Colored Image Segmentation Using K-Mean Clustering Algorithm For Different Color Models

Introduction To Image Segmentation:

Image segmentation is the process of diving an image into multiple parts. This is typically used to identify objects or other relevant information in digital images. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse.

Image segmentation can be done by several ways:

1. Thresholding
2. Region Growing
3. Clustering

Clustering:

The K-means algorithm is an iterative technique that is used to partition an image into K clusters.

The function k-mean partitions data into k mutually exclusive clusters, and returns the index of the cluster to which it has assigned each observation.

Each cluster in the partition is defined by its member objects and by its centroid, or centre.

The basic algorithm is:

1. Pick *K* cluster centre, either randomly or based on some heuristic method, for example K-means++
2. Assign each pixel in the image to the cluster that minimizes the distance between the pixel and the cluster centre.
3. Re-compute the cluster centre by averaging all of the pixels in the cluster
4. Repeat steps 2 and 3 until convergence is attained (i.e. no pixels change clusters)

The different color models that are used in this experimental analysis are: (a) RGB model, (b) HSV,and (c) YCbCr model.

(a) The RGB Model

In the RGB model, an image consists of three independent image planes, one in each of the primary colours: red, green and blue. A particular colour is specified by the amount of each of the primary components present. Fig 1 shows the geometry of the RGB colour model for specifying colours using a Cartesian coordinate system. The greyscale spectrum, i.e. those colours made from equal amounts of each primary, lies on the line joining the black and white vertices.

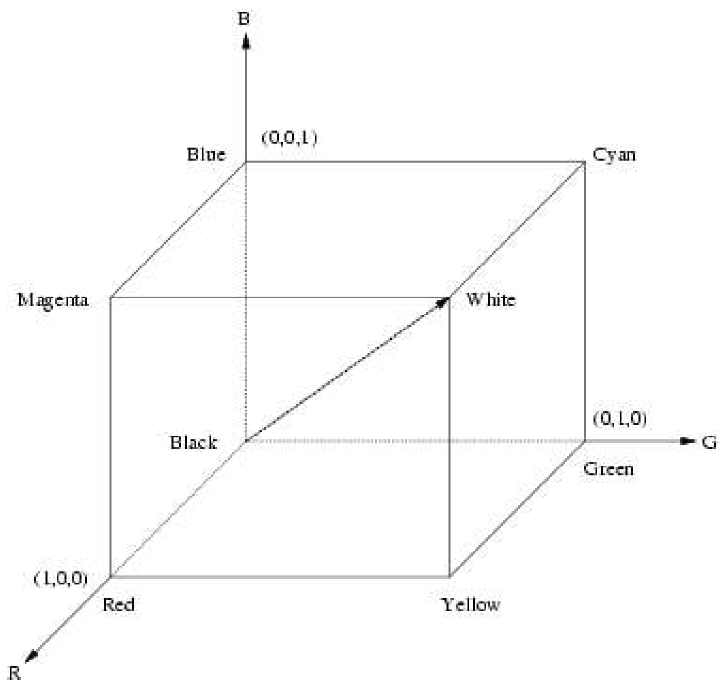


Fig 1: The RGB colour cube, the greyscale spectrum lies on the line joining the black and white vertices.

- This is an additive model, i.e. the colours present in the light add to form new colours, and is appropriate for the mixing of coloured light.
- The primary colors are added to form the three secondary colours yellow (red + green), cyan (blue + green) and magenta (red + blue), and white (red + green + blue).
- The RGB model is used for colour monitors and most video cameras.

(b) The HSV Model

HSV is closer to how humans perceive color. It has three components: hue, saturation, and value. This color space describes colors (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness value.

HUE: Hue is the color portion of the model, expressed as a number from 0 to 360 degrees: Red falls between 0 and 60 degrees, Yellow falls between 61 and 120 degrees, Green falls between 121-180 degrees, Cyan falls between 181-240 degrees, Blue falls between 241-300 degrees, Magenta falls between 301-360 degrees.

SATURATION: Saturation describes the amount of gray in a particular color, from 0 to 100 percent. Reducing this component toward zero introduces more gray and produces a faded effect. Sometimes, saturation appears as a range from just 0-1, where 0 is gray, and 1 is a primary color.

VALUE (OR BRIGHTNESS): Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0-100 percent, where 0 is completely black, and 100 is the brightest and reveals the most color.

(c) The YCbCr Model

YCbCr or Y'CbCr is a family of color spaces used as a part of the color pipeline in video and digital photography systems. Y is a luma component and Cb and Cr are the blue-difference and red-difference chroma components. Y′ is distinguished from Y, which is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries.

Y′CbCr color spaces are defined by a mathematical coordinate transformation from associated RGB color space.

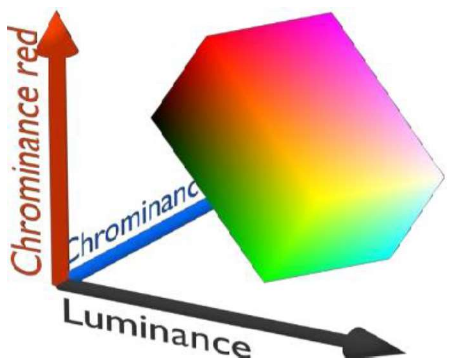
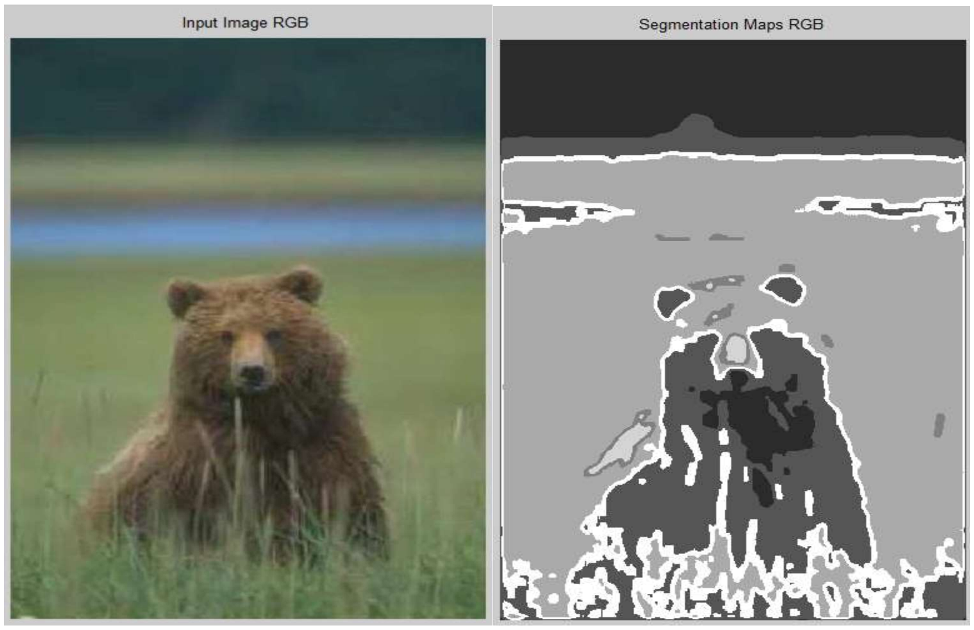


Fig 2: YCbCr color space

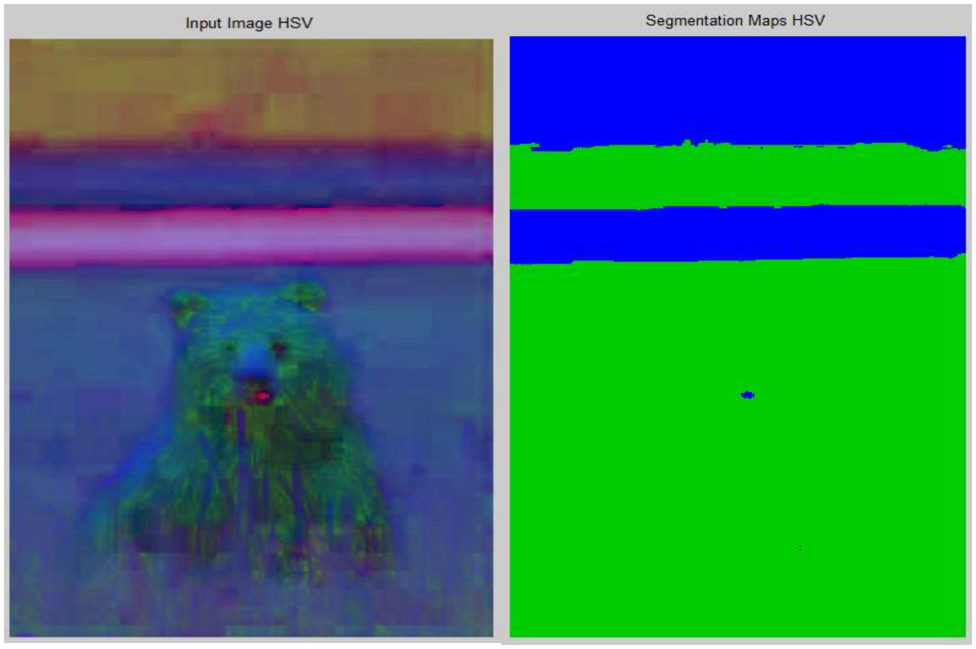
MATLAB code:

```
nBins=5;
winSize=7;
nClass=6;
inImg = imread('INPUT.JPG');
figure,imshow(inImg);title('Input Image RGB');
outImg = colImgSeg(inImg, nBins, winSize, nClass);
figure,imshow(outImg);title('Segmentation Maps RGB');
imghsv=rgb2hsv(inImg);
figure,imshow(imghsv);title('Input Image HSV');
I2 = im2double(inImg);
HSV = rgb2hsv(I2);
H = HSV(:,:,1); H = H(:);
S = HSV(:,:,2); S = S(:);
V = HSV(:,:,3); V = V(:);
idx = kmeans([H S V], 2);
figure,imshow(ind2rgb(reshape(idx, size(I2,1), size(I2, 2)), [0 0 1; 0 0.8 0]));
title('Segmentation Maps HSV');
img3=rgb2ycbcr(inImg);
figure,imshow(img3);title('Input Image YCBCR');
outImg3 = colImgSeg(img3, nBins, winSize, nClass);
figure,imshow(outImg3);title('Segmentation Maps ycbcr');
colormap('default');

% Function definition for image segmentation
function outImg = colImgSeg(inImg, nBins, winSize, nClass)

NbParam = nBins * nBins * nBins;
divis = 256 / nBins ;

s=size(inImg);
N=winSize;

n=(N-1)/2;
r=s(1)+2*n;
c=s(2)+2*n;
double temp(r,c,3);
temp=zeros(r,c,3);out=zeros(r,c,3);
coarseImg = zeros(r,c);
TabLabel = zeros(1,NbParam);
inrImg = rgb2gray(inImg);

temp((n+1):(end-n),(n+1):(end-n),1)=inImg(:,:,1);
temp((n+1):(end-n),(n+1):(end-n),2)=inImg(:,:,2);
temp((n+1):(end-n),(n+1):(end-n),3)=inImg(:,:,3);

temp_color = temp;

for x=n+1:s(1)+n
    for y=n+1:s(2)+n
        e=1;
        for k=x-n:x+n
            f=1;
            for l=y-n:y+n
                mat(e,f,1)=temp(k,l,1);
                mat(e,f,2)=temp(k,l,2);
                mat(e,f,3)=temp(k,l,3);
                f=f+1;
            end
            e=e+1;
        end

        sum_lab = 0;
        for i = 1 : winSize
            for j = 1 : winSize
                lab = floor(mat(i,j,1)/divis)*(nBins*nBins);
                lab = lab + floor(mat(i,j,2)/divis)*(nBins);
                lab = lab + floor(mat(i,j,3)/divis);
                lab = lab + 1;
                TabLabel(lab) = TabLabel(lab) + 1;
                sum_lab = sum_lab + lab;
            end
        end
        coarseImg(x,y) = floor(sum_lab / (winSize * winSize));

    end
end
trunCoarseImg(:,:) = coarseImg((n+1):(end-n),(n+1):(end-n));

tempVar = trunCoarseImg(:,:);
inImg_1D = double(tempVar(:));
fusedMap = kmeans(inImg_1D,nClass, 'EmptyAction', 'singleton');
fusedMapShow = uint8(fusedMap.*(255/nClass));
outImg = reshape(fusedMapShow,s(1),s(2));
```

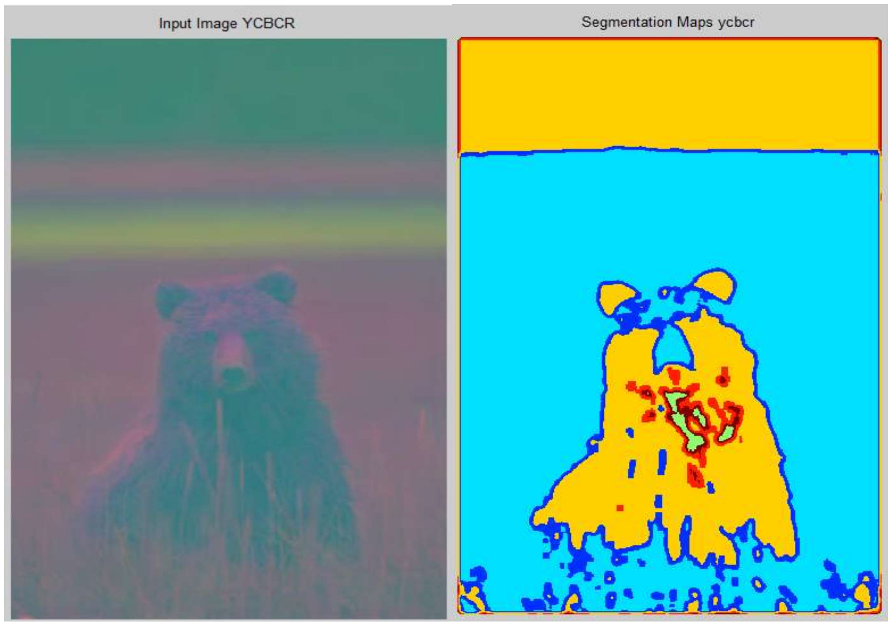
Outputs:


Input Image RGB
Segmentation Maps RGB

(a)


Input Image HSV
Segmentation Maps HSV

(b)



(c)

Fig 3: From left to right and top to bottom: Input image and their segmentation maps in different colour spaces (a) RGB, (b) HSV and (c) YCbCr

Conclusion:

By the simulation results, it can inferred that RGB color space gives the best color image segmentation compared to other color spaces.