RYERSON UNIVERSITY

Course Title:	Digital Image Processing
Course Number:	ELE 882
Semester/Year	Fall/2019

Instructor:	Ling Guan

Assignment/Lab Number:	1
Assignment/Lab Title:	Assignment 1

Submission Date:	October 1, 2019
Due Date:	October 1, 2019

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Baker	Raymond	500691429	03	R.B.
Bao	Doan	500733516	03	B.D.

Introduction

This lab introduced various methods of image processing. Different transforms will be applied to images and the effects will be examined.

Analysis

Question 1

To do an alpha mask you can let intensity

I(x,y) = (Ia(x,y) + Ib(x,y)*Normalize(M(x,y)))/2

- Where Ia is the intensity of image a
- Where Ib is the intensity of image b
- Where M is the intensity of the mask

This is taking the average of the two images intensities taking the mask into account

Question 2

- T(r) applies a contrast_stretch which normalizes the contrast in the image. In other words evenly distributing the intensities over the range 255 to 0.

Question 3

- LUT can also improve performance for computation intensive transforms.
- The LUT takes alot of memory which can be bad if running on a small machine.
- The LUT harder to modify than a function pointer.
- In the case of contrast stretching it depends on the image making it non reusable.
- LUT's can't be reused for translation transforms

Question 4

- Contrast stretching could be used to bring the higher intensity range down to the display range by normalizing all the values then multiplying them by 2ˆ8.
- Ie 2ˆ8*(r/(r$_m ax - r_m in) - r_m in/(r_m ax - r_m in)$
- Some considerations should be made if there is large amounts of intensities in a small range as the detail will be lost on the display.

Question 5

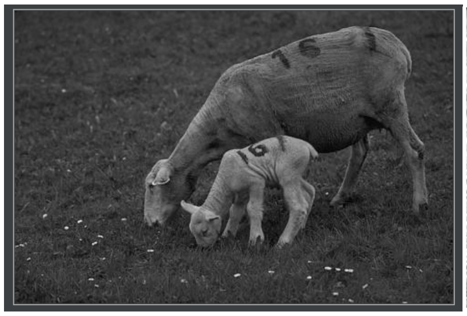
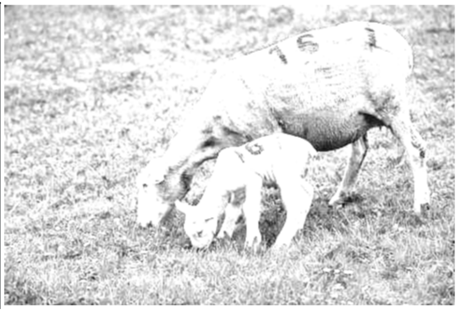
- If an image is very dim then the range of intensities would be low making the contrast stretching hard to impossible. As many distinct points would share intensities, they would be normalized to the same value.
- Sensor noise on a dim image will have the effect of disabling or significantly hindering the contrast stretch. This is due to the noise widening the range between r_min and r_max. With the upper bound having noise from 0 to 255 totally disabling the contrast stretch.

Conclusion

Throughout the lab, various forms of image processing were used and analyzed.

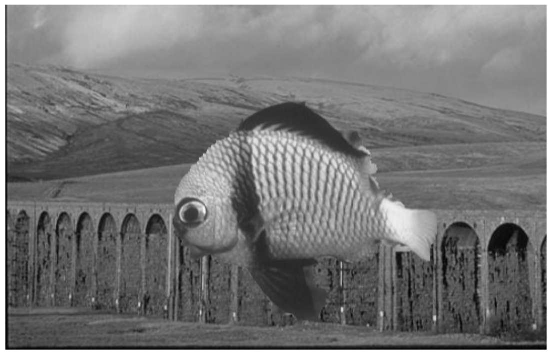
Section 2.1 - Q1

```
infile = "Images/Section2.1 - Q1/207056.jpg"
img = Image.open(infile)
img.load()
problem1 = np.asarray(img, dtype="int32")
def apply_point_tfrm(data, C, B):
out = copy.deepcopy(data)
for y in range(0, len(data)):
for x in range(0, len(data[y])):
out[y][x] = C * out[y][x] + B
return out
transform = apply_point_tfrm(problem1, 4, -10)
Image.fromarray(transform).show()
```

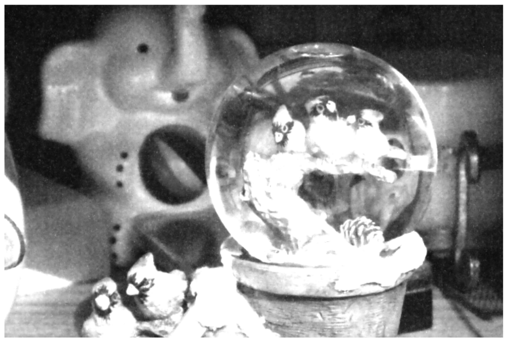
Section 2.1 - Q2

```
infile = "Images/Section2.1 - Q2/fish.png"
infile2 = "Images/Section2.1 - Q2/bridge.png"
maskfile = "Images/Section2.1 - Q2/mask.png"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
img2 = Image.open(infile2)
img2.load()
img2data = np.asarray(img2, dtype="int32")
mask = Image.open(maskfile)
mask.load()
mask_data = np.asarray(mask, dtype="int32")
def apply_mask(imga, imgb, imgmask):
out = copy.deepcopy(imgb)
for y in range(0, len(imgmask)):
for x in range(0, len(imgmask[y])):
if imgmask[y][x] > 0:
out[y][x] = imga[y][x]
return out
transform = applymask(imgdata, img2data, maskdata)
Image.fromarray(transform).show()
```



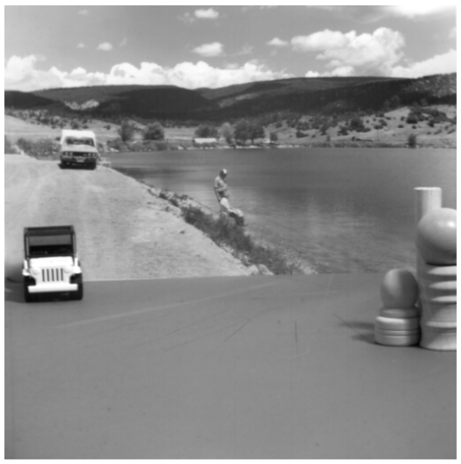
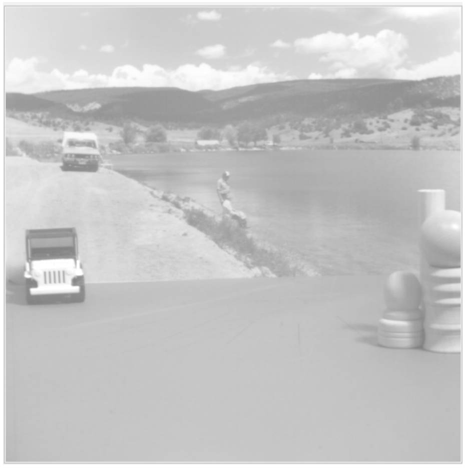
Section 2.1 - Q3

```
infile = "Images/Section2.1 - Q3/snowglobe_001.png"
infile2 = "Images/Section2.1 - Q3/snowglobe_002.png"
infile3 = "Images/Section2.1 - Q3/snowglobe_003.png"
infile4 = "Images/Section2.1 - Q3/snowglobe_004.png"
def average_images(*args):
img = Image.open(args[0])
img.load()
out_data = np.asarray(img, dtype="int32")
for image in args[1:]:
img = Image.open(image)
img.load()
img_data = np.asarray(img, dtype="int32")
for y in range(0, len(img_data)):
for x in range(0, len(img_data[y])):
out_data[y][x] += img_data[y][x]
return out_data
return np.vectorize(lambda x: x / len(args))(out_data)
transform = average_images(infile, infile2, infile3, infile4)
Image.fromarray(transform).show()
```



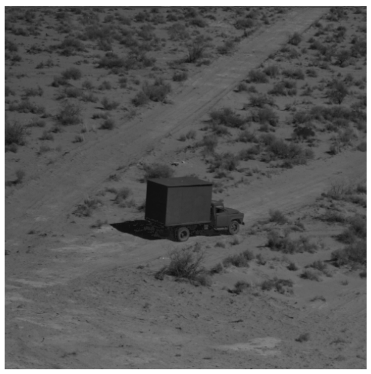
Section 2.2 - Q1

```
infile = "Images/Section2.2 - Q1/motion01.512.tiff"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
def contrast_stretch(img_data):
out_data = copy.deepcopy(img_data)
sorted_ = np.sort(out_data.flatten())
max_ = float(sorted_[-1])
min_ = float(sorted_[0])
return np.vectorize(lambda x: 255 * (float(x) / (max_ - min_) - min_ / (max_ - min_)))(out_data).astype("int32")
transform = contrast_stretch(img_data)
Image.fromarray(transform).show()
```

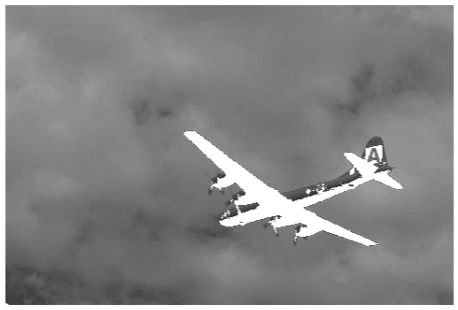
Section 2.2 - Q2

```
infile = "Images/Section2.2 - Q2/7.1.01.tiff"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
def contrast_piecewise(img_data, vec_a, vec_b):
out_data = copy.deepcopy(img_data)
x1, y1 = vec_a
x2, y2 = vec_b
func1 = lambda x: x * x1 / y1
func2 = lambda x: x * (y2 - y1) / (x2 - x1)
func3 = lambda x: x * (255 - y2) / (255 - x2)
def apply_funcs(brightness):
if brightness < x1:
return func1(brightness)
elif brightness > x2:
return func3(brightness)
else:
return func2(brightness)
return np.vectorize(apply_funcs)(out_data).astype("int32")
transform = contrast_piecewise(img_data, (20, 10), (200, 150))
```



Section 2.2 - Q3

```
infile = "Images/Section2.2 - Q3/3096.jpg"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
def contrast_highlight(img_data, A, B, I_min):
out_data = copy.deepcopy(img_data)
return np.vectorize(lambda x: I_min if x < A or x > B else x)(out_data).astype("int32")
transform = contrast_highlight(img_data, 50, 200, 255)
Image.fromarray(transform).show()
```



Section 2.2 - Q4

```
def apply_lut(img_data, lut):
return np.vectorize(lambda x: lut[x])(img_data)
```

Section 2.2 - Q5

```
def gen_non_generic_lut(func, img_data, *args):
after = func(img_data, *args)
out = {}
for y in range(0, len(img_data)):
for x in range(0, len(img_data[y])):
out[img_data[y][x]] = after[y][x]
return out
infile = "Images/Section2.2 - Q1/motion01.512.tiff"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
Q1_LUT = gen_non_generic_lut(contrast_stretch, img_data)
transform = apply_lut(img_data, Q1_LUT)
Image.fromarray(transform).show()
```

Section 2.2 - Q6

```
def gen_lut(func, *args):
orig = np.fromiter(range(0, 256), "int32")
after = func(orig, *args)
out = {}
for i in range(0, 256):
out[i] = after[i]
return out
```

Q2

```
infile = "Images/Section2.2 - Q2/7.1.01.tiff"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
Q2_LUT = gen_lut(contrast_piecewise, (20, 10), (200, 150))
print(Q2_LUT)
transform = apply_lut(img_data, Q2_LUT)
Image.fromarray(transform).show()
```

Q3

```
infile = "Images/Section2.2 - Q3/208001.jpg"
img = Image.open(infile)
img.load()
img_data = np.asarray(img, dtype="int32")
Q3_LUT = gen_lut(contrast_highlight, 50, 200, 255)
transform = apply_lut(img_data, Q3_LUT)
Image.fromarray(transform).show()
```