CSL7382: Programming Assignment

Saurabh Shashikant Burewar (B18CSE050)
Gourav Patidar (B18CSE015)

SLIC (Simple Linear Iterative Clustering) algorithm

Algorithm

This algorithm generates superpixels or clusters of pixels based on the similarity in color and proximity in the image. It is implemented in a five dimensional space 'labxy' where lab is color and xy is pixel coordinates.

Steps in the algorithm are as follows.

1. **Change the image to CIELAB color space**

   We use OpenCV to read the image and change it to the CIELAB color space. We define a class to define a superpixel or cluster. This makes it easier to maintain and update superpixels.

2. **Initialize cluster centers**

   We set up the initial cluster centers to start with which we then optimize. For this we just take centers at a regular distance S. This S is defined as -

   $$S = \sqrt{Total\ number\ of\ pixels\ /\ NUmber\ of\ superpixels}$$

3. **Reassign cluster center to pixel having lowest gradient**

   For every cluster, calculate the gradient of the cluster center. Then, calculate the gradient for each pixel to find the pixel with the lowest gradient. Make that pixel the new center. The gradients are calculated as -

   $$G(x,y) = \|I(x+1,y) - I(x-1,y)\|^2 + \|I(x,y+1) - I(x,y-1)\|^2$$

4. **Assign pixels to cluster**

   Now, we associate each pixel to its nearest cluster center and assign it to that cluster. This is done by calculating the distance D. It is defined as -

   $$D = \sqrt{(Dc/20)^2 + (Ds/S)^2}$$

   where,

   Ds is the euclidean distance between the pixel and cluster center in spacial

   $$Ds = \sqrt{(height_{pixel} - height_{center})^2 + (width_{pixel} - width_{center})^2}$$

   Dc is the euclidean distance between the pixel and cluster center in color

$$Dc = \sqrt{(l_{pixel} - l_{center})^2 + (a_{pixel} - a_{center})^2 + (b_{pixel} - b_{center})^2}$$

Now, we assign the pixel to the cluster whose center is closest (according to above calculated D) to it.

5. **Update the cluster center**

   For every cluster, calculate the mean of all pixels in that cluster. We do this by just taking the mean height and width coordinate of all pixels. Now, assign this mean pixel as the new cluster center.

6. **Repeat steps 5 and 6 for the required number of iterations**

   We update the cluster centers and assign all pixels to a cluster for a given number of iterations.

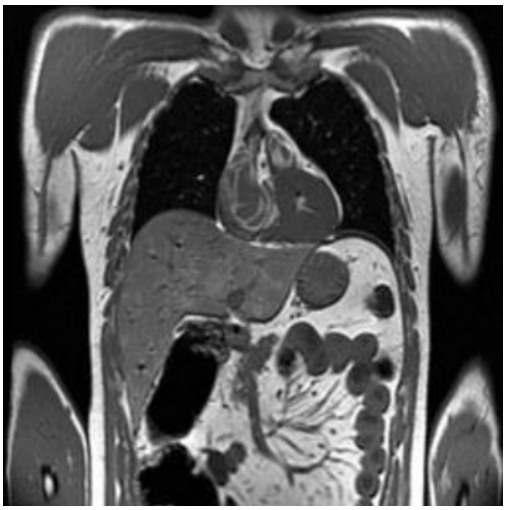
7. **Return the cluster list**

   Once the required number of iterations are over, we have the list of all the final clusters. We return this to create the final image.

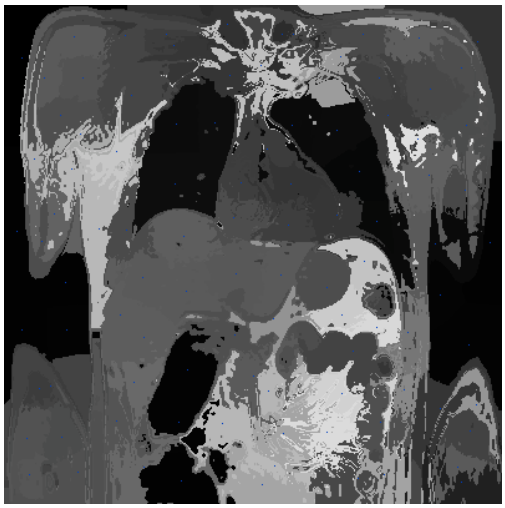
8. **Color every pixel in a cluster with the color of that cluster's center**

   We go through the list of clusters and for every cluster we color all the pixels of that cluster in the color of that cluster's center. Also, each cluster center is colored as blue (zoom in the resultant image below to see). Then we output the resultant image.

Results



Original



Superpixels

Thin Plate Spline algorithm

Algorithm

Thin plate splines (TPS) are a spline-based technique for data interpolation and smoothing, non-rigid alignment algorithm for aligning high-resolution range data in the presence of low frequency deformations, such as those caused by scanner calibration error. Traditional iterative closest points (ICP) algorithms, which rely on rigid-body alignment, fail in these cases because the error appears as a non-rigid warp in the data. Our algorithm combines the robustness and efficiency of ICP with the expressiveness of thin-plate splines to align high-resolution scanned data accurately.

The steps for this algorithm are as follows:

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} w_i U(|Pi - (x,y)|)$$

$$\begin{aligned}
\text{where: } f(x,y) &= \text{the surface} \\
v_i &= \text{the height of the surface at } (x_i, y_i) \\
i &= 1, 2, ..., n \\
n &= \text{number of control points} \\
U(r) &= r^2 log(r) \\
P(i) &= \text{the location of the ith control point } (x_i, y_i) \\
a, w &= \text{unknown variables}
\end{aligned}$$

There are two parts in the function f. The first three terms form an affine, or linear part, that represents the behavior at infinity. It is basically a flat plane that the surface approaches at infinity. The second term, which is a sum of functions U(r), provides the bending to the surface. So the unknown coefficients a and w can be obtained by solving the linear system of equations,

$$v_i = f(x_i, y_i), \ \sum_{i=1}^{n} w_i = 0, \ \sum_{i=1}^{n} w_i x_i = 0, \ \sum_{i=1}^{n} w_i y_i = 0.$$

These equations can be arranged nicely into a matrix as:

$$L[W|a_1, a_x, a_y]^T = Y,$$

where:

$$L = \begin{bmatrix} K & P \\ P^T & O \end{bmatrix},$$

$$K = \begin{bmatrix} 0 & U(r_{12}) & ... & U(r_{1n}) \\ U(r_{21}) & 0 & ... & U(r_{2n}) \\ ... & ... & ... & ... \\ U(r_{n1}) & U(r_{n2}) & ... & 0 \end{bmatrix},\ r_{ij} = |P_i - P_j|,$$

$$P = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ ... & ... & ... \\ 1 & x_n & y_n \end{bmatrix},$$

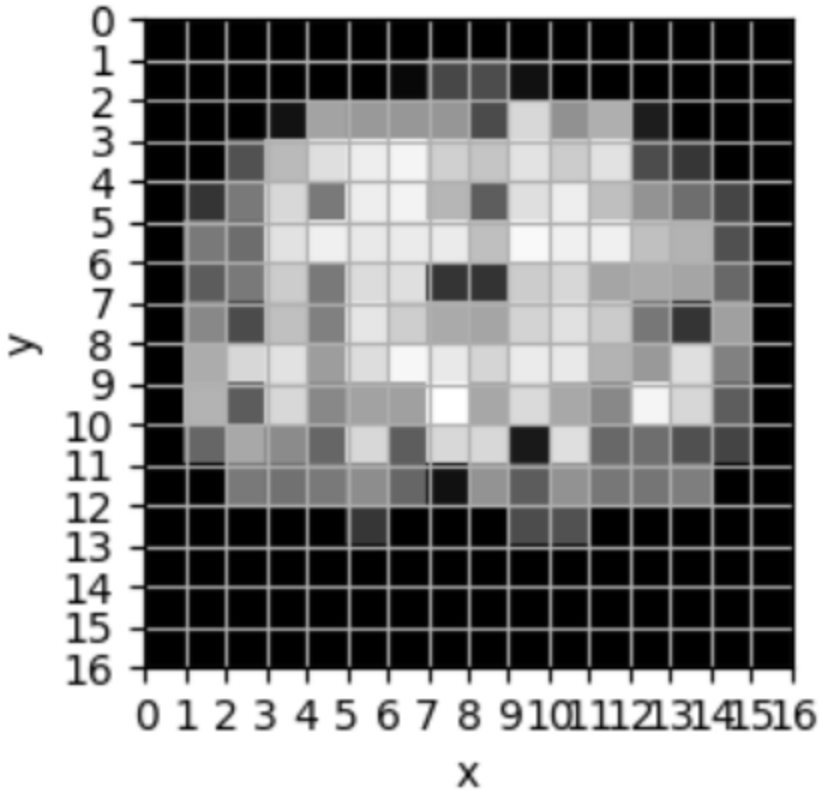
$$Y = [V|\ 0\ 0\ 0]^T,\ V = [v_1\ v_2\ ...\ v_n],\ W = [w_1\ w_2\ ...\ w_n].$$

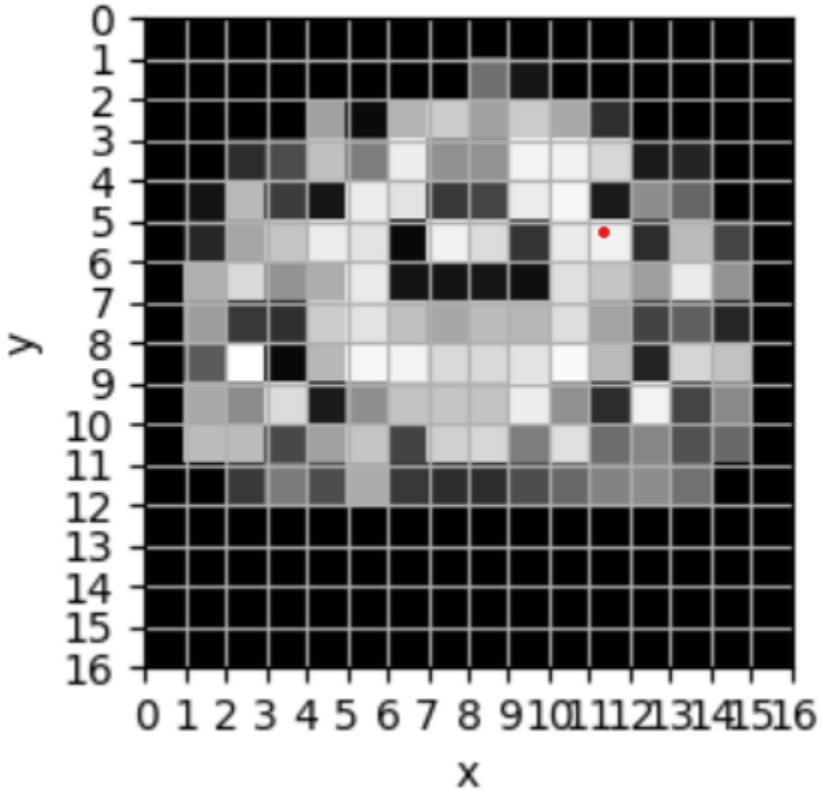
Then, once we know the values for w and a, we can interpolate v for arbitrary points (x, y) using:

$$v = f(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} w_i U(|Pi - (x, y)|)$$

Results:

We got our final output using the various inbuilt functions.


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
y
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
x


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
y
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
x

References

- https://infoscience.epfl.ch/record/177415/files/Superpixel_PAMI2011-2.pdf
- https://darshita1405.medium.com/superpixels-and-slic-6b2d8a6e4f08
- https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/belongie-iccv01.pdf
- "Manual Registration with Thin Plates" by Herve Lombaert
- Bookstein, F.L., 1989. Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Transactions on pattern analysis and machine intelligence, 11(6), pp.567-585.
- Thin Plate Splines (TPS) library