Labb 1 - AR

Elias Berglin

18$^{\text{th}}$ November 2021

1 Week 1

1.1 Assignment

1) Derived a camera's field of view θ as a function of focal length f and sensor width w

From figure 1 we can derive a function for θ as a function depending on α and we get the following equation:

$$\theta = 2\alpha \tag{1}$$

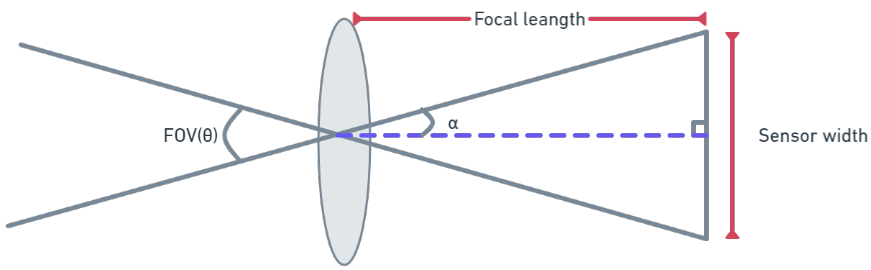

Figure 1: An illustration for deriving the formula for calculating field of view

With equation 1 we can see that we need to find an expression for α. From figure x we can derive α from the right angle triangle and with simple trigonometry we get the following formula:

$$\frac{w}{2 * f} = tan(\alpha) \tag{2}$$

With equation 2 we can now transform it to make it an expression of α. This resuls in the following equation:

$$\alpha = tan^{-1}\left(\frac{w}{2 * f}\right) \tag{3}$$

Now we can finally get an expression for θ with the help of equation 3 and equation 1. We get the following equation:

$$\theta = 2 * tan^{-1}\left(\frac{w}{2 * f}\right) \tag{4}$$

Equation 4 shows the derived formula for calculating the field of view (θ) as function of the focal length (f) and sensor width (w).

2) Allow two different cameras with sensor with w_1 and w_2 to have the same focal length f. Plot field of view θ_1 and θ_2 in a single graph.

The plot for θ as a function of f for the sensor widths w_1 and w_2 is shown in figure 2.

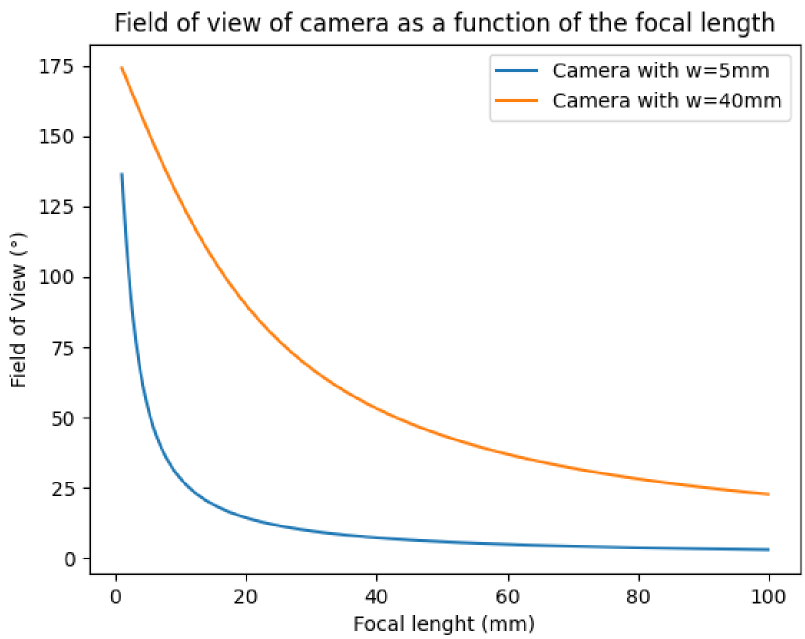

Figure 2: Field of view as a function of focal length plotted with sensor width 5mm and 40mm

3) Consider two world points x_1 and x_2 and their projected 2D points x_1' and x_2' on the sensor plane. Let the points x_1 and x_2 be located in world space such that $x_1 = (x, y, z)$ and $x_2 = (x + dx, y, z)$. Evaluate how the distance between the projected points $|x_2' - x_1'|$ varies as a function of focal lengths f and depth z

Given focal length f and coordinates (x, y, z) we can calculate a projected 2D point with the formula from equation 5.

$$\left(f\frac{x}{z}, f\frac{y}{z}\right) \tag{5}$$

Since the only differentiating factor between x_1 and x_2 is the coordinate x we can focus on that part and simplify the equation to:

$$|x_2' - x_1'| = \left|f\frac{x + dx}{z} - f\frac{x}{z}\right| \tag{6}$$

Simplifying the expression from equation 6 we get the following:

$$|x'_2 - x'_1| = \left| f\frac{dx}{z} \right| \tag{7}$$

Equation 7 shows the final formula for calculating the distance between x_1 and x_2.

1.2 Reflections

The first week was stressful with the SIMS course overlapping. I skipped reading the book and looked at the YouTube course. The YouTube course provided a good explanation on the part of the assignment that was a bit unclear. More specifically it provided good insight on the theory behind the third question and was helpful since lecture 1 overlapped to lecture 2. The seminar at the end of the week cleared up the structure of the course. The one weird thing I noticed is that there is not ethical discussion in the YouTube course, I think that this should be an important part of any computer vision course. The most interesting technique that I learned about this week was depth recovery from normal images.

2 Week 2

2.1 Assignment

The Python code can be found in Appendix A

2.2 Reflections

This week was a little better. It was good to use the seminar to catch up on the rest of the lecture. The assignment was interesting. I simplified the assignment a bit and used cv2 to pad the image instead of padding it myself. This week I got a bit more out of the seminar when I asked about the different color scales. This week I had time to read the chapters in the book and watch the YouTube course. In my opinion, the book is a bit all over the place. Figures are placed a few pages back from where they were first referenced. This made following the book harder. The YouTube course is easier to follow, and it cleared up some things I did not completely understand from the book. The most interesting for me this week was the different padding options when applying a kernel to keep the same size. It was interesting to see how different padding techniques affected the image.

Appendix A Python code for image filter

```
            from matplotlib import image
    import numpy as np
    from matplotlib.image import imread
    from matplotlib.image import imsave
    from PIL import Image
    from numpy.core.fromnumeric import shape
    import cv2
    from skimage.exposure import rescale_intensity
    from skimage.exposure.exposure import _output_dtype


    def convolve(img, kernel):
        iH, iW = img.shape[:2]
        _, kW = kernel.shape[:2]
        pad = (kW - 1) // 2
        img = cv2.copyMakeBorder(img, pad, pad, pad, pad, cv2.
            BORDER_REPLICATE)
        output = np.zeros((iH, iW), dtype="float32")

        for y in np.arange(pad, iH + pad):
            for x in np.arange(pad, iW + pad):
                roi = img[y - pad:y + pad + 1, x - pad:x + pad + 1]
                k = (roi * kernel).sum()
                output[y - pad, x - pad] = k
        output = rescale_intensity(output, in_range=(0, 255))

        output = (output * 255).astype("uint8")
        return output


    def convolveRGB(img, kernel):
        iH, iW = img.shape[:2]
        _, kW = kernel.shape[:2]
        pad = (kW - 1) // 2
        img = cv2.copyMakeBorder(img, pad, pad, pad, pad, cv2.
            BORDER_REPLICATE)
        output = np.zeros((iH, iW, 3), dtype="float32")

        for y in np.arange(pad, iH + pad):
            for x in np.arange(pad, iW + pad):
                roi = img[y - pad:y + pad + 1, x - pad:x + pad + 1]
                r = (roi[:, :, 0] * kernel).sum()
                g = (roi[:, :, 1] * kernel).sum()
                b = (roi[:, :, 2] * kernel).sum()
                k = [r, g, b]
                output[y - pad, x - pad] = k
        output = rescale_intensity(output, in_range=(0, 255))

        output = (output * 255).astype("uint8")
        return output


    roi = np.array(([
        [[250, 253, 251], [250, 253, 251], [250, 253, 251]],
        [[250, 253, 251], [250, 253, 251], [250, 253, 251]],
```

```
        [[250, 253, 251], [250, 253, 251], [250, 253, 251]]
    ]), dtype="int")


    sharp = np.array((
        [0, -1, 0],
        [-1, 5, -1],
        [0, -1, 0]), dtype="int")

    blurr = np.array(([
        [0.0625, 0.125, 0.0625],
        [0.125, 0.25, 0.125],
        [0.0625, 0.125, 0.0625]]), dtype=float)

    edge = np.array(([
        [-1, -1, -1],
        [-1, 8, -1],
        [-1, -1, -1]
    ]), dtype="int")

    rightSobel = np.array(([
        [-1, 0, 1],
        [-2, 0, 2],
        [-1, 0, 1]
    ]), dtype="int")

    emboss = np.array(([
        [-2, 1, 0],
        [-1, 1, 1],
        [0, 1, 2]
    ]), dtype="int")

    gblurr = np.array(([
        [1, 4, 6, 4, 1],
        [4, 16, 24, 16, 4],
        [6, 24, 36, 24, 6],
        [1, 4, 6, 4, 1],
        [4, 16, 24, 16, 4]
    ]), dtype="float")

    gblurr = gblurr * (1/256)

    umask = np.array(([
        [1, 4, 6, 4, 1],
        [4, 16, 24, 16, 4],
        [6, 24, -476, 24, 6],
        [1, 4, 6, 4, 1],
        [4, 16, 24, 16, 4]
    ]), dtype="float")

    umask = umask * (-1/256)

    img = cv2.imread("bilder/ill.png")
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    save = Image.fromarray(img)
    save.save('original.png')
```

```python


    print("Sharpening")
    sharpend = convolveRGB(img, sharp)
    save = Image.fromarray(sharpend)
    save.save('sharp.png')

    print("Blurring")
    blurred = convolveRGB(img, blurr)
    save = Image.fromarray(blurred)
    save.save('blurr.png')

    print("Detecting edges")
    edged = convolveRGB(img, edge)
    save = Image.fromarray(edged)
    save.save('edge.png')

    print("Right sobel")
    rSobel = convolveRGB(img, rightSobel)
    save = Image.fromarray(rSobel)
    save.save('right_sobel.png')

    print("Emboss")
    em = convolveRGB(img, emboss)
    save = Image.fromarray(em)
    save.save('embossed.png')

    print("G-blurr")
    gb = convolveRGB(img, gblurr)
    save = Image.fromarray(gb)
    save.save('gblurr.png')

    print("umask")
    um = convolveRGB(img, umask)
    save = Image.fromarray(um)
    save.save('umask.png')
```