TTK4255 Robotic Vision, Spring 2020

Grid Detection Dataset

Simen Haugo (simen.haugo@ntnu.no)

Dataset description

The dataset contains multiple image sequences of varying difficulty.

- `seq1/img0000-img0500.jpg`: Easy, majority of grid visible in each image.
- `seq2/img0000-img0310.jpg`: People walking past camera.
- `seq3/img0000-img0250.jpg`: Viewing from outside surrounding net.

The camera is approximately level with the ground in `seq1` and `seq2`, but substantially tilted in `seq3`. The camera height is between 1 and 2 meters.

Camera model

Modeling fisheye lenses is not part of the curriculum, so we provide an appropriate camera model. The model is equivalent to the OpenCV fisheye model with a single radial distortion coefficient. Python and Matlab implementations of the projection equations described below is included in the dataset.

Fisheye projection

The model is best understood by rewriting camera coordinates (X, Y, Z) using spherical coordinates:

$$X = \lambda \sin\theta \cos\phi \tag{1}$$
$$Y = \lambda \sin\theta \sin\phi \tag{2}$$
$$Z = \lambda \cos\theta \tag{3}$$

Here, λ is the distance of the point from the center of projection, θ is its angle against the optical axis, and ϕ is its angle around the optical axis. The projection of the point into the fisheye image is then given by

$$u = c_x + f\theta(1 + k\theta^2)\cos\phi \tag{4}$$
$$v = c_y + f\theta(1 + k\theta^2)\sin\phi \tag{5}$$

where f, c_x, c_y, k are the fisheye model parameters (included in the dataset).

Converting from and to rectilinear projection

If we rewrite the equations for a rectilinear projection (ideal pinhole) using spherical coordinates, the projection of a point into the image is given by

$$u = c_x + f \tan\theta \cos\phi \tag{6}$$
$$v = c_y + f \tan\theta \sin\phi \tag{7}$$

To convert a point in a rectilinear image into a point in the original fisheye image, you can use the procedure from exercise 2. First, invert the rectilinear projection to obtain $\theta, \cos\phi, \sin\phi$. Then, project back into the fisheye image using the projection equations.

To go the other way, converting a point in the fisheye image to a point in a rectilinear image, you need to invert the fisheye equations to find $\theta, \cos\phi, \sin\phi$. Doing this algebraically requires solving a cubic polynomial in θ, which can be tricky. Instead, you can solve for θ using numerical root finding, e.g. Newton-Raphson iteration (see Wikipedia).