

$$J = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Where N is the number of rounds. Therefore for each i from 1:N, the weight or partial exponential loss is:

$$e^{-y_i F(x_i)}$$

# *II - Mean Shift Segmentation*

For this part of the assignment the Mean-Shift Segmentation algorithm. which is an algorithm that decomposes an image in to a set of regions, was implemented and used. The algorithm was implemented using the given file `example_mean_shift.m`. The implementation was applied to the given image with a window size of 0.5. The resulting imaging can be seen below. Followed by several modifications of the `window_size` and `cluster_quantization` parameters.

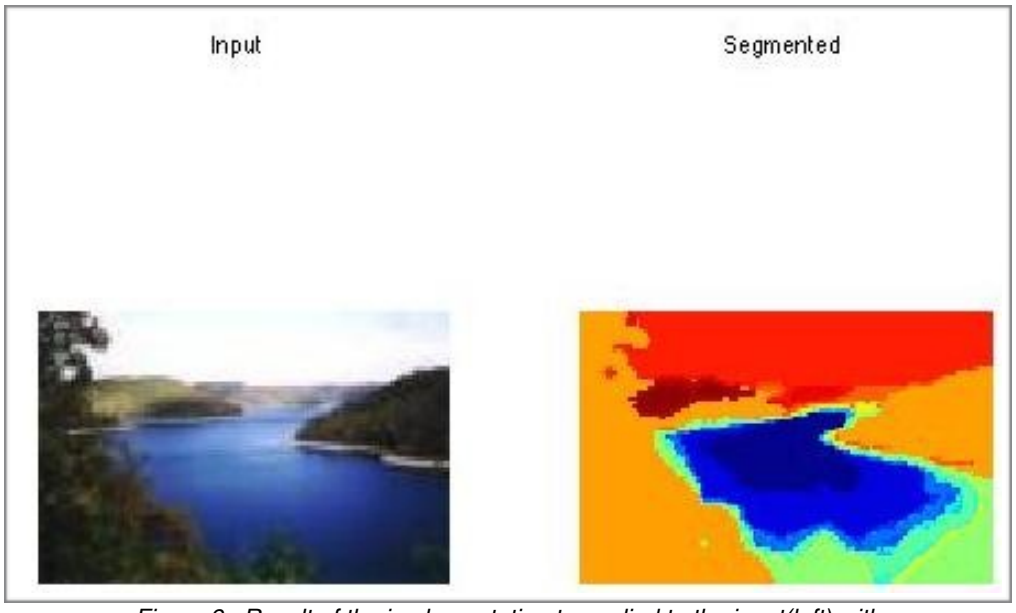


*Figure 9 : Result of the implementation to applied to the input(left) with window_size = 0.5, cluster quantization = 5*

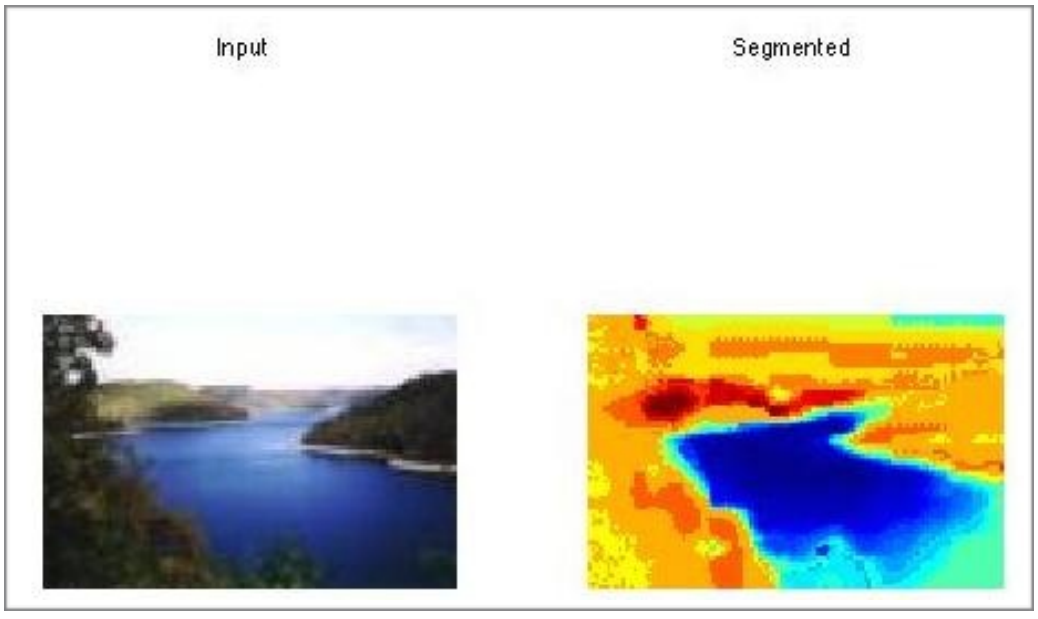


*Figure 10 : Result of the implementation to applied to the input(left) with window_size = 0.25, cluster quantization = 2*

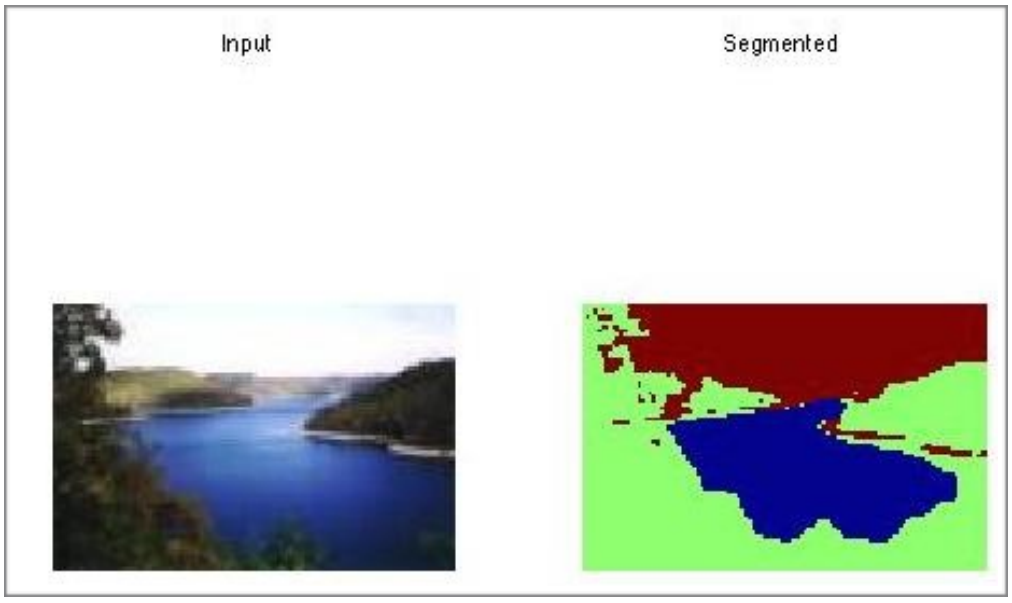


*Figure 11: Result of the implementation to applied to the input(left) with window_size = 2, cluster quantization = 10*

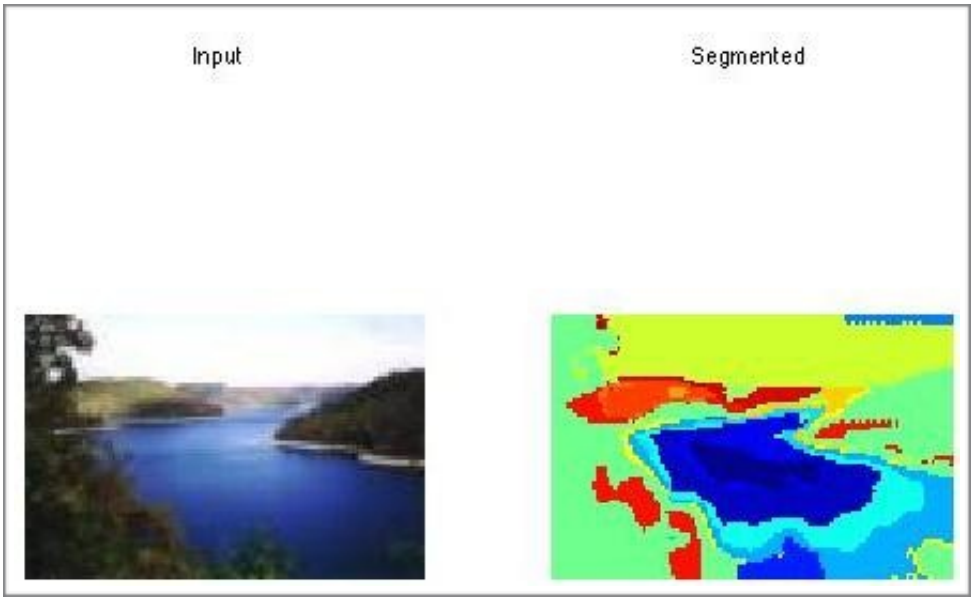


*Figure 12 : Result of the implementation to applied to the input(left) with window_size = 0.25, cluster quantization = 1*