Faculty of Engineering
Cairo University

Computer Vision

Task 4

Thresholding and Segmentation

Submitted by:

BN	Section	Name
10	1	Esraa Mohamed Saeed
12	1	Alaa Tarek Samir
15	1	Amira Gamal Mohamed
8	2	Fatma Hussain Wageeh
26	2	Mariam Mohamed Osama

Team 3

Submitted to:

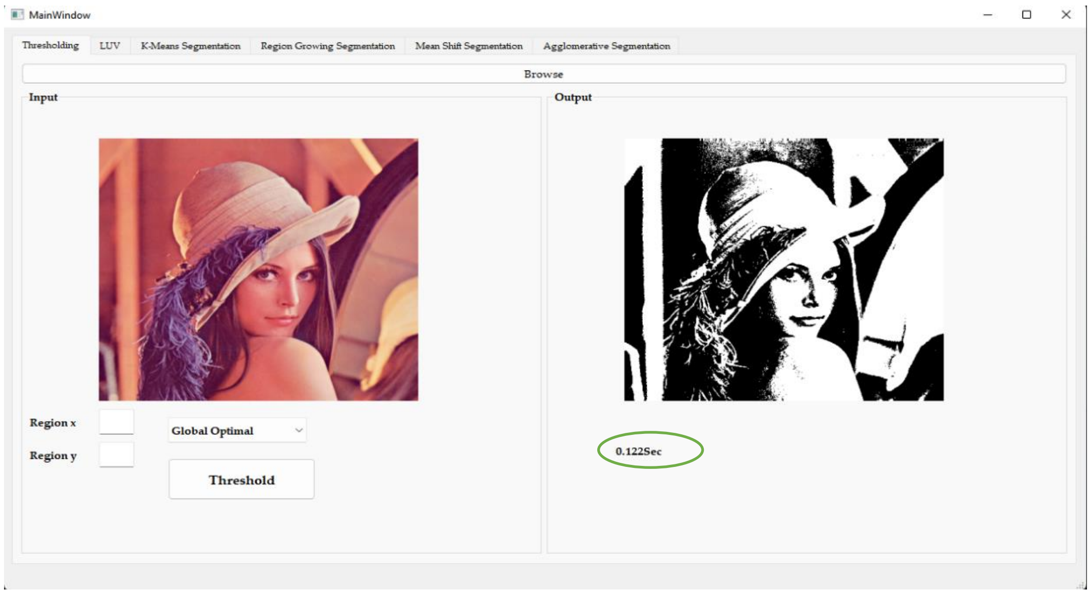
Eng. Laila Abbas

Contact Email: amira.gamal.ag22@gmail.com

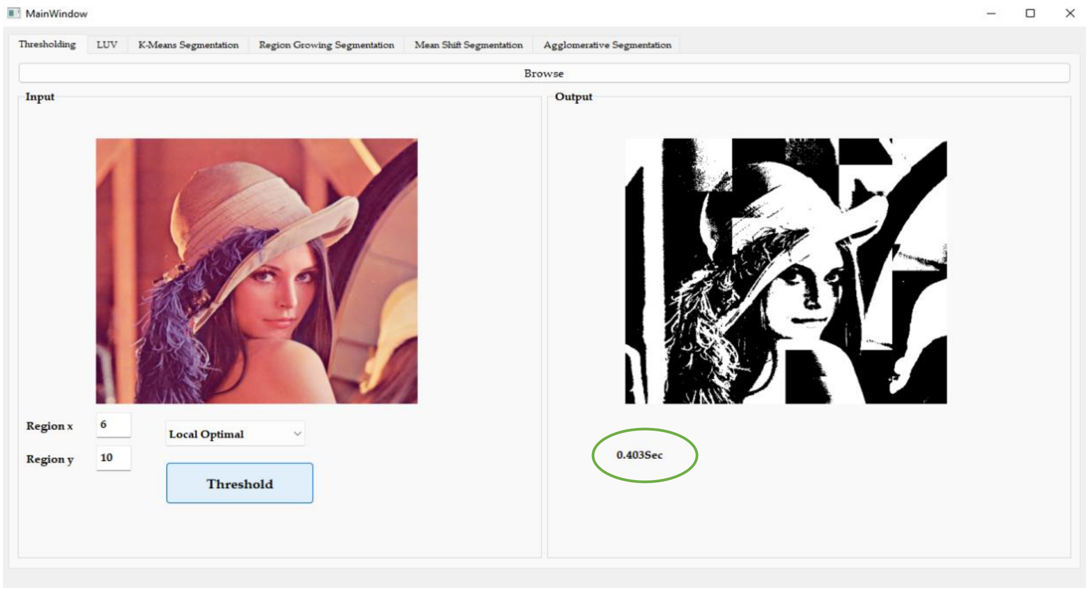
Thresholding

1- Optimal Thresholding:

To calculate optimal threshold we first calulate the initial threshold between background and foreground, and iterating to calculate the optimal threshold given the initial one.

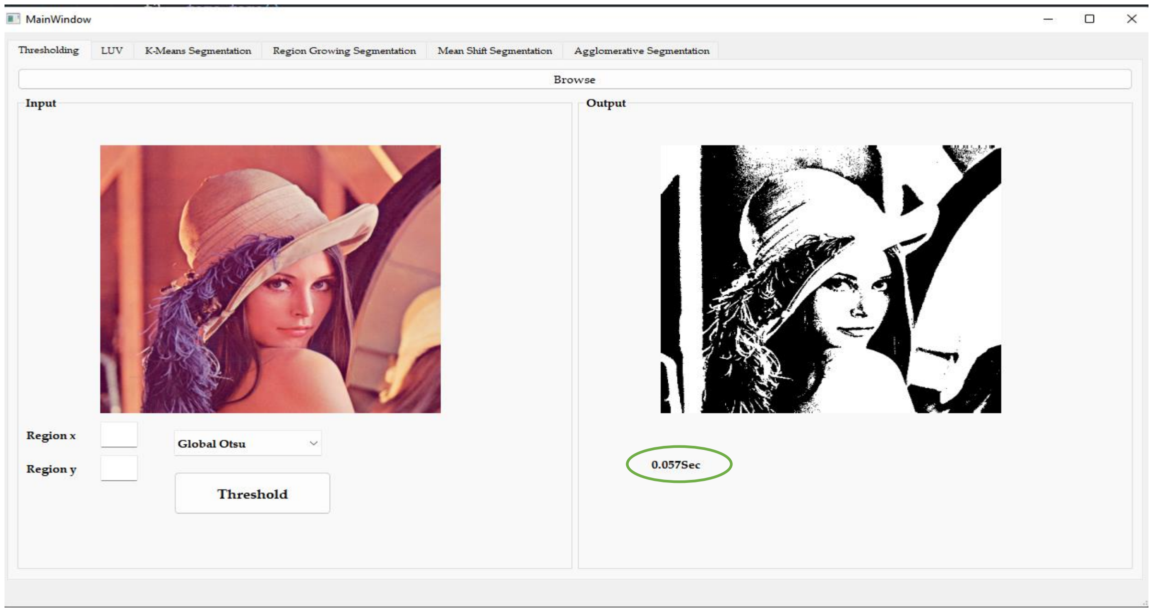


And for local thresholding we divide the image into regions and the number of regions depends on region_x and region_y inserted by the user and apply the global optimal thresholding for each region.

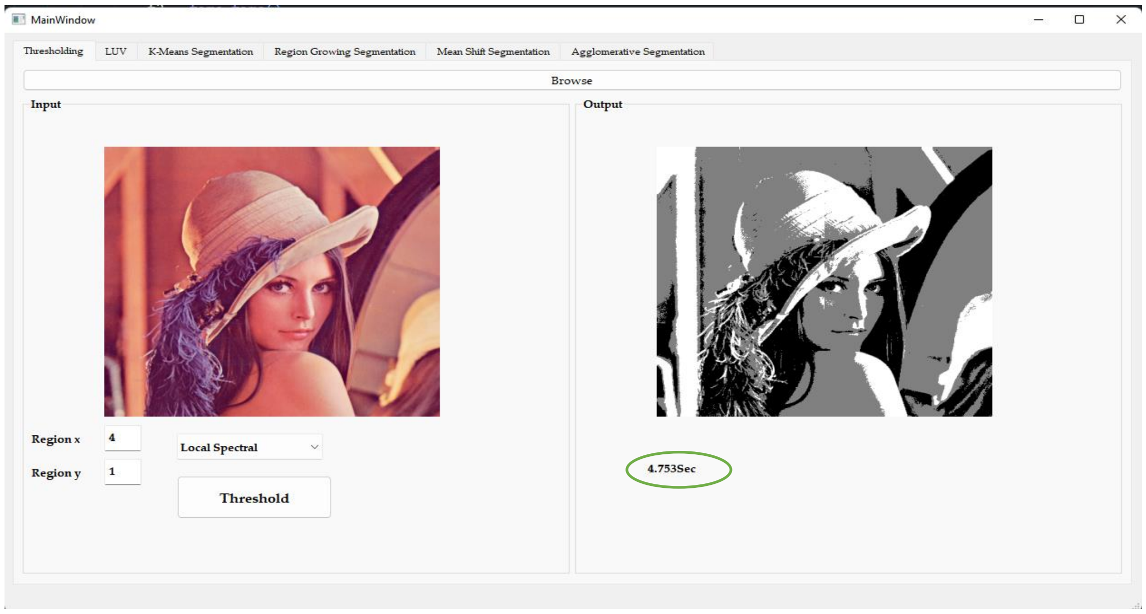


2- Otsu Thresholding:

We need to compute histogram, PDF, and CDF of each intensity level, Then loop over all possible thresholds, Calculate the variance between background and foreground for each threshold and select the threshold which have the maximum variance.



And for local thresholding we did the same for optimal thresholding.



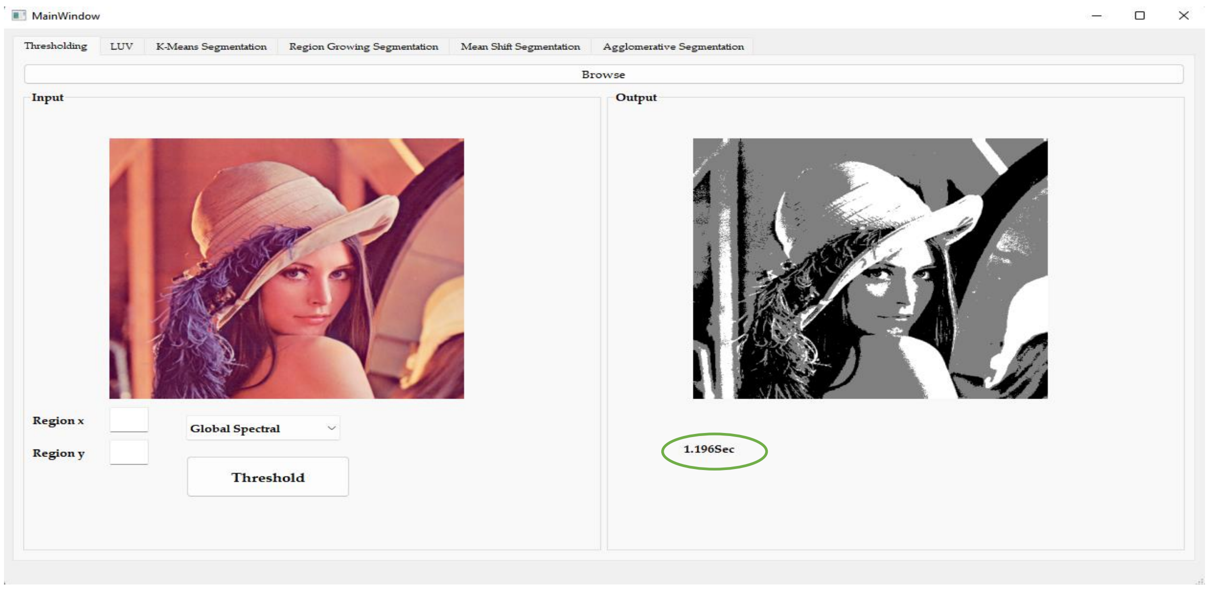
Region x: 4

Region y: 1

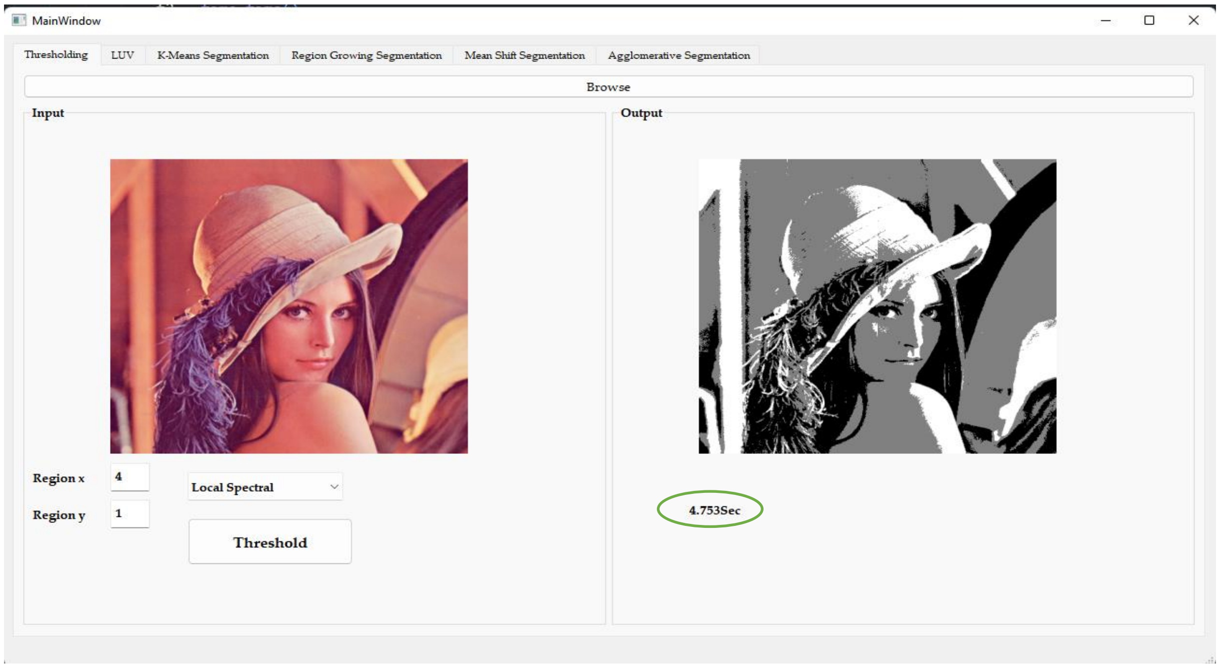
Local Spectral

3-Spectral Thresholding:

It looks like otsu thresholding but it uses double thresholding so we need to calculate high and low threshold, So we calculated two CDF one for high and the other for low intensties, we calculate the variance using both and choose those which give us the maximum variance.

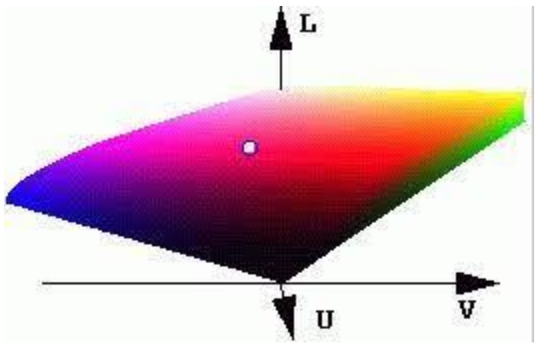


And for local thresholding we did the same for optimal and otsu thresholding.



Mapping from RGB to LUV

There are two reasons we might use non-RGB colorspaces, including LUV, in computer vision. The first reason is that differences in RGB space do not correspond well to perceived differences in color. That is, two colors can be close in RGB space but appear very different to humans and vice versa. The second reason is that spaces like LUV decouple the "color" (chromaticity, the UV part) and "lightness" (luminance, the L part) of color. In object detection, it is common to match objects just based on the UV part, which gives invariance to changes in lighting condition.



The transformation is done by three steps:

1. Convert RGB space to XYZ space by:

$RGB_{Rec709} \rightarrow XYZ$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412452 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

where RGB is rescaled to $[0\ldots1]$. After transformation, XYZ is scaled to $[0\ldots100]$.

2. Convert XYZ space to LUV space.

$$L \leftarrow \begin{cases} 116 * Y^{1/3} - 16 & \text{for } Y > 0.008856 \\ 903.3Y & \text{for } Y \leq 0.008856 \end{cases}$$

$$u' \leftarrow 4 * X/(X + 15 * Y + 3Z)$$

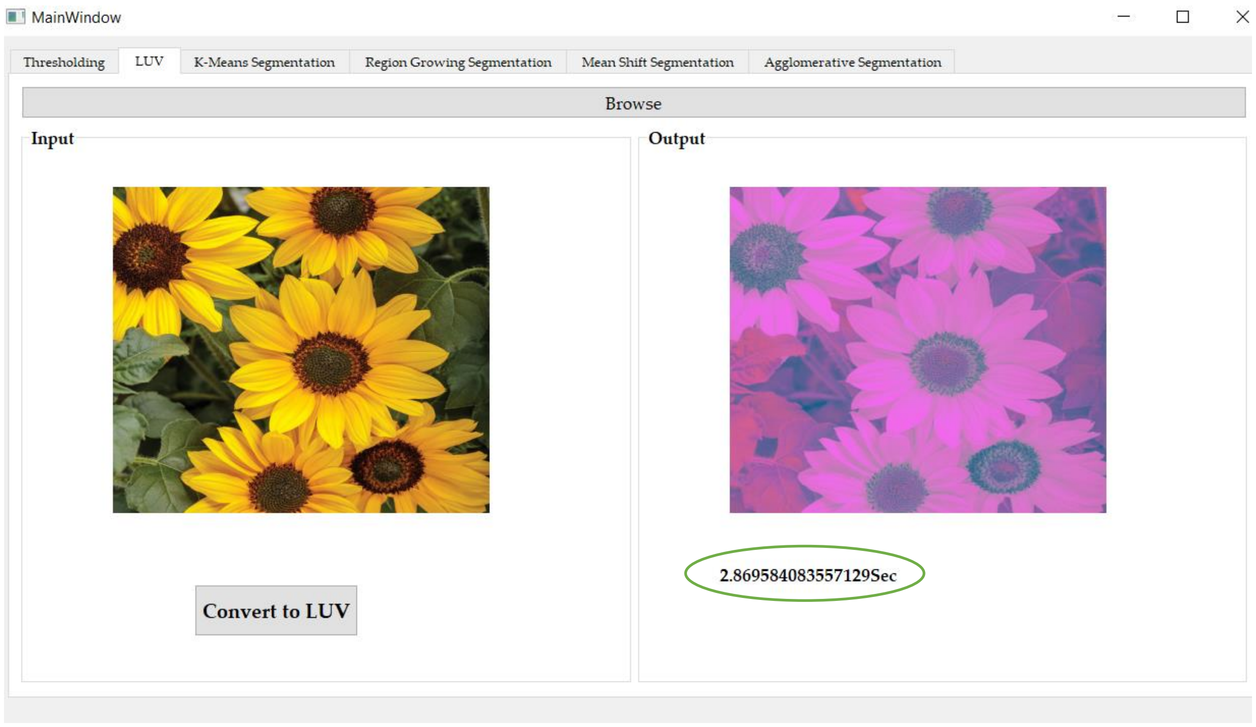
$$v' \leftarrow 9 * Y/(X + 15 * Y + 3Z)$$

$$u \leftarrow 13 * L * (u' - u_n) \quad \text{where} \quad u_n = 0.19793943$$

$$v \leftarrow 13 * L * (v' - v_n) \quad \text{where} \quad v_n = 0.46831096$$

3. Finally, the values will be converted to the destination datatype.

Results and Computation time

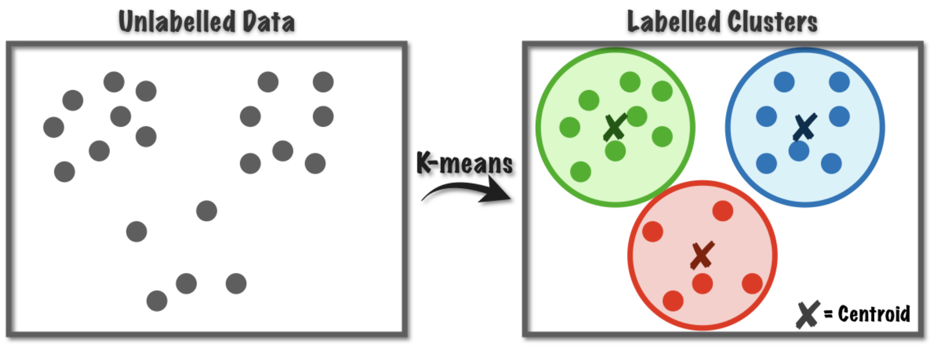


2.869584083557129Sec

Unsupervised Segmentation

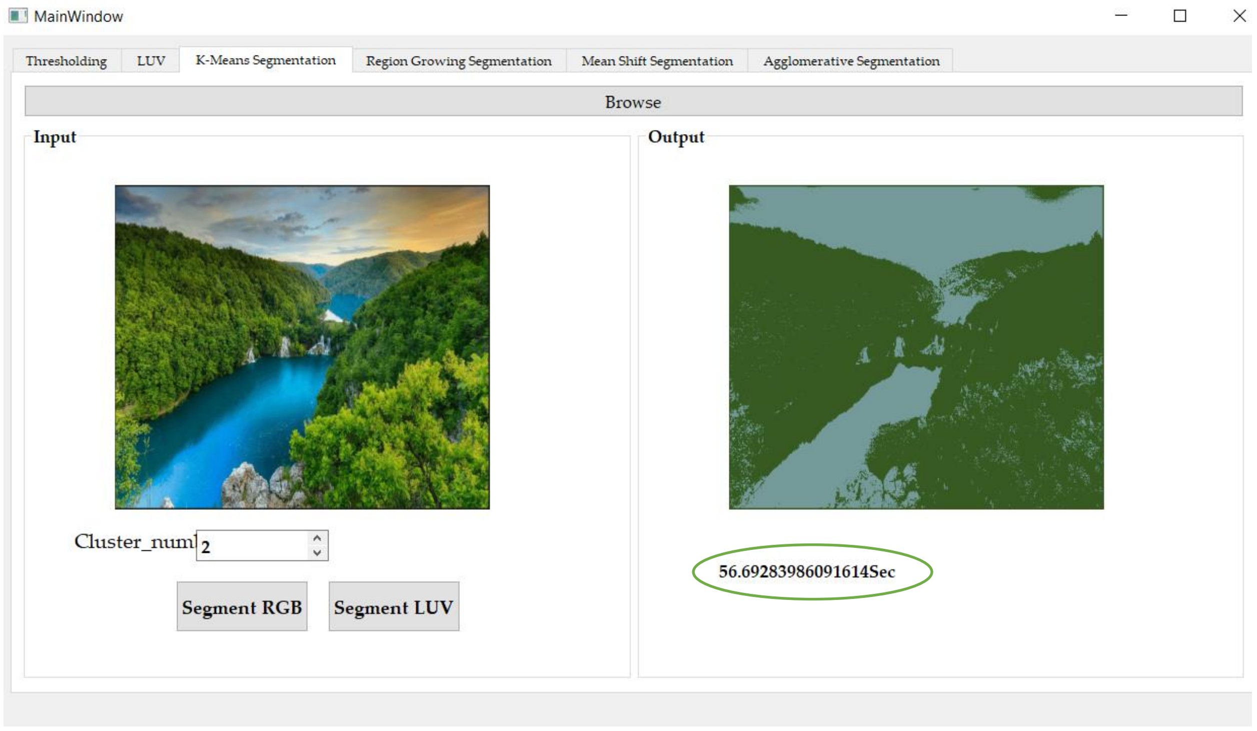
1. K-means

K-means is one of the simplest unsupervised learning algorithms. The algorithm follows a simple and easy way to group a given data set into a certain number of coherent subsets called as clusters.
The idea is to find K centers, called as cluster centroids, one for each cluster, hence the name K-means clustering.
The algorithm reports the closest cluster to which the point belongs based on a distance measure such as Euclidean distance.
It is easy to implement, simple and fast.
It is sensitive to cluster numbers, outliers, and initial points.



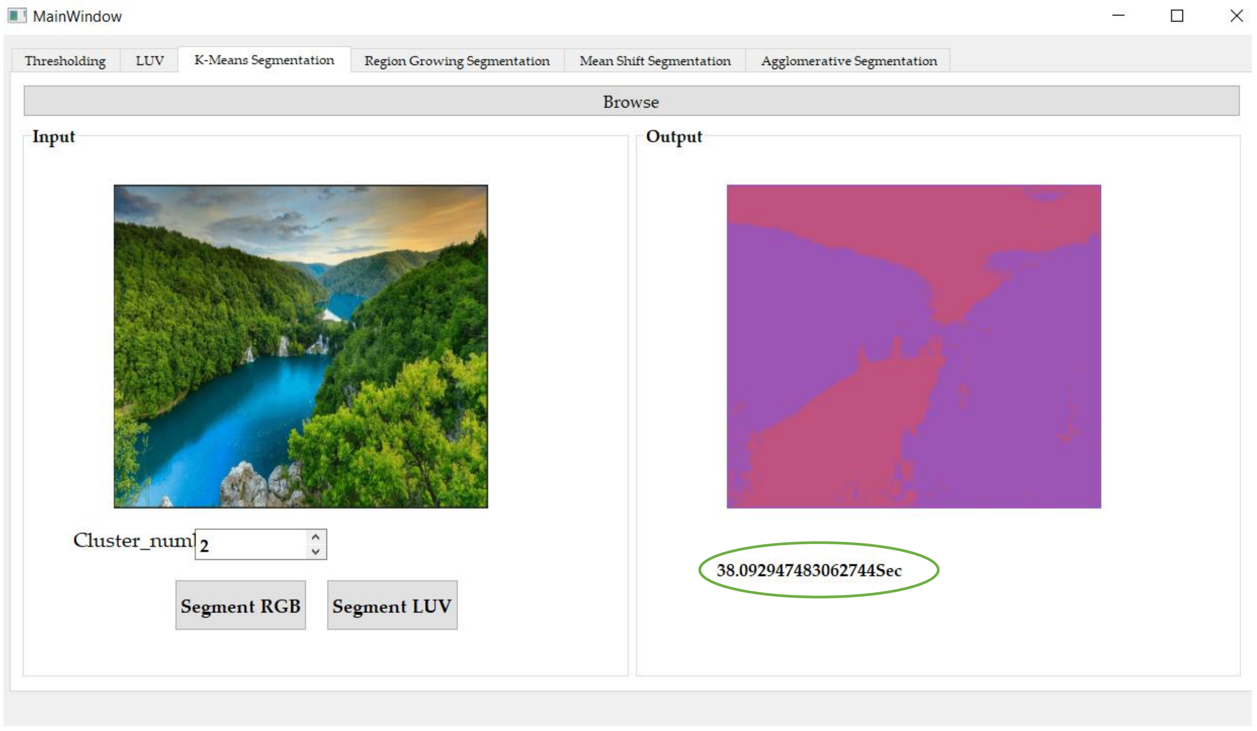
Results and Computation time

Clusters=2 for rgb and luv spaces



Cluster_num 2

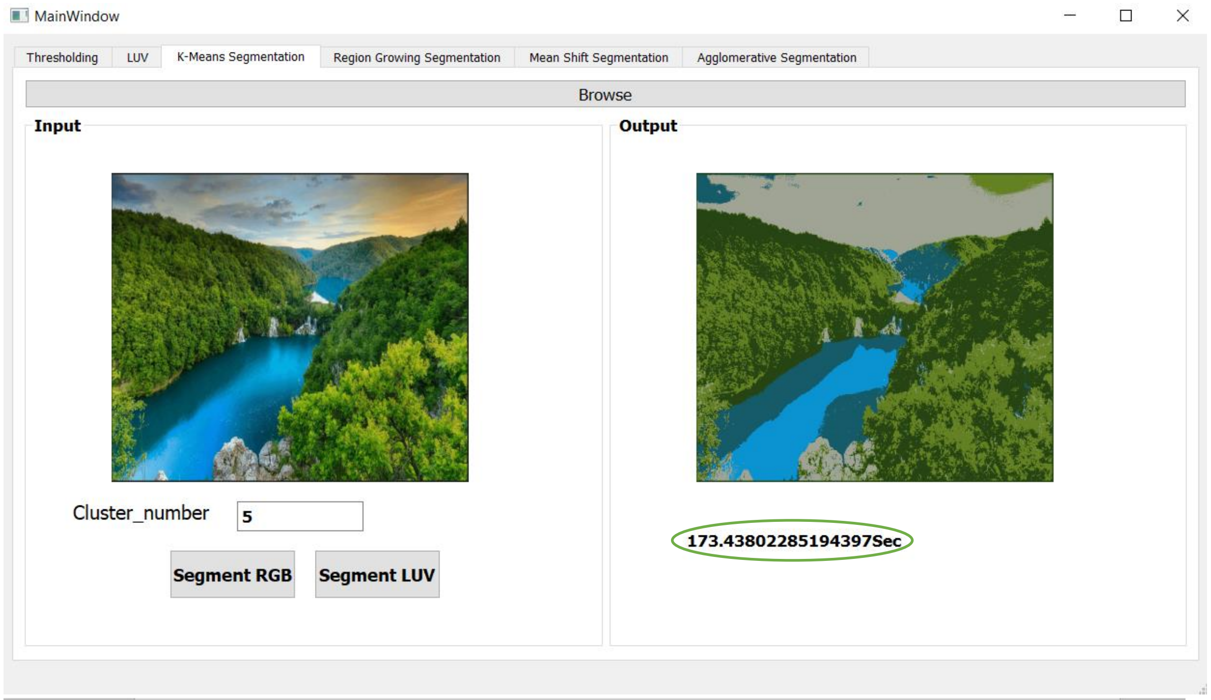
56.69283986091614Sec



Cluster_num 2

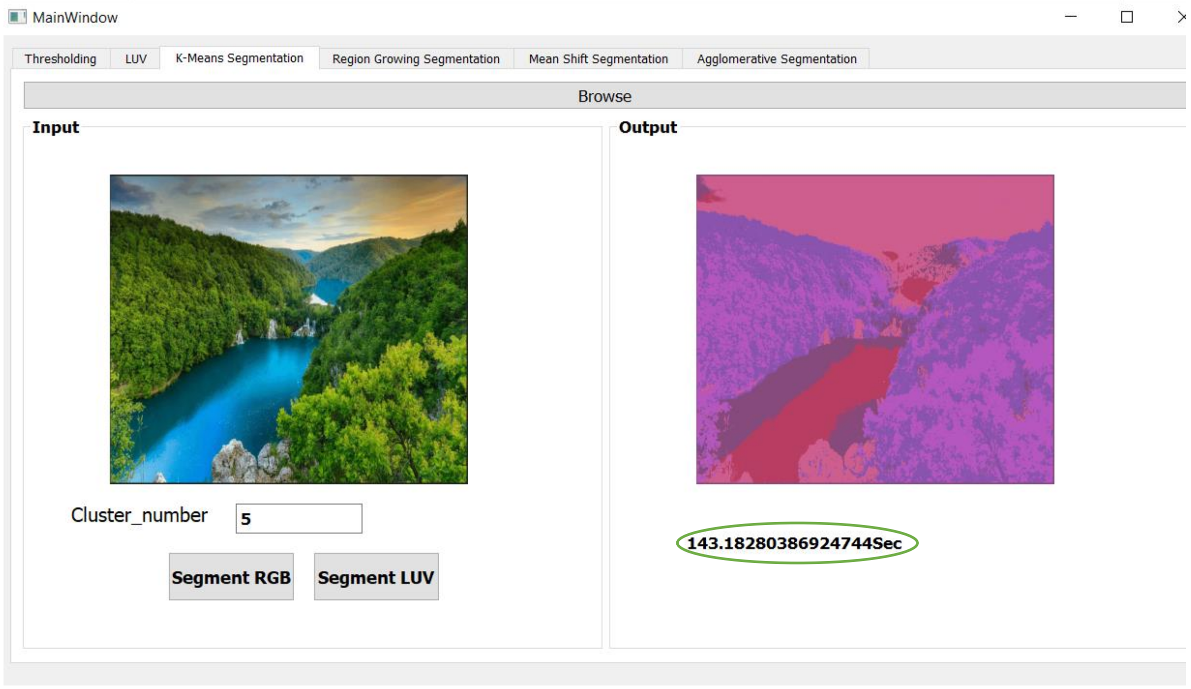
38.092947483062744Sec

Clusters=5 for rgb and luv space



Cluster_number 5

173.43802285194397Sec



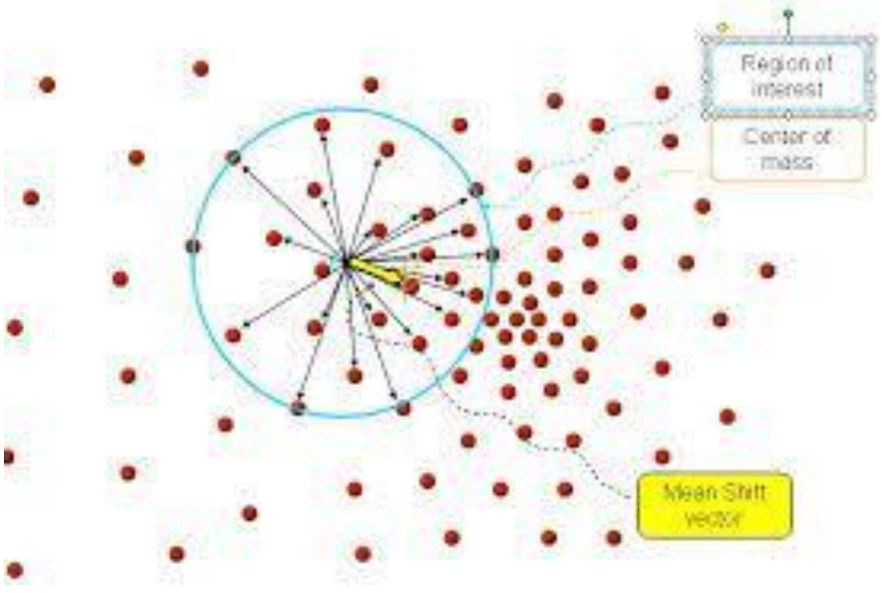
Cluster_number 5

143.18280386924744Sec

2. Mean Shift

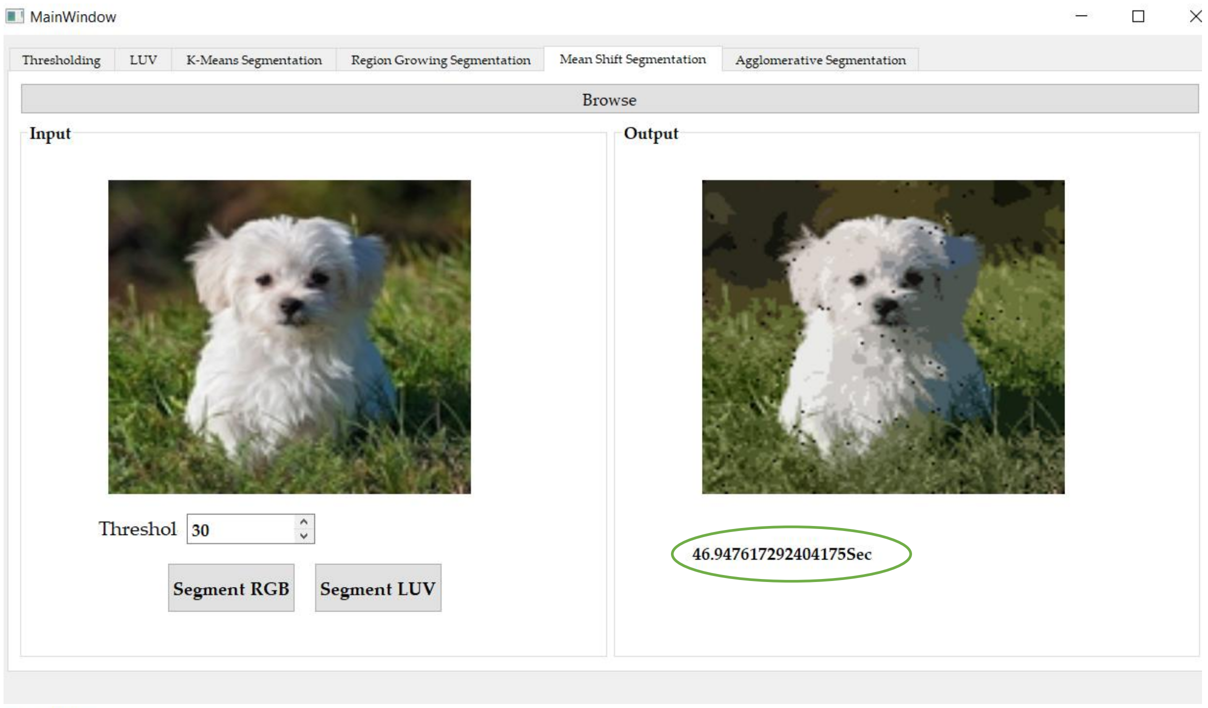
The mean shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters.

Mean shift treats the clustering problem by supposing that all points given represent samples from some underlying probability density function, with regions of high sample density corresponding to the local maxima of this distribution. To find these local maxima, the algorithm works by allowing the points to attract each other, via what might be considered a short-ranged “gravitational” force. Allowing the points to gravitate towards areas of higher density, one can show that they will eventually coalesce at a series of points, close to the local maxima of the distribution. Those data points that converge to the same local maxima are considered to be members of the same cluster.
It has good practice, robust to outliers and flexible in number & shape of regions.
It is not suitable for high dimensional features.



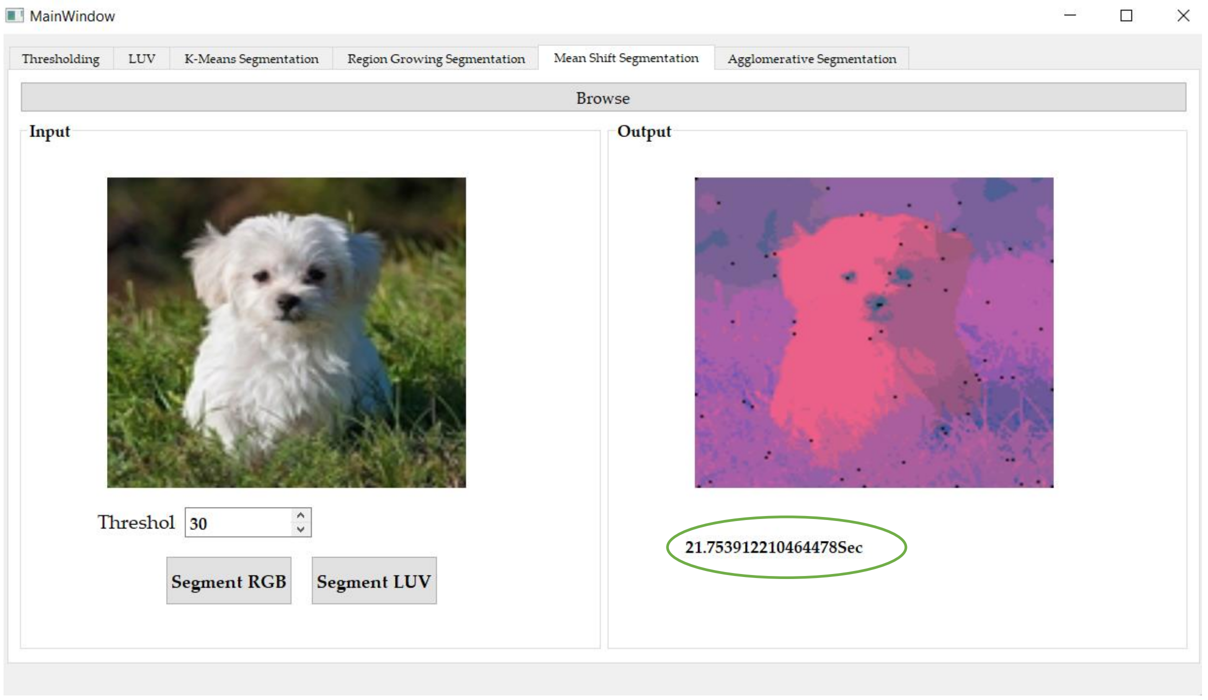
Results and Computation time

Threshold=30 for rgb and luv space



30

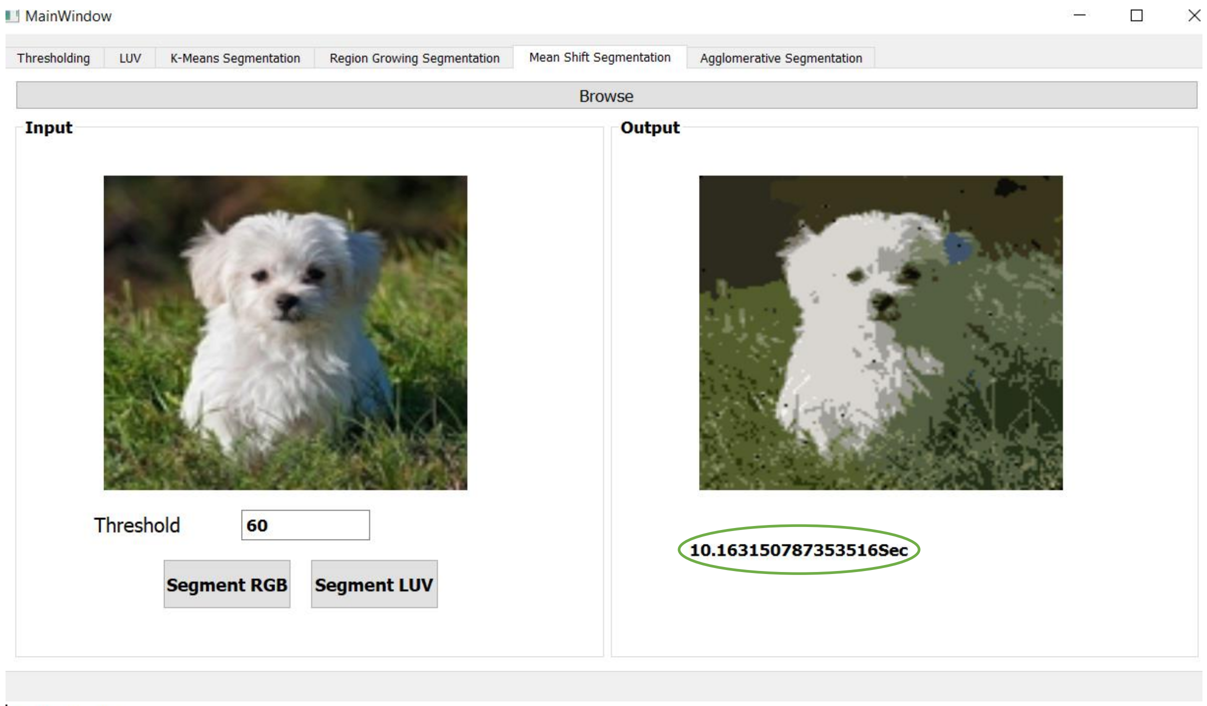
46.947617292404175Sec



30

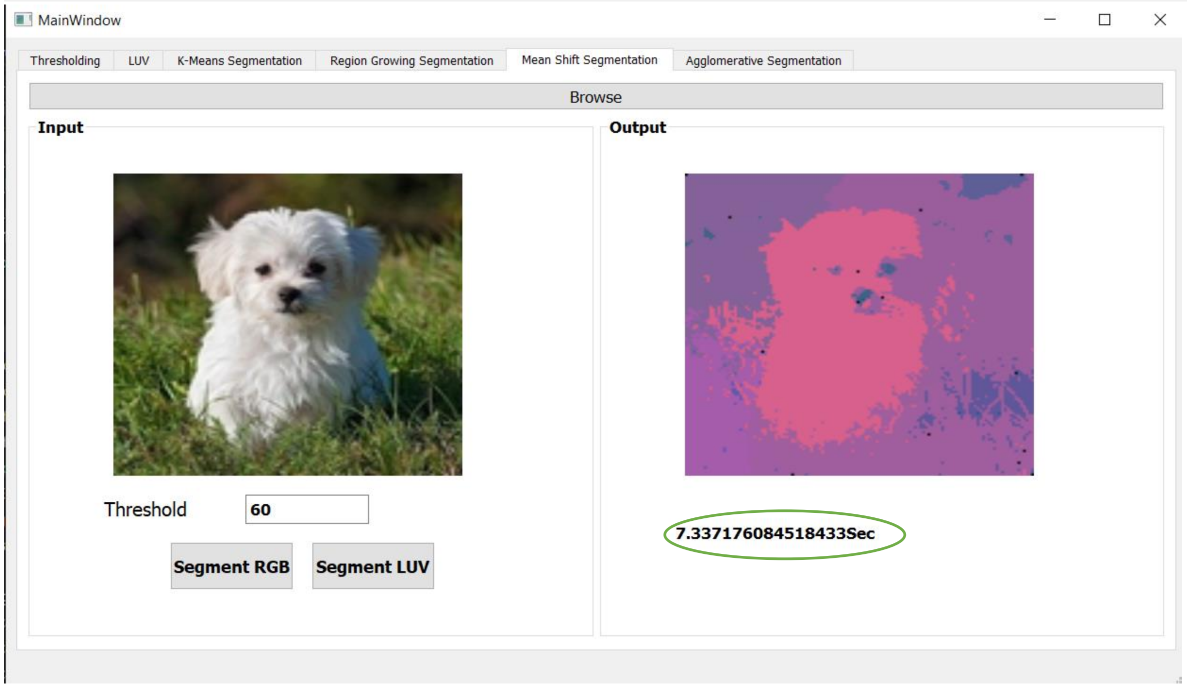
21.753912210464478Sec

Threshold=60 for rgb and luv



Threshold 60

10.163150787353516Sec

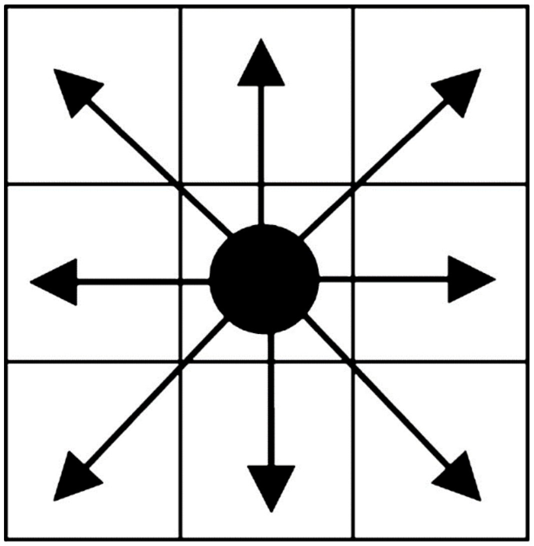


Threshold 60

7.337176084518433Sec

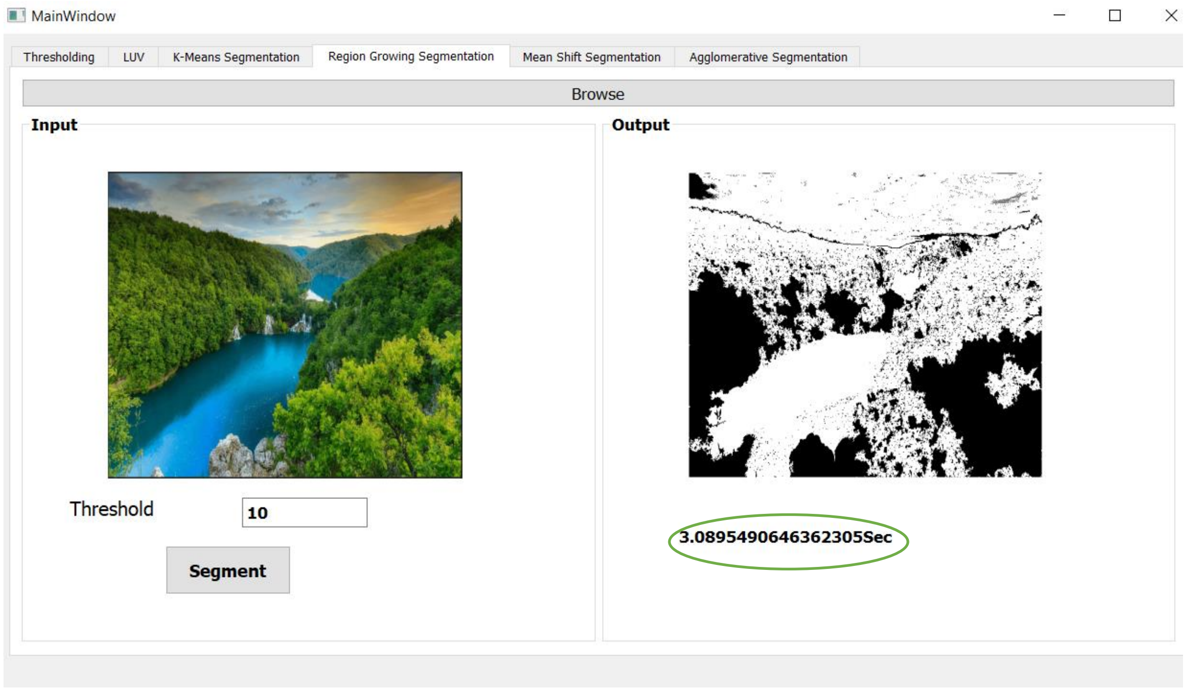
3. Region Growing

The basic idea of region growing is to assemble pixels with similar properties to form regions. Firstly, a seed pixel is found for each region to be segmented as the growth starting point, and then the seed pixel and the pixels in the surrounding neighborhood that have the same or similar properties as the seed pixel are merged into the region where the seed pixel is located. These new pixels are treated as new seeds to continue the above process until pixels that do not meet the conditions can be included. Such a region grows into.



Results and Computation time

Threshold=10



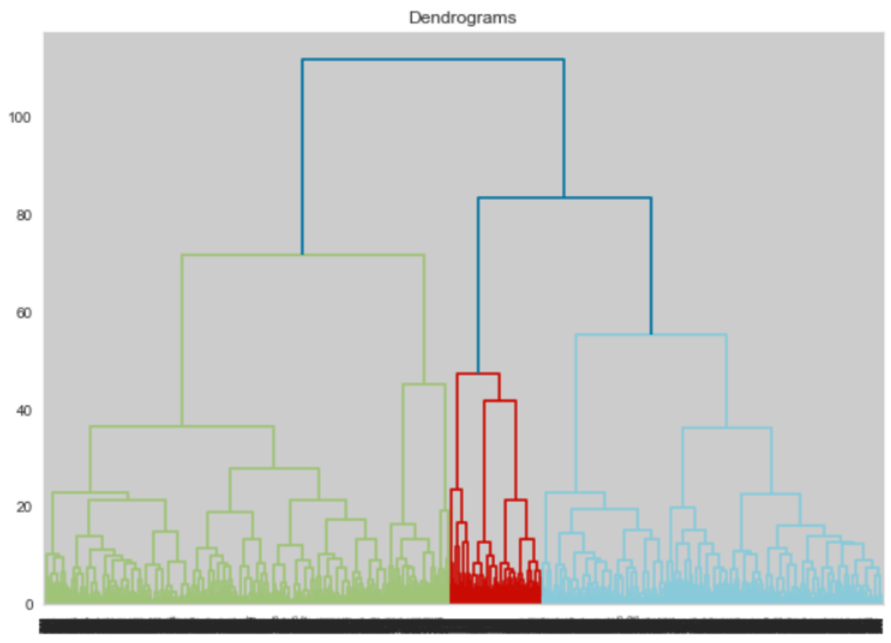
Threshold 10

3.0895490646362305Sec

4. Agglomerative

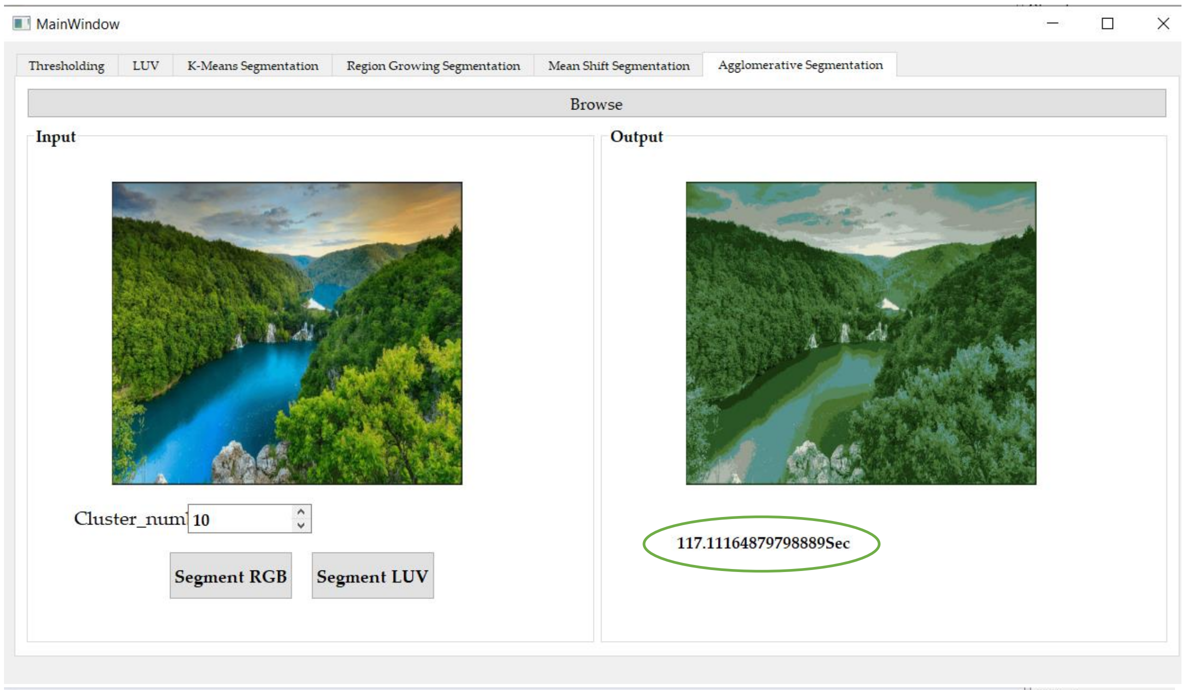
Agglomerative clustering is a general family of clustering algorithms that build nested clusters by merging data points successively. This hierarchy of clusters can be represented as a tree diagram known as dendrogram. The top of the tree is a single cluster with all data points while the bottom contains individual points.

We assign each data point to its own cluster, then find the closest pair of clusters using Euclidean distance and merge them into single cluster. Finally, we calculate the distance between two nearest clusters and combine till reach a single merged cluster.
It is simple and widespread and provides a hierarchy of clusters.
Clusters have adaptive shapes but we still have to choose number of clusters or threshold.

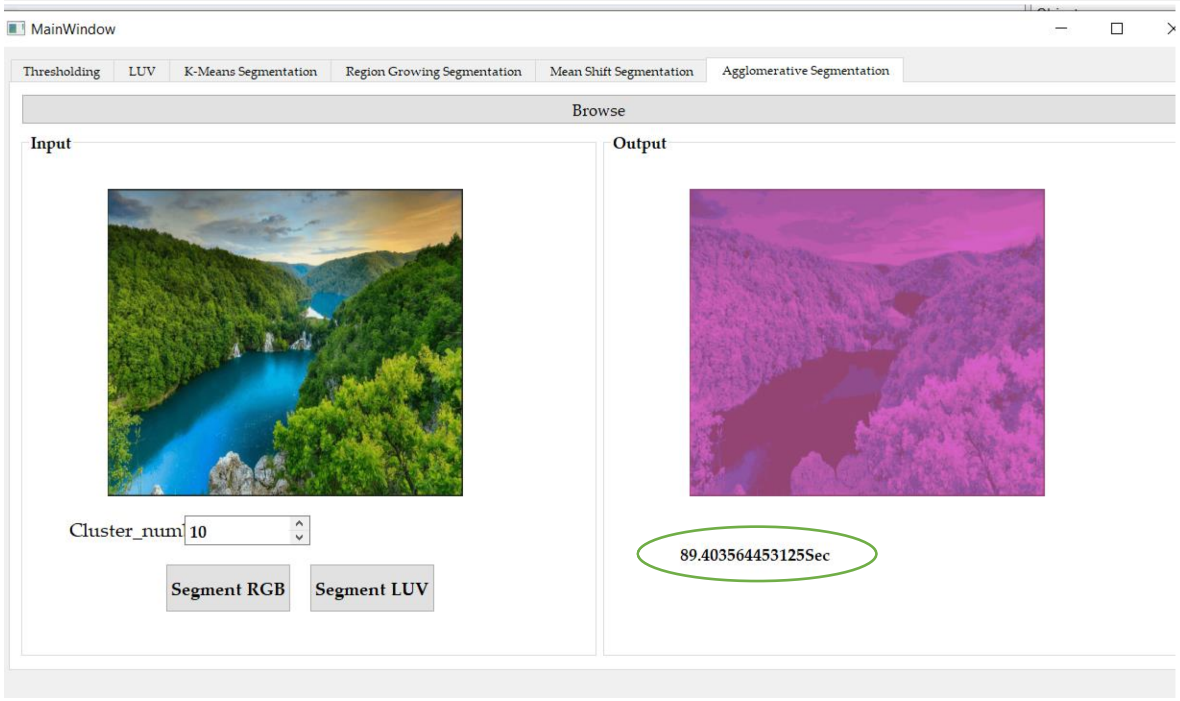


Results and Computation time

Clusters=10 for rgb and luv



117.11164879798889Sec



89.403564453125Sec

Note

For all segmentation results, threshold and number of clusters values have a big effect on the result.

The image size affects the computation time.