Colin Keenan

ECE 5470

4-13-2020

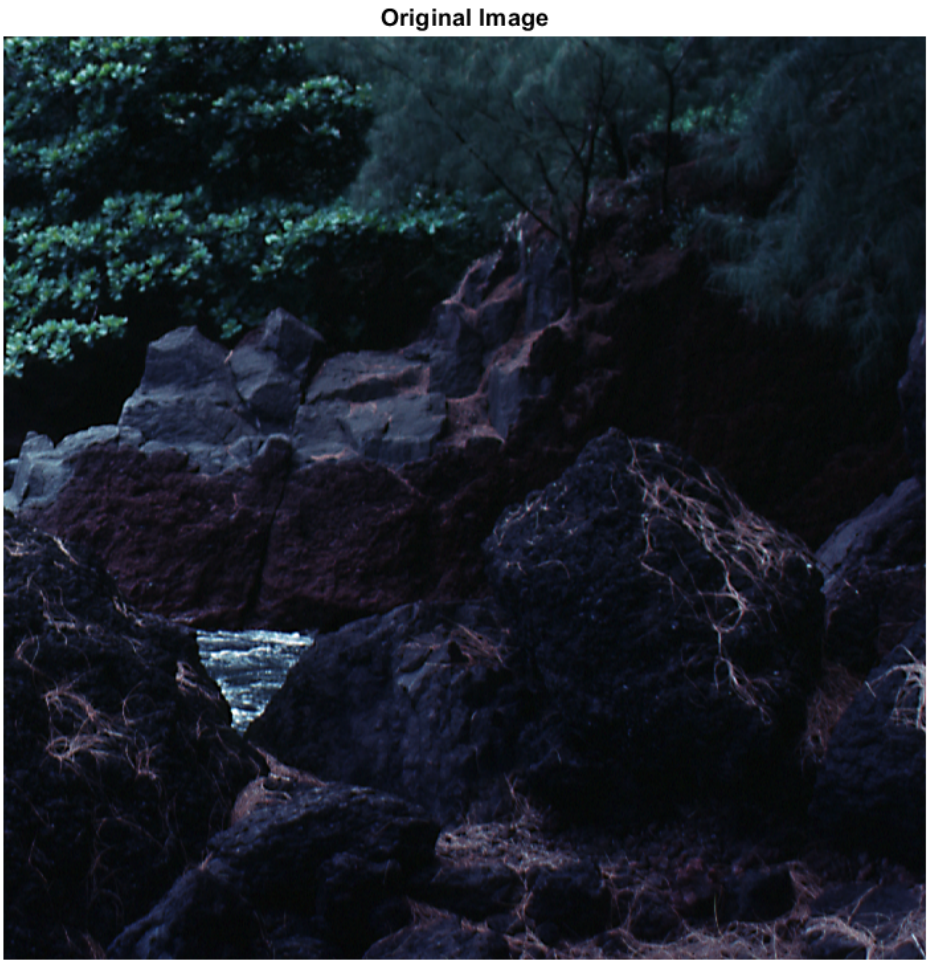
Homework 9

1. Tone and Color Correciton

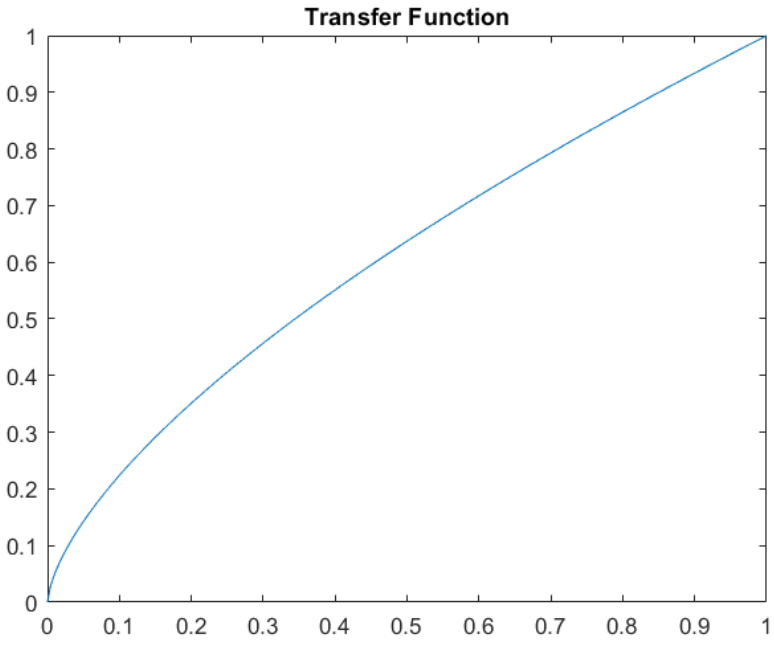
```
clear

fprintf("Question 1, Image 1:")
```

```
Question 1, Image 1:
```

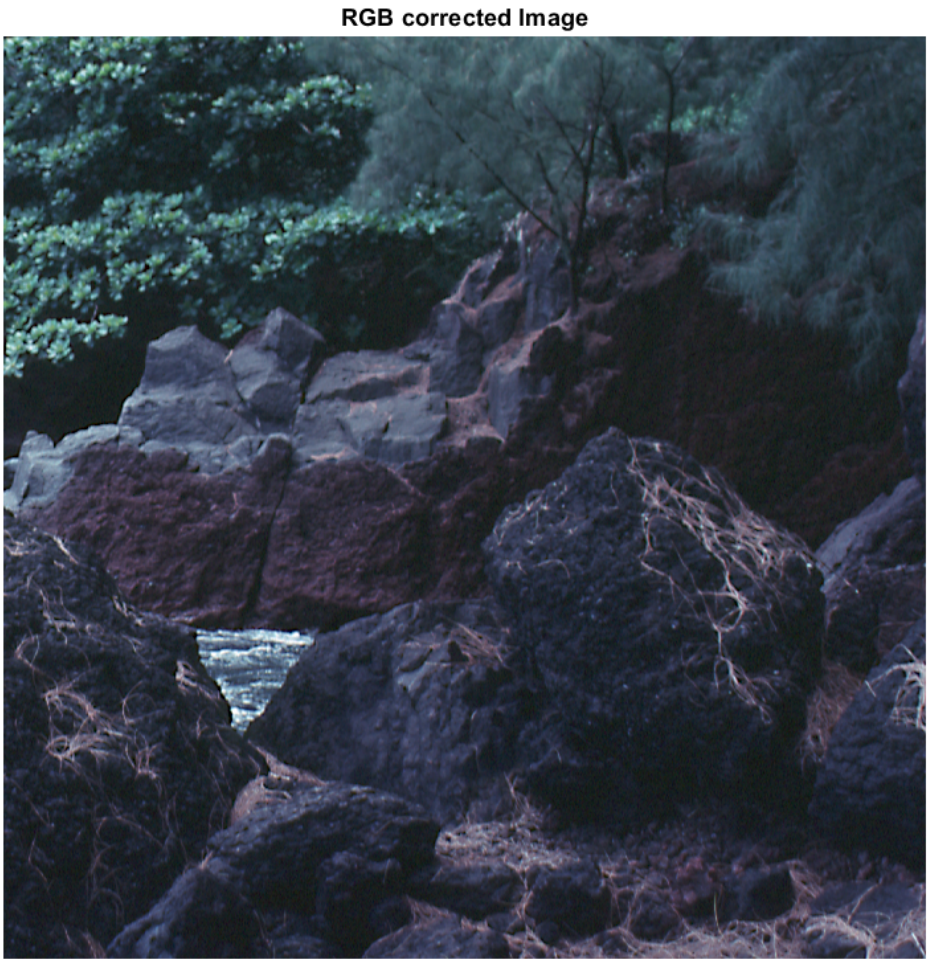
```
image = im2double(imread('Fig9-1a.tif'));

c = 1;      % c values from 0 to 10 with step size of 1
y = 0.65;    % gamma values from 0.01 to 2 with step size of 0.01

x = 0:1/255:1;
transform = c*(x.^y);
RGB = zeros(size(image));

for i = 1:size(image,1)
    for j = 1:size(image,2)
        RGB(i,j,1) = c * (image(i,j,1)^y);
        RGB(i,j,2) = c * (image(i,j,2)^y);
        RGB(i,j,3) = c * (image(i,j,3)^y);
    end
end

figure();
imshow(image);
title('Original Image');
```




```
figure();
plot(x,transform);
title('Transfer Function');
```



```
figure();
imshow(RGB);
title('RGB corrected Image');
```



Explanation: As you can see with the before and after images, given the shown transfer function for the power-law function, the darkest areas of the image is illuminated while no blowing out the other regions. One note, this image appears grainier now, unsure if this is due to the source image, or due to the processing of the power-law transform.

```
clear

fprintf("Question 1, Image 2:")
```

```
Question 1, Image 2:
```

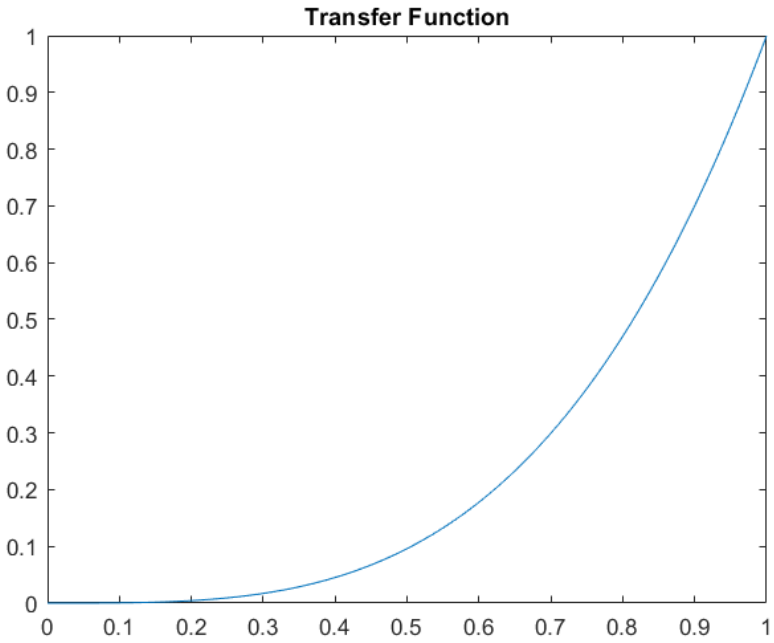
```
image = im2double(imread('Fig9-1b.tif'));

c = 1;      % c values from 0 to 10 with step size of 1
y = 3.38;    % gamma values from 0.01 to 5 with step size of 0.01

x = 0:1/255:1;
transform = c*(x.^y);
RGB = zeros(size(image));

for i = 1:size(image,1)
    for j = 1:size(image,2)
        RGB(i,j,1) = c * (image(i,j,1)^y);
        RGB(i,j,2) = c * (image(i,j,2)^y);
```

```
        RGB(i,j,3) = c * (image(i,j,3)^y);
    end
end

figure();
imshow(image);
```

```
Warning: Image is too big to fit on screen; displaying at 67%
```

```
title('Original Image');
```


```
figure();
plot(x,transform);
title('Transfer Function');
```




```
figure();
imshow(RGB);
```

```
Warning: Image is too big to fit on screen; displaying at 67%
```

```
title('RGB corrected Image');
```

Explanation: As you can see with the before and after images using a power-law transform, the brightness is turned down and the washed out look is eliminated. As a result, the new image is much richer in color.

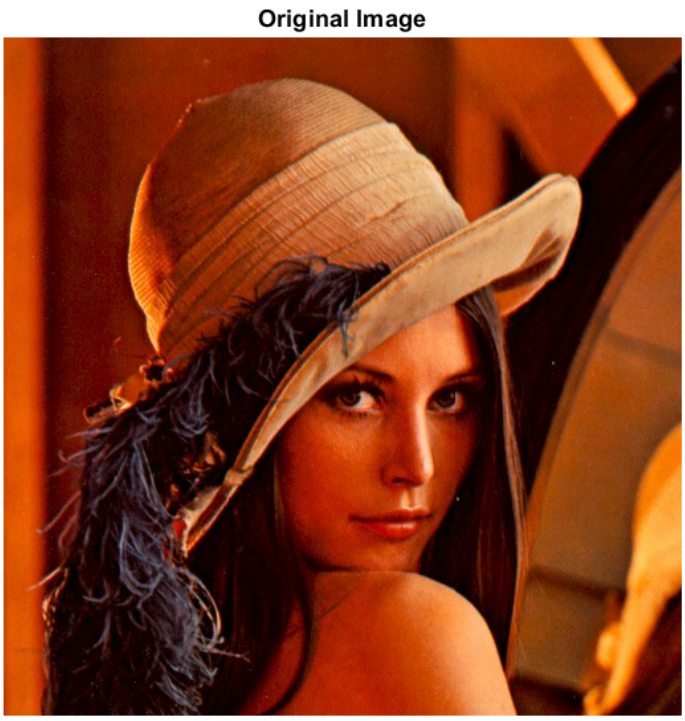
2. Smoothing

```
clear

fprintf("Question 2:")
```

```
Question 2:
```

```
image = im2double(imread('Fig9-2.tif'));

% Convert RGB image to HSI
[orig_H,orig_S,orig_I] = rgb2hsi(image(:,:,1), image(:,:,2), image(:,:,3));
HSI(:,:,1) = orig_H;
HSI(:,:,2) = orig_S;
HSI(:,:,3) = orig_I;

% Smooth both RGB and HSI
smooth_HSI = arithmeticMean5x5_3(HSI);
smooth_RGB = arithmeticMean5x5_3(image);

% Convert HSI back to RGB, note only Intensity part is using the smoothing
smooth_I = smooth_HSI(:,:,3);
[R,G,B] = hsi2rgb(orig_H,orig_S,smooth_I);
smooth_HSI(:,:,1) = R;
smooth_HSI(:,:,2) = G;
```

```
smooth_HSI(:,:,3) = B;

smooth_diff = abs(smooth_RGB - smooth_HSI);

figure();
imshow(image);
title('Original Image');
```

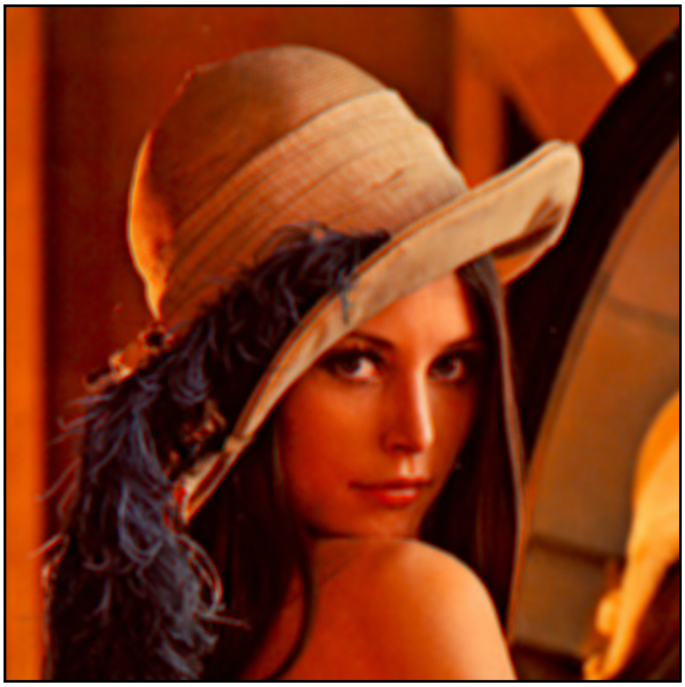



```
figure();
imshow(smooth_RGB);
title('RGB Smoothed Image');
```

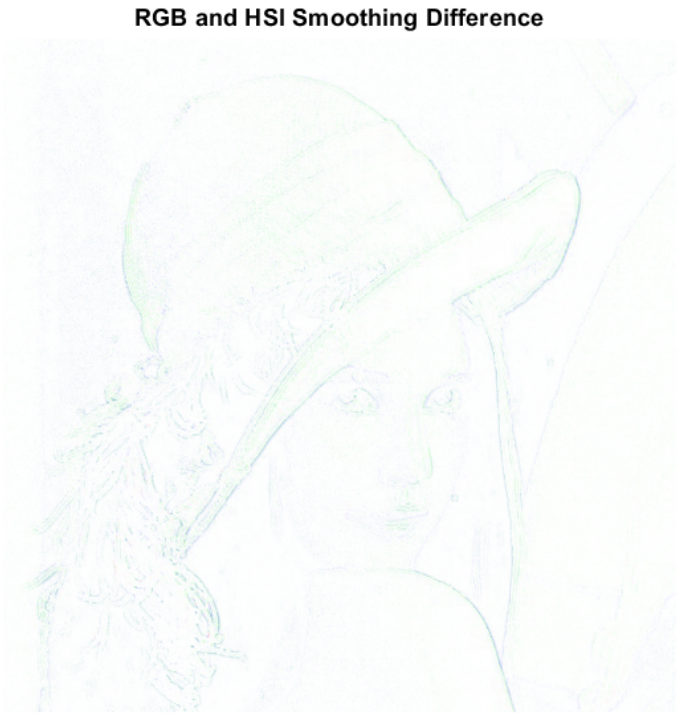
RGB Smoothed Image

```
figure();
imshow(smooth_HSI);
title('HSI Smoothed Image');
```

HSI Smoothed Image



```
figure();
imshow(1-smooth_diff);
title('RGB and HSI Smoothing Difference')
```



Explanation: For smoothing in the RGB domain, the 5x5 kernal was applied to every set of 25 pixels to each R, G, and B layers. For smoothing in the HSI domain, the 5x5 kernel was applied to only the Intensity layer and then reconverted back to RGB for display using the untouched H and S layers, along with the smoothed I layer. The inverse of their difference was shown to better illuminate their difference.

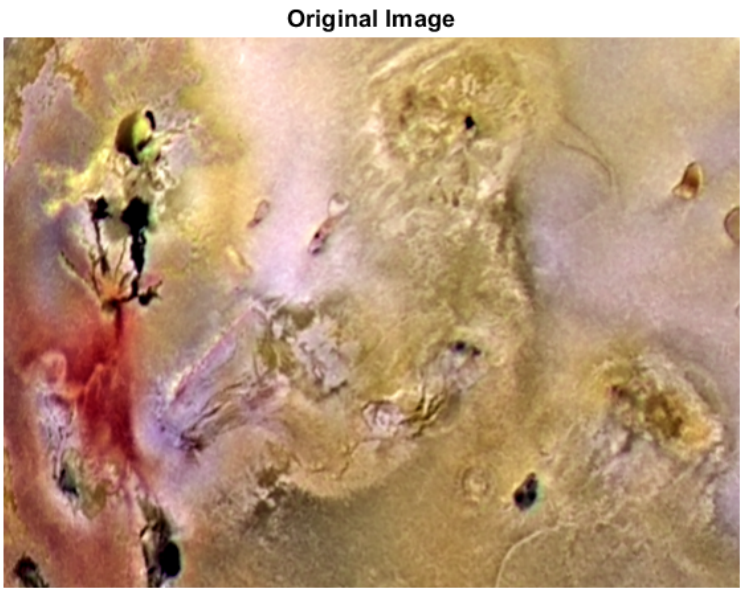
3. Image Segmentation

```
clear

fprintf("Question 3:")
```

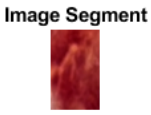
```
Question 3:
```

```
image = im2double(imread('Fig9-3.tif'));
segment = im2double(imread('Fig9-3b.tif'));

segment_avg = sum(sum(segment));
segment_avg = segment_avg ./ (size(segment,1) * size(segment,2));
aR = segment_avg(1,1,1);
aG = segment_avg(1,1,2);
aB = segment_avg(1,1,3);

D0 = 0.2;

image_mod = zeros(size(image));

for i = 1:size(image,1)
    for j = 1:size(image,2)
        zR = image(i,j,1);
        zG = image(i,j,2);
        zB = image(i,j,3);

        D = sqrt((zR-aR)^2 + (zG-aG)^2 + (zB-aB)^2);

        if(D <= D0)
```

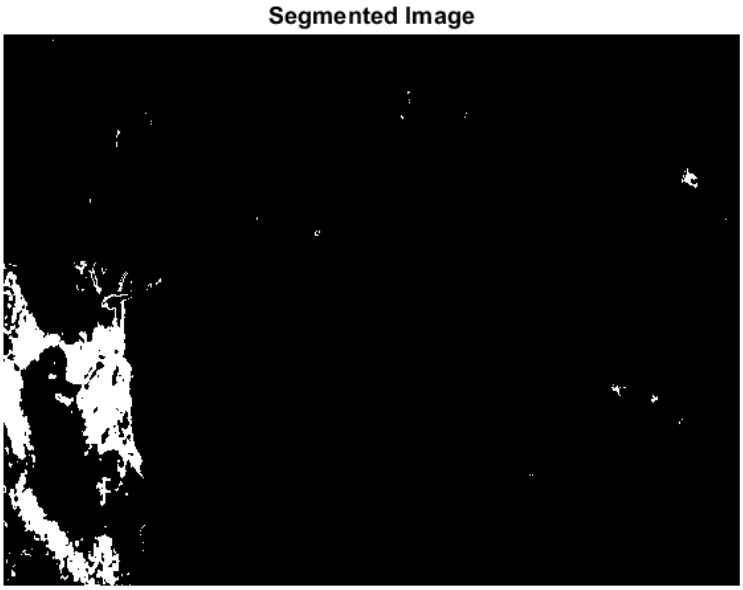
```
            image_mod(i,j,1) = 1;
            image_mod(i,j,2) = 1;
            image_mod(i,j,3) = 1;
        else
            image_mod(i,j,1) = 0;
            image_mod(i,j,2) = 0;
            image_mod(i,j,3) = 0;
        end
    end
end

figure();
imshow(image);
title('Original Image');
```




```
figure();
imshow(segment);
title('Image Segment');
```




```
figure();
imshow(image_mod);
title('Segmented Image');
```



Explanation: As you can see from the segmented image, only the regions that match the average color of the image segment are shown as white, everything else is turned to black. This can be helpful in illuminating any region in a picture that is similar to the color of what you want, or for a more practical application, this is how keying works when using/replacing a green-screen. It finds all of the colors similar to that green and replaces them with a new pixel which is usually an image.

Appendix (Functions Used)

```
function [H,S,I] = rgb2hsi(R,G,B)
    H = zeros(size(R));
    S = zeros(size(R));
    I = zeros(size(R));
    for x = 1:size(R,1)
        for y = 1:size(R,2)
            % Get RGB values for the given pixel
            Ri = R(x,y);
            Gi = G(x,y);
            Bi = B(x,y);

            % Calculate H
            num = 0.5*((Ri-Gi) + (Ri-Bi));
            den = sqrt(((Ri-Gi)^2 + (Ri-Bi)*(Gi-Bi)));
            theta = acos(num/den);
            thetaD = theta * (180/pi);

            if(Bi <= Gi)
                H(x,y) = thetaD;
            elseif(Bi > Gi)
                H(x,y) = 360 - thetaD;
            else
                print("error occurred in RGB to HSI processing")
            end

            % Calculate S
            S(x,y) = 1 - (((3)/(Ri+Gi+Bi)) * min([Ri Gi Bi]));

            % Calculate I
            I(x,y) = (1/3) * (Ri+Gi+Bi);

            if((Ri==Gi) && (Gi == Bi))
                H(x,y) = 0;
```

```
                S(x,y) = 0;
            end
            H(x,y) = H(x,y)/360;
        end
    end
end

function [R,G,B] = hsi2rgb(H,S,I)
    R = zeros(size(H));
    G = zeros(size(H));
    B = zeros(size(H));

    for x = 1:size(H,1)
        for y = 1:size(H,2)
            % Get HSI values for the given pixel
            Hi = H(x,y) * 360;
            Si = S(x,y);
            Ii = I(x,y);

            if((Hi >= 0) && (Hi <= 120))
                B(x,y) = Ii*(1-Si);
                R(x,y) = Ii * (1 + ((Si * cosd(Hi))/(cosd(60 - Hi))));
                G(x,y) = 3*Ii - (R(x,y)+B(x,y));

            elseif((Hi >= 120) && (Hi <= 240))
                Hi = Hi - 120;
                R(x,y) = Ii*(1-Si);
                G(x,y) = Ii * (1 + ((Si * cosd(Hi))/(cosd(60 - Hi))));
                B(x,y) = 3*Ii - (R(x,y)+G(x,y));

            elseif((Hi >= 240) && (Hi <= 360))
                Hi = Hi - 240;
                G(x,y) = Ii*(1-Si);
                B(x,y) = Ii * (1 + ((Si * cosd(Hi))/(cosd(60 - Hi))));
                R(x,y) = 3*Ii - (G(x,y)+B(x,y));

            end
        end
    end
end

function smoothed = arithmeticMean5x5_3(image)
    image = im2double(image);
    mask = [[1 1 1 1 1]
            [1 1 1 1 1]
            [1 1 1 1 1]
            [1 1 1 1 1]
            [1 1 1 1 1]];
    weight = sum(sum(mask));
    smoothed = zeros(size(image));

    for i = 1+2 : size(image,1)-2
        for j = 1+2 : size(image,2)-2

            frame_1 = [[image(i-2,j-2,1) image(i-2,j-1,1) image(i-2,j+0,1) image(i-2,j+1,1) image(i-2,j+2,1)]
                       [image(i-1,j-2,1) image(i-1,j-1,1) image(i-1,j+0,1) image(i-1,j+1,1) image(i-1,j+2,1)]
                       [image(i+0,j-2,1) image(i+0,j-1,1) image(i+0,j+0,1) image(i+0,j+1,1) image(i+0,j+2,1)]
                       [image(i+1,j-2,1) image(i+1,j-1,1) image(i+1,j+0,1) image(i+1,j+1,1) image(i+1,j+2,1)]
                       [image(i+2,j-2,1) image(i+2,j-1,1) image(i+2,j+0,1) image(i+2,j+1,1) image(i+2,j+2,1)]];

            frame_2 = [[image(i-2,j-2,2) image(i-2,j-1,2) image(i-2,j+0,2) image(i-2,j+1,2) image(i-2,j+2,2)]
                       [image(i-1,j-2,2) image(i-1,j-1,2) image(i-1,j+0,2) image(i-1,j+1,2) image(i-1,j+2,2)]
                       [image(i+0,j-2,2) image(i+0,j-1,2) image(i+0,j+0,2) image(i+0,j+1,2) image(i+0,j+2,2)]
                       [image(i+1,j-2,2) image(i+1,j-1,2) image(i+1,j+0,2) image(i+1,j+1,2) image(i+1,j+2,2)]
                       [image(i+2,j-2,2) image(i+2,j-1,2) image(i+2,j+0,2) image(i+2,j+1,2) image(i+2,j+2,2)]];
```

```
            frame_3 = [[image(i-2,j-2,3) image(i-2,j-1,3) image(i-2,j+0,3) image(i-2,j+1,3) image(i-2,j+2,3)]
                       [image(i-1,j-2,3) image(i-1,j-1,3) image(i-1,j+0,3) image(i-1,j+1,3) image(i-1,j+2,3)]
                       [image(i+0,j-2,3) image(i+0,j-1,3) image(i+0,j+0,3) image(i+0,j+1,3) image(i+0,j+2,3)]
                       [image(i+1,j-2,3) image(i+1,j-1,3) image(i+1,j+0,3) image(i+1,j+1,3) image(i+1,j+2,3)]
                       [image(i+2,j-2,3) image(i+2,j-1,3) image(i+2,j+0,3) image(i+2,j+1,3) image(i+2,j+2,3)]];

            masked_1 = frame_1 .* mask;
            masked_2 = frame_2 .* mask;
            masked_3 = frame_3 .* mask;
            new_level_1 = sum(sum(masked_1)) / weight;
            new_level_2 = sum(sum(masked_2)) / weight;
            new_level_3 = sum(sum(masked_3)) / weight;
            smoothed(i,j,1) = new_level_1;
            smoothed(i,j,2) = new_level_2;
            smoothed(i,j,3) = new_level_3;
        end
    end
end
```