Assignment 4 – Computer Vision 1 – WS18/19 17/12/2018

Prof. Stefan Roth
Junhwa Hur
Anne Wannenwetsch

This assignment is due on January 21st, 2019 at 13:00.

Please refer to the previous assignments for general instructions and follow the handin process described there.

Common function module

For this assignment please make use of the functions given in `Common.jl`, which is imported at the top of each Julia problem file.

Problem 1 - Fundamental Matrix (10 points)

In this problem, you will use the 8-point algorithm to compute the fundamental matrix between two images `a4p1a.png` and `a4p1b.png` which are shown in the following Figure:


Tasks:

The main function `problem1` already loads the images and point correspondences. Write a function `eightpoint`, which takes the 8 correspondences in homogeneous coordinates as arguments, and outputs the fundamental matrix. The function performs the following steps:

- Condition the image coordinates numerically, by moving their mean to $(0, 0)^T$ and scaling them such that coordinates are included in the interval between 1 and -1. Here, you can simply re-use the function `condition` that you already implemented for Assignment 3. Please note that this function will *not* be graded!
- Using the conditioned image points, compute the actual fundamental matrix in `computefundamental` by first building the homogeneous linear equation system for the elements of the fundamental matrix, and then solving it using singular value decomposition. Use `full=true` when calling `svd`.

- The preliminary fundamental matrix is not yet exactly of rank 2, due to noise and numerical errors. Enforce the rank constraint using SVD in `enforcerank2`. This function should be called in `computefundamental`.
- So far you still have the fundamental matrix for the conditioned image points. Transform it to obtain the one for the original pixel coordinates. This is done by matrix multiplication in the function `eightpoint`.

We now want to check the correctness of the computed fundamental matrix. First, draw the epipolar lines by implementing the function `showepipolar`. Second, verify that the epipolar constraints are satisfied for all pairs of corresponding points by computing the remaining residual in `computeresidual`.

Discussion

The conditioning of the image coordinates is an important step of your implementation. What happens if you do not condition the points? What makes the scaling necessary? Please answer both questions in `answersproblem1.txt`.

Submission: Please include `problem1.jl` and a file `answersproblem1.txt` in your submission.

Problem 2 - Stereo Matching (20 points)

In this problem, we perform stereo matching by estimating a disparity map between two front-parallel images. The two given images are already rectified, so we can estimate the disparity along horizontal scan-lines. We will try two cost functions, SSD (Sum of Squared Differences) and NC (Normalized Correlation), to measure the differences between the two window patches.

(a) Left image

(b) Right image

(c) Disparity map (Ground Truth)

Figure 1: Estimating the disparity map between the two front-parallel images

$$\begin{aligned} \mathrm{SSD}(x,y,d) &= \sum_{(x',y')\in w_L(x,y)} (I_L(x',y') - I_R(x'-d,y'))^2 \\ \mathrm{NC}(x,y,d) &= \frac{(\mathbf{w}_L(x,y) - \bar{\mathbf{w}}_L(x,y))^T(\mathbf{w}_R(x-d,y) - \bar{\mathbf{w}}_R(x-d,y))}{|\mathbf{w}_L(x,y) - \bar{\mathbf{w}}_L(x,y)||\mathbf{w}_R(x-d,y) - \bar{\mathbf{w}}_R(x-d,y)|}, \end{aligned} \tag{1}$$

where w_L is a image patch from the left image with the size of $m \times m$, and $\mathbf{w}_L$ the reshaped vector of w_L with the size of $m^2 \times 1$. See lecture slides for further details.

Disparity estimation based on window-based matching

The outline is given in `problem2`. Complete the following tasks:

1. The provided ground truth of the disparity map is sparse as the ground truth is captured by a laser scanner. Thus, the disparity map contains lots of invalid pixels which should not be taken into account when evaluating the estimated disparity map. Implement the function `loadGTdisaprity` that loads the ground truth disparity map as well as the validity mask. The validity mask is a binary array with the same size as the ground truth disparity map. In the validity mask, 1 denotes a valid disparity value and 0 an invalid one. After loading the ground truth disparity map, scale the values by a factor of 256.

2. Implement the function `computeDisparity` that estimates the disparity map between the two grayscale input images using a window size of 5×5. The maximum search range of the disparity is 100 pixels. For calculating the cost between two image patches, we will try two cost functions, SSD (Sum of Squared Differences) and NC (Normalized Correlation). Implement the two functions, `computeSSD` and `computeNC`, that estimate SSD and NC respectively. The output of the two functions is a scalar value.

3. Implement the function `calculateError` that calculates an average end-point error between the estimated disparity and the ground truth, where the end-point error is $EPE(d, d_{gt}) = \|d - d_{gt}\|_2$. Note that the average end-point error should be calculated only for valid pixels. Also output the error map that visualizes the end point error.

4. Depending on your implementation, the function `computeDisparity` takes more than 30 seconds to run. Using vectorized functions can effectively reduce the running time. Implement the function `computeDisparityEff` in your own preferred way and speed up your previous implementation in `computeDisparity` as much as you can. Use the SSD criteria for estimating the disparity. For the flexibility of programming, we only prepare the minimal setup of input and output parameters. You can implement and use your own custom functions inside of the function `computeDisparityEff`.

Discussion

In window-based matching methods, the results are sensitive to the applied window size.

1. Test various square window sizes ($w \times w$) and determine which window size yields the lowest average end-point error for each cost function (SSD and NC).

2. Have a close look at the estimated disparity and error maps and discuss pros and cons of using a bigger window size (*e.g.* 27×27) and smaller window size (*e.g.* 5×5) in a qualitative manner.

Summarize your answers in `answersproblem2.txt`

Submission: Please include only `problem2.jl` and `answersproblem2.txt` in your submission.