0.1 Camera Geometries for a birds-eye perspective

The goal of these calculations is to project the captured floor to a birds-eye view. To make a perspective transformation (in python) we need to define points in the image, input-points, which will be the outer corners of the newly transformed image, output-points. For this purpose, we need to determine certain geometric values of the FOV, field of view, and the surrounding to crop accordingly.

We assume having a camera at a given height h and a given tilt angle β, as well as the sensor size SS in mm. In the following we respect the sensor being rectangular, thus we have a horizontal sensor size SS_h and a vertical SS_v. Remember that usually, image sensors have $SS_h > SS_v$. Furthermore, we know die final image size in pixels with width P_x and height P_y. In the sketches 1 and 2, space with white background is visible in the camera. The fat horizontal line marks the floor. The vertical line at the right at point B which is cutting the FOV represents a wall, the horizon or the end of the usable floor footage, respectively. In the following, we refer to this as the *horizon wall*.

0.1.1 Geometrics

Given (0.1.1) and (0.1.2) we get the FOV angles of the camera. In the case that the users assume the image to be quadratic, you can ignore the specific FOV angles and replace them all by α. In the following calculations please take reference in image 1.

$$\frac{\alpha_h}{2} = \arctan\left(\frac{SS_h}{2 \cdot f}\right) \tag{0.1.1}$$

$$\frac{\alpha_v}{2} = \arctan\left(\frac{SS_v}{2 \cdot f}\right) \tag{0.1.2}$$

0.1.1.1 Side view geometrics

Given now the FOV angle $\alpha/2$ we can calculate some basic geometries of the FOV by the camera's position. In doing so, using β,h and α we get the diagonal distance where the FOV first hits the floor d_i in (0.1.3). Furthermore we compute d_{hi} as the horizontal distance of d_i in (0.1.4) and d_m in (0.1.5), which is the horizontal distance where the middle of the FOV hits the floor.

$$d_i = \frac{h}{\cos(\beta - \alpha_v/2)} \tag{0.1.3}$$

$$d_{hi} = \tan\left(\beta - \frac{\alpha_v}{2}\right) \cdot h \tag{0.1.4}$$

$$d_m = \tan(\beta) \cdot h \tag{0.1.5}$$

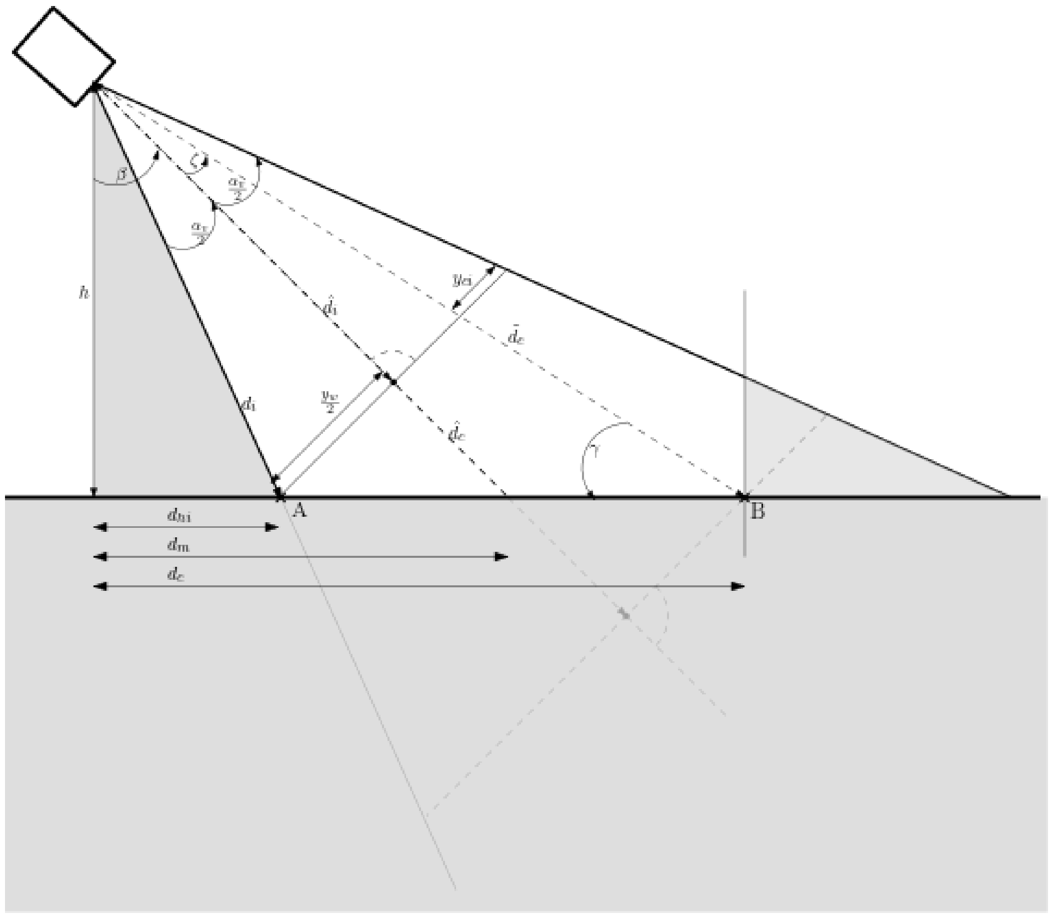

Figure 1: side view

To facilitate the user interaction to adjust the horizon distance we claim the wall distance d_c to be a multiple of d_m, thus $d_c = a \cdot d_m$ with $a \in \mathbb{R}^+$ given by the user. The imaginary view line from the camera to the horizon wall at B is given by $\tilde{d}_c$ with its floor angle γ.

$$\gamma = \arctan\left(\frac{h}{d_c}\right) \tag{0.1.6}$$

$$\tilde{d}_c = \frac{h}{\sin(\gamma)} \tag{0.1.7}$$

For calculations, we then introduce the angle ζ between the line of $\tilde{d}_c$ and the middle FOV line.

$$\zeta = (\pi/2 - \gamma) - \beta \tag{0.1.8}$$

For later use, we also need the orthogonal distance of B to the camera. This is given by $\hat{d}_c$.

$$\hat{d}_c = \cos(\zeta) \cdot \tilde{d}_c \tag{0.1.9}$$

Now we assume an imaginary, orthogonal image plane in the FOV. Its distance to the camera is given by $\hat{d}_i$ and its width by y_w.

$$\hat{d}_i = cos\left(\frac{\alpha_v}{2}\right) \cdot d_i \tag{0.1.10}$$

$$\frac{y_w}{2} = \cos(\beta) \cdot (d_m - d_{hi}) \tag{0.1.11}$$

As we want to crop the image vertically at point B, we need to determine its equivalent point in the imaginary image plane. To do so, we use the angle ζ to obtain the cropping measure y_{ci} with

$$y_{ci} = \frac{y_w}{2} - \tan(\zeta) \cdot \hat{d}_i \tag{0.1.12}$$

0.1.1.2 Top view geometrics

Now we go on with some measures visible in the top-down view, see image 2. First the orientation: for visibility we assume that the vertical line at point A is where the FOV hits the floor, compare image 1. At this point, the image effectively begins.

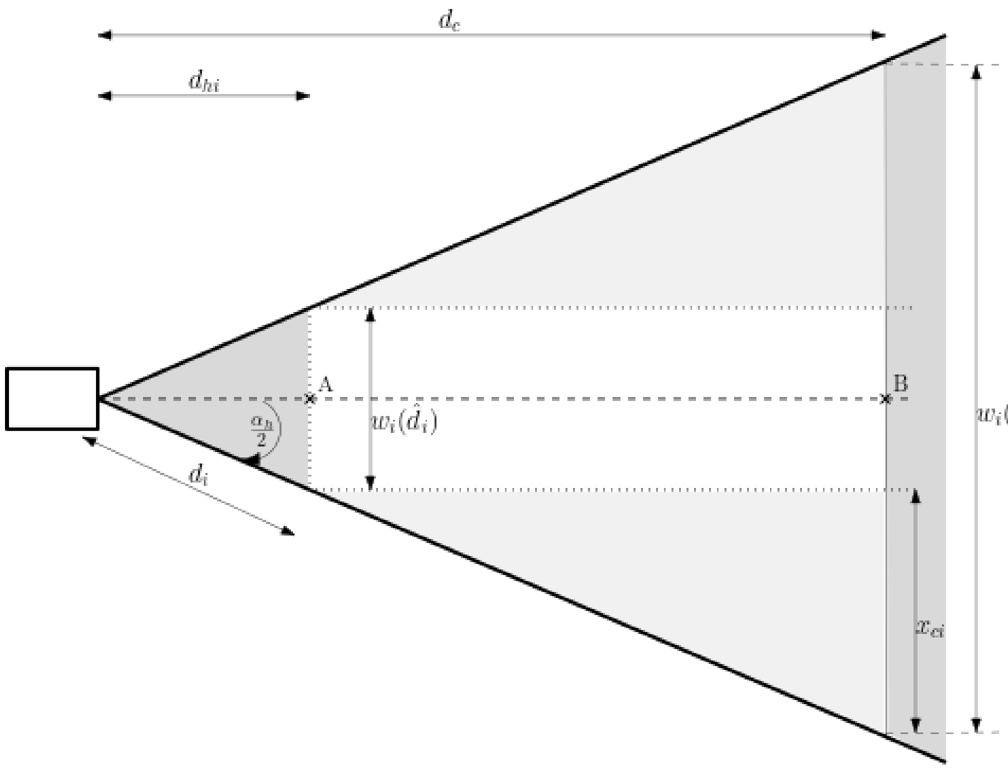

Figure 2: top view

Point A is thus located at a horizontal distance from the camera at d_{hi}. How wide the FOV of the image at this point is, however, is defined by another variable, $\hat{d_i}$. This is due to the fact that the FOV is mainly affected by the angles α_v and α_h. This means that the opening angle of the camera lense at this point A is determined by the orthogonal distance $\hat{d_i}$. So we call this FOV width $w_i(\hat{d_i})$ and can compute it with:

$$w_i(\hat{d_i}) = 2 \cdot \tan\left(\frac{\alpha_h}{2}\right) \cdot \hat{d_i} \tag{0.1.13}$$

Equivalently we do at point B which lays at distance d_c but has a FOV width affected by the orthogonal distance $\hat{d_c}$.

$$w_i(\hat{d_c}) = 2 \cdot \tan\left(\frac{\alpha_h}{2}\right) \cdot \hat{d_c} \tag{0.1.14}$$

Note that $w_i(\hat{d_c})$ is the complete width of the lense opening at the horizontal distance at d_c. This is the width in the taken image, but not yet the width shown in the projected image. As we want the projection of the floor from a birds-eye perspective, we need to determine the orthogonal rectangle on the floor. As its width is defined by $w_i(\hat{d_i})$ we now need to calculate the image width we need to crop according to the FOV width $w_i(\hat{d_c})$ at B. Thus we define x_{ci}, which, for the FOV angle, depends on $\hat{d_i}$ and $\hat{d_c}$.

$$x_{ci} = \tan\left(\frac{\alpha_h}{2}\right) \cdot (\hat{d_c} - \hat{d_i}) \tag{0.1.15}$$

0.1.2 Cropping

As the real FOV first hits the floor at A, with no obstacles in between, we claim that the bottom line of the real image also starts there. Thus, we only need to cut the y-dimension for the horizon wall. For this purpose, we assume the relation

$$\frac{y_{cp}}{P_y} = \frac{y_{ci}}{y_w} \tag{0.1.16}$$

with y_{cp} as the pixel equivalent to y_{ci}. This means that the image height to crop, y_{ci}, has to have the same relation to $w_i(\hat{d_c})$ as the pixel height to crop, y_{cp}, to P_y. Thus we get

$$y_{cp} = \frac{P_y \cdot y_{ci}}{y_w} \tag{0.1.17}$$

For the x-dimension of the input-points we state the same condition for the bottom image line. Equivalently to the y-direction, we get the relation

$$\frac{x_{cp}}{P_x} = \frac{x_{ci}}{w_i(\hat{d_c})} \tag{0.1.18}$$

with the pixel width to crop

$$x_{cp} = \frac{P_x \cdot x_{ci}}{w_i(\hat{d}_c)} \tag{0.1.19}$$

0.1.3 Aspect-ratio

Finally, we have to adjust the aspect-ratio of the output-points to fit the real-world distances and ratios. The real-world ratio is given by the floor rectangle width and its height. Therefore we have to equal this ratio to the relation of the pixel width to height. Note that we now have to use d_c and d_{hi} as they define the floor rectangle size.

$$\frac{P_x}{P_{yn}} = \frac{w_i(\hat{d}_i)}{d_c - d_{hi}} \tag{0.1.20}$$

So we get the corrected pixel height P_{yn}

$$P_{yn} = \frac{P_x(d_c - d_{hi})}{w_i(\hat{d}_i)} \tag{0.1.21}$$

0.2 Extended width projection

However, in some cases the simple planar area pictured in the middle of an image might be too narrow. Thus, the area of interest is of extended width. For visualization see the blue area in 3. As follows from the mathematical derivation in 0.1 one needs to extend the image's width artificially since otherwise it lacks pixels being projected. This extension is colored green in 3 while the necessary pixels are in the overlapping of blue and green. For further comprehension see 4 with the same colors.
In the following, variables with the footnote r mean that they are reverse engineered. They still carry the same meaning like in the sections before, but are now calculated differently.
The difficulty in this manner is that we do not know all the points necessary for the projection. The upper points are given by the user input. However, the lower points, depending on the extension, are unknown. Note that before we got the input in meters and calculated to corresponding pixels in the image. Now it's the reverse.

In the following, we can assume that a user input that completely defines the upper points with $[(x_1, y_1), (x_2, y_2)] = [(x_{cp,i}, y_{cp,i}), (P_x - x_{cp,i}, y_{cp,i})]$. In the previous attempt, the distance d_c, in meters, is passed by the user. Now, the user passes x_{cp}, which is the corresponding measure in pixels. To revers engeneer d_c one needs ζ and γ, which then depend on y_{ci}. This can be obtained by $y_{cp,i}$ from (0.1.16) with

$$y_{ci,r} = \frac{y_w \cdot y_{cp,i}}{P_y} \tag{0.2.1}$$

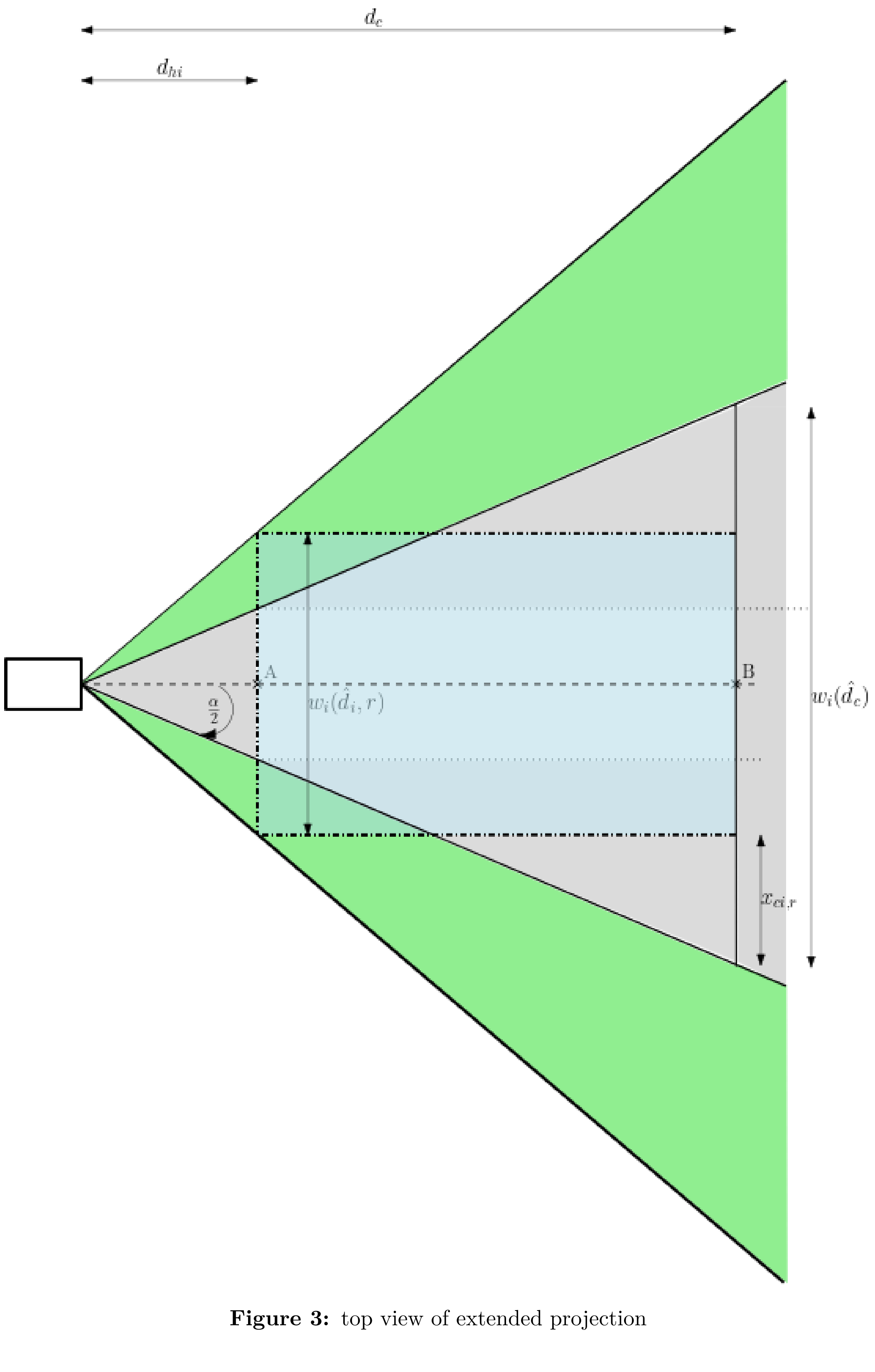

Figure 3: top view of extended projection

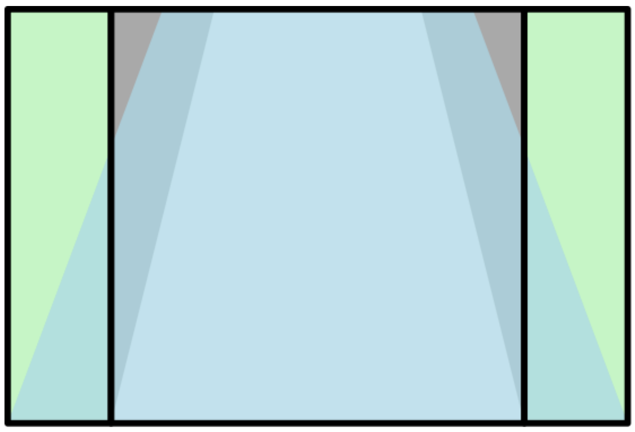
Figure 4: extended image dimensions

This gives for *zeta* and γ and thus d_c

$$\zeta = \arctan\left(\frac{y_{wh} - y_{ci,r}}{d_{id}}\right) \tag{0.2.2}$$

$$\gamma = \pi/2 - \gamma - \beta \tag{0.2.3}$$

$$d_c = \frac{h}{\tan(\gamma)} \tag{0.2.4}$$

The measures $\tilde{d}_c$, $\hat{d}_c$, $w_i(\hat{d}_i)$ and $w_i(\hat{d}_c)$ stay the same. To get the image extension we need the pixel width of the desired area. Thus, first x_{ci}. From (0.1.18) we get

$$x_{ci,r} = \frac{w_i(\hat{d}_c) \cdot x_{cp,i}}{P_x} \tag{0.2.5}$$

The new planar width is then given as

$$w_i(\hat{d}_i, r) = w_i(\hat{d}_c) - 2 \cdot x_{ci,r} \tag{0.2.6}$$

allowing to define the new image width

$$\frac{w_i(\hat{d}_i, r)}{w_i(\hat{d}_i)} = \frac{P_{x,r}}{P_x} \tag{0.2.7}$$

$$P_{x,r} = \frac{P_x \cdot w_i(\hat{d}_i, r)}{w_i(\hat{d}_i)} \tag{0.2.8}$$

With these results, we now have the width of the extension on either side. The bottom corners of the image are thus the lower points of the projection.

$$ext_x = 0.5(P_{x,r} - P_x) \tag{0.2.9}$$

The newly calculated P_{yn} stays the same as before since this one determines the original aspect ratio which remains unchanged.