Image & Video Processing

Assignment 1

Albert Cerfeda

Contents

19 March 2023
Università della Svizzera italiana
Faculty of Informatics
Switzerland

1 Point Operations [20 points]

1.1 Tone mapping & Linearization [2 points]

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

1.2 Color correction [4 points]

1.3 Histograms [1 point]

1.4 Histogram Equalization [5 points + 2 Bonus]

1.5 Manual histogram equalization [2 points]

1.6 Thresholding & matting [4 points]

1.7 How many bits is enough [2 points]

2 Bonus: use your own pictures [3 points]

(a) Original `ferrari.jpg`.

(b) `ferrari_lin.jpg`. Linearized contrast

(c) `ferrari_bri.jpg`. Scaled brightness

(d) `ferrari_con.jpg`. Exponential contrast.

Figure 1: Tonemapping procedure comparison.

```
im = imread("./media/ferrari.JPG");
im = double(im)./255;
imwrite(uint8(255.*im), "./out/1.ferrari.jpg");

% 1.1 Map pixels to 0-1 range and linearize image back
im_lin = im.^2.2;
imwrite(uint8(255.*im_lin), "./out/1.ferrari_lin.jpg");

% 1.2 Increase brightness
im_bri = im.*2;
imwrite(uint8(255.*im_bri), "./out/1.ferrari_bri.jpg");

% 1.3 Enhance contrast using exponential function
im_con = im.^0.7;
imwrite(uint8(255.*im_con), "./out/1.ferrari_con.jpg");
figure()
imshow([im, im_lin,im_bri,im_con], []);
```

Figure 2: Matlab code performing various intensity transformations on the image `ferrari.jpg`.

3 Reproducing the results

In the `src` folder you can find a `Makefile`. Run `make` to execute the matlab scripts and store the results in the `src/out` folder.
Alternatively you can run individual scripts e.g `make src/script1.m`.

(a) Original image (b) Pixel-based correction (c) Gray-world assumption correction

Figure 3: Color correction procedures comparison.

```
im = imread("./media/white_balance_input.jpg");
im = (double(im)./255);
imwrite(uint8(255.*im), "./out/2.wb.jpg");

channelmean = mean(mean(im))
% 2.1 Pixel-based correction
figure()
imshow(im);
coords = int32(ginput(1));
px_color = im(coords(2), coords(1), 1:3);
gain = px_color./channelmean;
im_pxcorr = im.*gain;
imwrite(uint8(255.*im_pxcorr), "./out/2.wb_pxcorr.jpg");

% 2.2 Gray-world assumption
gain = 0.5./channelmean;
im_gwa = im.*gain;
imwrite(uint8(255.*im_gwa), "./out/2.wb_gwa.jpg");

figure()
imshow([im, im_pxcorr, im_gwa])
```

Figure 4: Matlab code performing various intensity transformations on the image `ferrari.jpg`.

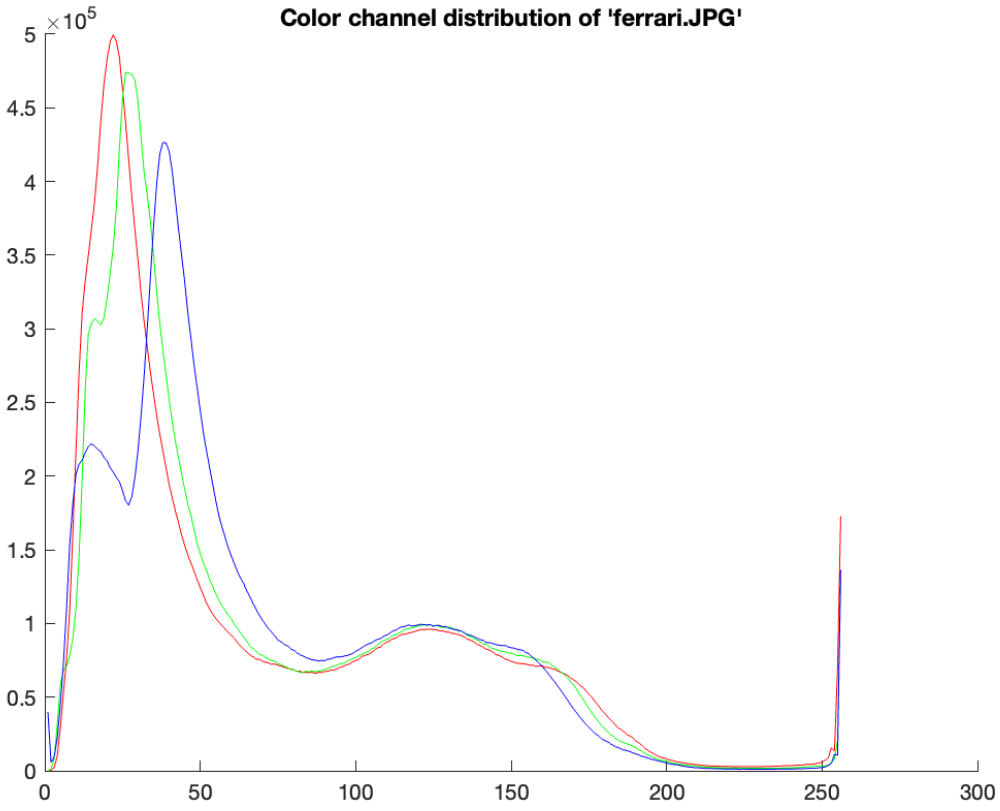


Figure 5: Color channel distribution of the image `ferrari.JPG`.

(a) Original image.

(b) Chroma Key mask.

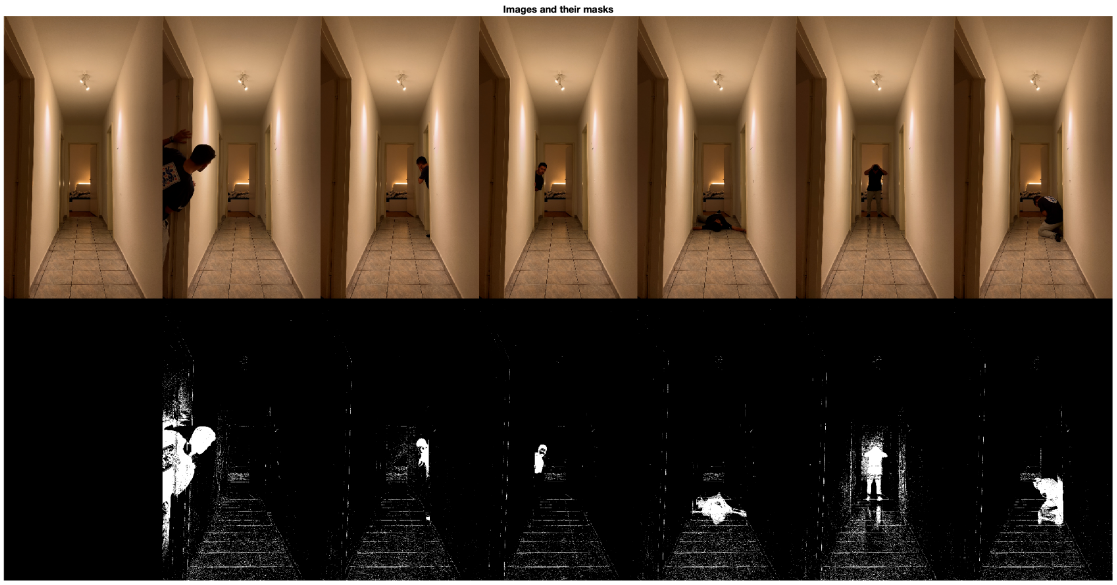
(c) Background image.

(d) Composited result image.

Figure 6: Chroma key compositing.

```
im = imread("./media/cat.jpg");
imhsv = rgb2hsv(im);
im_spacebg = imread("./media/spacebg.jpg");

imwrite(im, "./out/6.cat.jpg")
imwrite(im_spacebg, "./out/6.spacebg.jpg")
im = (double(im)./255);
im_spacebg = (double(im_spacebg)./255);

% Lets the user select a pixel in the image to key out.
figure()
imshow(im);
coords = int32(ginput(1));
px_hsv = imhsv(coords(2), coords(1), 1:3);

%  Each channel may deviate of these amounts from the selected pixel.
hue_deviation = 0.05;
saturation_deviation = 0.1; % += 10% saturation
brightness_deviation = 0.4; % += 40% darker/lighter

% Creates the mask for each pixel whose channels are within the deviations
mask = ~((imhsv(:,:,1) > px_hsv(1) - hue_deviation) & (imhsv(:,:,1) < px_hsv(1) + hue_deviation) & .
        (imhsv(:,:,2) > px_hsv(2) - saturation_deviation) & (imhsv(:,:,2) < px_hsv(2) + saturation_d
        (imhsv(:,:,3) > px_hsv(3) - brightness_deviation) & (imhsv(:,:,3) < px_hsv(3) + brightness_d

im_mask = comp(ones(size(im)),mask,zeros(size(im)));
im_comp = comp(im,mask,im_spacebg);

imshow([im, im_mask, im_comp], []);
title("Original image, mask and composited image")
saveas(gcf,'out/6.image_mask_comp.png')

imwrite(uint8(255.*im_mask), "./out/6.mask.jpg");
imwrite(uint8(255.*im_comp), "./out/6.comp.jpg");


function res = comp(a,mask,b)
    % Masks image 'a' on top of image 'b'
    res = a.*mask + b.*(~mask);
end
```

Figure 7: Matlab code performing background replacement on image `cat.jpg`.



(a) Pictures and relative masks.

(b) Final composited image

Figure 8: Compositing myself mutiple times on the same cleanplate.

```
im = imread("./media/cat.jpg");
imhsv = rgb2hsv(im);
im_spacebg = imread("./media/spacebg.jpg");

imwrite(im, "./out/6.cat.jpg")
imwrite(im_spacebg, "./out/6.spacebg.jpg")
im = (double(im)./255);
im_spacebg = (double(im_spacebg)./255);

% Lets the user select a pixel in the image to key out.
figure()
imshow(im);
coords = int32(ginput(1));
px_hsv = imhsv(coords(2), coords(1), 1:3);

%  Each channel may deviate of these amounts from the selected pixel.
hue_deviation = 0.05;
saturation_deviation = 0.1; % += 10% saturation
brightness_deviation = 0.4; % += 40% darker/lighter

% Creates the mask for each pixel whose channels are within the deviations
mask = ~((imhsv(:,:,1) > px_hsv(1) - hue_deviation) & (imhsv(:,:,1) < px_hsv(1) + hue_deviation) & .
        (imhsv(:,:,2) > px_hsv(2) - saturation_deviation) & (imhsv(:,:,2) < px_hsv(2) + saturation_d
        (imhsv(:,:,3) > px_hsv(3) - brightness_deviation) & (imhsv(:,:,3) < px_hsv(3) + brightness_d

im_mask = comp(ones(size(im)),mask,zeros(size(im)));
im_comp = comp(im,mask,im_spacebg);

imshow([im, im_mask, im_comp], []);
title("Original image, mask and composited image")
saveas(gcf,'out/6.image_mask_comp.png')

imwrite(uint8(255.*im_mask), "./out/6.mask.jpg");
imwrite(uint8(255.*im_comp), "./out/6.comp.jpg");


function res = comp(a,mask,b)
    % Masks image 'a' on top of image 'b'
    res = a.*mask + b.*(~mask);
end
```

Figure 9: Matlab code performing background replacement on image `cat.jpg`.