ECE661: Homework 10

Fall 2020
Due Date: Nov 16, 2020 (11:59 PM)

Turn in typed solutions via BrightSpace. Additional instructions can be found at BrightSpace.

1 Introduction

This homework consists of two parts:

1. Creating a 3D reconstruction from a pair of images recorded with an uncalibrated camera (such as your own cellphone). Such reconstructions are related to the actual scene structure by a 4x4 homography as you will see.

2. Given two rectified images, calculating the disparity map for the left image vis-a-vis the right image. Note that rectification means that the two images have been subject to appropriate homographies so that their epipolar lines correspond to the rows in the images and the corresponding rows in two images are also in epipolar correspondence.

2 Task 1: Projective Stereo Reconstruction

A 3D reconstruction is called projective if it is related by a 4x4 homography to the real scene. Obviously, this means that what we obtain from projective reconstruction might appear distorted when compared to the actual scene. In practice, depending on how rich the scene structure is and how much prior knowledge one has about the objects in the scene, one may be able to use additional constraints derived from the reconstruction to remove the projective, the affine, and the similarity distortions. In this task, however, our focus is on just creating a projective reconstruction of a scene.

A 2008 submission for this homework contains a very nice summary of what it takes to create a projective reconstruction of a scene from its two images recorded with an uncalibrated camera:

https://engineering.purdue.edu/kak/courses-i-teach/ECE661.08/solution/hw9_s1.pdf

2.1 Reading

1. **Image Rectification:** For image rectification using 8-point algorithm, refer to page 21-6 of Lecture 21 scroll or page 282 of the text [1]. For non-linear refinement of the fundamental matrix **F**, refer to pages 22-1 and 22-2 of Lecture 22.

2. **Projective Reconstruction:** Pages 22-2 and 22-3 on Lecture 22 and Section 10.3 and 12.2 of the text [1].

3. **Displaying Projective Distortion:** The previous years' solutions have some examples of how to display projective distortions with the help of 3D plots. You can also refer to Fig. 10.3 on page 267 of the text [1].

2.2 Programming Task

Take a pair of stereo images with your camera, with no particular constraints on how the second image is recorded vis-a-vis the first, as long as the two views are of the same scene. Using the two stereo images perform thefollowing tasks : -

2.2.1 Image Rectification

Manually extract a set of corresponding points (a minimum of 8) between the two images. Use these correspondences to estimate the fundamental matrix F using the 8-point algorithm. See page 21-6 of the Lecture 21 scroll or page 282 of the text for the 8-point algorithm for estimating F from these correspondences. (The comment on page 21-6 regarding you needing 40 correspondences does not apply to manually selected correspondences.) After the linear least-squares estimation of F, you must also refine F by nonlinear optimization as described on pages 22-1 and 22-2 of Lecture 22 scroll. And, yes, do not forget to enforce the rank constraint on the fundamental matrix. Rectify the images using the estimated fundamental matrix.

2.2.2 Interest Point Detection

Your next goal is to construct automatically a large set of correspondences between the two images. Toward that end:

- Use Canny edge detector to extract edge features in your scene. Obtain pixel coordinates from the Canny edge mask. Use these points as your interest points.

- If your image rectification is"perfect", given a pixel in the first image, all you'd need to do for finding the corresponding pixel in the second image is to look at the same row in the latter. Ordinarily, you'd look in the same row and in a small number of adjoining rows for the correspondences. If your rectification procedure is not working at all, directly use the epipolar constraint to establish correspondences between the extracted interest points between the two images.

- In most cases, for any given pixel in the first image, you will end up with multiple candidates in the second image. In such cases, use the SSD or the NCC metric to select the best candidate for each pair of correspondences.

- You also need to ensure that the ordering of the interest points on an epipolar line (which would be a row if your rectification procedure is working well) in the second image is the same as that of the corresponding points in the second image.

2.2.3 Projective Reconstruction

For triangulating from the correspondences, use the procedure described on pages 22-2 and 22-3 of Lecture 22 scroll or in Sections 10.3 and 12.2 of the text. In light of the fact that you'll be reconstructing your scene with uncalibrated camera, your reconstruction will assume a canonical configuration for the cameras.

2.2.4 3D Visual Inspection

- Make a 3D plot of the reconstructed points. Draw some 'pointer lines' between the corresponding pixels in the two images, on the one hand, and between those pixels and the reconstructed 3D points on the other.

- Remember to include a display of the projective distortion in a fashion similar to the one on page 267 of the text. You can also refer to the previous years' solutions too for that

3 Task 2 : Dense Stereo Matching

Given a stereo rectified pair, the dense stereo matching problem essentially involves finding as many accurate pixel correspondences as possible and

mark occluded regions invalid. For this task, you will implement a rudimentary dense stereo matching algorithm using Census Transform, you won't be doing the additional test for finding the "invalid" pixels. The goal of this task is not to outperform the state-of-the-art approaches. The focus is to gain some insights on the challenges that are involved in the dense stereo matching.

Before diving into the programming task, you need to understand the notions of the disparity map and Census Transform.

3.1 What is a Disparity Map?

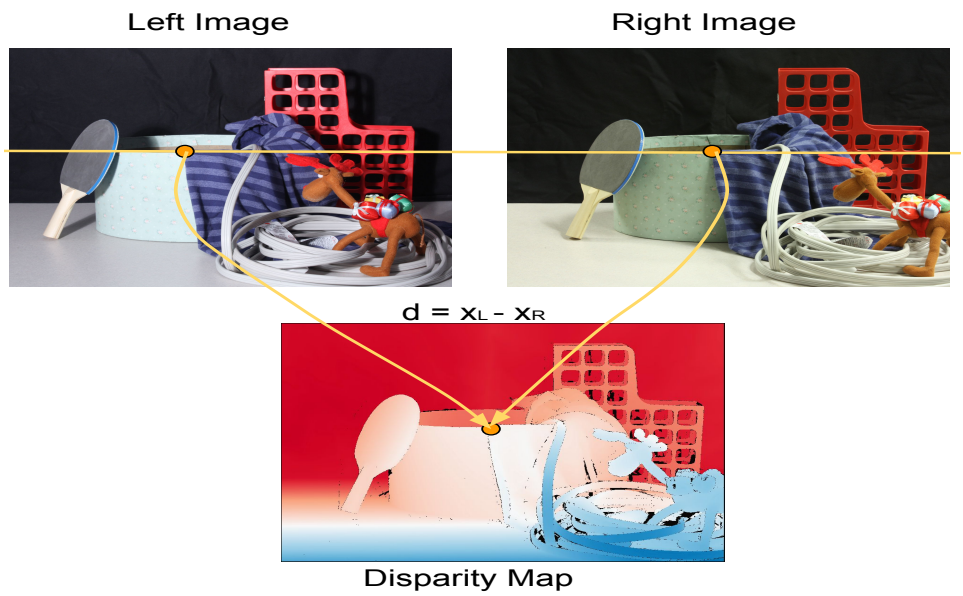
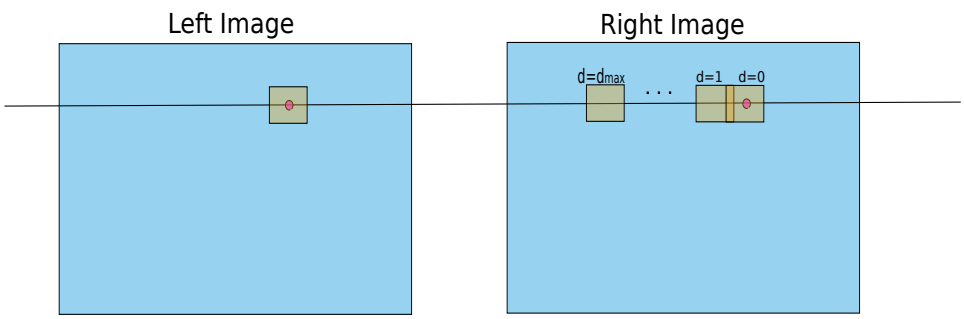


Figure 1: Example ground truth Disparity map, the disparity map stores per-pixel difference in x–coordinates. The black pixels in the disparity map are occluded and therefore invalid pixels. The stereo pair is a part of the Middlebury 2014 stereo dataset.

Dense stereo matching problem essentially boils down to finding per-pixel correspondences in the given stereo rectified pair. Concretely speaking to compute the disparity map for the left image vis-a-vis the right image, for every pixel in the left image we scan all the potential matching candidates along the same row in the right image. After establishing the correspondences, we simply store the difference in the x-coordinates as the output

disparity map. As one can guess, in untextured regions you would find many possible correspondences, such as the black background in the scene in Fig. 1. To overcome this ambiguity, there are more sophisticated approaches that aggregate global or semi-global information from a pair. However, for this homework you will only use the local context around each pixel to find the correspondences in the right image, as explained in the next section 3.2. The example in Fig. 1 is from Middlebury 2014 high resolution stereo dataset. For this homework you will use a low-resolution stereo pair from Middlebury 2001 dataset.

3.2 Census Transform



(a) Scanning a stereo pair for dense stereo matching

50	100	51
60	70	80
85	75	125

40	30	50
90	72	80
85	70	125

0 1 0 0 0 1 1 1 1	XOR	0 0 0 1 0 1 1 0 1
= 0 1 0 1 0 0 0 1 0 = 3 (bit count)		

(b) Census Transform. Note that the red cells are the center pixels and the numbers in each cell represent pixel intensity value. The corresponding M^2 bitvector is computed by setting a bit one wherever the pixel value is strictly greater than the center pixel value.

Figure 2: Dense stereo matching using Census Transform

Fig. 3b shows the summary of computing the Census transform. On the left hand side, you have a local pixel intensity distribution within the $M \times M$

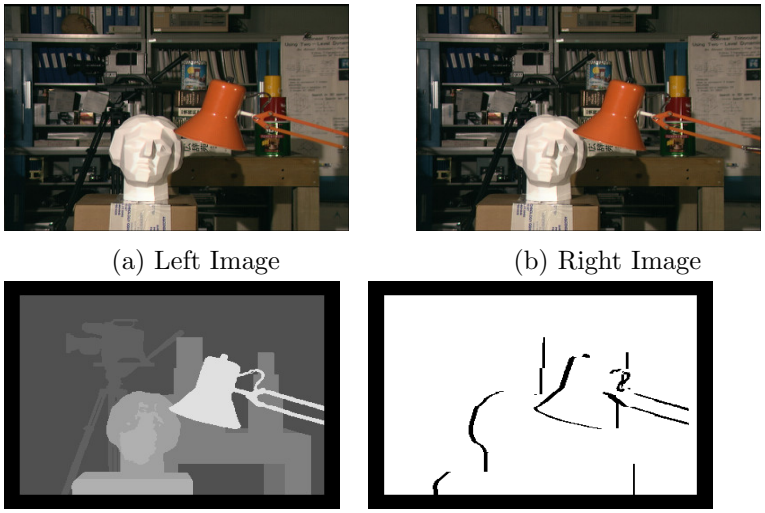
window in the left image centered at pixel $\mathbf{p}$ and on the right you have the corresponding local view of a pixel $\mathbf{q} = [\mathbf{p}_x - d, \mathbf{p}_y]$ at disparity d. Note that the pixel position $\mathbf{q}$ is defined in the right image. We compute a bitvector of size M^2 , wherever the pixel intensity value is strictly grater than the center pixel value we make that bit one. This gives us two M^2 bitvectors at pixel $\mathbf{p}$ in the left image and at pixel $\mathbf{q}$ in the right image. After the bitwise XOR operation between the two bitvectors, we simply compute the number of ones in the output bitvector. This gives us the data cost between the two pixels. Note that $d \in \{0, \cdots, d_{max}\}$. We pick the disparity value d such the data cost is minimized. In the case of multiple minima, pick the first disparity value that results in the minimum data cost. The assumption here is that we iterate from 0 to d_{max}.

Compute the disparity maps for at least two different values of M. Note that it doesn't have to be a square window, e.g., 9×7 is also a valid size for the census window.

3.3 Programming Task

You're provided with a stereo rectified image pair and the corresponding ground truth disparity map for evaluating your stereo matching accuracy. Fig. 3 shows the given stereo pair for Task 2. Note that all the images are of the same size. You can read the input images using OpenCV's imread function.

- Estimate disparity maps using the Census transform for at least two different values of M and evaluate the accuracy as explained in the next point.
- Let $\mathbf{D}_{gt}$ be the given ground truth disparity map and $\mathbf{D}$ be the estimated disparity map. Let N be the total number of valid pixels, i.e., wherever you see the white pixels in the ground truth non-occlusion mask in Fig. 3d. Report the accuracy as the percentage of N in the $\mathbf{D}_{gt}$ such that $|\mathbf{D}_{\mathbf{p}} - \mathbf{D}_{gt\mathbf{p}}| \leq \delta$, for $\delta \in \{1, 2\}$
- In order to highlight challenging regions in the given stereo pair, also show binary error masks with the value of 255 for pixels where the disparity error is $\leq \delta$ and 0 otherwise for different values of M.
- **Special note:** After reading the ground truth disparity map, convert it into Float32 and divide the disparity values by 16, then convert it back to int16 value. You can obtain the d_{max} value from the ground



(a) Left Image (b) Right Image

(c) Ground Truth Disparity (d) Ground truth non-occlusion mask

Figure 3: Input stereo pair for the task 2 and the ground truth disparity map and non-occlusion binary mask.

truth disparity map, after the aforementioned adjustment. We need this adjustment because the ground truth disparity map was generated for a higher resolution stereo pair. This is how it's available in the Middlebury 2001 dataset.

4 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks.

1. Turn in a zipped file, it should include (a) a typed self-contained pdf report with source code and results and (b) source code files (.py, .cpp,.c). (c) Your output disparity maps for Task2. Rename your .zip file as hw10_<First Name><Last Name>.zip and follow the same file naming convention for your pdf report too.

2. Your pdf must include the source code for the both tasks and a description of

   - Task1

  - A good description how you implemented each of the subtasks with relevant equations.
  - The manually selected points used to estimate the fundamental matrix as well as the matched correspondences between interest points.
  - The reconstructed results with some marker or guide lines to understand the scene structure.
  - Any improvements that you see by using the LM method.
- Task 2
  - Explain in your own words the dense stereo matching algorithm using Census transform.
  - Accuracy and error masks for $\delta \in \{1, 2\}$ and for different values of M
  - Your estimated disparity maps, in other words, specify a colormap with matplotlib's plotting function.
  - Your observations on the output quality for different values of M.
- Your source code. Make sure that your source code files are adequately commented and cleaned up.

References

[1] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.