Master Project ASL

Frederike Dmbgen

June 8, 2016

Chapter 1

Rectification

1.1 Rectification test

The following test was performed to verify the correctness of the pixel associations with respect to the rectified images given by *image_proc* of `ROS`. The difference of intensities at corresponding pixels in the original images I_j and rectified images I_j^{rect} are given by:

$$\Delta I_{i,j} = I_j(u_{i,j}) - I_j^{rect}(w_{i,j}) \quad \text{for } i = 1 \dots n, \quad j = 1, 2 \quad , \tag{1.1}$$

where $u_{i,j}$, $w_{i,j}$ denote the pixel location of the object point X_i in the original and rectified image of camera j respectively.

If the rectified pixels are computed in accordance with the rectification process given by `ROS`, this difference should be approximately zero or

$$I_j(u_{i,j}) = I_j^{rect}(w_{i,j}) + \epsilon_{i,j} \quad \text{for } i = 1 \dots n, \quad j = 1, 2 \quad , \tag{1.2}$$

with the error $\epsilon_{i,j}$ arising solely from interpolation.

The pixels of the rectified images are computed using the projection matrices $P_j = K' \left[I|t_j\right]$ and rotation matrices R_j obtained from the *CameraInfo* message with [2]

$$\begin{aligned} w_{i,j} &= K'(R_j T_j(X_i)) \quad \text{, where} \qquad (1.3) \\ T_j(X_i) &= \pi({}_{C_j}r_{C_j X_i}) \qquad (1.4) \\ &= \pi(C_{C_j M}(X_i - {}_M r_{MC_j})) \quad \text{and} \qquad (1.5) \\ \pi(\left(x, y, z\right)^T) &= \left(x/z, y/z, 1\right)^T \quad . \qquad (1.6) \end{aligned}$$

The parameters related to the pose of each camera are obtained from

$$\begin{aligned} C_{C_j M} &= C_{C_j S} C_{SM} = R_j^T C_{SM} \quad \text{and} && (1.7)\\ {}_M r_{MC_j} &= C_{MS}({}_S r_{SC_j} - {}_S\, r_{SM}) && (1.8)\\ &= C_{MS}(-t_j - {}_S\, r_{SCM}) \quad . && (1.9) \end{aligned}$$

The pose of the stereo frame with respect to the map frame (C_{MS}, ${}_M r_{MS}$) is estimated such that all map point projections lie inside both images.

The last information missing for calculating the pixel correspondences are the distorted pixels which are simply given by

$$u_{i,j} = K_j D_j(T_j(X_i))) \quad \text{for } i = 1 \dots n, \quad j = 1, 2 \quad , \tag{1.10}$$

with K_j and D_j the camera matrix and distortion function and T_j the same projection function as given above.

A look at Figures 1.1a and 1.1b shows that the resulting error between the rectified image and the original image is indeed uniformly small and with only very light patterns observable in both left and right images. These patterns get stronger as we approach the image border, where interpolation effects become more important. The pixel correspondence equations developed above are thereby validated for the rectified images provided by *image_proc*.

1.2 Transformation test

Similar considerations can be applied for validating the interpretation of the camera parameters to compute the transformation between the left and the right camera.

The following assumptions need to be tested [3], [4, p. 523f].

$$C_{SC_j} = R_j \quad \text{and} \quad {}_S r_{SC_j} = -t_j \quad \text{for j=1,2} \quad . \tag{1.11}$$

If the above assumptions hold, then one can obtain the point coordinates in the second camera frame from the ones in the first camera frame by applying

$$\begin{align} {}_{C_2}r_{C_2X_i} &= C_{C_2C_1}({}_{C_1}r_{C_1X_i} + {}_{C_1}r_{C_1C_2}) \quad \text{for } i = 1 \ldots n\text{, with} \tag{1.12}\\ C_{C_2C_1} &= R_2^T R_1 \quad \text{and} \tag{1.13}\\ {}_{C_1}r_{C_1C_2} &= C_{C_1S}({}_S r_{SC_2} - {}_S r_{SC_1}) \tag{1.14}\\ &= R_1^T(t_1 - t_2) \quad . \tag{1.15} \end{align}$$

The same transformations as before are then applied to project these points in the respective rectified images.

As can be seen from Figure 1.1c, the resulting intensity difference image is identical with the one from the preceding procedure where the pixels were directly projected from the world frame to the left and right image (Figure 1.1b). Therefore, the transform between the left and right image is computed correctly. *No! This only tells me that I was consistent with my transformations and not that the transformations are indeed right... (Check with examples with correct depth information if the transformations are right.)*

The photometric error in the original and rectified image are is shown in Figures 1.2a and 1.2b. Since at this stage, no reliable depth information is available, low photometric errors cannot be expected, which explains the relatively high average value. However, we know that the photometric errors should be approximately the same in the rectified and non-rectified images. This behaviour is indeed observable, as shown by the difference image (Figure 1.2c).

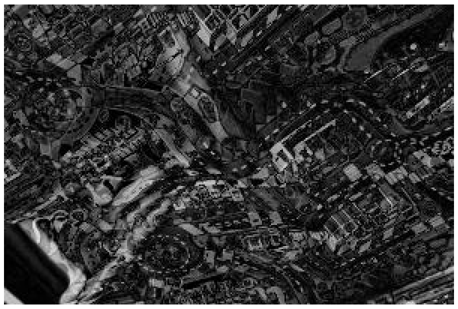
(a) Left image (mean: 4.78).

(b) Right image (mean: 4.34).

(c) Right image obtained with transformation (mean: 4.34).

Figure 1.1: Rectification intensity errors $\Delta I_{i,j}$ with $n_x = 300$ and $n_y = 200$.

(a) Original image (mean: 44.97).


(b) Rectified image (mean: 46.23).

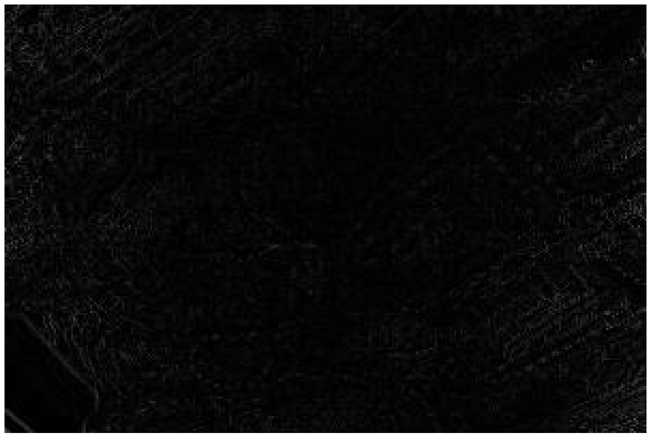

(c) Difference of original and rectified image (mean: 6.71).

Figure 1.2: Rectification residual errors with $n_x = 300$ and $n_y = 200$.

Chapter 2

Optimization

Why add bending energy as regularizer?Because when there are not many measurements, the hessian Matrix (A'A) would be singular and not invertible. (Which is a problem when trying to solve b =(A'A)A'x. The bending energy further offers the advantage over other choices that it leads to a smooth transition between well-populated regions if their in between is not well populated (as opposed to other choices that would tune the "in between"to zero instead).

Optimization procedure:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}\in\mathbb{R}^M}{\arg\min} \sum_{i=0}^{N} r_i(\mathbf{a})^2 + \beta(\mathbf{a}) + \dots \tag{2.1}$$

$$\mathbf{r} : \mathbb{R}^M \to \mathbb{R}^N \tag{2.2}$$

$$\mathbf{a} \in \mathbb{R}^M \tag{2.3}$$

$$J(\mathbf{a}) = \begin{bmatrix} \nabla r_1(\mathbf{a})^T \\ \nabla r_2(\mathbf{a})^T \\ \vdots \\ \nabla r_N(\mathbf{a})^T \end{bmatrix} \tag{2.4}$$

$$\nabla f(\mathbf{a}) = J(\mathbf{a})^T \mathbf{r}(\mathbf{a}) \tag{2.5}$$

$$\nabla^2 f(\mathbf{a}) \approx J(\mathbf{a})^T J(\mathbf{a}) \tag{2.6}$$

$$\hat{\mathbf{a}} = \underset{\mathbf{a}\in\mathbb{R}^M}{\arg\min} \frac{1}{2} \sum_{i=0}^{N} r_i(\mathbf{a})^2 \tag{2.7}$$

$$J_k(\mathbf{a})^T J_k(\mathbf{a}) \mathbf{p}_k^{GN} = -J_k(\mathbf{a})^T \mathbf{r}_k(\mathbf{a}) \tag{2.8}$$

$$\mathbf{a}_{k+1} = \mathbf{a}_k + \alpha_k \mathbf{p}_k^{GN} \tag{2.9}$$

Leastsquares: [1, p.245-269]
Line search: [1, p.30f]
Gauss newton: [1, p.254]
Armijo backtracking: (not suited for quasi newton and conjugate gradient methods!) [1, p.37]

Bibliography

[1] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag New York, Inc., second ed., 1999.

[2] ROS, *wiki CameraInfo*. `http://wiki.ros.org/image_pipeline/CameraInfo`, 2011.

[3] ——, *docs CameraInfo*. `http://docs.ros.org/api/sensor_msgs/html/msg/CameraInfo.html`, 2016.

[4] B. Siciliano and O. Khatib, *Springer, Handbook of Robotics*, Springer-Verlag New York, Inc., first ed., 2007.