CMU Fall24 16820 Homework 5

Patrick Chen

November 10, 2024

Q1-a at page 3

Ans:

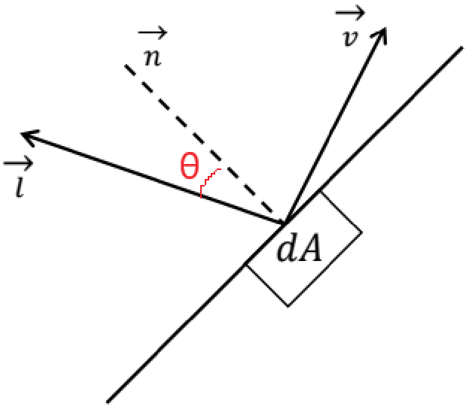
According to Lambertian's cosine law, the intensity of light observed from a Lambertian surface is propational to the cosine of the angle θ between the surface normal, $\vec{n}$, and the direction of the incoming light, $\vec{l}$.

$$\vec{n} \cdot \vec{l} = |\vec{n}||\vec{l}|\cos(\theta) \tag{1}$$

Typically in the context of n-dot-l lighting, both $\vec{n}$ and $\vec{l}$ are unit vectors, so we have:

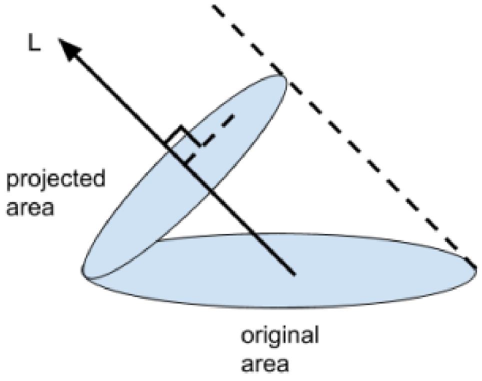
$$\vec{n} \cdot \vec{l} = \cos(\theta) \tag{2}$$

As shown in Fig. 2a as below, the θ is the dot product between $\vec{n}$ and $\vec{l}$. The dot product, θ, quantifies the amount of incident light that is effectively contributing to the brightness of the surface. As the angle increases, less light is effectively reaching the surface, as the angle decrease, more light hits the surface, reaching maximum incoming light when θ is 0, identical to the case that $\vec{l}$ is aligned with $\vec{n}$.



(a) Geometry of photometric stereo

Fig. 2a



(b) Projected area

Fig. 2b

The projected area, $\mathbf{dA_{projected}}$, in Fig. 2b is the original area, $\mathbf{dA}$, times $\cos(\theta)$:

$$\begin{aligned} \mathbf{dA_{projected}} &= \mathbf{dA} \cdot \cos(\theta) && (3) \\ &= \mathbf{dA} \cdot \vec{n} \cdot \vec{l} && (4) \\ \frac{\mathbf{dA_{projected}}}{\mathbf{dA}} &= \cos(\theta) && (5) \\ &= \vec{n} \cdot \vec{l} && (6) \end{aligned}$$

As θ becomes larger, the effective area that light directly expose to becomes smaller, and on the opposite, as θ becomes smaller, the effective area becomes larger, reaching maximum $\mathbf{dA}$ as $\theta = 0$. Hence, the projected area comes into the equation as part of the Lambertian reflectance model through the $\cos(\theta)$ factor.

The reason that the viewing direction does not matter is that we assume the surface is Lambertial surface, which reflects light equally in all directions, $\vec{v}$. Specifically, the light reflected from each point on the surface has the same intensity value no matter where the observer is located. It also means that the intensity of light that the observer can observe is based solely on the $\vec{n} \cdot \vec{l} = \cos(\theta)$ term.

Q1-b at page 4

Ans:

Following Figure 3 - Figure 5 shows the rendered result, and the Figure 6 shows the code snippet in q1.py.

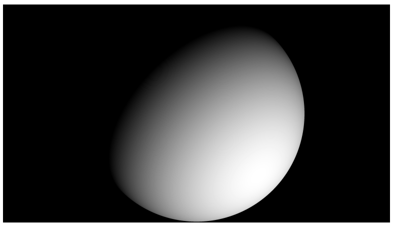


Figure 3: Rendered Result with (1, 1, 1)/$\sqrt{3}$ Light Vector

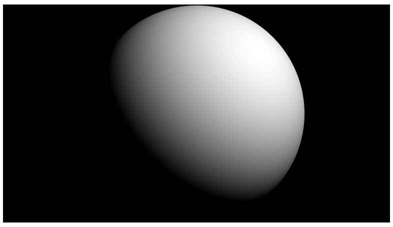


Figure 4: Rendered Result with (1, -1, 1)/$\sqrt{3}$ Light Vector

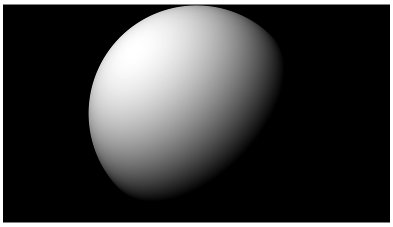
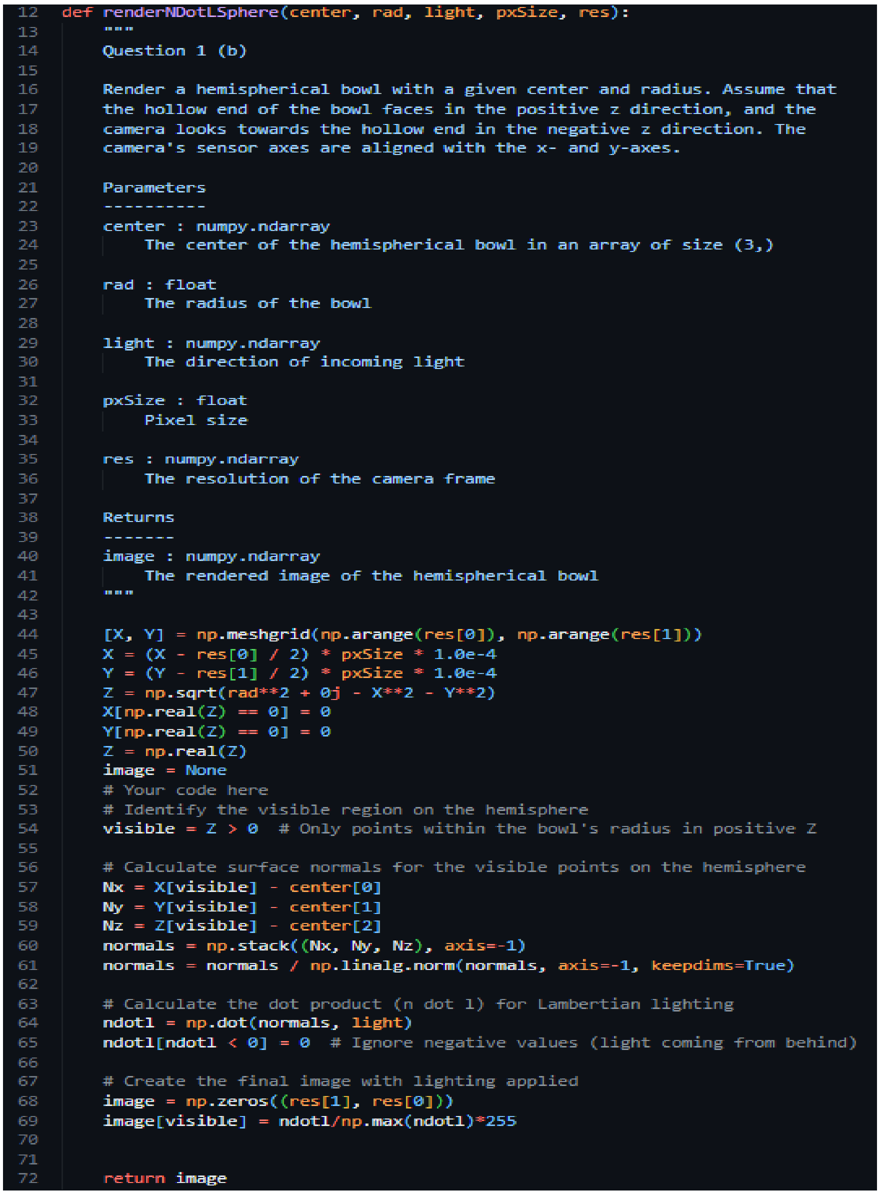


Figure 5: Rendered Result with (-1, -1, 1)/$\sqrt{3}$ Light Vector



```python
def renderNDotLSphere(center, rad, light, pxSize, res):
    """
    Question 1 (b)

    Render a hemispherical bowl with a given center and radius. Assume that
    the hollow end of the bowl faces in the positive z direction, and the
    camera looks towards the hollow end in the negative z direction. The
    camera's sensor axes are aligned with the x- and y-axes.

    Parameters
    ----------
    center : numpy.ndarray
        The center of the hemispherical bowl in an array of size (3,)

    rad : float
        The radius of the bowl

    light : numpy.ndarray
        The direction of incoming light

    pxSize : float
        Pixel size

    res : numpy.ndarray
        The resolution of the camera frame

    Returns
    -------
    image : numpy.ndarray
        The rendered image of the hemispherical bowl
    """

    [X, Y] = np.meshgrid(np.arange(res[0]), np.arange(res[1]))
    X = (X - res[0] / 2) * pxSize * 1.0e-4
    Y = (Y - res[1] / 2) * pxSize * 1.0e-4
    Z = np.sqrt(rad**2 + 0j - X**2 - Y**2)
    X[np.real(Z) == 0] = 0
    Y[np.real(Z) == 0] = 0
    Z = np.real(Z)
    image = None
    # Your code here
    # Identify the visible region on the hemisphere
    visible = Z > 0  # Only points within the bowl's radius in positive Z

    # Calculate surface normals for the visible points on the hemisphere
    Nx = X[visible] - center[0]
    Ny = Y[visible] - center[1]
    Nz = Z[visible] - center[2]
    normals = np.stack((Nx, Ny, Nz), axis=-1)
    normals = normals / np.linalg.norm(normals, axis=-1, keepdims=True)

    # Calculate the dot product (n dot l) for Lambertian lighting
    ndotl = np.dot(normals, light)
    ndotl[ndotl < 0] = 0  # Ignore negative values (light coming from behind)

    # Create the final image with lighting applied
    image = np.zeros((res[1], res[0]))
    image[visible] = ndotl/np.max(ndotl)*255


    return image
```

Figure 6: Code Snippet

Q1-c at page 4

Ans:

Following.

Collaborations

Ans:

Though I do not have collaborators, I found the following websites helpful on understanding the concepts in this homework.

1. `https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2003_3/baker_simon_2003_3.pdf.`
2. `https://pytorch.org/tutorials/beginner/pytorch_with_examples.html`
3. `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imsave.html`
4. `https://www.geeksforgeeks.org/how-to-load-cifar10-dataset-in-pytorch/`
5. `https://www.geeksforgeeks.org/how-do-you-use-pytorchs-dataset-and-dataloader-classes-for-custom`
6. `https://github.com/facebookarchive/fb.resnet.torch/issues/180`
7. `https://pytorch.org/tutorials/beginner/data_loading_tutorial.html#afterword-torchvision`
8. `https://www.kaggle.com/code/satriasyahputra/flowers-17`