ELE510 Image Processing with robot vision: LAB, Exercise 2, Image Formation.

Purpose: *To learn about the image formation process, i.e. how images are projected from the scene to the image plane.*

The theory for this exercise can be found in chapter 2 and 3 of the text book [1]. Supplementary information can found in chapter 1, 2 and 3 in the compendium [2]. See also the following documentations for help:

- OpenCV
- numpy
- matplotlib

IMPORTANT: Read the text carefully before starting the work. In many cases it is necessary to do some preparations before you start the work on the computer. Read necessary theory and answer the theoretical part frst. The theoretical and experimental part should be solved individually. The notebook must be approved by the lecturer or his assistant.

Approval:

The current notebook should be submitted on CANVAS as a single pdf file.

To export the notebook in a pdf format, goes to File -> Download as -> PDF via LaTeX (.pdf).

Note regarding the notebook: The theoretical questions can be answered directly on the notebook using a *Markdown* cell and LaTex commands (if relevant). In alternative, you can attach a scan (or an image) of the answer directly in the cell.

Possible ways to insert an image in the markdown cell:

```
![image name]("image_path")
```

```
<img src="image_path" alt="Alt text" title="Title text" />
```

Under you will find parts of the solution that is already programmed.

You have to fill out code everywhere it is indicated with `...`

The code section under `######## a)` is answering subproblem a) etc.

Problem 1

a) What is the meaning of the abbreviation PSF? What does the PSF specify?

The PSF - Point Spread Function, more generally known as a systems impulse response, is a function that
describes how how an object will look like when processed by an imaging system. It specifies the
shape that a point will take on the image pane.

b) Use the imaging model shown in Figure 1. The camera has a lens with focal length $f = 40\text{mm}$ and in the image plane a CCD sensor of size $8\text{mm} \times 8\text{mm}$. The total number of pixels is 4000×4000. How many lines per mm will this camera resolve at a distance of $z_w = 1\text{m}$ from the camera center?

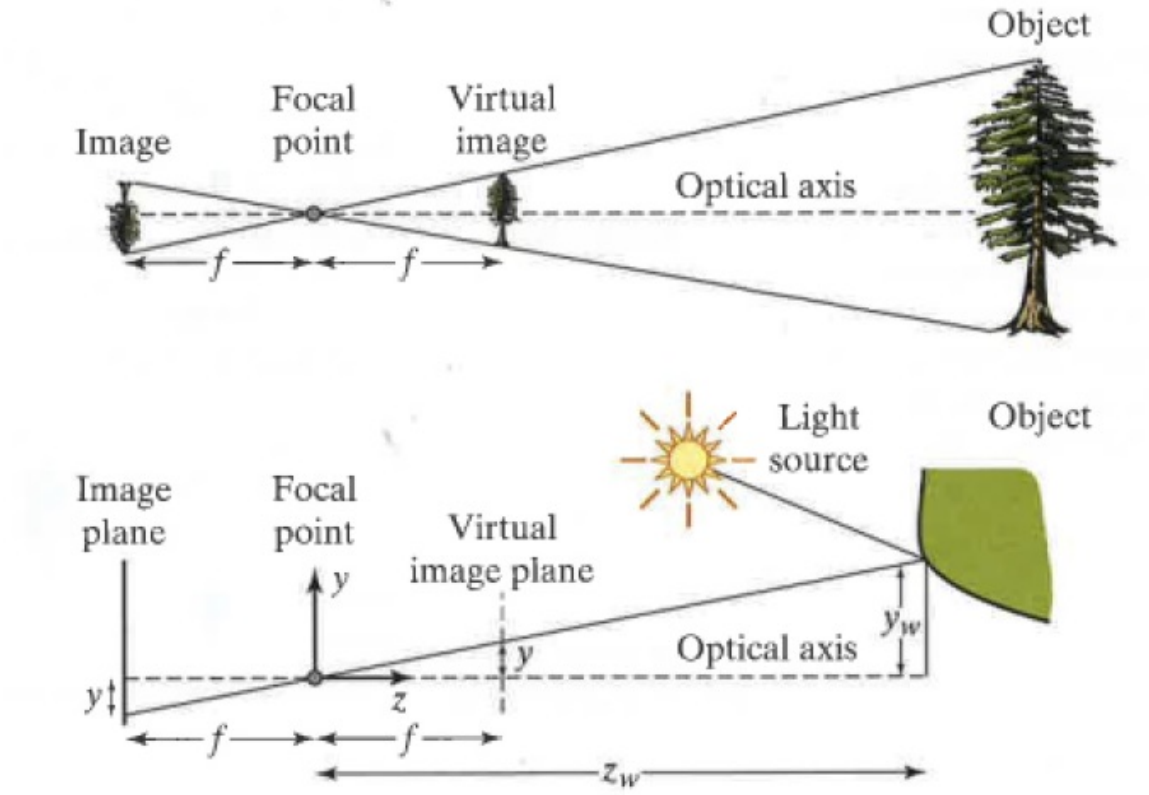


Figure 1: Perspective projection caused by a pinhole camera. Figure 2.23 in [2].

$$d = 8mm/4000pixels = 2\mu m/pixel(line)$$

$$\frac{D}{1m} = \frac{d}{f} = \frac{d}{40mm}$$

$$D = \frac{2 \cdot 10^{-6}}{40 \cdot 10^{-6}} \cdot 1m = \frac{1}{20} mm/pixel(line)$$

The camera will resolve 20 lines per mm, at a distance of, z_w = 1m from the camera center

c) Explain how a Bayer filter works. What is the alternative to using this type of filter in image acquisition?

A Bayer filter is a sensor overlay (or color filter array) which is used to filter light

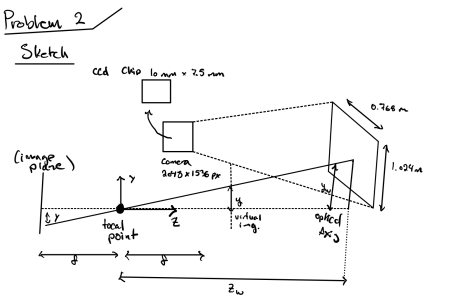
wavelengths. The filter is arranged in a grid-like pattern, where half of the squares are used to sense green, one quarter to sense red and one quarter to sense blue. This method is a compromise between the cost of the sensor and the quality of the image, given that other methods such as using three sensors would be more expensive.

Problem 2

Assume we have captured an image with a digital camera. The image covers an area in the scene of size $1.024\mathrm{m} \times 0.768\mathrm{m}$ (The camera has been pointed towards a wall such that the distance is approximately constant over the whole image plane, *weak perspective*). The camera has 2048 pixels horizontally, and 1536 pixels vertically. The active region on the CCD-chip is $10\mathrm{mm} \times 7.5\mathrm{mm}$. We define the spatial coordinates (x_w, y_w) such that the origin is at the center of the optical axis, x-axis horizontally and y-axis vertically upwards. The image indexes (x, y) is starting in the upper left corner. For simplicity let the optical axis meet the image plane at $(x_0 = 1024, y_0 = 768)$. The solutions to this problem can be found from simple geometric considerations. Make a sketch of the situation and answer the following questions:

a) What is the size of each sensor (one pixel) on the CCD-chip?

b) What is the scaling coefficient between the image plane (CCD-chip) and the scene? What is the scaling coefficient between the scene coordinates and the image indexes?



a) Scene Size = 1024 mm × 768 mm CCD Active Region = 10 mm × 7.5 mm
Camera resolution = 2048 × 1536 px

$$\text{Pixel Size}_x = \frac{10\ \text{mm}}{2048\ \text{px}} = 0.0048\ \text{mm} = 4.8\ \mu\text{m}$$

$$\text{Pixel Size}_y = \frac{7.5\ \text{mm}}{1536\ \text{px}} = 0.0048\ \text{mm} = 4.8\ \mu\text{m}$$

Pixel size = 4.8 μm × 4.8 μm

b) (1) $C_x = \frac{10\ \text{mm}}{1024\ \text{mm}} = 0.097$, $C_y = \frac{7.5\ \text{mm}}{768\ \text{mm}} = 0.097$

(2) $C_x = \frac{1024}{2048} = 0.5$, $C_y = \frac{768}{1568} = 0.5$

The scaling coefficient between the image plane and the scene (1), and between the scene coordinates and the image pixels (2), are 0.097 and 0.5 respectively.

Problem 3

Translation from the scene to a camera sensor can be done using a transformation matrix, T.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} \tag{1}$$

where

$$T = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

α_x and α_y are the scaling factors for their corresponding axes.

Write a function in Python that computes the image points using the transformation matrix, using the parameters from Problem 2. Let the input to the function be a set of K scene points, given by a $2 \times K$ matrix, and the output the resulting image points also given by a $2 \times K$ matrix. The parameters defining the image sensor and field of view from the camera center to the wall can also be given as input parameters.

Test the function for the following input points given as a matrix:

$$\mathbf{P}_{in} = \begin{bmatrix} 0.512 & -0.512 & -0.512 & 0.512 & 0 & 0.3 & 0.3 & 0.3 & 0.6 \\ 0.384 & 0.384 & -0.384 & -0.384 & 0 & 0.2 & -0.2 & -0.4 & 0 \end{bmatrix}. \tag{3}$$

Comment on the results, especially notice the two last points!

In [ ]:
```
# Import the packages that are useful inside the definition of the weakPersp
import math
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
"""
Function that takes in input:
- FOV: field of view,
- sensorsize: size of the sensor,
- n_pixels: camera pixels,
- p_scene: K input points (2xK matrix)

and return the resulting image points given the 2xK matrix
"""
def weakPerspective(FOV, sensorsize, n_pixels, p_scene):
    # pixel size
    p_x, p_y = sensorsize[0] / n_pixels[0], sensorsize[1] / n_pixels[1]

    # scale factors
    a_x, a_y = sensorsize[0] / FOV[0], sensorsize[1] / FOV[1]
    s_x, s_y = a_x / p_x, a_y / p_y
    x_0, y_0 = n_pixels[0] / 2, n_pixels[1] / 2

    # transformation matrix
    T = np.array([[s_x, 0, x_0], [0, s_y, y_0], [0, 0, 1]])
    coords = np.vstack([p_scene, np.ones((1, p_scene.shape[1]))])
    return np.matmul(T, coords)[:2, :]
```

In [ ]:
```
# The above function is then called using the following parameters:
```

```
# Parameters
FOV = np.array([1.024, 0.768])
sensorsize = np.array([10e-3, 7.5e-3])
n_pixels = np.array([2048, 1536])
p_scene_x = [0.512, -0.512, -0.512, 0.512, 0, 0.3, 0.3, 0.3, 0.6]
p_scene_y = [0.384, 0.384, -0.384, -0.384, 0, 0.2, -0.2, -0.4, 0]
```

In [ ]:
```
p_scene = np.array([p_scene_x, p_scene_y])

# Call to the weakPerspective() function
pimage = weakPerspective(FOV, sensorsize, n_pixels, p_scene)

# Result:
print(pimage)
```

```
[[2048.    0.    0. 2048. 1024. 1624. 1624. 1624. 2224.]
 [1536. 1536.    0.    0.  768. 1168.  368.  -32.  768.]]
```

Delivery (dead line) on CANVAS: 16-09-2022 at 23:59

Contact

Course teacher

Professor Kjersti Engan, room E-431

E-mail: kjersti.engan@uis.no

Teaching assistant

Tomasetti Luca, room E-401 E-mail: luca.tomasetti@uis.no

Saul Fuster Navarro, room E-401 E-mail: saul.fusternavarro@uis.no

References

[1] S. Birchfeld, Image Processing and Analysis. Cengage Learning, 2016.

[2] I. Austvoll, "Machine/robot vision part I," University of Stavanger, 2018. Compendium, CANVAS.