EECS 4422/5323 Assignment 1

Due: 2 October, 2019

Part 1: Written Questions

Your answers to these questions should be provided in writing (either on paper or as a PDF file emailed to me according to the instructions posted on the course website). All answers must be your own work.

Question 1: BRDFs

Stefan is working on a graphics engine and is designing bidirectional reflectance distribution functions for a number of different materials. He shares several of them with you and asks you what you think of them. For each BRDF, state whether or not it is physically realistic. If not, briefly justify why.

Note that for equations ***(a.)-(c.)***, f_r is the BRDF, θ_i is the zenith angle of incident light, ϕ_i is the azimuth angle of incident light, θ_r is the zenith angle of reflected light, and ϕ_r is the azimuth angle of reflected light. All angles are in radians.

[7] marks each.

(a.)

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{1}{\pi}\sin(\theta_r)$$

(b.)

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r) = \begin{cases} \frac{0.1}{\pi} + 0.2(\cos(\phi_i + \pi - \phi_r)\cos(\theta_r - \theta_i)) & |\phi_i + \pi - \phi_r| < 0.1 \wedge |\theta_r - \theta_i| < 0.1 \\ \frac{0.1}{\pi} & \text{otherwise} \end{cases}$$

(c.)

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r) = \begin{cases} \frac{0.2}{\pi} + 0.1(\cos(\phi_i + \pi - \phi_r) + 0.2\cos(\theta_r - \theta_i)) & \phi_i < \pi \wedge |\theta_r - \theta_i| < 0.1 \\ \frac{0.3}{\pi} & \text{otherwise} \end{cases}$$

Question 2: Propeller Image

Fernanda was flying in a turboprop airplane and took several pictures out the window. She was surprised to notice something weird going on in her photos. Knowing that you are studying computer vision, she showed you one of her pictures (on the right) and asked if you could explain what was happening.

Fernanda's image of a propeller.

(a.) [9 points] Write a brief explanation for the strange appearance of the aircraft propeller, including what specific aspect of image capture is responsible.

(b.) [4 points] What can you likely conclude about the camera Fernanda was using when she took this photo?

(c.) [6 points] Fernanda says that in person the propeller looked to her like a "ghostly circle" (*i.e.* a uniform semi-transparent circle). Can you briefly explain why it would appear that way to a human observer?

Question 3: Filter Ordering

Anastasia and Sailesh are performing a number of back-to-back filtering operations on their images, and were surprised to discover that sometimes they get the same final output and sometimes they do not. For the following sets of operations, state whether or not the output should be the same, and why.

Note that for all code listed below, you can assume that the following import command has been completed: `import cv2`

[5] marks each.

(a.)

Listing 1: Anastasia's Code

```
ksize1 = 10
sigma = 3
ksize2 = 5

gkern = cv2.getGaussianKernel(ksize1, sigma)

fimgA = cv2.filter2D(img, -1, gkern)
fimgA = cv2.medianBlur(fimgA, ksize2)
```

Listing 2: Sailesh's Code

```
ksize1 = 5
sigma = 3
ksize2 = 10

fimgS = cv2.medianBlur(img, ksize1)

gkern = cv2.getGaussianKernel(ksize2, sigma)
fimgS = cv2.filter2D(fimgS, -1, gkern)
```

(b.)

Listing 3: Anastasia's Code

```
ksize1 = 3
ksize2 = 5

fimgA = cv2.medianBlur(img, ksize1)
fimgA = cv2.medianBlur(fimgA, ksize2)
```

Listing 4: Sailesh's Code

```
ksize1 = 5
ksize2 = 3

fimgS = cv2.medianBlur(img, ksize1)
fimgS = cv2.medianBlur(fimgS, ksize2)
```

(c.)

Listing 5: Anastasia's Code

```
ksize1 = 8
sigma1 = 2

ksize2 = 10
sigma2 = 3

gkern1 = cv2.getGaussianKernel(ksize1, sigma1)
gkern2 = cv2.getGaussianKernel(ksize2, sigma2)
fimgA = cv2.filter2D(img, -1, gkern1)
fimgA = cv2.filter2D(fimgA, -1, gkern2)
```

Listing 6: Sailesh's Code

```
ksize1 = 8
sigma1 = 2

ksize2 = 10
sigma2 = 3

gkern = cv2.getGaussianKernel(ksize1, sigma1)
fimgS = cv2.filter2D(img, -1, gkern)
gkern = cv2.getGaussianKernel(ksize2, sigma2)
fimgS = cv2.filter2D(fimgS, -1, gkern)
```

Coding Questions

Your answers to these questions should be provided in a zip file which contains a Python file `Assignment_One.py`. For each question, this file should contain a defined function `Question_[Question Number]_[Question Part]` which follows the specified input and output requirements given in the question. Any additional files or functions which are not part of standard Python packages should be included in the folder. If you use any code which is not your own (for example, the connected component plotting function which I showed in the extended morphology demo which I modified from a StackOverflow answer), you should note this in comments in your code and include a `readme.txt` file which details appropriate credit.

Question 4: Fancy Fonts

You have been hired as an image processing consultant for a marketing company, and your first task is to build them a set of functions for customizing the appearance of letters and shapes. You can assume your input is a grayscale input with dark objects on a white background; your job is to manipulate that image in one of the following ways:

(a.) Borders [20 marks]

Write a function which outputs all shapes in the input image as two-tone outputs, with a coloured outline (the border) and a coloured interior (the font colour). Your function should take the form `img_out = Question_4_a(img_in, thick, border_colour, font_colour)`, where

- `img_out` is an RGB image containing the function output
- `img_in` is a grayscale image with dark text or shapes on a white background
- `thick` is an integer parameter specifying the thickness of the border in pixels
- `border_colour` is a 3-tuple specifying the RGB colour of the border
- `font_colour` is a 3-tuple specifying the RGB colour of the non-border portions of the input shapes

You can assume that all shapes will be at least `thick` pixels away from the nearest other shape.

(b.) Shadows [30 marks]

Write a function which projects a directed shadow for all shapes in the input image. Your function should take the form `img_out = Question_4_b(img_in, shadow_size, shadow_magnitude, orientation)`, where

- `img_out` is a grayscale image containing the function output
- `img_in` is a grayscale image with dark text or shapes on a white background
- `shadow_size` is a parameter which tells you the maximum number of pixels which should be potentially affected by the shadow
- `shadow_magnitude` is a parameter, expressed in terms of the standard deviation of a Gaussian, which controls how intense the shadow effect should be
- `orientation` is a parameter, in degrees, which dictates the direction the shadow should project in

Question 5: Pokemon AR

Your friend Darrel is a big fan of the Pokemon franchise, but has been disappointed with his ability to compose nice augmented reality images from the Pokemon GO game. You decide to help him out by building a desktop tool in Python which can be used to insert Pokemon into images.

Note that for this question you have been provided with a set of images of Pokemon set against a green screen (`Q5_train.zip`); your functions should be able to work with any of the Pokemon in this training set. You have also been provided with a set of sample scenes depending on whether you are EECS4422 or EECS5323, `Q5_samples_4422.zip` or `Q5_samples_5323.zip`

EECS4422 Students [45 marks]

Write a function which places a Pokemon into an image scene at a specified size and position. Your function should take the form `img_out = Question_5(scene_img, pokemon_string, location, width)`, where

- `img_out` is an RGB image containing the composite output
- `scene_img` is an RGB image into which the Pokemon should be inserted
- `pokemon_string` is a string consisting of the name of one of the available Pokemon, `Flygon`, `Jigglypuff`, `Muk`, or `Pikachu`.
- `location` is a tuple in (x, y) format which specifies the horizontal and vertical position, respectively, to which the centroid of the Pokemon should be inserted
- `width` is a parameter which specifies how many pixels wide the bounding box for the Pokemon should be. The Pokemon's height should be scaled accordingly to maintain its aspect ratio

EECS5323 Students [45 marks]

Write a function which places a Pokemon into an image scene at a specified size and position and takes into account occluding objects. Your function should take the form `img_out = Question_5(scene_img, scene_depth, pokemon_string, location, width)`, where

- `img_out` is an RGB image containing the composite output
- `scene_img` is an RGB image into which the Pokemon should be inserted
- `scene_depth` is a 2D matrix specifying the depth channel for the scene image. Values are in meters
- `pokemon_string` is a string consisting of the name of one of the available Pokemon, `Flygon`, `Jigglypuff`, `Muk`, or `Pikachu`.
- `location` is a tuple in (x, y, z) format which specifies the horizontal and vertical position in pixels and the depth in meters, respectively, to which the centroid of the Pokemon should be inserted
- `width` is a parameter which specifies how many pixels wide the bounding box for the Pokemon should be. The Pokemon's height should be scaled accordingly to maintain its aspect ratio

You can assume all Pokemon are flat, so for overlapping Pokemon and scene elements any pixel with depth less than the z coordinate in `location` should occlude the Pokemon, whereas pixels with depth greater than z should be occluded by the Pokemon.

Note that depth values are stored in `.mat` files. These can be loaded into Python using the following code:

```
from scipy.io import loadmat

annots = loadmat(FILENAME)
```