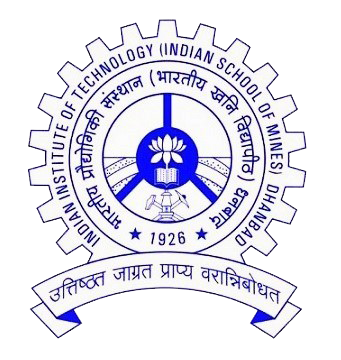
**INDIAN INSTITUTE OF TECHNOLOGY (INDIAN SCHOOL OF MINES) DHANBAD**

****

**PROJECT ON COLOR IMAGE SEGMENTATION**

**SUBMITTED TO,**

**DR. SUSHANTA MUKHOPADHYAY**

**ASSOCIATE PROFESSOR**

**DEPARTMENT OF COMPUTER SCIENCE**

**IIT(ISM) DHANBAD**

**SUBMITTED BY,**

**N NAGA SAI KRISHNA(ADMN NO. 14JE000157)**

**SAMANWAY DEY(ADMN NO. 14JE000237)**

**ABHINAV SRIHARI R(ADMN NO. 14JE000373)**

Certificate

This is to certify that this report is submitted by N Naga Sai Krishna, Samanway Dey,Abhinav Srihari.R students of Computer Science and Engineering, Indian Institute of Technology(Indian School of Mines), Dhanbad and they have successfully completed a project on Color Image Segmentation in 5th Semester of Academic year 2016-2017.

Dr. Sushanta Mukhopadhyay

…………………………

Associate Professor

Department of CSE

ISM Dhanbad

Acknowledgement

We express our deep sense of gratitude and respect towards our project mentor, Dr. Sushanta Mukhopadhyay, Associate Professor, Department of Computer Science and Engineering, , Indian Institute of Technology(Indian School of Mines), Dhanbad. We are very grateful for the generosity, expertise and guidance We have received from him while working on the project. Expressing our sincere thanks to our mentor, we would like to convey our sincere regards and gratitude for encouraging us to undertake this project and to work on it at time and again, it would not have been possible without him. We have all but enjoyed working on this project under his wings and We shall keep so for the times to come.

**Image segmenation:**

Image segmentation is a classic inverse problem that consists of achieving a compact region-based description of the image scene by decomposing it into meaningful or spatially coherent regions sharing similar attributes. This low-level vision task is often the preliminary (and also crucial) step in many video and computer vision applications, such as object localization or recognition, data compression, tracking, image retrieval, or under-standing

**Color image segmentation:**

Color image segmentation is useful in many applications. From the segmentation results, it is possible to identify regions of interest and objects in the scene, which is very beneficial to the subsequent image analysis or annotation .Recent work includes a variety of techniques:

For example, stochastic model based approaches morphological watershed based region growing, energy diffusion , and graph partitioning .Quantitative evaluation methods have also been suggested. However, due to the difficult nature of the problem , there are few automatic algorithms that can work well on a large variety of data.

**Application of color image segmentation:**

Some of the practical applications of image segmentation are:

Content-based image retrieval

[Machine vision](https://en.wikipedia.org/wiki/Machine_vision)

[Medical imaging](https://en.wikipedia.org/wiki/Medical_imaging)

Locate tumors and other pathologies

Measure tissue volumes

Diagnosis, study of anatomical structure

Surgery planning

Intra-surgery navigation

[Object detection](https://en.wikipedia.org/wiki/Object_detection)

[Face detection](https://en.wikipedia.org/wiki/Face_detection)

Brake light detection

Locate objects in satellite images (roads, forests, crops, etc.)

Recognition Tasks

[Face recognition](https://en.wikipedia.org/wiki/Face_recognition)

[Fingerprint recognition](https://en.wikipedia.org/wiki/Fingerprint_recognition)

[Iris recognition](https://en.wikipedia.org/wiki/Iris_recognition)

Traffic control systems

[Video surveillance](https://en.wikipedia.org/wiki/Video_surveillance)

Several general-purpose algorithms and techniques have been developed for image segmentation. To be useful, these techniques must typically be combined with a domain's specific knowledge in order to effectively solve the domain's segmentation problems.

**Various approach for color image segmentation:**

Image thresholding methods are popular due to their simplicity and efficiency. However, traditional histogram-based thresholding algorithms cannot separate those areas which have the same gray level but do not belong to the same part. In addition, they cannot process images whose histograms are nearly unimodal, especially when the target region is much smaller than the background area. Template

matching method becomes time consuming when the image becomes more complex or larger in size. Clustering method, viewing an image as a set of multi- dimensional data and classifying the image into different parts according to certain homogeneity criterion, can get much better results of segmentation. But over-segmentation is the problem that must be settled and feature extraction is an important factor for clustering. The edge detection method is one of the widely used approaches to the problem of image segmentation. It is based on the detection of points with abrupt changes at gray levels. The main disadvantages of the edge detection technique are that it does not work well when images have many edges, and it cannot easily identify a closed curve or boundary. Region growing algorithms deal with spatial repartition of the image feature information. In general, they perform better than the thresholding approaches for several sets of images. However, the typical region growing processes are inherently sequential. The regions produced depend both on the order in which pixels are scanned and on the value of pixels which are first scanned and gathered to define each new segment. In view of the problems mentioned above, plenty of approaches and their corresponding improvements have been proposed to ensure the accuracy and rapidity of image segmentation. But there is still much work to be done to overcome their drawbacks, and attempts at utilizing knowledge on other domains, especially artificial intelligence,should be highly appreciated.

**K-nearest neighbour algorithm:**

In our project , here k-NN algorithm is used. In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), the ***k*-Nearest Neighbor algorithm** (or ***k*-NN** for short) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification)and [regression](https://en.wikipedia.org/wiki/Regression_analysis).[[1]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-1) In both cases, the input consists of the *k* closest training examples in the [feature space](https://en.wikipedia.org/wiki/Feature_space). The output depends on whether *k*-NN is used for classification or regression:

In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

*k*-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

**CIELAB:**

CIELAB is the second of two systems adopted by CIE in 1976 as models that better showed uniform color spacing in their values. CIELAB is an opponent color system based on the earlier (1942) system of Richard Hunter called L, a, b. Color opposition correlates with discoveries in the mid-1960s that somewhere between the optical nerve and the brain, retinal color stimuli are translated into distinctions between light and dark, red and green, and blue and yellow. CIELAB indicates these values with three axes: L\*, a\*, and b\*. (The full nomenclature is 1976 CIE L\*a\*b\* Space.)

The central vertical axis represents lightness (signified as L\*) whose values run from 0 (black) to 100 (white). This scale is closely related to Munsell's value axis except that the value of each step is much greater. This is the same lightness valuation used in CIELUV.The color axes are based on the fact that a color can't be both red and green, or both blue and yellow, because these colors oppose each other. On each axis the values run from positive to negative. On the a-a' axis, positive values indicate amounts of red while negative values indicate amounts of green. On the b-b' axis, yellow is positive and blue is negative. For both axes, zero is neutral gray:

Therefore, values are only needed for two color axes and for the lightness or grayscale axis (L\*), which is separate (unlike in RGB, CMY or XYZ where lightness depends on relative amounts of the three color channels).CIELAB has become very important for desktop color. Like all CIE models, it is device independent (unlike RGB and CMYK), is the basic color model in Adobe PostScript (level 2 and level 3), and is used for color management as the device independent model of the ICC (International Color Consortium) device profiles.

**Code:**

**%Step 1: Acquire Image**

figure(1), imshow(fabric), title('fabric');

**%Step 2: Calculate Sample Colors in L\*a\*b\* Color Space for Each Region**

load regioncoordinates;

nColors = 6;

sample\_regions = false([size(fabric,1) size(fabric,2) nColors]);

for count = 1:nColors

sample\_regions(:,:,count) = roipoly(fabric,region\_coordinates(:,1,count),

region\_coordinates(:,2,count));

end

imshow(sample\_regions(:,:,2)),title('sample region for red');

lab\_fabric = rgb2lab(fabric);

a = lab\_fabric(:,:,2);

b = lab\_fabric(:,:,3);

color\_markers = zeros([nColors, 2]);

for count = 1:nColors

color\_markers(count,1) = mean2(a(sample\_regions(:,:,count)));

color\_markers(count,2) = mean2(b(sample\_regions(:,:,count)));

end

fprintf('[%0.3f,%0.3f] \n',color\_markers(2,1),color\_markers(2,2));

**%Step 3: Classify Each Pixel Using the Nearest Neighbor Rule**

color\_labels = 0:nColors-1;

a = double(a);

b = double(b);

distance = zeros([size(a), nColors]);

for count = 1:nColors

distance(:,:,count) = ( (a - color\_markers(count,1)).^2 + ...

(b - color\_markers(count,2)).^2 ).^0.5;

end

[~, label] = min(distance,[],3);

label = color\_labels(label);

clear distance;

**%Step 4: Display Results of Nearest Neighbor Classification**

rgb\_label = repmat(label,[1 1 3]);

segmented\_images = zeros([size(fabric), nColors],'uint8');

for count = 1:nColors

color = fabric;

color(rgb\_label ~= color\_labels(count)) = 0;

segmented\_images(:,:,:,count) = color;

end

**Input Image:**



imshow(segmented\_images(:,:,:,6)), title('green objects');



imshow(segmented\_images(:,:,:,2)), title('blue objects');



imshow(segmented\_images(:,:,:,1)), title('red objects');

