AUTOMATIC IMAGE SEGMENTATION VIA DYNAMIC REGION MERGING

Team Noisy Pixel

Shubham Dokania (2020701016)
Shanthika Shankar Naik (2020701013)
Sai Amrit Patnaik (2020701026)

TA
Aditya Arun

Github link:
https://github.com/Digital-Image-Processing-IIITH/project-noisypixel

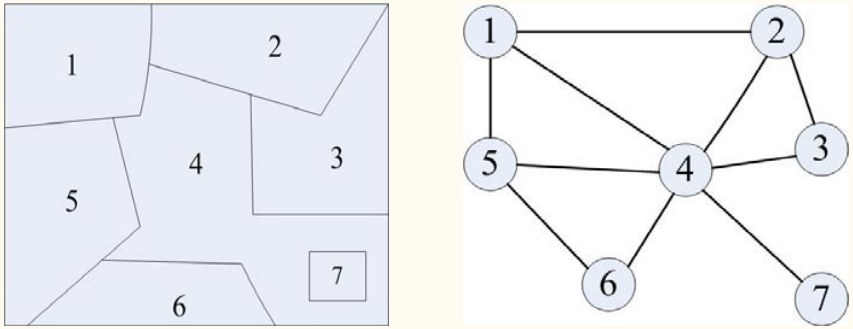
Segmentation with Region Graphs

Segmentation is a difficult problem to solve in image processing and computer vision, yet contains high potential for image understanding, analysis, and is useful in several downstream tasks. This work takes the region merging approach to segmentation. Following is the pipeline of the implementation :

- We start with an initially over-segmented image with many regions (can contain vertices or super-pixels).
- Formulate the image as a graph and segmentation over this graph as an optimization problem.
- Iteratively merge regions according to some statistical tests.
- Build the objective function and propose a solution based on dynamic programming.
- Show that local segmentation using this method also satisfies global segmentation properties.

Building the Region Graph

Given an image, an undirected graph G = (V , E) , where V is set of vertices representing the pixels or regions and E are set of edges with weight given by minimum dissimilarity between two regions.



Region Adjacency Graph formation from parts of Image

Region Merging

$$P(R_1, R_2) = \begin{cases} true & \text{if (a) } S(R_1, R_2) = \min_{R_i \in \Omega_1} S(R_1, R_i) = \min_{R_j \in \Omega_2} S(R_2, R_j); \text{ and} \\ & \text{(b) } R_1 \text{ and } R_2 \text{ are consistent} \\ false & \text{otherwise} \end{cases}$$

Conditions for region Merging:

Region Merging is essentially depends on:

- Compatibility : Depends on the weights associated with edges of adjacent regions
- Stopping Criteria: Sequential Probability Ratio Test (SPRT)

Preset λ_1;
Let λ_2=1, α=0.05, β=0.05;
Compute parameters:
N_0: be a constant greater than $\max\{E\{\delta|\theta_0\}, E\{\delta|\theta_1\}\}$;
$A=\log(1-\beta)/\alpha$, $B=\log\beta(1-\alpha)$;
$P_0(x|\theta_0)$, $P_1(x|\theta_1)$ are computed using Eq. (4).

Input: a pair of neighboring regions.
Output: the decision D that the two regions are "consistent" (D=1) or "inconsistent" (D=0).

1. Set evidence accumulator δ and the trials counter n to be 0.
2. Randomly choose m pixels in each of the pair of regions, where m equals the half size of the region.
3. Calculate the distributions of visual cues x using Eq. (4) based on these pixels.
4. Update the evidence accumulator $\delta = \delta + \log \frac{P_1(x|\theta_1)(1-\beta)}{P_0(x|\theta_0)(1-\alpha)}$.
5. If $n \le N_0$
   If $\delta \ge A$, return D=1 (consistent)
   If $\delta \le B$, return D=0 (inconsistent)
   If $n > N_0$
   If $\delta \ge 0$, return D=1 (consistent)
   If $\delta < 0$, return D=0 (inconsistent)
6. Go back to step 2.

Algorithm for consistency test (The SPRT test)

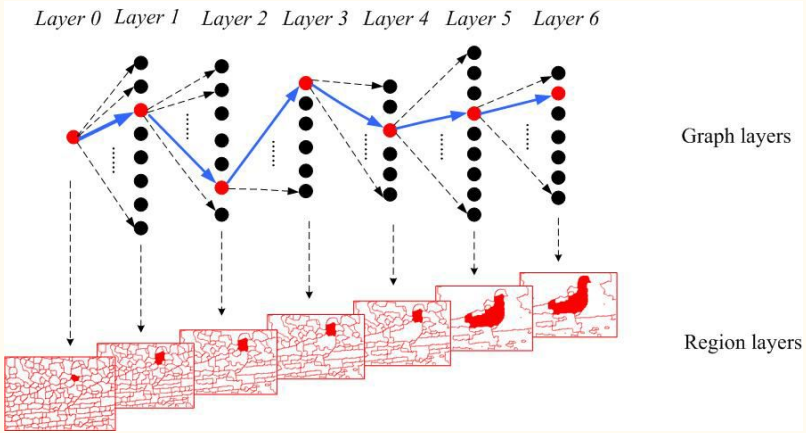
Dynamic Region Merging

- As stated before, we start with an over-segmented image. We can use watershed, SLIC, or K-means based simple methods to obtain such an initial segmentation easily.
- Considering the region merging as a labelling problem, we see that each region in the graph changes from an initial label to a new label on each iteration (could be same or repetitive as well), and we want to find the overall final label for these regions.
- Ideally, we want to find the optimal sequence of merges which produces the union of optimal labeling for all the regions, and hence minimizes the cost of such transitions.

$$
\begin{aligned}
\min F_i(l_i^0, \cdots, l_i^n) &= \min F_i(l_i^0, l_i^{n-1}) + d_{n-1,n} \\
&= \min F_i(l_i^0, l_i^{n-2}) + d_{n-2,n-1} + d_{n-1,n} \\
&= \cdots \\
&= \sum_{k=0}^{n-1} d_{k,k+1}
\end{aligned}
$$

We define the cost for one region, based on the above assumptions as shown (left). Here, each parameter in the final sum is the cost of transition over two labels in the same region.

Dynamic Programming Problem



We can assume each transition as a layer in a graph, and the weight on the path to be the cost. This way, the changes happening across the entire RAG through time becomes a big graph problem where we need to find the shortest path.

And for this task, the dijkstra's algorithm is taken into application.

Overall algorithm as a block

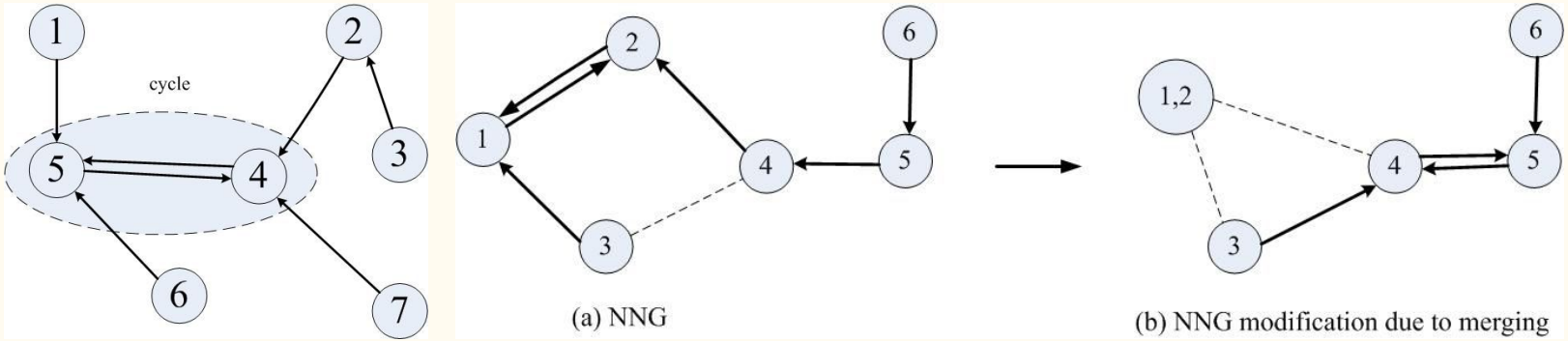
Input: the initially over segmented image S_0.
Output: region merging result.

1. Set i=0.
2. For each region in segmentation S_i, use **Algorithm 1** to check the value of predicate P with respect to its neighboring regions.
3. Merge the pairs of neighboring regions whose predicate P is true, such that segmentation S_{i+1} is constructed.
4. Go back to step 2 until $S_{i+1} = S_i$.
5. Return S_i.

Nearest Neighbour Graph

It is easy to see that the previous algorithm has a high time complexity due to the evaluation of the minimum weights between regions for every change and requires a search over all edges at each iteration.

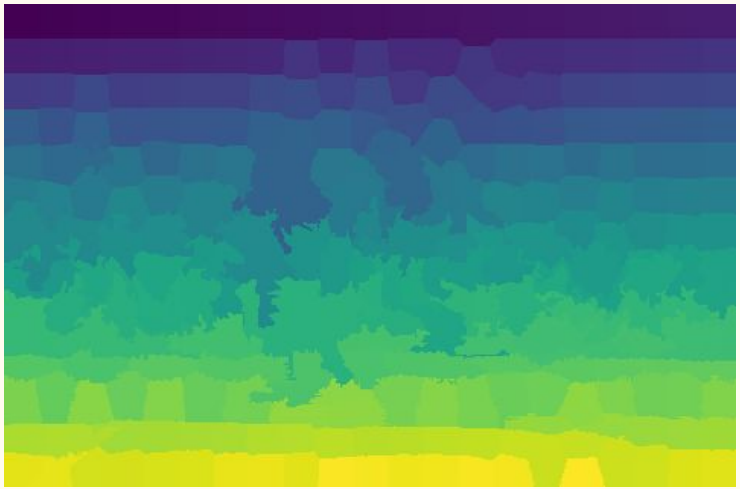
Such an evaluation is very slow and even for regions which have been merged, it becomes a redundant operation. To cope with this problem, the authors propose a variation to the graph which is formed by creating a Nearest neighbour alternative called the NNG (Nearest neighbour graph) *(left)*.

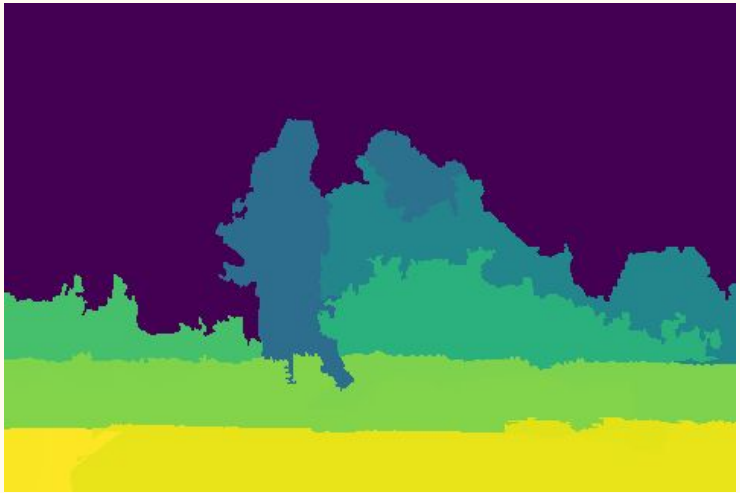


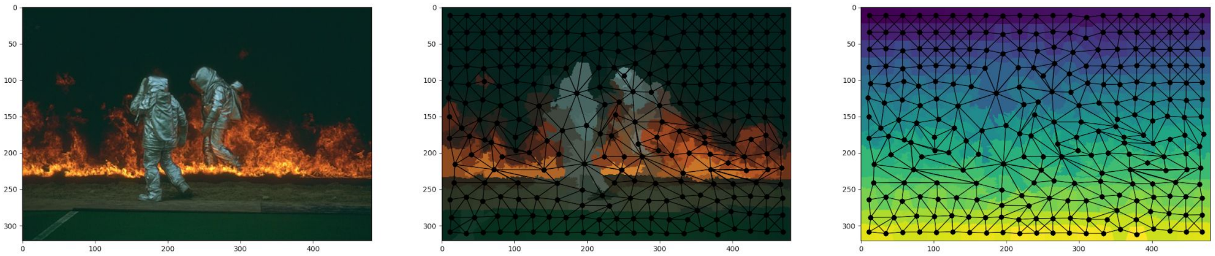
The nearest neighbour graph approach *(right)* follows the below algorithm by merging the region nodes into one and keeping track of the weight changes. Due to this, we also notice some desirable features in the search for new regions to be merged as shown below:

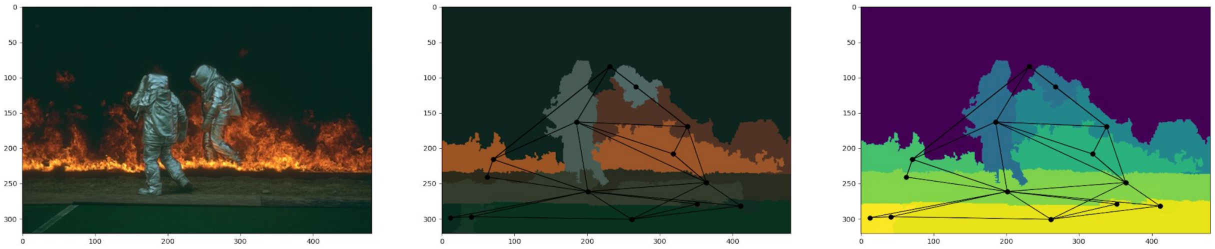
EXPERIMENTS

Results and Analysis

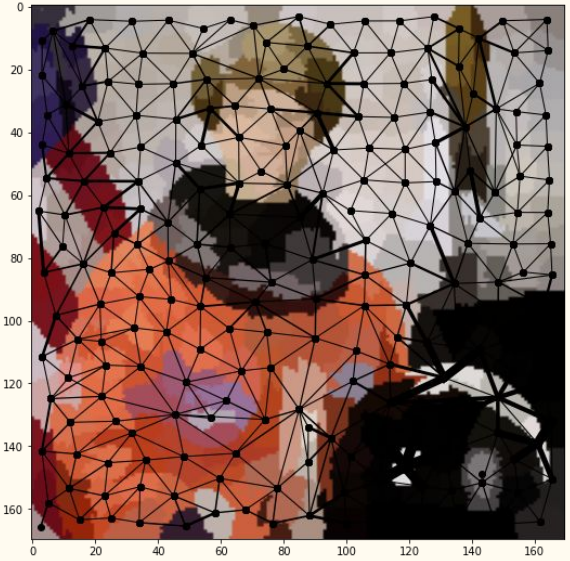


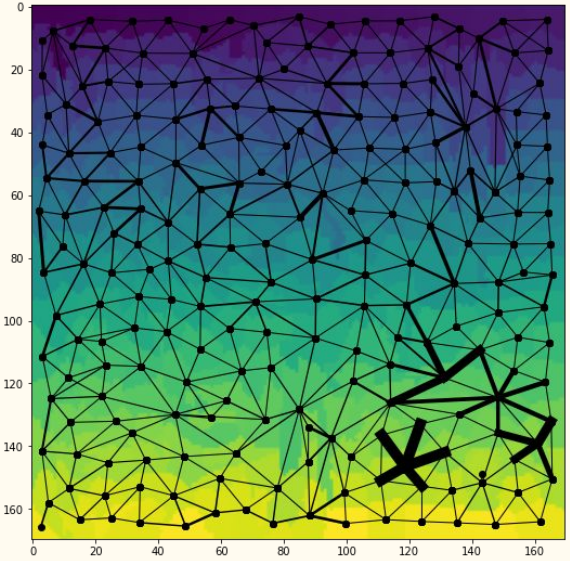


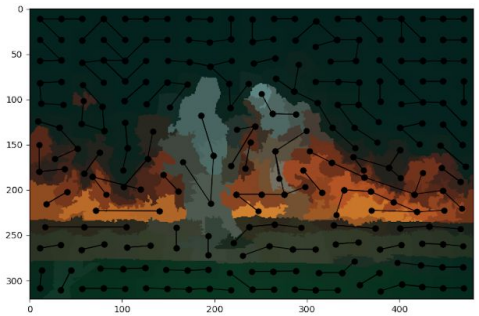
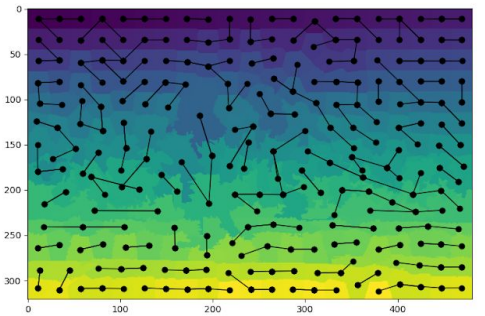

0
50
100
150
200
250
300
0 100 200 300 400

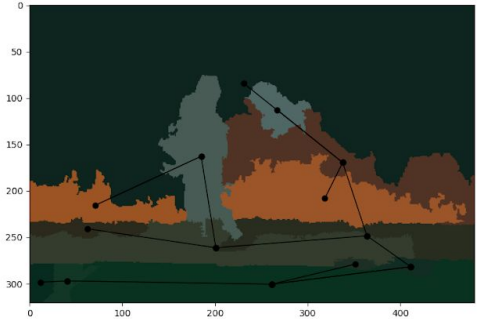
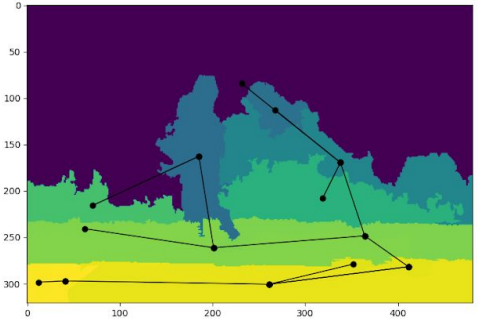

0
50
100
150
200
250
300
0 100 200 300 400

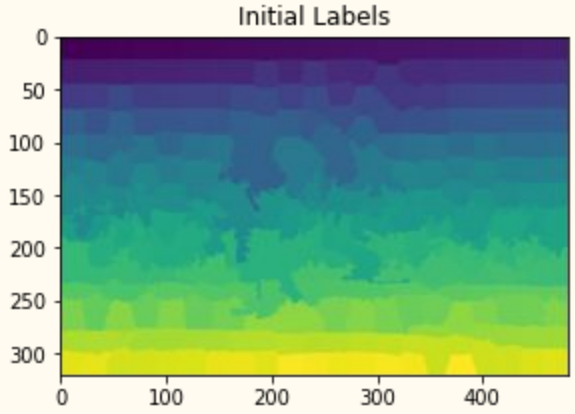

0
20
40
60
80
100
120
140
160
0
20
40
60
80
100
120
140
160

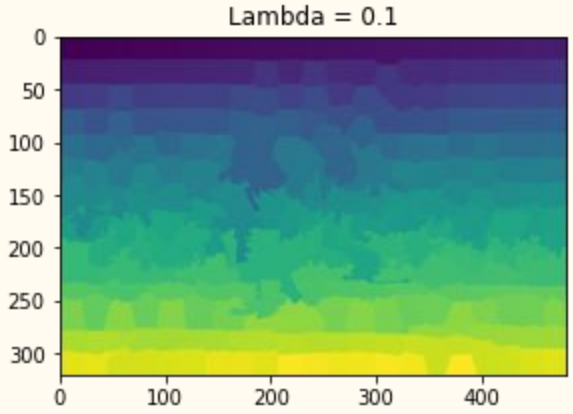

0
20
40
60
80
100
120
140
160
0
20
40
60
80
100
120
140
160

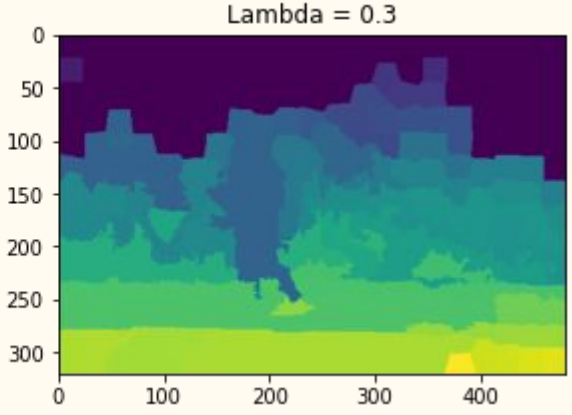

0
20
40
60
80
100
120
140
160
0
20
40
60
80
100
120
140
160

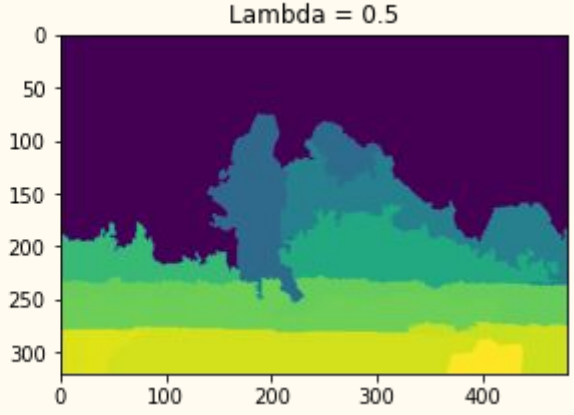




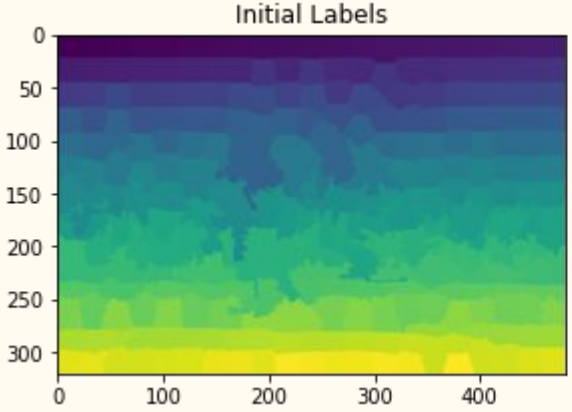



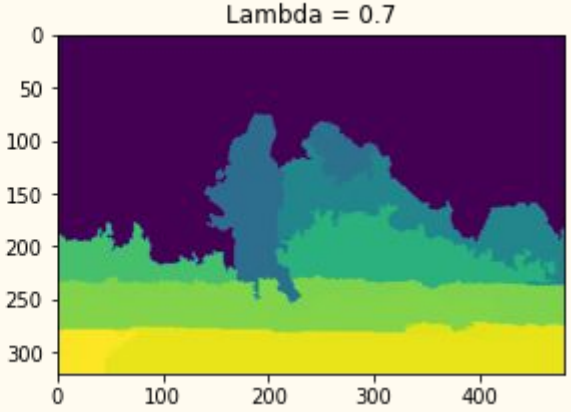

Initial Labels
0
50
100
150
200
250
300
0
100
200
300
400

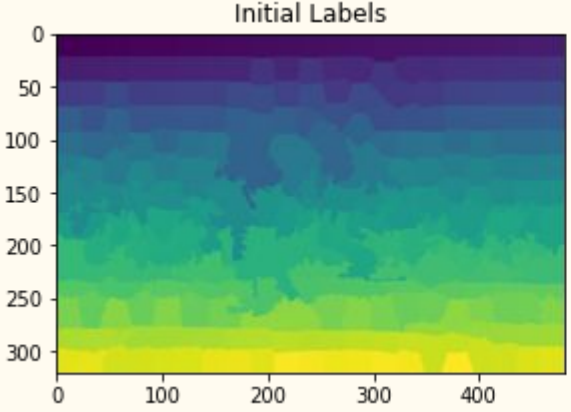

Lambda = 0.1
0
50
100
150
200
250
300
0
100
200
300
400

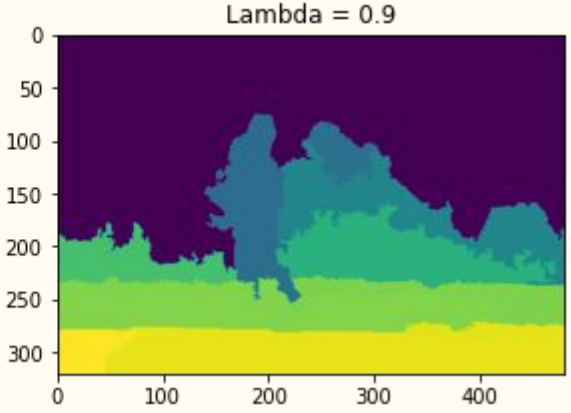

Lambda = 0.3
0
50
100
150
200
250
300
0
100
200
300
400


Lambda = 0.5
0
50
100
150
200
250
300
0
100
200
300
400

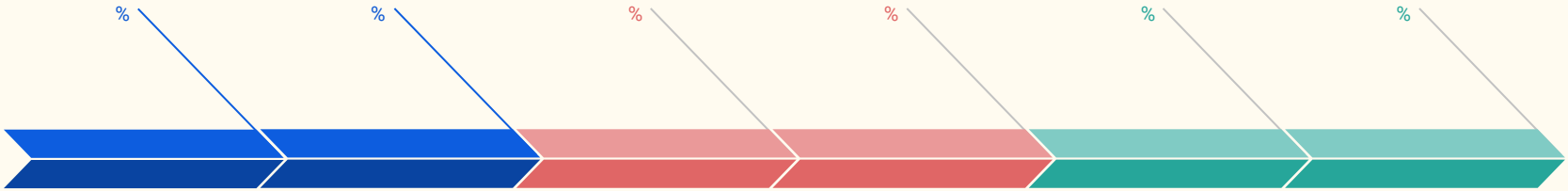

Initial Labels
0
50
100
150
200
250
300
0
100
200
300
400


Lambda = 0.7
0
50
100
150
200
250
300
0
100
200
300
400


Initial Labels
0
50
100
150
200
250
300
0
100
200
300
400


Lambda = 0.9
0
50
100
150
200
250
300
0
100
200
300
400

Division of Work



Watershed algorithm

Shanthika

RAG implementation

Shubham

SPRT Test

Sai Amrit

Dynamic Region Merging

Shubham

NNG for Region Merging

Shanthika
Shubham

Results

Sai Amrit

Thank You

Fin.