Lab Course: Robot Vision WS 2013 / 2014

Philipp Heise, Brian Jensen

Assignment 4 - Due: 18.12.2013

In this assignment sheet, you are again working with stereo data and this time you will generate dense disparity maps. On the previous sheet we used feature-matches between the stereo images to triangulate the depth. Here we will try to triangulate the depth for every pixel using algorithms belonging to the class of local stereo methods. These dense stereo algorithms are referred to as local methods because each pixel in the source image picks the best match in the destination image without consideration of its neighbours' matches. Rectified stereo images simplify the matching problem since the corresponding point in the other image must be on the same horizontal scanline. Further the range of the disparity values d can be assumed to be in a certain range $[0, d_{\max}]$. In the following I_l and I_r refer to the left and right images respectively.

Exercise 1 Dense Stereo Matching

The basic idea for stereo matching is that the optimal disparity value d^* for a pixel $\mathbf{p} = [x, y]^\top$ is given by

$$d^*(\mathbf{p}) = \underset{d \in [0, d_{\max}]}{\arg\min}\, \Phi(I_l, I_r, \mathbf{p}, d), \tag{1}$$

where Φ is a function measuring the similarity between the pixel at position $\mathbf{p}$ in the left image and $\mathbf{p} - [d, 0]^\top$ in the right image using the intensity information of I_l and I_r. Two possible functions for measuring the similarity are

$$\begin{aligned} \Phi_{\text{SD}}(I_l, I_r, \mathbf{p}, d) &= (I_l(\mathbf{p}) - I_r(\mathbf{p} - [d, 0]^\top))^2 \quad \text{and} \\ \Phi_{\text{AD}}(I_l, I_r, \mathbf{p}, d) &= |I_l(\mathbf{p}) - I_r(\mathbf{p} - [d, 0]^\top)|. \end{aligned} \tag{2}$$

Both functions should return very high values when the tested position is out of the image bounds. Comparing individual pixel intensities is not very robust, therefore it makes sense to additionally compare small patches with side length $2 * r + 1$ centred on the pixel and sum of the total difference. In the literature this is often referred to as aggregating support. The Φ function for patch comparison are given by

$$\begin{aligned} \Phi_{\text{SSD}}(I_l, I_r, \mathbf{p}, d) &= \sum_{x=-r}^{r} \sum_{y=-r}^{r} (I_l(\mathbf{p} + [x, y]^\top) - I_r(\mathbf{p} - [d, 0]^\top + [x, y]^\top))^2 \quad \text{and} \\ \Phi_{\text{SAD}}(I_l, I_r, \mathbf{p}, d) &= \sum_{x=-r}^{r} \sum_{y=-r}^{r} |I_l(\mathbf{p} + [x, y]^\top) - I_r(\mathbf{p} - [d, 0]^\top + [x, y]^\top)|. \end{aligned} \tag{3}$$

1. *(Cost-Volume)* The fist step for dense stereo matching is to implement a cost-volume data structure CV that holds all the evaluations for the Φ function.

$$\mathrm{CV}(\mathbf{p}, d) = \Phi(I_l, I_r, \mathbf{p}, d). \tag{4}$$

   Create a class StereoCV that can store a cost-volume. It makes sense to store the evaluations in an array holding $d_{\max}$ `cv::Mat*` elements. Add the methods:

   - `createSD( const cv::Mat& left, const cv::Mat& right, int dmax)`
   - `createAD( const cv::Mat& left, const cv::Mat& right, int dmax)`

   These methods should allocate the cost-volume and evaluate Φ_{SD} and Φ_{AD} for all pixels and disparity values $d \in [0, d_{\max}]$.

2. *(Winner Takes All)*
   In equation 1 the disparity value with the smallest Φ is assumed to be the correct the disparity value. This strategy is called the ***w**inner **t**akes **a**ll*. Create a method `WTA( cv::Mat& disparityimage )` that implements this strategy and returns the $d*$ parameter with the smallest value in the cost-volume ($d* = \arg\min_d CV(\mathbf{p}, d)$) for every pixel $\mathbf{p}$ in the image.

3. *(Patches and Cost-Volume Filtering)*
   As already mentioned the pixel-wise comparsion is not roboust and patches should be used. To simplify the implementation we can now just apply `cv::boxFilter` or `cv::GaussianBlur` on every `cv::Mat` for the whole disparity range. Why is this equivalent to patch comparison?

4. *(ROS-Node)*
   Make a new ROS node that takes stereo images and outputs a disparity map. Make the Φ function and the box- and gaussian-smoothing and their size accessible via dynamic reconfigure.

5. *(ROS/PCL Point Cloud)*
   The node should now use the disparity map and the left image to publish a colored point cloud.

Exercise 2 Dense Stereo Refinement (Optional)

1. *(Left/Right Consistency Checking)*
   Instead of matching from the left to the right image we could also evaluate everything from the right to the left image. We should get the same matches but in reality e.g. due to occlusions etc. this is not the case. To keep only the consistent matches we check for consistency. Given both disparity maps D_l and D_r the result of pixel $\mathbf{p}$ should only differ by a small amount $D_r(\mathbf{p} - [D_l(\mathbf{p}), 0]^\top) - D_l(\mathbf{p}) <$ threshold.
   Instead of implementing right to left matching additionally we use a trick and flip the images along the y axis using `cv::flip` and swap them to calculate the flipped disparity map D_r. To compensate for the flipping we can use the function $flip(x) = w - x$ given the width w of the image. LR-Checking should also be accessible using dynamic reconfigure.

2. *(Subpixel Disparity Estimation)* The point cloud of the the previous task exhibit strong layering artefacts. This is a direct result of the discrete evaluation of the possible disparity values. Instead of evaluating Φ at even more positions we use one Newton step to get subpixel disparity results. For a minimum the derivative should be zero. We perform one Newton step to improve our current estimate for the minimum disparity value. Given the current estimate for minimum value $d_c = \arg\min_d CV(\mathbf{p}, d)$ at the position $\mathbf{p}$, we would like to improve d_c such that $\frac{\partial CV(\mathbf{p}, d)}{\partial d} = 0$. This results in the following Newton step to

get the subpixel precise disparity value $d*$

$$\begin{aligned} d^* &= d_c - \frac{\frac{\partial CV(\mathbf{p},d)}{\partial d}}{\frac{\partial^2 CV(\mathbf{p},d)}{\partial d^2}} \\ &= d_c - \frac{\nabla CV(\mathbf{p},d)}{\nabla^2 CV(\mathbf{p},d)}. \end{aligned}$$

For the first and second order derivative we can use their discrete approximations

$$\begin{aligned} \nabla CV(\mathbf{p},d) &= CV(\mathbf{p},d+1) - CV(\mathbf{p},d-1) \\ \nabla^2 CV(\mathbf{p},d) &= CV(\mathbf{p},d+1) - 2 \cdot CV(\mathbf{p},d) + CV(\mathbf{p},d-1). \end{aligned}$$