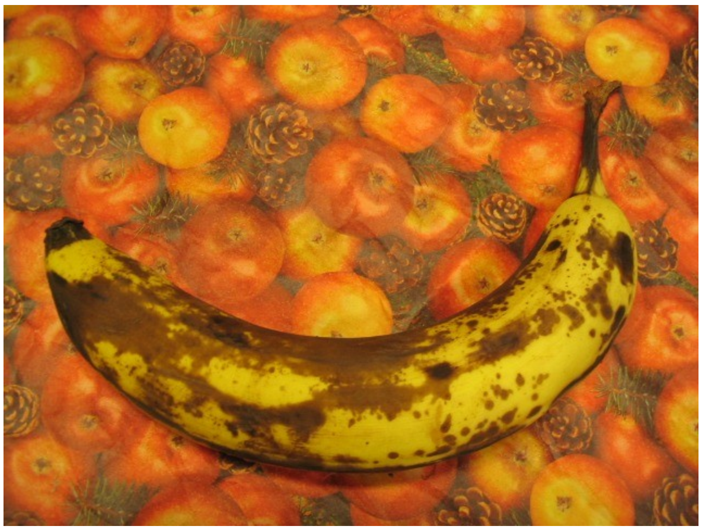
Computer Vision
Assignment 4

Udit Singh Parihar
2018701024

Image 1:



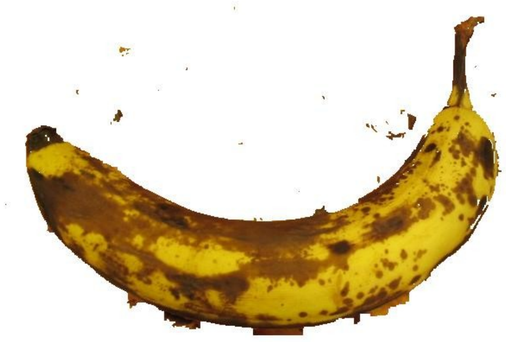
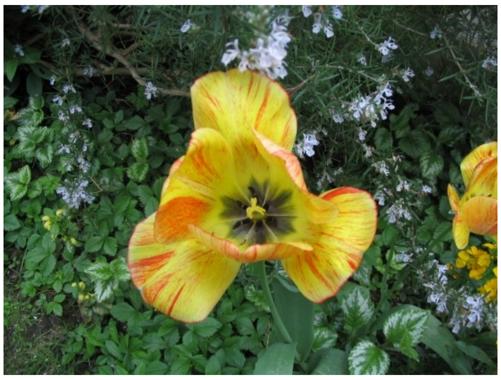


Image 2:



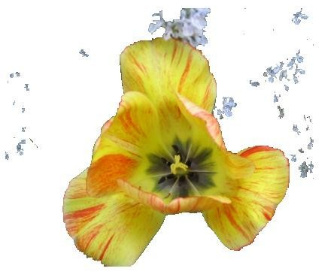


Image 3:

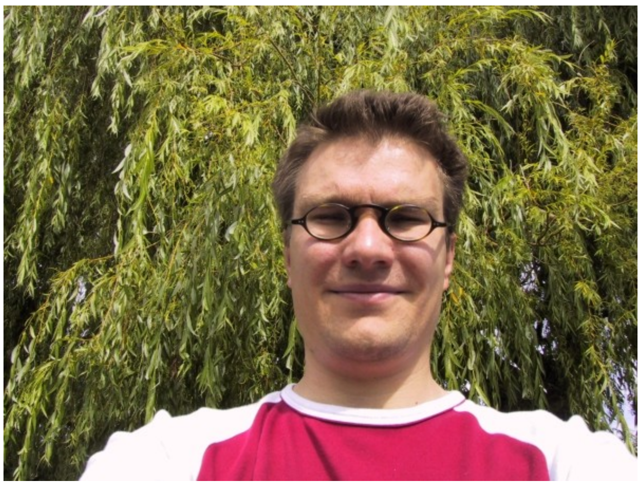
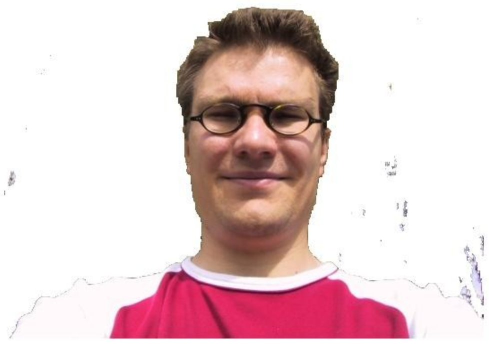
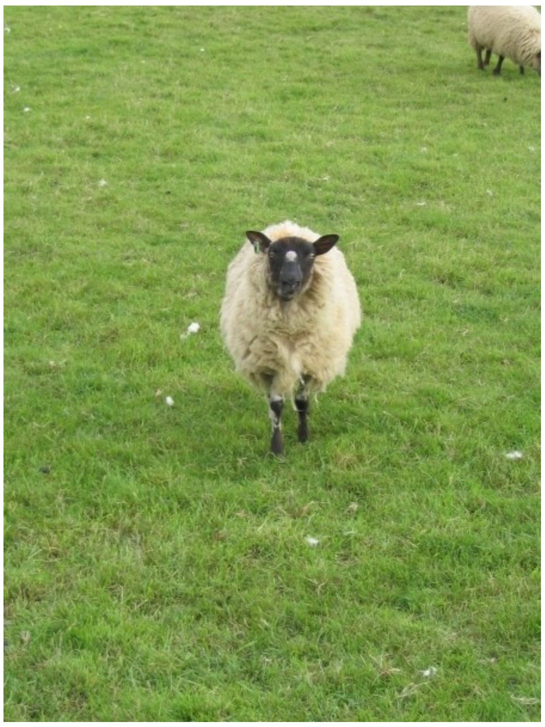
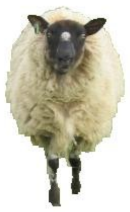



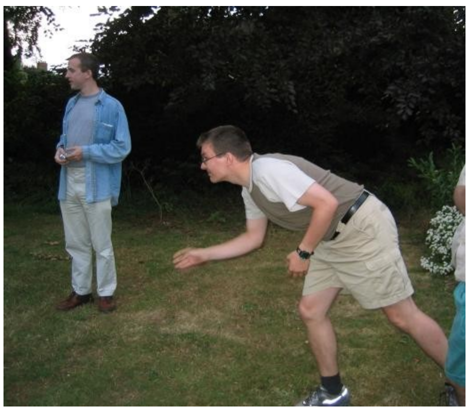
Image 4:

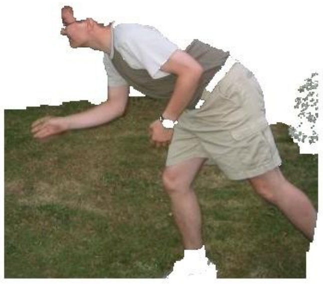
Effects of various parameters:

1. Color space:

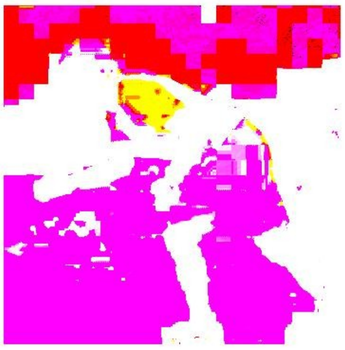
   1. Bull



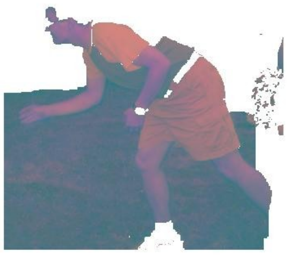
RGB



LAB

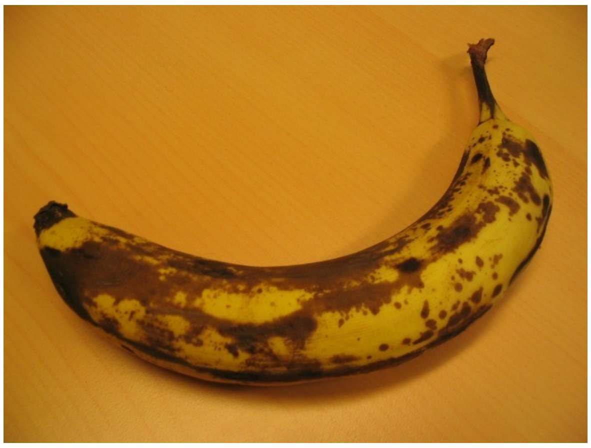


YCBCR

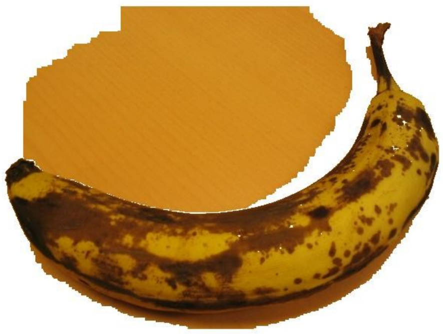


Summary : Here LAB and YcbCr shows a little bit deviation from RGB, but since his hair and background are of same color, so they are treated as background in gaussian model also. So a user interaction step can improve on this.

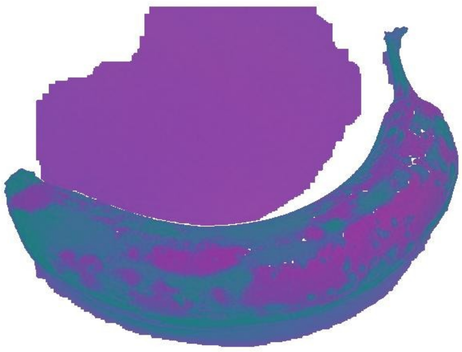
2. banana1



RGB



YcbCr:

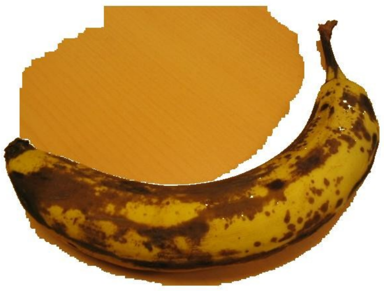


Summary: Here YcbCr is showing very small improvement(reduction) in background reduction. But still some region remains, because banana inside color also is same as the background. Hence small part of background is kept as in uncertain region and not made equal to 0(white).

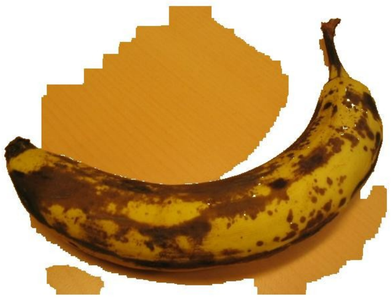
2. Gaussian Mixture Components:

1. Banana

5 Components

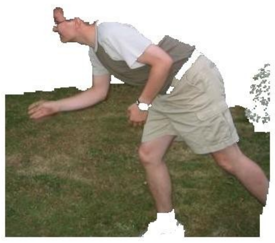


10 Components

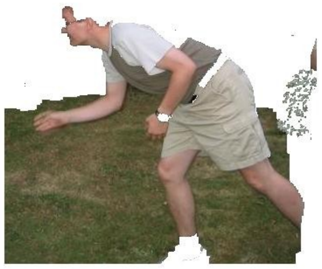


2. Bool

5 Components



10 Components



Summary : Changing gaussian mixture components reduce the iterations require to arrive at threshold point. Since we can say that after some number of gaussian mixture models, the further addition of components wont change much to area reduction.

Code:
Below is attached the main code in MATLAB, the remaining helper functions are in code directory in the folder attached.

```
function im_out = grabcut(img_name)

gamma = 20;
img = imread(img_name);

% Convergence criterion
E_CHANGE_THRES = 0.0001;

%%% Get image dimensions
[im_h, ~, ~] = size(img);

% im_in = rgb2lab(img);
im_in = rgb2ycbcr(img);
% im_in = img;
%-------------------------- I. Initialization

%%% User indicates background
[im_1d, alpha, im_sub] = select_back(im_in);

pix_U = alpha==1;

T_U = im_1d(pix_U, :);
pix_B = ~pix_U;
T_B = im_1d(pix_B, :);

%%% Initialize GMM
no_gauss = 5; % 5 Gaussians in each GMM
% Background
k_B = kmeans(T_B, no_gauss, 'Distance', 'cityblock', 'Replicates', 5);
gmm_B = fit_gmm(T_B, k_B);
% Foreground
k_U = kmeans(T_U, no_gauss, 'Distance', 'cityblock', 'Replicates', 5);
gmm_U = fit_gmm(T_U, k_U);

%-------------------------- II. Iterative Minimization

%%% Compute pairwise in one shot
pairwise = compute_pairwise(im_sub, gamma);
fprintf('Pairwise terms computed in one shot\n');

isConverged = 0;
E_prev = +Inf;
```

```
iter = 0;
while ~isConverged

    %------- 1. Assign GMM components to pixels

    [k_U, k_B] = assign_gauss(im_1d, pix_U, gmm_U, pix_B, gmm_B);

    %------- 2. Learn GMM parameters from data

    [gmm_U, gmm_B] = update_gmm(im_1d, pix_U, k_U, pix_B, k_B);

    %------- 3. Estimate segmentation: use min cut to solve

    [pix_U, E] = cut_Tu(pix_U, im_sub, alpha, gmm_U, gmm_B, pairwise);

    %%% Report progress
    E_change = (E_prev-E)/E_prev;
    iter = iter+1;
    fprintf('\n');
    fprintf('Iter %i done, E drops by %.3f%% (converged when < %.3f%%)\n', iter, E_change*100,
E_CHANGE_THRES*100);

    %%% Check convergence
    if E_change < E_CHANGE_THRES
        isConverged = 1;
    end

    %%% Update for next iteration
    pix_B = ~pix_U;
    E_prev = E;

    %------- Display current result

    im_out = im_in;
    im_out_1d = im_1d;
    % Set background to white
    im_out_1d(pix_B, :) = 255;
    % Assemble the 1D image back into 2D
    for idx = 1:size(im_out, 2)
        im_out(:, idx, :) = im_out_1d((idx-1)*im_h+1:idx*im_h, :);
    end
    imshow(im_out);
    drawnow;

end
```