Jerry Chen

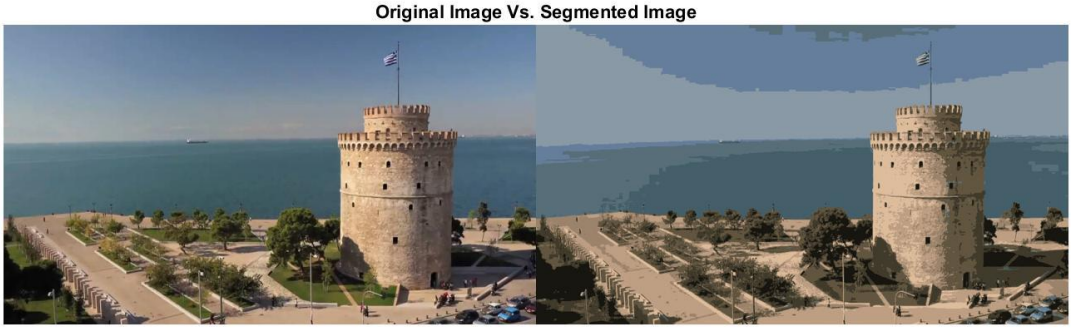
CS558

Computer Vision

Prof. Dunn

HW 3

For k means, picked 10 random triplets as seeds and computed the distance between the 10 points and every other point on the image. Picked the minimum distance to assign each point to a center and then recomputed new center using the average. Repeated until centers have converged and stopped changing after iteration. For slic, initialized with 50*50 blocks and shifted centers in 3*3 windows using least gradient. Then computed 5d distance for every point on the image and assigned each to its centroid with least distance. Repeated until convergence or max iterations.




```
%run k-means on image
```

```
I = imread('white-tower.png');
I = im2double(I);
k = 10;
[RGB_map,Seg] = kmeans(I,k);

montage({I,Seg})
title("Original Image Vs. Segmented Image")

%slic segmentation
Im = imread('wt_slic.png');
Im = im2double(Im);
max_it = 3;
[li,slic_seg] = slic(Im,max_it);
montage({slic_seg})
title("SLIC")
function [RGB_map,seg] = kmeans(img,k)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
seg = img;
%randomly choose 10 points
[ht,wd,ch] = size(img);
x = randi(ht,1,k);
y = randi(wd,1,k);
RGB_map = zeros(k,ch);
new_RGB = zeros(k,ch);
map = zeros(ht,wd,k);
for i = 1:k
    RGB_map(i,:) = img(x(i),y(i),:);
end
converging = true;
while converging
    %calculate distance
    for i = 1:k
        for h = 1:ht
            for w = 1:wd
                map(h,w,i) = sqrt((img(h,w,1)-
RGB_map(i,1)).^2+(img(h,w,2)-RGB_map(i,2)).^2+(img(h,w,3)-
RGB_map(i,3)).^2);
            end
        end
    end
    [~,I] = min(map,[],3);
    %new center
    for i = 1:k
        count = 0;
        for h = 1:ht
            for w = 1:wd
```

```
                if I(h,w) == i
                    new_RGB(i,1)  = new_RGB(i,1) + img(h,w,1);
                    new_RGB(i,2)  = new_RGB(i,2) + img(h,w,2);
                    new_RGB(i,3)  = new_RGB(i,3) + img(h,w,3);
                    count = count + 1;
                end
            end
        end
        new_RGB(i,:) = new_RGB(i,:)/count;
    end
    difference = abs(RGB_map-new_RGB);
    %check if converged
    if sum(difference) < 0.05
        converging = false;
        %produce segmented image
        for i = 1:k
            for h = 1:ht
                for w = 1:wd
                    if I(h,w) == i
                        seg(h,w,:) = RGB_map(i,:);
                    end
                end
            end
        end
    else
        RGB_map = new_RGB;
    end
end

end

function [I_map,seg] = slic(img,max_it)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
[ht,wd,~] = size(img);
seg = img;
%Divide into 50*50 blocks
b_num = (ht*wd)/2500;
%[x,y,R,G,B]
c_map = zeros(b_num,5);
converging = true;
it = 1;
for i = 1:ht/50
    x = 25+50*(i-1);
    for j = 1:wd/50
        y = 25+50*(j-1);
        idx = 15*(i-1)+j;
```

```
            c_map(idx,1) = x;
            c_map(idx,2) = y;
            c_map(idx,3:5) = img(x,y,:);
        end
    end



    while converging
        %compute gradient
        for j = 1:b_num
            window = zeros(3,3);

            for x = -1:1
                for y = -1:1
                    window(x+2,y+2) =
rgb_grad(c_map(j,1)+x,c_map(j,2)+y,img);
                end
            end
            [minw,idx] = min(window);
            [~,miny] = min(minw);
            minx = idx(miny);
            c_map(j,1:2) = c_map(j,1:2) + [minx-2,miny-2];
            c_map(j,3:5) = img(c_map(j,1),c_map(j,2),:);
        end
        d_map = zeros(ht,wd,b_num);
        for i = 1:b_num
            for h = 1:ht
                for w = 1:wd
                    d = c_map(i,:);
                    d_vect = [(h-d(1))/2,(w-d(2))/2,(img(h,w,1)-
d(3)),(img(h,w,2)-d(4)),(img(h,w,3)-d(5))];
                    d_map(h,w,i) = norm(d_vect);
                end
            end
        end
        [~,I_map] = min(d_map,[],3);
        newc_map = zeros(b_num,5);
        cluster_rgb = zeros(b_num,3);
        for i = 1:b_num
            count = 0;
            for h = 1:ht
                for w = 1:wd
                    if I_map(h,w) == i
                        rgb = img(h,w,:);
                        newc_map(i,1)  = newc_map(i,1) + h;
                        newc_map(i,2)  = newc_map(i,2) + w;
```

```
                    cluster_rgb(i,1) = cluster_rgb(i,1) +
rgb(1);
                    cluster_rgb(i,2) = cluster_rgb(i,2) +
rgb(2);
                    cluster_rgb(i,3) = cluster_rgb(i,3) +
rgb(3);
                    count = count + 1;
                end
            end
        end
        newc_map(i,1:2) = floor(newc_map(i,1:2)/count);
        colors = img(newc_map(i,1),newc_map(i,2),:);
        newc_map(i,3:5) = colors;
        cluster_rgb(i,:) = cluster_rgb(i,:)/count;
    end


    eq = isequal(c_map,newc_map);

    if ~eq && it < max_it
        c_map = newc_map;
        it = it + 1;
    else
        converging = false;
        for i = 1:b_num
            for h = 1:ht
                for w = 1:wd
                    if I_map(h,w) == i
                        if border(I_map,h,w)
                            seg(h,w,:) = [0,0,0];
                        else
                            seg(h,w,:) = cluster_rgb(i,:);
                        end
                    end
                end
            end
        end
    end
end



end


function grad = rgb_grad(x,y,img)
%UNTITLED4 Summary of this function goes here
```

```
%   Detailed explanation goes here

r_g = norm(img(x+1,y,1)-img(x-1,y,1),img(x,y+1,1)-img(x,y-1,1));
g_g = norm(img(x+1,y,2)-img(x-1,y,2),img(x,y+1,2)-img(x,y-1,2));
b_g = norm(img(x+1,y,3)-img(x-1,y,3),img(x,y+1,3)-img(x,y-1,3));
grad = norm([r_g,g_g,b_g]);
end

function bool = border(i_map,x,y)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
bool = false;
x_range = x+1;
y_range = y+1;
[ht,wd] = size(i_map);
if x_range > ht
    x_range = ht;
end
if y_range > wd
    y_range = wd;
end


pixel1 = i_map(x_range,y);
if pixel1 ~= i_map(x,y)
    bool = true;
end
pixel2 = i_map(x,y_range);
if pixel2 ~= i_map(x,y)
    bool = true;
end
end
```