

# Image & Video Processing

## Assignment 1

Albert Cerfeda

## Contents

<b>1</b>	<b>Point Operations [20 points]</b>	<b>1</b>
1.1	Tone mapping & Linearization [2 points] . . . . .	1
1.2	Color correction [4 points] . . . . .	2
1.3	Histograms [1 point] . . . . .	3
1.4	Histogram Equalization [5 points + 2 Bonus] . . . . .	3
1.5	Manual histogram equalization [2 points] . . . . .	3
1.6	Thresholding & matting [4 points] . . . . .	3
1.7	How many bits is enough [2 points] . . . . .	3
<b>2</b>	<b>Bonus: use your own pictures [3 points]</b>	<b>3</b>
<b>3</b>	<b>Reproducing the results</b>	<b>3</b>



# 1 Point Operations [20 points]

## 1.1 Tone mapping & Linearization [2 points]



Figure 1: Tonemapping procedure comparison.

---

```

1 im = imread("./media/ferrari.JPG");
2 im = double(im)/255;
3 imwrite(uint8(255.*im), "./out/1.ferrari.jpg");
4
5 % 1.1 Map pixels to 0-1 range and linearize image back
6 im_lin = im.^2.2;
7 imwrite(uint8(255.*im_lin), "./out/1.ferrari_lin.jpg");
8
9 % 1.2 Increase brightness
10 im_bri = im.*2;
11 imwrite(uint8(255.*im_bri), "./out/1.ferrari_bri.jpg");
12
13 % 1.3 Enhance contrast using exponential function
14 im_con = im.^0.7;
15 imwrite(uint8(255.*im_con), "./out/1.ferrari_con.jpg");
16 figure()
17 imshow([im, im_lin, im_bri, im_con], []);

```

---

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

## 1.2 Color correction [4 points]



(a) Original image

(b) Pixel-based correction

(c) Gray-world assumption correction

Figure 3: Color correction procedures comparison.

```

1 im = imread("./media/white_balance_input.jpg");
2 im = (double(im)./255);
3 imwrite(uint8(255.*im), "./out/2.wb.jpg");
4
5 channelmean = mean(mean(im))
6 % 2.1 Pixel-based correction
7 figure()
8 imshow(im);
9 coords = int32(ginput(1));
10 px_color = im(coords(2), coords(1), 1:3);
11 gain = px_color./channelmean;
12 im_pxcorr = im.*gain;
13 imwrite(uint8(255.*im_pxcorr), "./out/2.wb_pxcorr.jpg");
14
15 % 2.2 Gray-world assumption
16 gain = 0.5./channelmean;
17 im_gwa = im.*gain;
18 imwrite(uint8(255.*im_gwa), "./out/2.wb_gwa.jpg");
19
20 figure()
21 imshow([im, im_pxcorr, im_gwa])

```

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

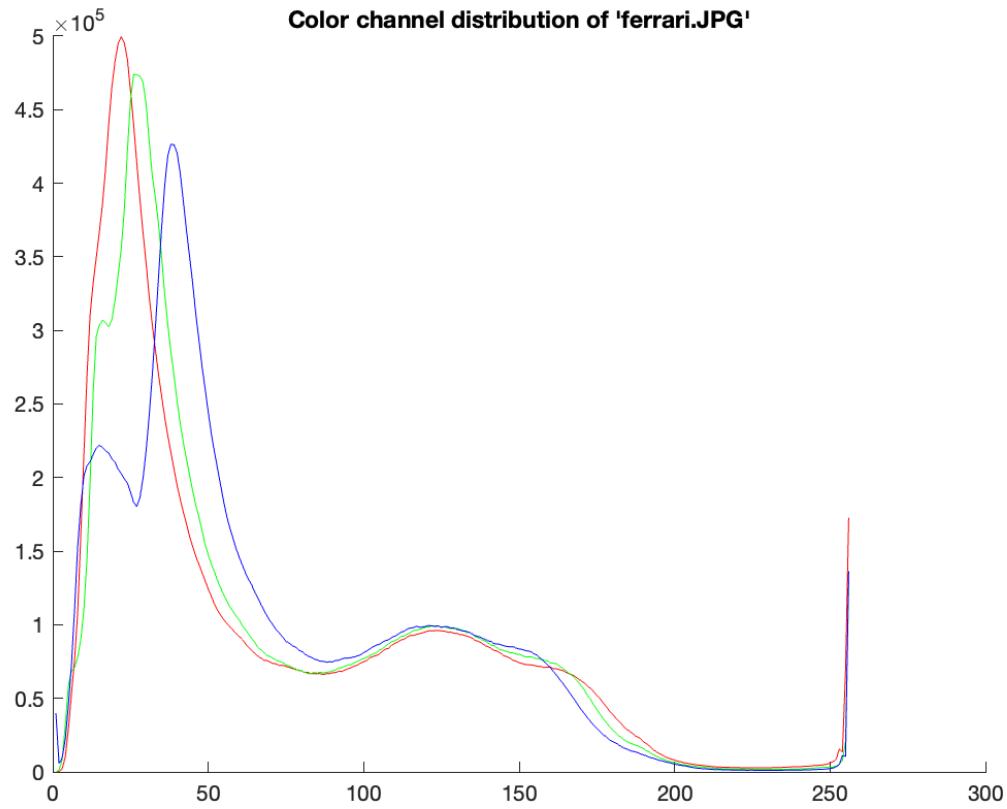


Figure 5: Color channel distribution of the image `ferrari.JPG`.

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]

### 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`.

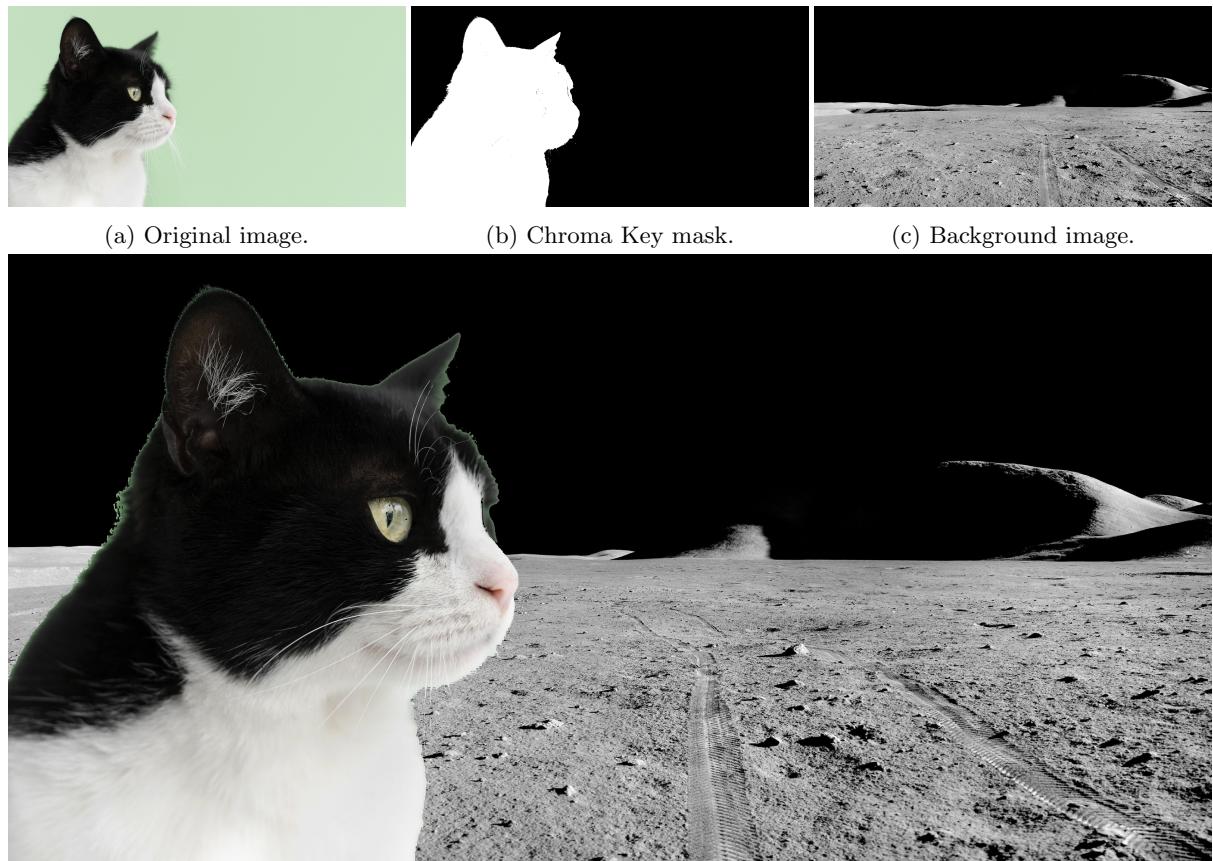


Figure 6: Chroma key compositing.

---

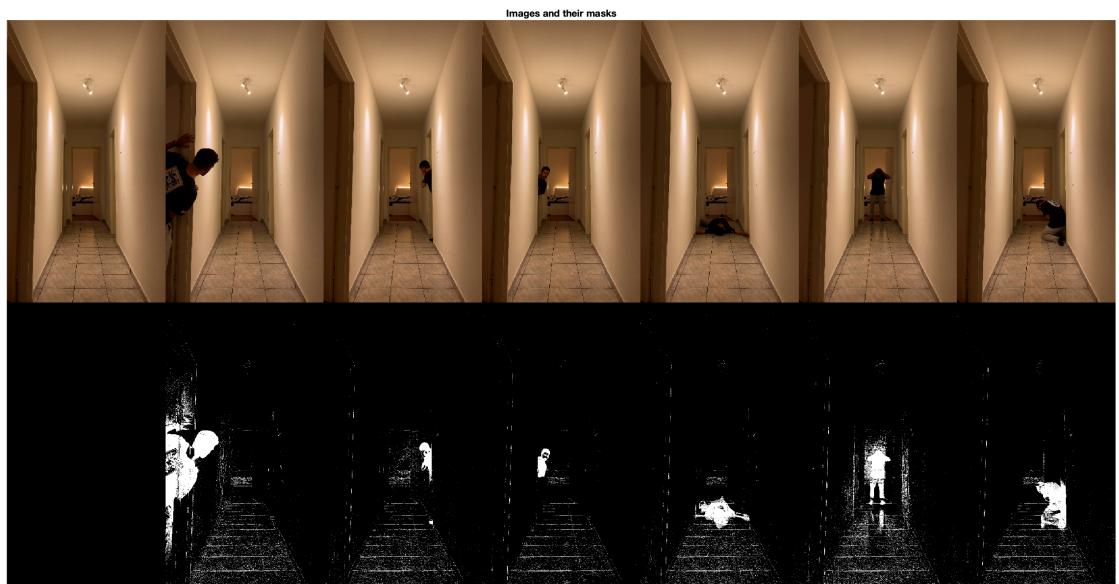
```

1 im = imread("./media/cat.jpg");
2 imhsv = rgb2hsv(im);
3 im_spacebg = imread("./media/spacebg.jpg");
4
5 imwrite(im, "./out/6.cat.jpg")
6 imwrite(im_spacebg, "./out/6.spacebg.jpg")
7 im = (double(im)./255);
8 im_spacebg = (double(im_spacebg)./255);
9
10 % Lets the user select a pixel in the image to key out.
11 figure()
12 imshow(im);
13 coords = int32(ginput(1));
14 px_hsv = imhsv(coords(2), coords(1), 1:3);
15
16 % Each channel may deviate of these amounts from the selected pixel.
17 hue_deviation = 0.05;
18 saturation_deviation = 0.1; % += 10% saturation
19 brightness_deviation = 0.4; % += 40% darker/lighter
20
21 % Creates the mask for each pixel whose channels are within the deviations
22 mask = ~((imhsv(:,:,1) > px_hsv(1) - hue_deviation) & (imhsv(:,:,1) < px_hsv(1) + hue_deviation) & .
23           (imhsv(:,:,2) > px_hsv(2) - saturation_deviation) & (imhsv(:,:,2) < px_hsv(2) + saturation_devi
24           (imhsv(:,:,3) > px_hsv(3) - brightness_deviation) & (imhsv(:,:,3) < px_hsv(3) + brightness_devi
25
26 im_mask = comp(ones(size(im)),mask,zeros(size(im)));
27 im_comp = comp(im,mask,im_spacebg);
28
29 imshow([im, im_mask, im_comp], []);
30 title("Original image, mask and composited image")
31 saveas(gcf,'out/6.image_mask_comp.png')
32
33 imwrite(uint8(255.*im_mask), "./out/6.mask.jpg");
34 imwrite(uint8(255.*im_comp), "./out/6.comp.jpg");
35
36
37 function res = comp(a,mask,b)
38     % Masks image 'a' on top of image 'b'
39     res = a.*mask + b.*(~mask);
40 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 multiple times on the same cleanplate.

---

```

1 im = imread("./media/cat.jpg");
2 imhsv = rgb2hsv(im);
3 im_spacebg = imread("./media/spacebg.jpg");
4
5 imwrite(im, "./out/6.cat.jpg")
6 imwrite(im_spacebg, "./out/6.spacebg.jpg")
7 im = (double(im)./255);
8 im_spacebg = (double(im_spacebg)./255);
9
10 % Lets the user select a pixel in the image to key out.
11 figure()
12 imshow(im);
13 coords = int32(ginput(1));
14 px_hsv = imhsv(coords(2), coords(1), 1:3);
15
16 % Each channel may deviate of these amounts from the selected pixel.
17 hue_deviation = 0.05;
18 saturation_deviation = 0.1; % += 10% saturation
19 brightness_deviation = 0.4; % += 40% darker/lighter
20
21 % Creates the mask for each pixel whose channels are within the deviations
22 mask = ~((imhsv(:,:,1) > px_hsv(1) - hue_deviation) & (imhsv(:,:,1) < px_hsv(1) + hue_deviation) & .
23           (imhsv(:,:,2) > px_hsv(2) - saturation_deviation) & (imhsv(:,:,2) < px_hsv(2) + saturation_devi
24           (imhsv(:,:,3) > px_hsv(3) - brightness_deviation) & (imhsv(:,:,3) < px_hsv(3) + brightness_devi
25
26 im_mask = comp(ones(size(im)),mask,zeros(size(im)));
27 im_comp = comp(im,mask,im_spacebg);
28
29 imshow([im, im_mask, im_comp], []);
30 title("Original image, mask and composited image")
31 saveas(gcf,'out/6.image_mask_comp.png')
32
33 imwrite(uint8(255.*im_mask), "./out/6.mask.jpg");
34 imwrite(uint8(255.*im_comp), "./out/6.comp.jpg");
35
36
37 function res = comp(a,mask,b)
38     % Masks image 'a' on top of image 'b'
39     res = a.*mask + b.*(~mask);
40 end

```

---

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