Advanced Graphics and Image Processing

Practical Exercise 1 Report

Yifan Chen

yc462@cam.ac.uk

Michaelmas 2019

Task 1: Image Enhancement

Histogram Equalisation for grayscale images

Original image 1

Resulting image 1

Original image 2

Resulting image 2

Original image 3

Resulting image 3

Histogram Equalisation for colour images

Strategy 1:

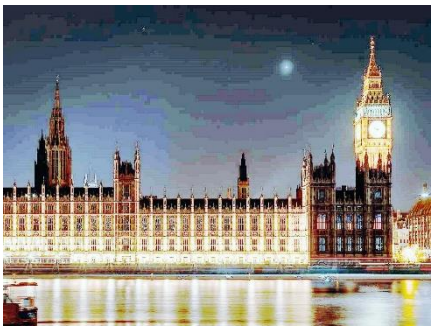
Original image 1

Resulting image 1

Original image 2

Resulting image 2

Original image 3



Resulting image 3

Strategy 2:

Original image 1

Resulting image 1

Original image 2

Resulting image 2

Original image 3

Resulting image 3

Strategy 3:

Original image 1

Resulting image 1

Original image 2

Resulting image 2

Original image 3

Resulting image 3

Screenshots for comparing clearly are shown in Figure1-1 to 1-3:

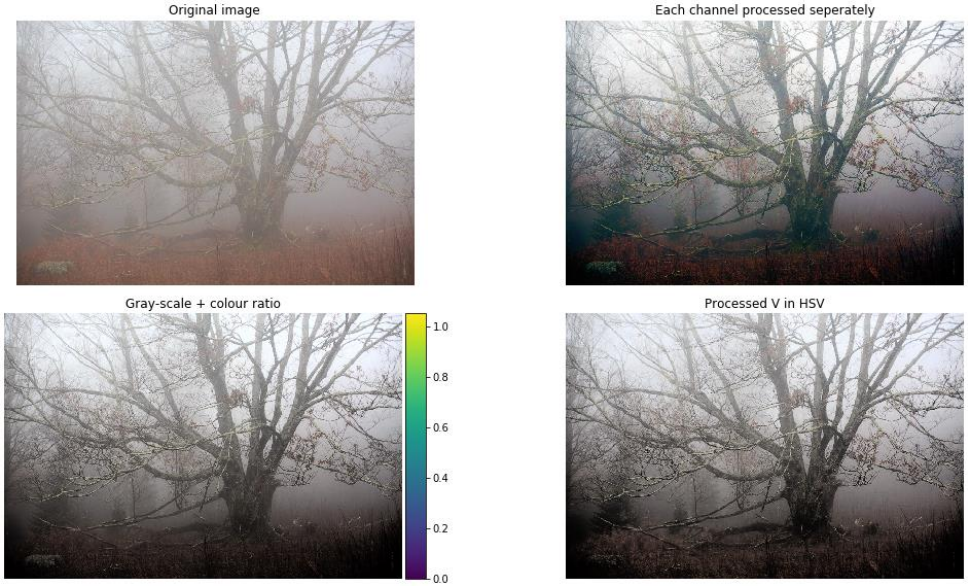


Figure1-1:Group1

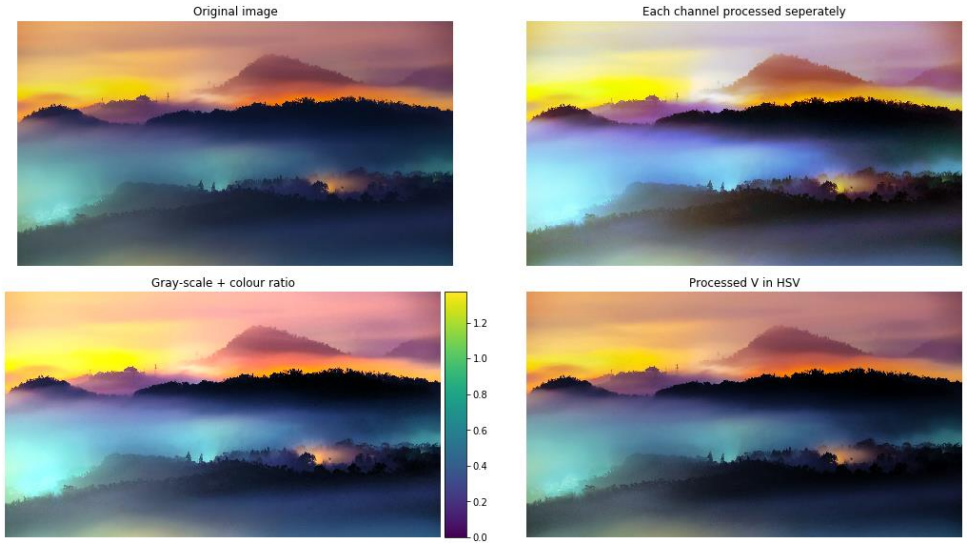


Figure1-2:Group2

Figure1-3:Group3

The noticeable differences between three strategies

1. When orginal images have low contrast, the brightest results always come from strategy1 and in contrast, the “darkest” ones come from strategy2. In this aspect, strategy3 has the best result.

2. When the colours of images are vivid and saturated, strategy1 would be brighter than any other, and some colours even seem unreal, like over-exposure in the flash light(e.g. Group2), or showing Mach Band in Group3; In comparison, strategy2 always generates darker results, as it may not constract bright colours from dark ones; strategy3 holds different colours well and enhances the original by given some details and naturally brightening images.

To conclude, the best strategy should be strategy 3, equalising the histogram from a HSV channel.

Task 2: Alpha blending

The alpha mask

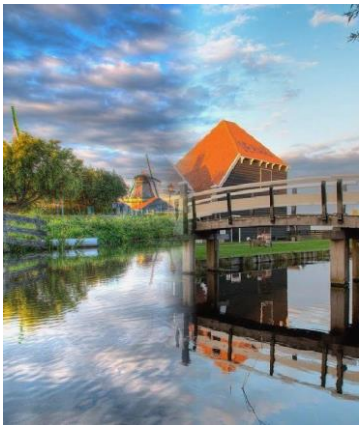
```
#The default setting is horizontal stitching and blending
def get_mask(img,start,stop):
  #define alpha mask array
  img_mask = np.zeros((height,width,len(start)),dtype=np.float)
  #define the window size:mid_width represents the width of blending area,
  #whereas left_width and right_width are widths of two original images.
  mid_width = np.int(window_size*width)
  left_width = np.int(((1-window_size)/2)*width)
  right_width = width-mid_width-left_width

  #create the alpha blending mask
  for i,(start,stop) in enumerate(zip(start,stop)):
      #set the value of alpha
      window = np.linspace(start,stop,mid_width)
      #set the value for non-blending area
      left_side = np.tile(start,left_width)
      right_side = np.tile(stop,right_width)
      #concatenate them
      temp_line = np.concatenate((left_side,window,right_side))
      img_mask[:,:,i] = np.tile(temp_line,(height,1))

  return img_mask
```

Figure2-1: Alpha mask code

Results

Group1:


Original images



The result image(in window_size=0.2)

Group 2:

Original images

The result image(in window_size=0.2)

Task 3: Pyramid Blending

Results

Test images are showed below in group1 and group2, pyramid blending in window_size=[0.2,0.2,0.3,0.3,0.4,0.4].

Group1:


Original images

The resulting image

Group2:


Original images

The resulting image

The differences observed between alpha blending and pyramid blending

1.As pyramid blending applies alpha blending on each layer of the laplacian pyramid, it can avoid blended “ghost”, see Figure3-1 below.

Figure3-1:When window_size is 0.3 in alpha blending(left) and window_size from 0.1 to 0.3 in pyramid blending(right).

2.The image applied pyramid blending tend to be smoother and more natural compared to alpha blending. We can see from Figure3-2 and Figure3-3.

Figure3-2:Details comparison.Window sizes are 0.2 and [0.2,0.2,0.3,0.3,0.4,0.4] respectively.

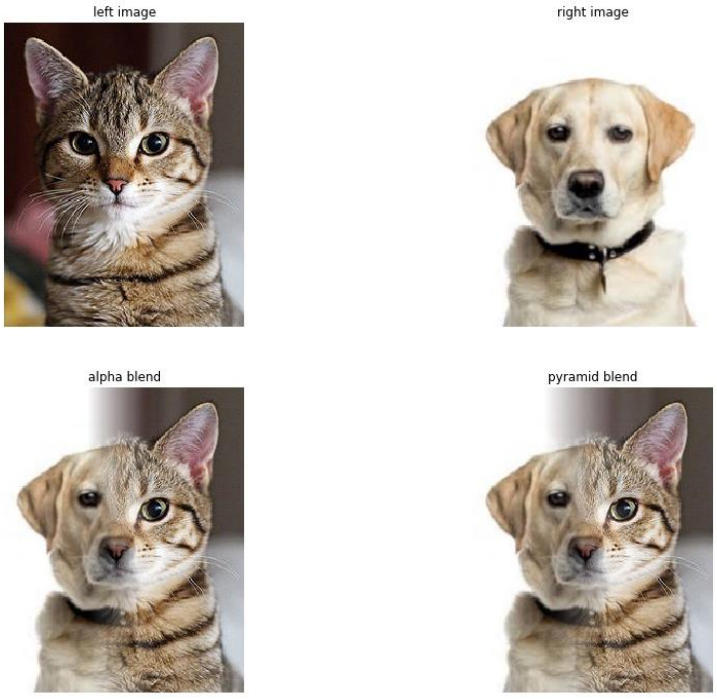


Figure3-3 Cat and dog facial blending given in lecture, applied by window_size=[0.2,0.2,0.3,0.3,0.4,0.4]

To conclude, pyramid blending is more flexible and can customize desired blended images, like smoother ways or less ‘ghost’; but alpha blending is more limited to the window size.

Task 4: Gradient domain reconstruction

Results

Figure4-1:”Pout.jpg” by spsolve solver

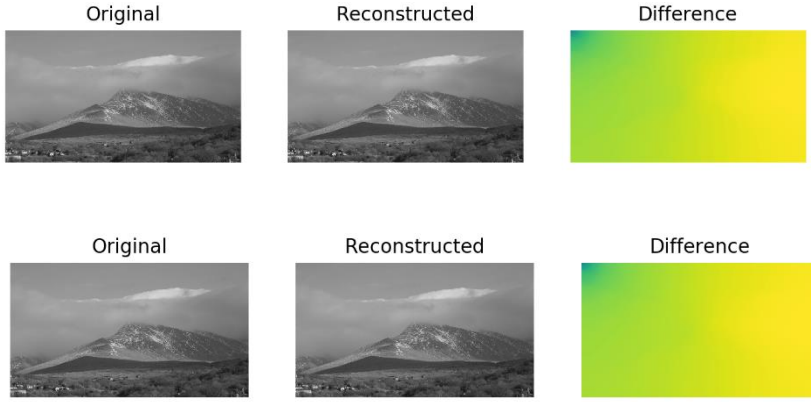


Figure4-2:”mountain.jpg” by spsolve(below) and Cholesky solver(above)

Solver comparison

The table below compares the running time between sparse.linalg.spsolve and Cholesky solver on Macbook Pro 2017(i5 CPU):

Table4-3: running time comparison

Image(size)	Solver	Running time(0.001)
Mountain.jpg(1200*675)	spsolve	11.937
	cholesky	2.133

Task 5: Gradient domain image enhancement

Results

The pairwise-linear function applied to task5 as shown in Figure5-1. Although I tried other linear functions, the differences were unnoticeable.

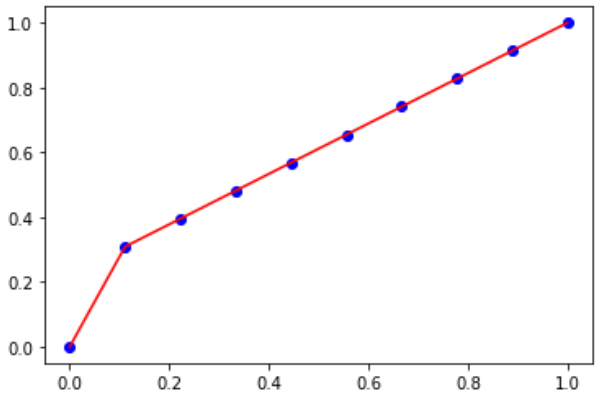


Figure5-1: Linear function in quandrant 1 for enhancing the gradient magnitude

I apply Strategy3 in task1 to recover colors. See Figure5-2.

```
def enhancedImage_gray2RGB_hsv(im,imr):
    """Implement task1(equalising histogram for color images by hsv) to recover colors
    im - The original image
    imr - The reconstructed image
    """
    #convert grayscale image to RGB image

    row,col,channel = im.shape
    hsv_img = color.rgb2hsv(im)
#    #create a array for the recovered image
    new_img = np.zeros((row,col,channel))

    for x in range(row):
          for y in range(col):
              hsv_img[x,y,2] = imr[x,y]

    new_img = color.hsv2rgb(hsv_img)

    return new_img
```

Fugure5-2:Recovering color Strategy code.

Two groups below show the resulting images:

Figure5-3:Image “lake.jpg” is enhanced in right.

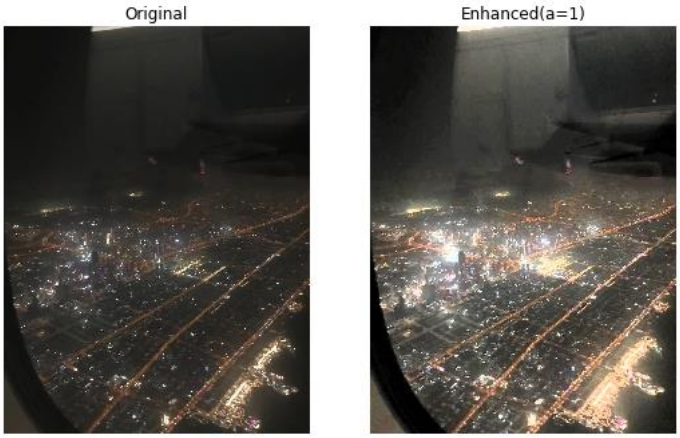


Figure5-4:Image “dubai.jpg” is enhanced in right.

Processing time comparison(Table5-5) in different solvers(related to task4):

Table5-5:Processing time comparison using functions and variables above(Same tech spec. as Table4-3)

*Due to the CPU limitation of the laptop, “field.jpg” can not be enhanced.

Solution(spsolve/ Cholskey)	Image name	size	Processing time(0.001)
spsolve	Field.jpg	4032*3024	(Overflow)*
	Dubai.jpg	1440*1080	52.433
	Mountain.jpg	1200*675	15.642
	Pout.jpg	840*837	11.687
	Lake.jpg	590*290	1.819
	Blue_sky.jpg	300*200	0.424
cholesky	Field.jpg	4032*3024	219.560
	Dubai.jpg	1440*1080	12.475
	mountain	1200*675	1.766

	Pout.jpg	840*837	1.801
	Lake.jpg	590*290	0.265
	Blue_sky.jpg	300*200	0.142

In conclusion, Cholesky obviously has more powerful calculation ability.

The ‘alpha’ problem

The alpha in least-weighted formulation, to avoid pinching artefacts, shows in Figure5-6 may have little effect to the results. See Figure5-7. But it does influence processing time, shown in Table5-8.

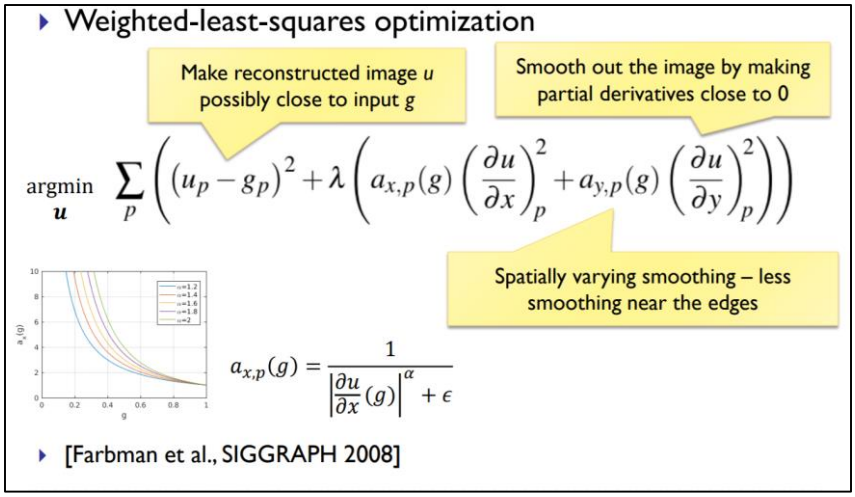


Figure5-6:Weighted-least-squares optimization

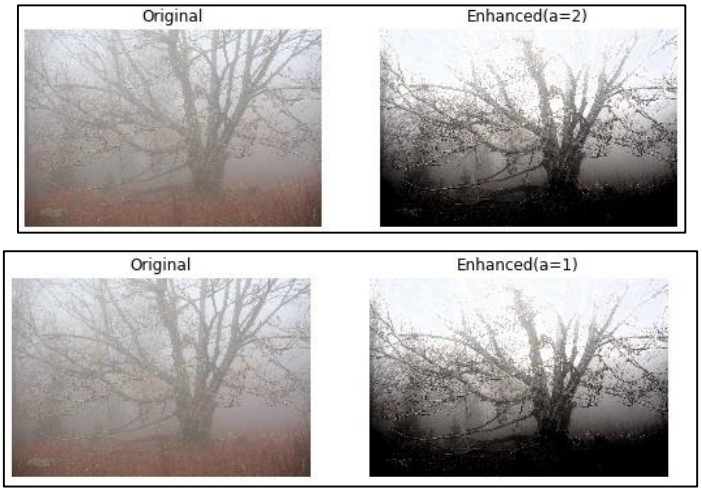


Figure5-7: ‘Identical’ images on different alpha values

The table shows according to different images, processing time varies:

Table5-8: Processing time on different images and different solvers.(Same tech spec. as Table4-3)

Solution	Image name	size	Processing time(0.001)		
			a=1	a=1.5	a=2.0
spsolve	Dubai.jpg	1440*1080	52.433	43.999	44.359
	Mountain.jpg	1200*675	15.642	15.401	15.412
	Pout.jpg	840*837	11.687	12.829	14.197
	Lake.jpg	590*290	1.819	2.228	2.011
	Blue_sky.jpg	300*200	0.424	0.409	0.391