IMAGE SEGMENTATION USING PUSH-RELABEL ALGORITHM WITH SUPERPIXELS

AARYAN GUPTA- 200050002

DHAIRYA PAREKH- 200050097

NAMAN SINGH RANA- 200050083

PROBLEM STATEMENT

- Given a black and white greyscale image, segment it into two clusters based on locality and color value.
- This problem is particularly important in medical images such as brain MRI scans, where segmentation is based on white matter and gray matter.
- Another possible area of application is to cut out relevant images with a green screen in the background in the film industry.
- Problem can be generalized to full color images and multiple $K>2$ segments, but we simplify as above to make use of graph flows.

CURRENT APPROACH IN TEXTBOOKS

- An image with N*N pixels is given to us with each pixel being characterized by location x, y and greyscale g (g is between 0 and 1).
- Assume an MRF(Markov Random Field) prior with quadratic clique potential with fixed parameters.
- Consider a Gaussian noise model for segmentation characterized by unknown parameters.
- We obtain the segmentation and parameter estimates for noise model by maximizing posterior function.
- The MAP estimation can be mathematically written as below with x denoting the segmentation, y denoting the pixel intensities, and beta depending on the MRF prior and clique.

$$\max_x P(x|y,\theta) = \max_x \left(\sum_i \lambda_i x_i + 0.5 \sum_i \sum_j \beta_{ij}(2x_i x_j - x_i - x_j) \right)$$

where $\lambda_i := \log P(y_i|x_i = 1,\theta) - \log P(y_i|x_i = 0,\theta) = \log \frac{P(y_i|x_i=1,\theta)}{P(y_i|x_i=0,\theta)}$

CURRENT APPROACH IN TEXTBOOKS

- Construct an s-t graph with vertices as pixels with a source and a sink with three types of edges:

1. if lambda>0, add an edge from source to vertex with cost lambda
2. if lambda<0, add an edge from vertex to sink with cost -lambda
3. for every pair of neighboring vertices add corresponding edge beta between them

- The MAP estimation problem described below corresponds to finding the min cut in the graph below to get the segmentation.
- We find the max flow using the Ford-Fulkerson algorithm and find the corresponding min cut using the duality theorem.

OUR APPROACH

- Our strategy is based on the introduction of superpixels and replacing the network flow algorithm by the push-relabel algorithm.

STEP 1: Creating Superpixellized image from the original image by clustering similar pixels together

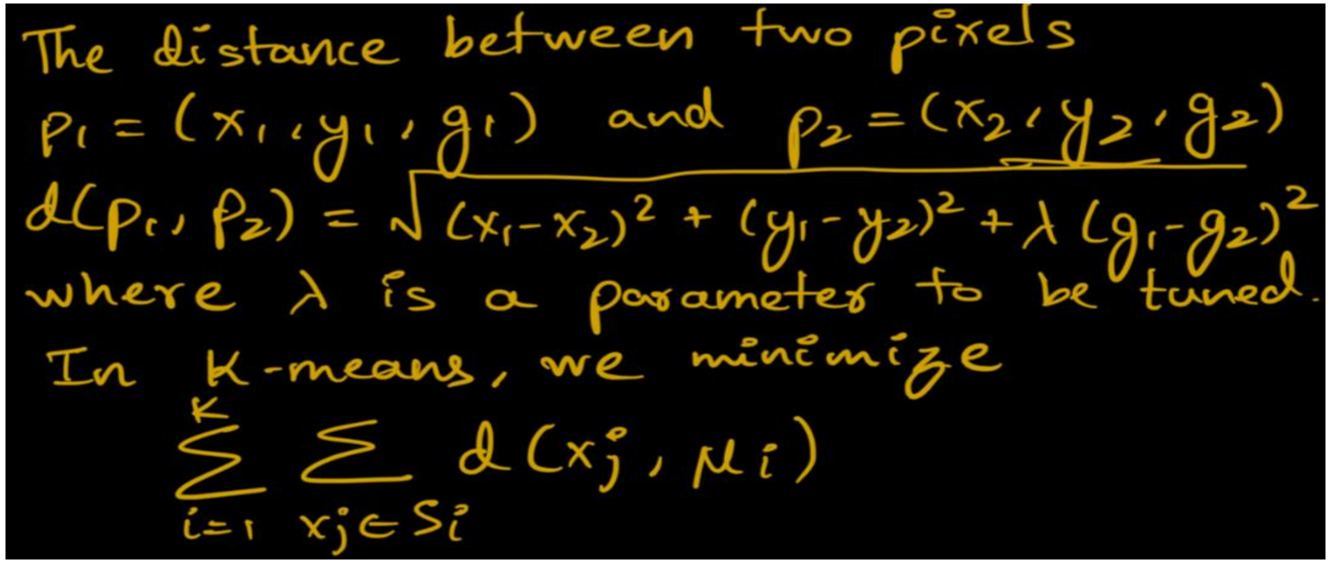
STEP 2: Creating an initial s-t flow graph from these superpixels with some aribitrary parameters for noise model

STEP 3: Find the max flow of above graph using the push-relabel algorithm and find the min cut of the graph and segment the image according to the cut.

STEP 4: Find optimal parameters for noise model given the segmentation found in step 3. Go back to step 2 and repeat steps 2,3,4 iteratively until segmentation values do not change between two consecutive iterations.

STEP 1: CREATE SUPERPIXELS

- Inspiration from Wang and Zhang's (2009) paper led us to realize that superpixels(or a similar creation) could be used to cluster similar pixels before the network flow algorithm to minimize the complexity.
- Superpixel: Cluster of pixels with similar values of position and color.
- In our approach k superpixels are created using K-means algorithm with random initialization.
- After this algorithm is over, we get k means containing x,y and average g corresponding to the k



The distance between two pixels $p_1=(x_1,y_1,g_1)$ and $p_2=(x_2,y_2,g_2)$

$$d(p_1,p_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+\lambda(g_1-g_2)^2}$$

where λ is a parameter to be tuned.

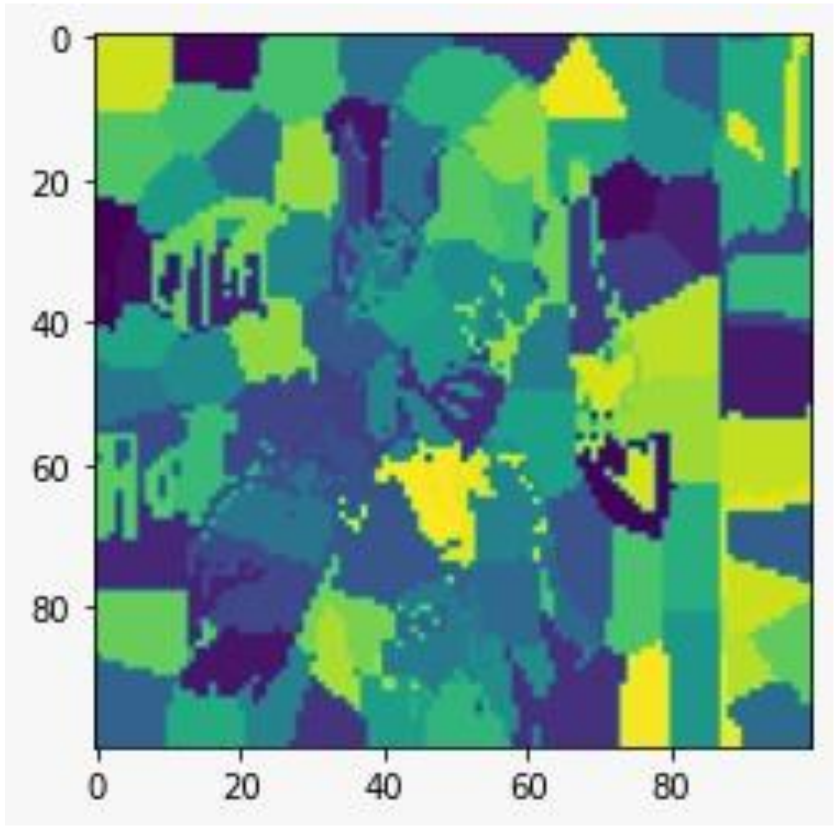
In K-means, we minimize

$$\sum_{i=1}^{K}\sum_{x_j\in S_i} d(x_j,\mu_i)$$

ANALYSIS: CREATION OF SUPERPIXELS EXAMPLE 1

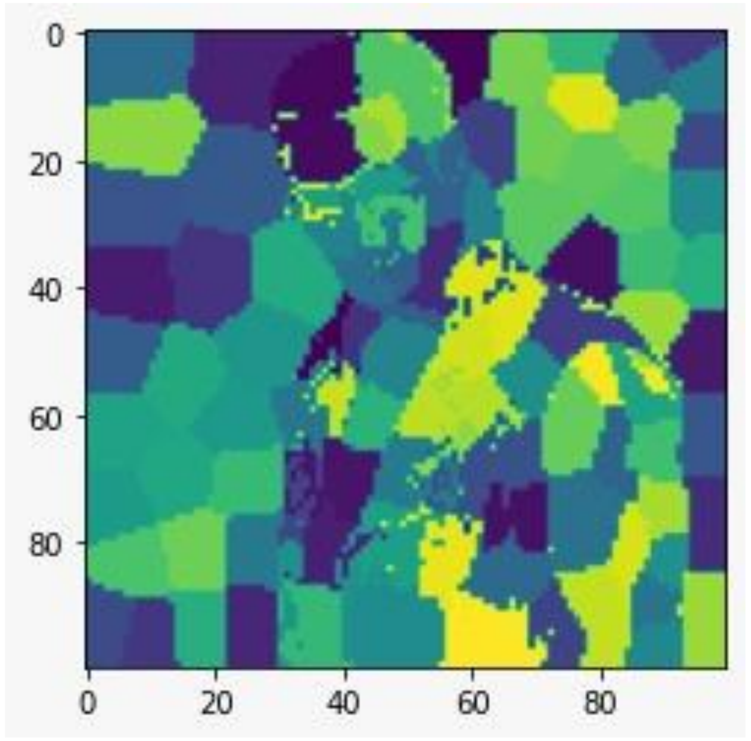
- Original image

Superpixel segmentation



ANALYSIS: CREATION OF SUPERPIXELS EXAMPLE 2

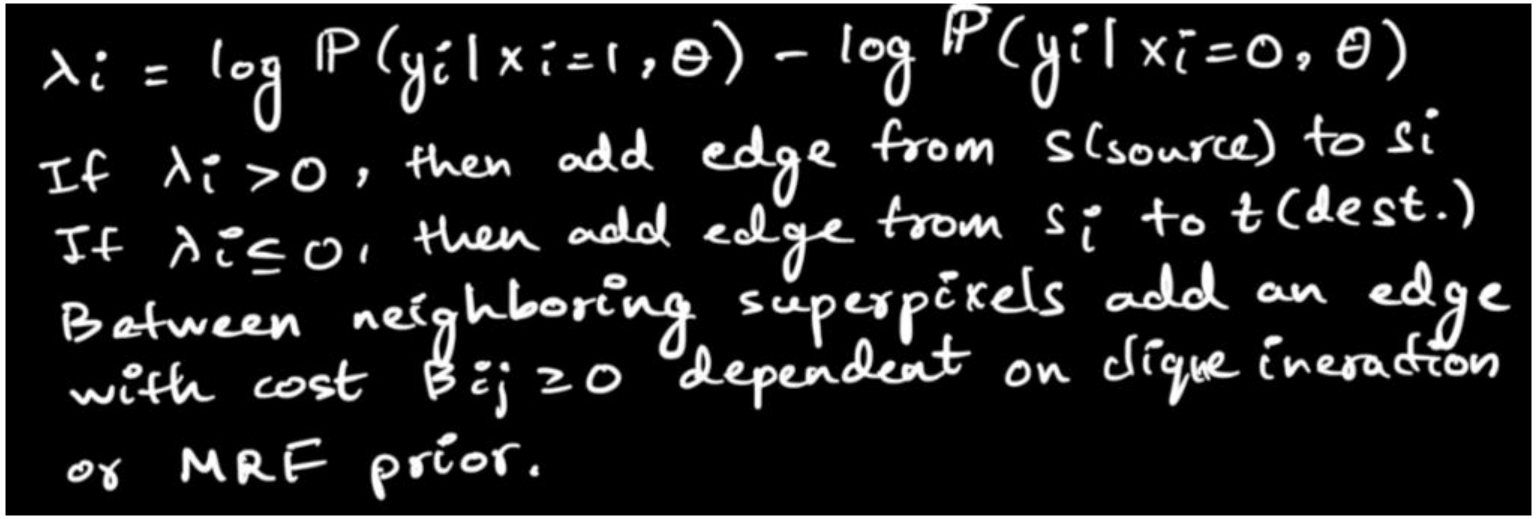
- Original Image



Superpixelized image
with Superpixels given different
colors for clarity

STEP 2: CONSTRUCTING GRAPH FROM SUPERPIXELS

- We get k superpixels q_i where i is in 1,2,...k. For each superpixel q_i we get the tuple (x_i, y_i, g_i) from the last step. We calculate lambda for each pixel given the current estimates for theta (random for the first iteration).
- Two super pixels are considered neighbors if d(q_i, q_j)<= c where c is a constant depending on n and k. (Note that this is different from the 4-neighbor system we usually follow as the superpixels can be blob shaped and have more or less than 4 neighbors)
- K-means algorithm generally gives superclusters of similar size, so c can be theoretically calculated to be c=2n/sqrt(k).
- We introduce a source s and a sink t in the graph defined on user input (ideally source source and sink should be in different segments)
- The edge weights(depending on calculated parameters) are defined in the same way as the textbook approach given below-



$$\lambda_i = \log \mathbb{P}(y_i \mid x_i = 1, \theta) - \log \mathbb{P}(y_i \mid x_i = 0, \theta)$$

If $\lambda_i > 0$, then add edge from s(source) to s_i

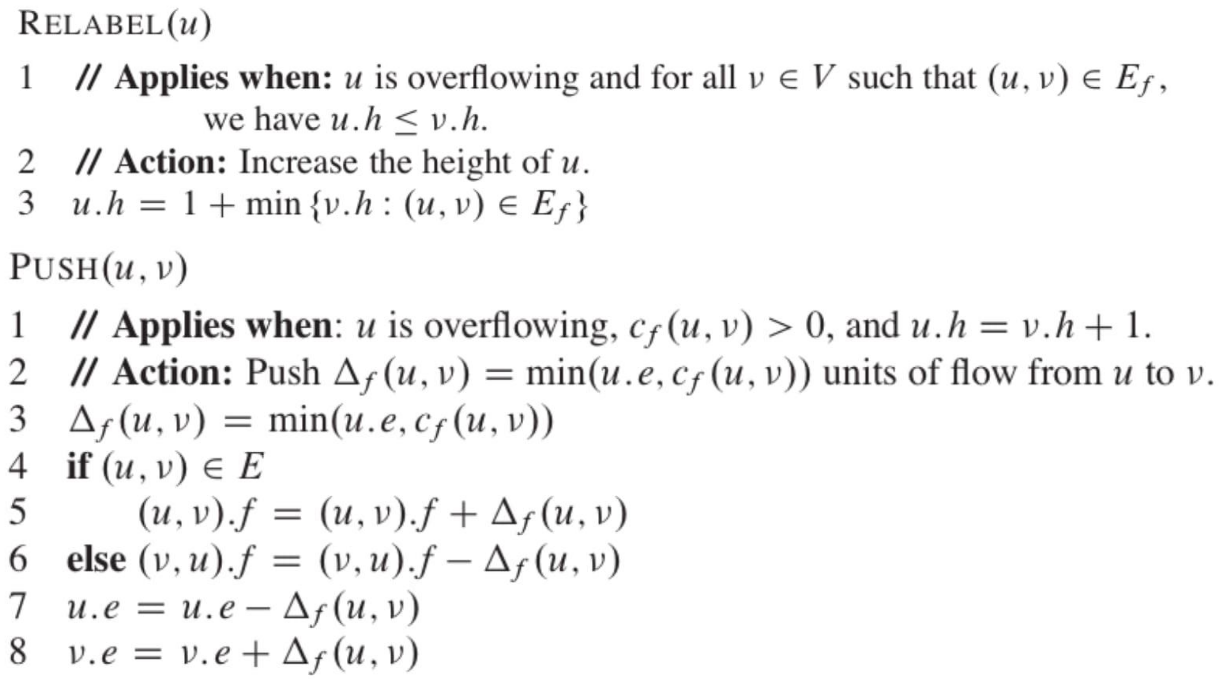
If $\lambda_i \leq 0$, then add edge from s_i to t(dest.)

Between neighboring superpixels add an edge with cost $B_{ij} \geq 0$ dependent on clique ineraction or MRF prior.

Note that the xi here denotes the segmentation and yi denotes grayscale intensity, not positions.

STEP 3: SEGMENTATION USING PUSH-RELABEL ALGORITHM

- We use push-relabel algorithm instead of Ford-Fulkerson(or equivalently Edmond-Karp) because it reduces complexity to O(EV^2) from O(VE^2) and turns out to be much faster in implementation.
- This algorithm maintains a height h and an excess Delta at each node at every iteration.
- There are two main actions, push and relabel defined as below:



```
RELABEL($u$)
// Applies when: $u$ is overflowing and for all $v \in V$ such that $(u, v) \in E_f$,
        we have $u.h \leq v.h$.
// Action: Increase the height of $u$.
$u.h = 1 + \min\{v.h : (u, v) \in E_f\}$

PUSH($u, v$)
// Applies when: $u$ is overflowing, $c_f(u, v) > 0$, and $u.h = v.h + 1$.
// Action: Push $\Delta_f(u, v) = \min(u.e, c_f(u, v))$ units of flow from $u$ to $v$.
$\Delta_f(u, v) = \min(u.e, c_f(u, v))$
if $(u, v) \in E$
    $(u, v).f = (u, v).f + \Delta_f(u, v)$
else $(v, u).f = (v, u).f - \Delta_f(u, v)$
$u.e = u.e - \Delta_f(u, v)$
$v.e = v.e + \Delta_f(u, v)$
```

STEP 3: SEGMENTATION USING THE PUSH-RELABEL ALGORITHM

- We initialise the preflow and heights given as follows:

INITIALIZE-PREFLOW creates an initial preflow f defined by

$$(u, v).f = \begin{cases} c(u, v) & \text{if } u = s \text{ ,} \\ 0 & \text{otherwise .} \end{cases}$$

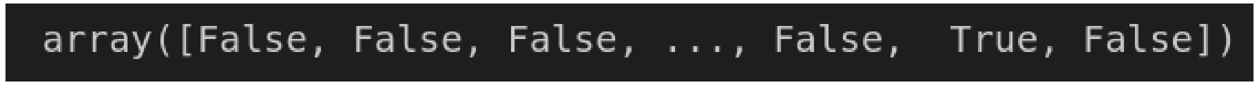
$$u.h = \begin{cases} |V| & \text{if } u = s \text{ ,} \\ 0 & \text{otherwise .} \end{cases}$$

The main algorithm to obtain max flow is given as below:

```
GENERIC-PUSH-RELABEL(G)
1  INITIALIZE-PREFLOW(G, s)
2  while there exists an applicable push or relabel operation
3      select an applicable push or relabel operation and perform it
```

STEP 3: SEGMENTATION USING THE PUSH-RELABEL ALGORITHM

- Find set of vertices reachable from source in the final residual graph from step 3 using a simple graph search algorithm such as DFS (implemented here)
- All edges which are from a reachable vertex to non-reachable vertex are minimum cut edges.
- The set of reachable vertices is the minimum cut.
- The two sets of reachable and non-reachable vertices give the segmented image.



```
array([False, False, False, ..., False,  True, False])
```

This is the set of boolean values given by the program on a particular iteration to tell us if a pixel is in the min cut (i.e a particular segment)

STEP 4: CALCULATION OF NOISE MODEL PARAMETERS

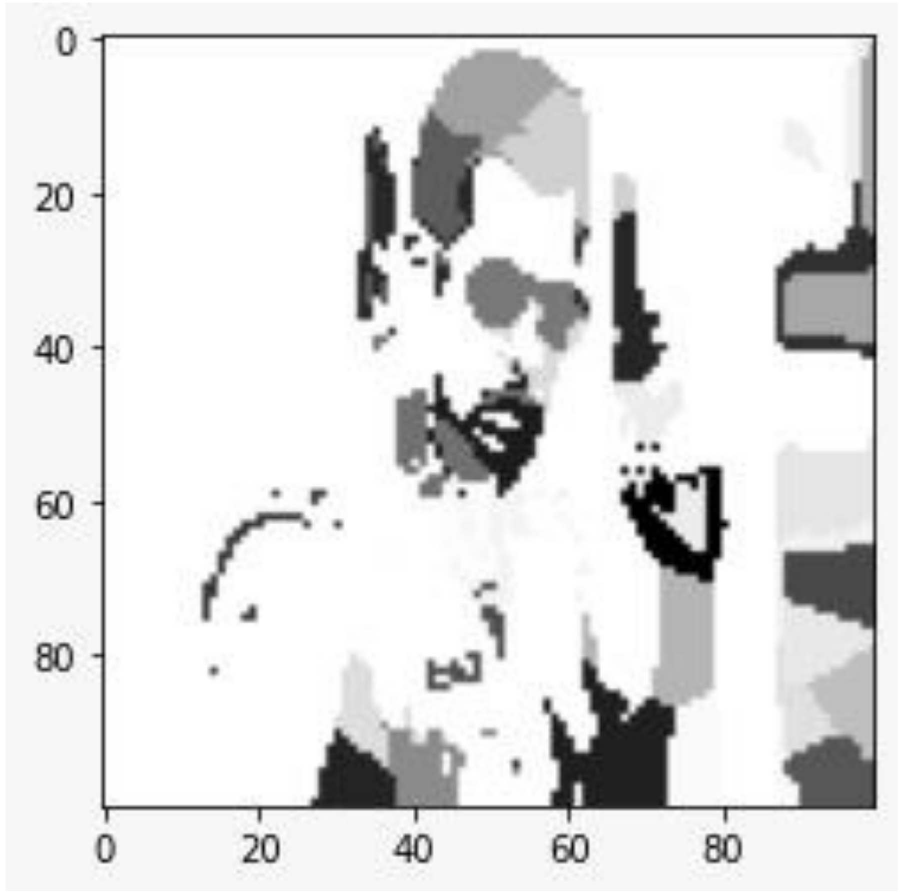
- Given the segmentation we had from step 3 (the true/false array), we need to find the optimal parameters theta for the noise model to maximize posterior function.
- The noise model was assumed to be Normal with parameters (mean1, variance1) for the first segment and parameters (mean2, variance2) for the second segment.
- A standard maximum likelihood estimation gives us the parameter estimates as follows:

$$\mu_1 = \frac{\sum_{i \in \{x=0\}} \mu_i}{|\{x=0\}|} \qquad \mu_2 = \frac{\sum_{i \in \{x=1\}} \mu_i}{|\{x=1\}|}$$

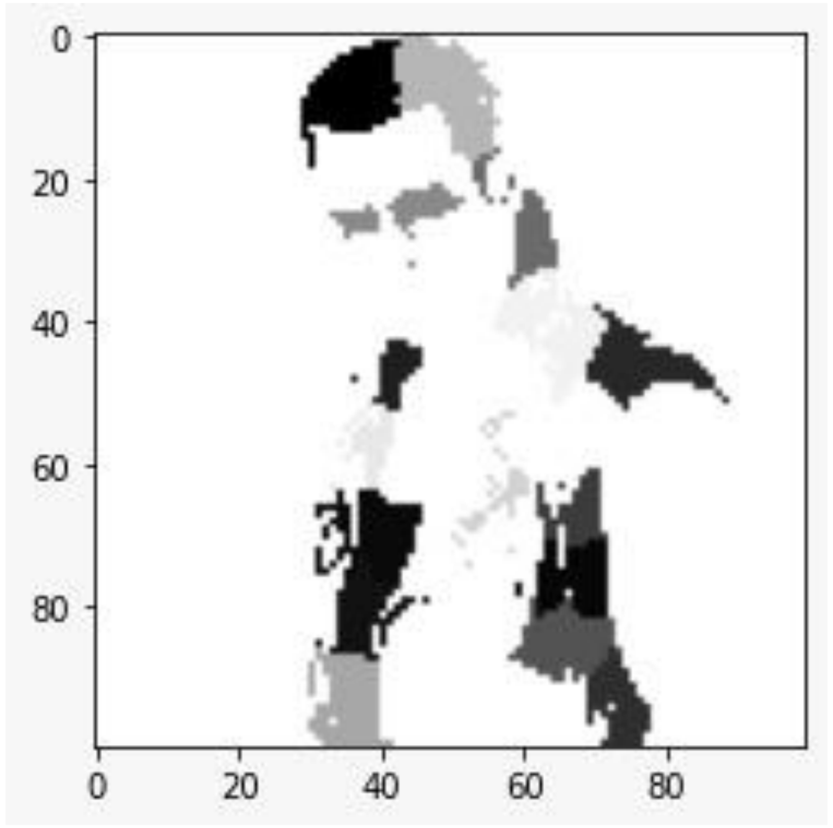
$$\sigma_1 = \frac{\sum_{i \in \{x=0\}} (\mu_i - \mu_1)^2}{|\{x=0\}|} \qquad \sigma_2 = \frac{\sum_{i \in \{x=1\}} (\mu_i - \mu_2)^2}{|\{x=1\}|}$$

After the parameter estimates are calculated, we go back to step 2 and create a graph with these parameter estimates to obtain a suitable segmentation. We iterate this process a fixed number of times (say 20), or we iterate until segmentation values do not change between two consecutive iterations.

ANALYSIS: FINAL SEGMENTED IMAGE EXAMPLE 1



ANALYSIS: FINAL SEGMENTED IMAGE EXAMPLE 2



ANALYSIS: PERFORMANCE

- The superpixellisation reduces the calculations the graph flow algorithm needs to perform significantly more than the textbook method.
- After superpixellisation, we were able to segment 1000*1000 pixel images whereas we were only able to segment 30*30 pixel images at most.

CONCLUSION: PROBLEMS FACED

- The creation of superpixels faced a lot of problems. The superpixels created were sometimes not a continuous blob but a disjoint set of shapes. This led to us choosing a low value of lambda to allow for more spatial preference in clusters.
- When deciding if two nodes are neighbors, no matter what value of c we pick, some neighbors are inadvertently left or too many extra neighbors are included.
- We had quite a lot of difficulty in getting the segmentation(min cut) from the flow obtained from step 3. This actually gave away the time complexity benefits we were theoretically expecting.
- The loop invariant while iterating between parameter estimates and segmentation was first taken to be that segmentation between two consecutive images changes. However, this was taking too much time to segment because of the "ping-pong" effect.

CONCLUSION: POSSIBLE FIXES TO PROBLEMS

- Another procedure called SLIC exists to generate the superpixels which avoids the problems we faced.
- Some condition other than just distance could be used to decide if two nodes are neighbors.
- We can just implement Edmond-Karp/Ford-Fulkerson instead of the push-relabel algorithm.
- We fixed the number of iterations we took to 20. The images seemed to be well segmented by this point.

REFERENCES

- X. Wang and X. Zhang, "A new localized superpixel Markov random field for image segmentation," 2009 IEEE International Conference on Multimedia and Expo, 2009, pp. 642-645, doi: 10.1109/ICME.2009.5202578.
- Cormen, Thomas H., and Thomas H. Cormen. 2001. *Introduction to algorithms*. Cambridge, Mass: MIT Press.
- https://en.wikipedia.org/wiki/Push%E2%80%93relabel_maximum_flow_algorithm