1 Theoretical Exercises

1.1 Light and Matter

a) Please label on the following illustration the name of the effects that can occur during interaction between light and a solid object.

b) Complete the following figure such that it illustrates reflection on a Lambertian surface. Draw the direction(s) of the reflected light beam.
1.2 Image Acquisition / Camera Projection

1.2.1 Acquisition of an image with a convex lens

You want to take a picture of an object with the following simple camera model (a lens of focal length \( f \) and a screen). Where should you place the screen in order to have a sharp image? How large will the object be on the screen?

Draw and label the position of the screen and the size of the object onto the sketch below:

1.2.2 Minimal distance between object and screen

Using Gauss’s formula (see script), compute the minimum distance between the object and the screen for the above camera setup. Assume that the object is in focus and that the focal length is constant. To simplify the calculation, you may introduce positive variables for both the object distance \( g = Z_0 \) and the image distance \( b = -Z_i \). Hence, you have to minimize \( g + b \).
1.2.3 Camera Projection

You have a (pinhole-) camera, for which the internal calibration parameters are already known. Now you take a picture with this camera, in which you know the exact position of $N$ points in the “World Coordinate System”.

How large must $N$ be to determine all external calibration parameters?

Try to answer the question in two ways:

a) Consider how many unknowns this problem has, and how many equations you obtain from each point. The problem is solved if there are at least as many equations as there are unknowns.

b) Consider which degrees of freedom are lost for each additionally considered point. If no more degrees of freedom exist, i.e. the location and orientation of the camera are fixed, then the camera is externally calibrated.
2 Practical Exercise

This first programming exercise addresses the acquisition of digital, discretely sampled images and helps you to understand how to access individual pixels in a given input image. Part 1 treats spatial sampling techniques, while part 2 considers pixel intensity values. To complete this exercise, you may use either the Python script templates sample.py and requant.py or the Jupyter notebook templates sample.ipynb and requant.ipynb, all located in the folder `~/cvcourse/materials/`. Just copy the templates into your local folder by typing:

cp ~/cvcourse/materials/* .

(Do not forget the point at the end)

2.1 Sampling

Digital images are sampled representations of continuous space. Each sample in the image is a discrete entity known as a pixel. This part of the exercise explores the properties of spatial sampling by considering an already sampled input image and resampling it with a lower sampling rate, which is known as downsampling or decimation. **Your first task** is to implement a function that extracts every \( n \)-th pixel from a given input image (where \( n \) is an integer) both in the horizontal and vertical direction to generate an output downsampling image for which the total number of pixels is reduced by a factor of approximately \( n^2 \). Parameter \( n \) is known as the **decimation factor**. For \( n = 3 \), the following output image (right part) will result from the input image (left part).

![Sampling Process](image)

Using your function, experiment with various decimation factors \( n \) on different images in the directory `~/cvcourse/pics`.

1. What happens if you resample an input image containing fine regular structures (e.g., `~/cvcourse/pics/carpet.png`) with a much lower sampling rate, i.e., using very large \( n \)?

2. What is aliasing and why does it occur?

3. The higher the decimation factor \( n \), the lower the sampling rate. Without modifying the input image, what is the highest decimation factor \( n_{\text{max}} \) that one can apply to a given image without any information loss incurred?
A standard *anti-aliasing* step is to *pre-filter* the input image with a low-pass filter, in order to suppress its high-frequency content, before downsampling it. In this way, it is possible to use quite large decimation factors $n$ and still obtain a meaningful output. **Your second task** is to implement a function that filters the input image with a Gaussian filter, which is a good approximation of a low-pass filter. Selecting a proper built-in Python function to achieve this is highly recommended. After that, experiment with large decimation factors $n$ for downsampling the filtered image and compare to the case of no pre-filtering.

4. How is the scale parameter $\sigma$ of the Gaussian filter related to the bandwidth of an ideal low-pass filter?
2.2 Modification of the pixel intensities

In this part of the exercise, you will manipulate the actual pixel values. For example, if you take a look at the provided stereo images `cvcourse/pics/cube_left.pgm` and `cvcourse/pics/cube_right.pgm`, you will see that image acquisition was not optimized and both images are underexposed. Furthermore, the acquisition hardware used, i.e. the camera, performs a non-linear mapping from the captured irradiance values (light coming from the scene) to resulting pixel intensities. In the following step you will learn how to adjust the range of pixel values.

Requantization of the pixel intensities: Digital images are discrete not only in space but also in intensity. The process of mapping a continuous intensity range to a finite number of intensity levels is called quantization. **Your task** is to complete the provided code skeleton in order to uniformly requantize an input image down to a given number $n_g$ of gray levels. After requantization, the $n_g$ gray levels should span the entire intensity range $[0, 255]$ of the output image that corresponds to 8 bits per pixel; note that this is not the same as having 256 gray levels.

Experiment with different quantization levels from 8 bits ($2^8 = 256$ gray levels) down to 1 bit (2 gray levels) and observe the output images in comparison to the corresponding input images.

1. As the number of quantization levels decreases, the quality of the image degrades. Why would one want to use fewer quantization levels then?

2. Is it possible to decrease the error of quantization—measured as the mean of some norm of the difference between input and output—while keeping the same number of quantization levels?