



Multiple View Geometry: Exercise Sheet 8

Prof. Dr. Daniel Cremers, Robert Maier, Rui Wang, TU Munich

<http://vision.in.tum.de/teaching/ss2016/mvg2016>

Exercise: June 30th, 2016

Part II: Practical Exercises

In this exercise you will implement direct image alignment as Gauss-Newton minimization on $SE(3)$. Download the package `ex8.zip` provided on the website. It contains a code framework, test images and the corresponding camera calibration.

1. Implement a function `[Id, Dd, Kd] = downscale(I, D, K, level)` which (recursively) halves the image resolution of the image I , the depth map D and adjusts the corresponding camera matrix K per pyramid level (according to the slides). For an input frame of dimensions 640×480 (level 1), level 2 corresponds to 320×240 pixels, level 3 corresponds to 160×120 pixels and so on. For the intensity image, downscaling is performed by averaging the intensity, that is

$$I_d(x, y) := 0.25 \sum_{x', y' \in O(x, y)} I(x', y') \quad (1)$$

where $O(x, y) = \{(2x, 2y), (2x + 1, 2y), (2x, 2y + 1), (2x + 1, 2y + 1)\}$.

For the depth map, downscaling is performed by averaging the depth of all valid pixels (invalid depth values are set to zero), that is

$$D_d(x, y) := \left(\sum_{x', y' \in O_d(x, y)} D(x', y') \right) / |O_d(x, y)| \quad (2)$$

where $O_d(x, y) := \{(x', y') \in O(x, y) : D(x', y') \neq 0\}$.

2. Complete the function `r = calcErr(I1, D1, I2, xi, K)` that takes the images and their (assumed) relative pose, and calculates the per-pixel residual $\mathbf{r}(\xi)$ as defined in the slides. \mathbf{r} should be a $n \times 1$ vector, with $n = w \times h$. Visualize the residual as image for $\xi = \mathbf{0}$.
Hint: work on a coarse version of the image (e.g. 160×120) to make it run faster.

3. Implement a function `[J, r] = deriveNumeric(I1, D1, I2, xi, K)` that **numerically** derives $\mathbf{r}(\xi)$ on the manifold, i.e., for each pixel i computes

$$\frac{\partial r_i(\xi)}{\partial \xi} = \left(\frac{r_i((\epsilon \mathbf{e}_1) \circ \xi) - r_i(\xi)}{\epsilon}, \dots, \frac{r_i((\epsilon \mathbf{e}_6) \circ \xi) - r_i(\xi)}{\epsilon} \right) \quad (3)$$

where ϵ is a small value (for Matlab $\epsilon = 10^{-6}$), and \mathbf{e}_j is the j 'th unit vector. \mathbf{J} should be a $n \times 6$ matrix. Additionally, the per-pixel residuals $\mathbf{r}(\xi)$ are returned as \mathbf{r} .

4. Implement Gauss Newton minimization for the photometric error $E(\xi) = \|\mathbf{r}(\xi)\|_2^2$ as derived in the slides. Use only one pyramid level (160×120) in the beginning, and then add the others.
5. Implement a function `J = deriveAnalytic(I1, D1, I2, xi, K)` which **analytically** derives $\mathbf{r}(\xi)$ (see slides). Using it instead of the numeric derivatives in the minimization from the previous task should result in a significant speed-up.