Assignment 3: Stereo Correspondence Analysis

Cristian Mitroi, Kaela DeAngelis, Terne Thorn Jakobsen

January 8, 2018

1 Introduction

The problem important to the subject of stereo vision is being able to construct 3D visualizations based on 2D images of a given scene. A disparity map between two 2D images of the same scene can be viewed as an estimation of depth of that scene, and hence provide valuable information for a later 3D reconstruction. This is called disparity computation. Analyzing the two pictures resulting from two cameras pointing at the same scene "and exploiting the difference (or disparity) between them allows us to gain a strong sense of depth" [3]. It is this "depth" map that we are trying to generate in this assignment's algorithm.

This report is split into three parts:

1. A discussion of our methods and of the theory underlining the process
2. Results: images and numerical analysis
3. A concluding discussion

2 Methods

2.1 Disparity map

A disparity map is computed as an *estimation* of the depth of the different elements in the image. It is not true depth, since the disparities are only calculated horizontally, i.e. between `x` coordinates. The disparities provide an estimation of depth because disparity and depth are inversely related. The disparity itself is the shift in object placement when looking at the scene with the right eye alone, and then with the left eye alone (or from two cameras positioned along the same horizontal axis, but in slightly different positions). The inverse relationship between disparity and depth is given by the fact that objects far away in the scene shift less than objects that are nearby [4].

Disparity between two corresponding points x and x' in an image plane (where images lie on the same epipolar plane, as our test images do) is therefore the distance of this shift, and thus given by:

$$x - x' = \frac{Bf}{Z}$$

where B is the distance between two cameras, f the focal length of the camera, and Z gives the depth of the scene point in 3D [4].

2.2 Intensity matching

In this assignment, we base our matching procedure on intensity matching, where Intensity I is defined by:

$$I = \frac{(R + G + B)}{3}$$

For calculating this we used the `cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)` function [1]. This creates a single-channel image from the original three-channel one. `OpenCV` loads images as `BGR` images. In order to convert to grayscale, we pass the parameter `cv2.COLOR_BGR2GRAY`. We don't simply compare the intensity of one single pixel, but rather we use a window (of size `N`) around said pixel. This is just one part of the process we developed in this exercise.

2.3 Normalized cross-correlation

We use normalized cross-correlation to compute the similarity between pixels, taking into account the surrounding area of said pixel (window size `N`). Normalized cross-correlation is a method used in template (also called patch) matching [2], and we use this method to obtain best matches of intensity values between the two image, representing left and right view. We then use the difference in position on the horizontal axis as the disparity of the pixel between the two images.

Cross-correlation is similar to convolution but where the convolution acts as an addition between variables, cross-correlation acts as the difference between variables [7]. Discrete normalized cross-correlation, in 2D, is defined by:

$$\frac{1}{n} \sum_{(x,y)\in\Omega} \frac{(f(x,y) - \bar{f}) \cdot (g(x+\alpha, y+\beta) - \bar{g})}{\sigma_f \sigma_g}$$

where n is the number of pixels in Ω, and $\bar{f}$ and $\bar{g}$ are the mean values of functions f and g respectively. Normalized cross-correlation means suppressing the effect of possible outliers given by large variation between intensity values, which is done by dividing by the standard deviation (σf and σg is the standard deviation of the functions f and g, respectively). To perform normalized cross-correlation, we use the `OpenCV` function `cv2.matchTemplate()` [6]. Template matching, which is what the function performs, is a way of searching for identical signals in two positions, and only two, i.e. we do not allow several matches, making it a fast matching procedure [6].

2.4 Image pyramids

For searching efficiency, and for handling numerically large disparities, a coarse-to-fine approach is used; we down-sample the images before calculating disparity values, and then up-sample afterwards. Image pyramids of the left and right images are constructed with the OpenCV function `Cv2.pyrDown()` [5]. The `cv2` function constructs Gaussian pyramids where each pixel in the down-sampled images is constructed by convolving five pixels in the higher resolution image with Gaussian weights. The images are down-sampled by a factor of two, where each MxN image becomes an M/2xN/2 image, and we do this for each image (left and right) three times. This gives us four images in total, including the original.

The process is then to compute the disparity matrix at the bottom level of the pyramid, where the search space is significantly smaller. We then up-sample that disparity matrix and use it to calculate the new disparity matrix at the previous level of the pyramid. In this case, up-sampling yields a matrix that is doubled on the X and Y dimensions, and where every pixel is doubled in value and used to narrow the search space for computing the new disparity matrix ($\pm M$).

2.5 Other methods

The concept of two-way matching is similar to that of a concept utilized in assignment two, that being left-to-right matching and right-to-left matching. For example, the idea is that the algorithm searches for matching pixels of "right image" in "left image". However, we cannot ascertain at that point that the output candidate matches are true matches so we apply the algorithm in the reverse direction to search for matching pixels of "left image" in "right image". After producing results from both directions and checking for equivalent matches, the number of false matches are reduced and we can then accept the candidate matches that agree.

It is important to mention that false matches of stereo images can occur from occlusions or depth discontinuities. For example, the angle at which "right image" is taken may reveal an object previously hidden behind a foreground object in "left image". When comparing these two images the algorithm cannot match the pixels of the previously hidden object to any in that of "right image". These occluded areas and discontinuities need to be dealt with to produce a smoother image. Therefore two-way matching can help identify poor matches related to occlusions and depth discontinuities. In our approach, these false matches are tagged with -1 in this step, converted to black for the disparity map, and replaced with a local average. Therefore, we exclude areas that were previously identified as poor matches from all local mean calculations and we replace these pixels with the local average.

Even matches that are accepted by two-way matching may be inaccurate if they deviate significantly from the local mean disparity. In our implementation, these deviations are replaced with

the local average. From experimenting with different values for the threshold we have chosen the value of 25 as yielding decent results.

3 Results

In this section, the results of our algorithm for dense stereo correspondence analysis is presented. For each stereo dataset of the *Map, Tsukuba*, and *Venus* images, we present only:

1. The pyramid representation of the right image, which was created as described in section 2.4
2. The true disparity map
3. The disparity map created by our algorithm, with methods described above, and with a patch size of 11x11.

Not included in this report is disparity maps created with different patch sizes (of 5x5 and 7x7) as well as results from experimenting with threshold values, i.e. we only present the best results from our experimentation. For each set, a small subjective analysis of the generated disparity map follows, while the numerical comparison is to be found in section 3.4, and a discussion of results in section 4.

3.1 Map set

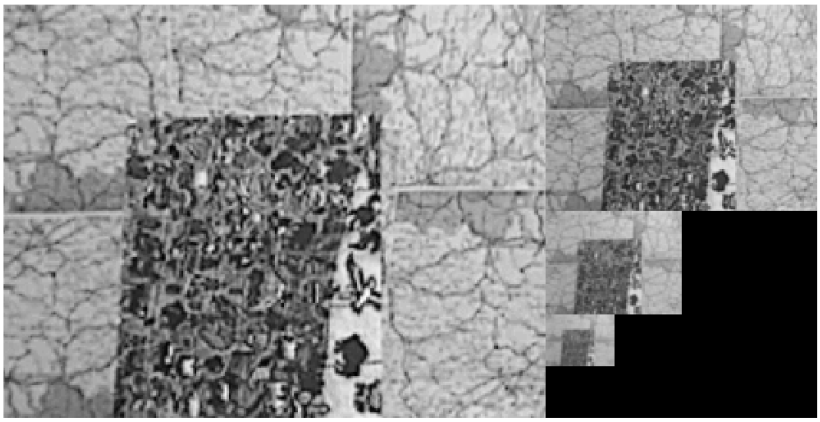
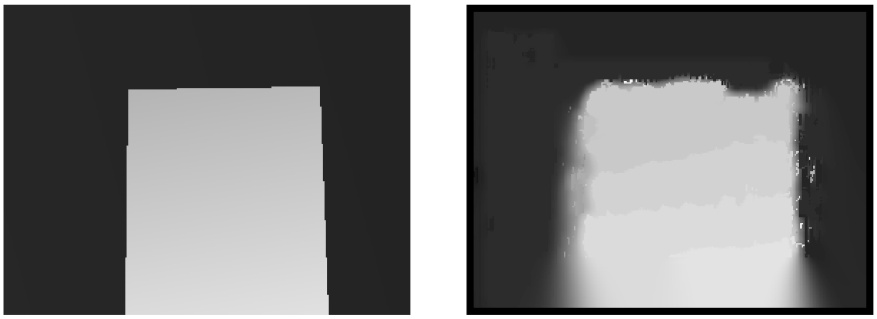


Figure 1: Pyramid representation of right *Map* image



(a) True disparity map (b) Our disparity map

Figure 2: Disparity map of *Map* image

The *Map* image is expected to be the simplest image to test because there is only one object in the foreground and the pattern is helpful for template matching. Therefore, the algorithm is expected to perform well with this one. Comparing the true disparity map (figure 2a) with the one generated by our algorithm (figure 2b) it is visible that the square shape of the map is captured through the visualization of disparities. However, the edges of the shape appear smudged and uneven compared to the goal output. The smudging effect results from the averaging of bad matches. This appears to happen mainly in occluded areas.

3.2 Tsukuba set

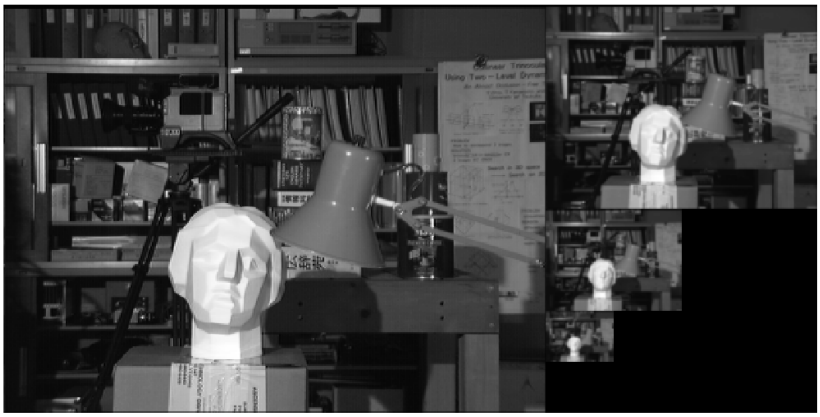
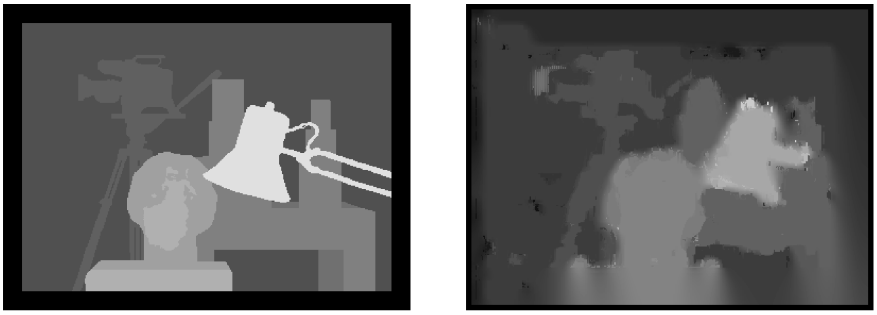


Figure 3: Pyramid representation of right *Tsukuba* image



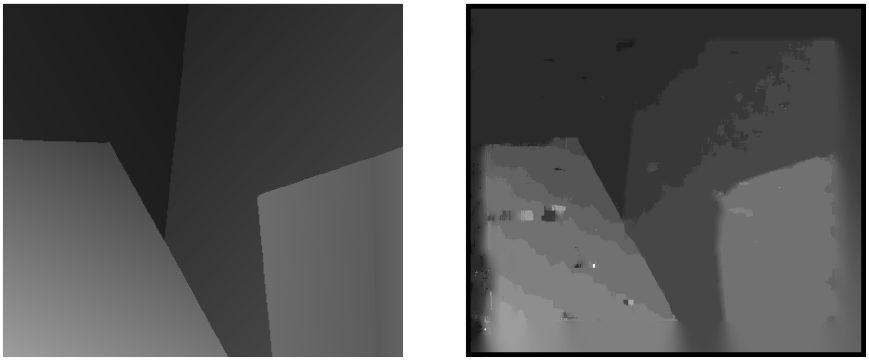
(a) True disparity map (b) Our disparity map

Figure 4: Disparity map of *Tsukuba* image

The *Tsukuba* image is a more complex image to test as it contains more layers of depth and objects within it. Based off our previous test with the *Map* image, we expect that the our disparity map of the *Tsukuba* image will produce an image in likeness with its true disparity map but with errors and fuzzy edges. As we saw in the *Map* image, the edges of the shapes are uneven and smudged, when comparing the true disparity map (figure 4a) with ours (figure 4b). Additionally, there is more variance within the background. Again, many of these errors may occur due to occlusion areas between stereo images.

3.3 Venus set

Figure 5: Pyramid representation of right *Venus* image



(a) True disparity map (b) Our disparity map

Figure 6: Disparity map of *Venus* image

Testing with the *Venus* image posed similar results seen in the previous *Map* image and *Tsukuba* image disparity maps but to a lesser extent. The edges of the foreground shapes are distinct in our disparity map (figure 6b) as they are in the true disparity map (figure 6a); they present a more choppy change in grayscale intensity compared to the more solid color of the shapes in the true disparity map. Take for example, the leftmost foreground object in Figure 6b. This object displays abrupt changes in grayscale color with the lightest color being at the bottom and the darkest color at the top corner of the object. This change does occur within the true disparity map (figure 6a) but gradually and subtly. This block-like ombré effect in our disparity map may occur because of how the object is angled away at the top from the camera. The algorithm detects the change in depth as evident by its change in color but the effect may occur because the distance between matches in pixels is increasing by integer values and we do not have a high enough resolution to display a more accurate gradient.

3.4 Numerical Comparison

In order to verify our results, we compared our final disparity maps (of patch size 11 and after doing two-way matching) with the *true* disparity maps provided in the datasets. We computed the ab-

solute difference (`abs(final_disp - true_disp)`) between the two matrices and then computed the following statistics, summarized in Table 1:

1. mean difference
2. standard deviation
3. number of pixels where the difference was greater or equal to 3
4. percentage of pixels where the difference was greater or equal to 3

	Tsukuba	Map	Venus
Mean diff.	135.96	38.80	36.82
Std. dev.	104.64	75.37	72.74
>= 3	684916	614892	597114
>= 3 (%)	74.41	67.75	74.95

Table 1: Comparison with ground truth

4 Discussion

We can see from the numbers and the comparison figures that our solution worked pretty well for the *Map* and *Venus* datasets. In fact, both the mean difference and the standard deviation of the difference are surprisingly close to each other. For the *Map* dataset we can see that we have slightly less pixels where the difference was large (greater than or equal to 3). The images themselves are jagged and uneven-shaped in comparison with the benchmark ones, but we can definitely tell the outlines.

The `Tsukuba` dataset posed a challenge for our implementation. We think this is due to the images in this set contain more elements of more intricate shapes - it is more *complex*.

We believe the results could have been improved if we had employed some further steps in our algorithm:

1. blurring the images in the pyramid (Gaussian blur) would have reduced some of the detail. This could have provided us with smoother images.
2. ignoring the pixels where the ratio between the best match and the second best was too high. We could have then replaced these pixels with the local mean (like we did with the two-way matching). This is application we used in assignment 2.

All in all, we consider the assignment a success. We have managed to implement a basic algorithm for computing disparity map images of two stereo images of the same scene. In this we have researched and implemented different techniques commonly used in the image processing field. While the results were not up to par with the benchmarks provided, we think our solutions provided decent outcomes.

References

[1] Opencv: changing colorspaces, 2018. https://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html.

[2] W. contributors. Cross-correlation — wikipedia, the free encyclopedia, 2017. https://en.wikipedia.org/w/index.php?title=Cross-correlation&oldid=817992773.

[3] D. A. Forsyth and J. Ponce. *Computer Vision*. Pearson Education Limited, 2011.

[4] S. Ghosh. Generating dense disparity maps using orb descriptors, 2018. http://sourishghosh.com/2016/dense-disparity-maps-orb-descriptors/.

[5] OpenCV. Image pyramids, 2018. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_pyramids/py_pyramids.html.

[6] OpenCV. Template matching tutorial, 2018. https://docs.opencv.org/trunk/d4/dc6/tutorial_py_template_matching.html.

[7] stackexchange. What's the difference between convolution and crosscorrelation?, 2018. https://math.stackexchange.com/questions/353272/whats-the-difference-between-convolution-and-crosscorrelation.