Lecture 11, Oct 10, 2025

Camera Calibration

- *Camera calibration* is the process of characterizing the geometric and photometric properties of the imaging system in use
  - Geometric detection, e.g. determining $\boldsymbol{K}$, is what we are primarily concerned with; photometric calibration (e.g. getting consistent brightness and colour values across pixels) is less important since we use techniques such as feature detection, which are invariant to photometric properties
  - This is done by using a calibration pattern with a known geometry, then solving for the camera parameters using the known quantities of the target
- The most common target for geometric calibration is a flat checkerboard pattern
  - Critical to keep the checkerboard flat and include a variety of views (multiple angles, distances, fully covering the image plane)
  - Other patterns exist, e.g. intersecting orthogonal planes, circle patterns, etc
  - Checkerboard is good because the cross junctions are not affected by foreshortening, and the simple square pattern is easy to detect, allowing for pixel or subpixel-level accuracy
  - For a circle, the effect of foreshortening distorts it into an ellipse, but critically the centre of this ellipse is not the same as the original centre of the circle
- Recall our forward imaging model: $\boldsymbol{x}_{ij} = f(\boldsymbol{p}_i; \boldsymbol{C}_j, \boldsymbol{t}_j, \boldsymbol{K}, \boldsymbol{\kappa}, \boldsymbol{\tau})$ for point i in camera j
  - For calibration, we want to recover all of these parameters; most often we're interested in $\boldsymbol{K}, \boldsymbol{\kappa}, \boldsymbol{\tau}$ but all parameters need to be estimated to solve the problem
  - The n-planes approach is to take n images of our known calibration target, so for each image we get a different set of extrinsic parameters
    - This means we need at least a minimum number of points per image for this to work
- Targetless calibration and online calibration (as the robot is operating) is also possible, but much harder; most often we want to solve the calibration problem given known positions of landmark points in some frame (usually the target's frame)
  - DLT can be used, but it does not handle radial or tangential distortion, so it will be a very bad approximation
  - DLT also doesn't work when all points are coplanar, which is often the case
- We can often get an initial guess and use NLS, using all feature points from all views in a big single optimization process
  - The structure of the Jacobian can be used for some speedups, but practically computers are already fast enough
  - For NLS, we optimize the sum of squared reprojection errors, i.e. predicted location minus observed location
  - $E_{NLS}(\delta\boldsymbol{\theta}) = \sum_{ij} \left\| \frac{\partial f}{\partial \boldsymbol{C}_j}\delta\boldsymbol{C}_j + \frac{\partial f}{\partial \boldsymbol{t}_j}\delta\boldsymbol{t}_j + \frac{\partial f}{\partial \boldsymbol{K}}\delta\boldsymbol{K} + \frac{\partial f}{\partial \boldsymbol{\kappa}}\delta\boldsymbol{\kappa} + \frac{\partial f}{\partial \boldsymbol{\tau}}\delta\boldsymbol{\tau} - \boldsymbol{r}_{ij} \right\|^2$
    - Note the partials should be taken with respect to each parameter individually (a bit of abuse of notation, especially for the matrix...)
- In the Jacobian, only the extrinsics for one measurement affects that measurement, but the intrinsics and distortion parameters affect all measurements, so we get a sparse structure like the following figure
- There are many existing toolboxes for calibration: MATLAB, OpenCV, ROS1/ROS2
- For stereo calibration, we need to add a rigid body transformation between the cameras to our optimization parameters
  - We can either do intrinsics for the 2 cameras separately, or include both intrinsics along with the transformation between cameras in a big optimization
  - For stereo the projection matrices have the following form:
    - $\tilde{\boldsymbol{P}}_L = \begin{bmatrix} \boldsymbol{K}_L & 0 \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{C}_{L,W} & \boldsymbol{t}_{L,W} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \tilde{\boldsymbol{K}}\boldsymbol{T}_{L,W}$
    - $\tilde{\boldsymbol{P}}_R = \begin{bmatrix} \boldsymbol{K}_R & 0 \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{C}_{R,L} & \boldsymbol{t}_{R,L} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{C}_{L,W} & \boldsymbol{t}_{L,W} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \tilde{\boldsymbol{K}}\boldsymbol{T}_{R,L}\boldsymbol{T}_{L,W}$

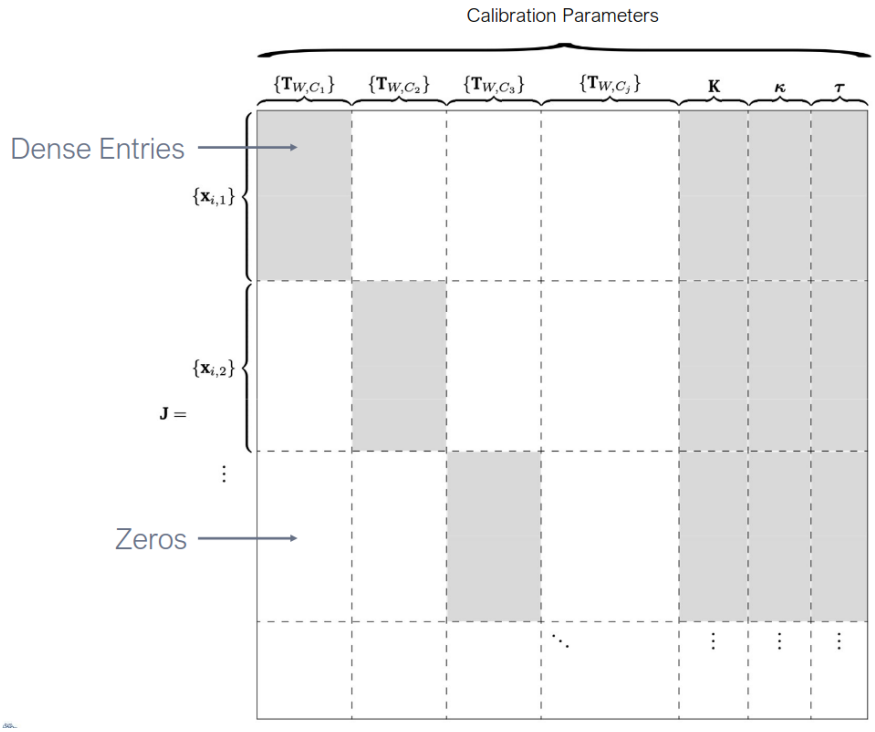

Figure 1: Structure of the Jacobian matrix.

* Multiply a world point $\tilde{\boldsymbol{p}}_i$ by these matrices to get the corresponding pixel locations in the left or right camera

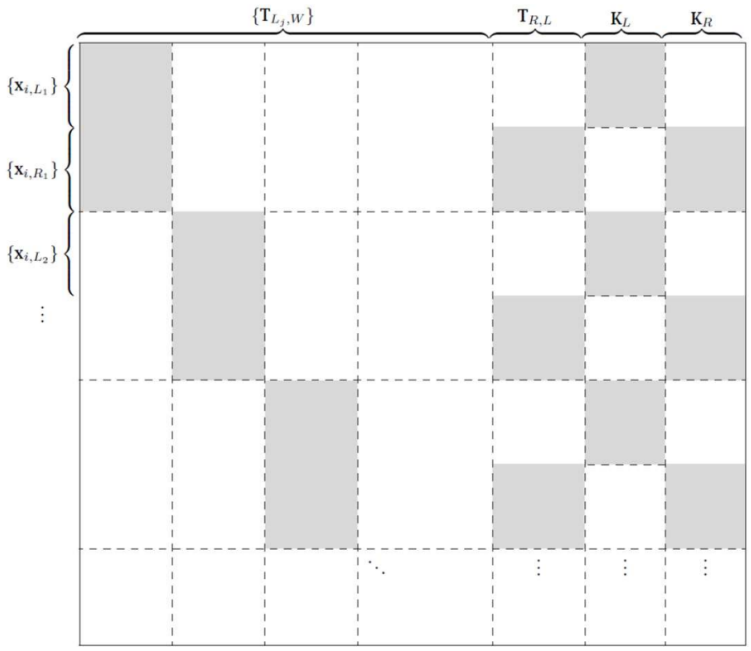

Figure 2: Structure of the Jacobian matrix for stereo calibration (note distortion parameters are left out).