# Project 1 - Stereo Vision

Christian Miranda
christianmoryah@gmail.com

Departamento de Ciência da Computação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil

**Abstract**

Stereo vision is the computer vision branch that allows the discovery of dimensions, shapes and positions of objects based on a pair of images.

In this project, some stereo vision techniques are explored for: 1. depth maps extraction using camera calibration parameters, both in parallel and convergent cameras; 2. object measure estimation in the 3D world.

A fast, quality depth map may have several applications, like object avoidance systems for self driving cars, autonomous drones, or other kind of moving robots.

## 1 Introduction

In stereo vision, two cameras are aligned in the y axis and displaced horizontally from one another, in a way similar to human eyes. These cameras obtain two images, with different points of view of the scene.

By comparing these different images we can obtain a disparity map that encodes the difference in the x axis between corresponding image points [1]. The values in the map are inversely proportional to the depth of the scene, at the corresponding pixel location.

## 2 depth map estimation from rectified stereo images

In this section we use the 2014 Middlebury's stereo image dataset [6]. According to Hartley and Zisserman [1], image depth can be estimated from disparity maps. To obtain a disparity map, we need to process both left and right image from a stereo camera set, and compare the translation of points from one view to another. Points with bigger translation (disparity) will be closer to the cameras.

### 2.1 Methodology

For disparity and depth estimation, we can do the following steps:

- Point matching: For each point in the left image, find the same point on the right image;

- Using the matched points, generate the disparity map, in which each point (pixel) stores the distance value, in pixels, from the left image to the right one, along the epipolar line;
- Generate the depth map, using disparity and camera parameters. In the depth map each pixel stores the Z coordinate value, in millimeters, from the observed points;
- Finally, compare results with ground truth from Middlebury's 2014 dataset, to check the method performance.

When working with stereo data, points will keep the same y in both images. This allows the simplification of the problem and reduces the point search space to a 1D vector. Because of that, the first point matching experiment is made with the naive approach: 1D vector convolutions along the epipolar line, with a 1D window to find the least squared differences for point matching. This is not a robust method. Multiple pixels in a line can have the same pixel intensity.

After that, a well known method for block matching is tested: Hirschmuller's Semi-Global Matching and Mutual Information [2]. Unlike the previous method, this one searches neighboring pixels as well and is known for having good performance overall. The SGBM algorithm is already available in OpenCV.

Lastly, OpenCV's Weighted Least Square (WLS) Filter [4] is used to further improve SGBM results.

Image Depth can finally be obtained with Z = baseline * f / (d + doffs). Where baseline is the distance between the two lenses, f is the focal length, d is the disparity value per pixel and doffs is the x-difference of principal points of the cameras.

## 2.2 Results

Figure 1: Original left image

Figure 2: line based disparity map

Results were better than nothing for the first method, but the disparity map got streaking effects because of ambiguity in the line match. Original image resolution wasn't reduced to avoid loss of information, so it took hours to process the entire image without the proper 'convolution tricks'.

StereoGBM, in turn, gave visible results. It became clear why 2d area matching solves ambiguity problems. But results were still noisy and had several artifacts (Figure 3).

To address these new problems, a WLS filter was tested. WLS uses left-right consistency check to improve the results of uniform areas or areas with occlusion. It also applies smoothing to the disparity map. After introducing the post filtering, the result got considerably better (Figure 4).

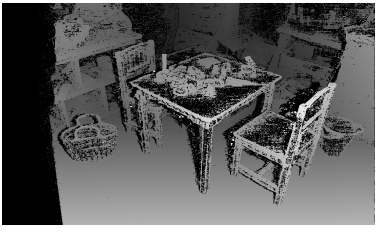

Figure 3: SGBM based disparity map

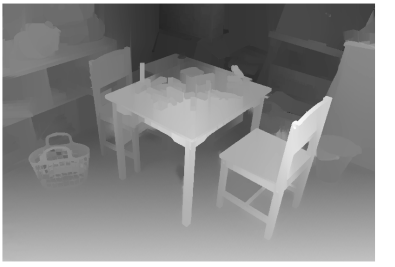

Figure 4: SGBM + post filtering

Since the disparity is inversely proportional to the distance Z for each pixel, it gets converted into a depth map of the scene with Z = baseline * f / (d + doffs).

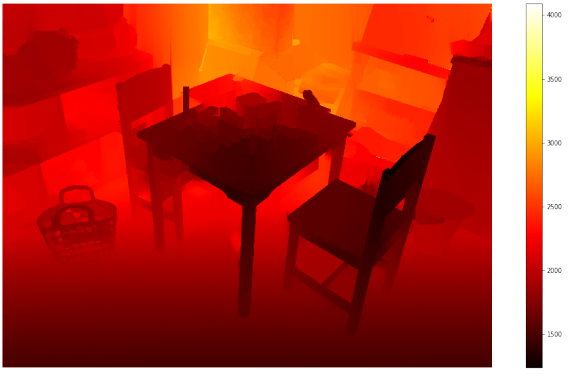


Figure 5: Image Depth with HOT color map, distance is expressed in millimeters

The disparity map is then compared to ground truth result from MiddleBury's dataset, where Bad 2.0 metric is calculated for benchmarking reasons.

- Playtable bad 2.0: **0.45**.
- Jadeplant bad 2.0: **0.61**.

At the end, a new pair of images were generated for this experiment. They can be found at the 'data/Middlebury/Chris-undistorted' directory and will run automatically when the code is executed. The setup was made with a single webcam, sliding across the computer screen top with a baseline of 10cm. Calibration parameters follow the Middlebury's dataset standard, but not all camera parameters were saved on the calib.txt file. Results are available in Figures 7 and 8.

# 3 depth map estimation from converged stereo cameras

For this task a pair of Morpheus images from the Furukawa and Ponce dataset is used. This time, cameras converge to the object, the projection plans are not parallel and yL is different from yR.

Figure 6: Left image taken from webcam

Figure 7: Disparity map


Figure 8: Depth map

Figure 9: Furukawa and Ponce Morpheus images 0 and 1

## 3.1 Methodology

There are several techniques for depth estimation from converged stereo cameras. This is approach intends to rectify the images, making the epipolar lines parallel, and then apply the same techniques used in last section.

The process as a whole can be summarized in these steps:

- For each point in the left image, find the same point on the right image (OpenCV's

SIFT + FLANN);

- Find the homography of the images from the point matches and use it to project them to a plane parallel to the baseline (rectification);
- After rectified, repeat the steps from Section 2.

For rectification, camera parameters were not used to calculate the homography between the two images planes. Instead, Scale Invariant Feature Transform (SIFT) [3] is used along with Fast Library for Approximate Nearest Neighbor (FLANN) [5] for point matching.

SIFT's patent expired on 2020, so its functionalities are now part of OpenCV. It finds potential keypoints using Scale Space Extrema Detection. With this framework, features become scale invariant, so we are able to 'detect blobs' from several sizes and several distances in the image, these blobs become our keypoints.

Orientation histograms of blocks surrounding the keypoints are then calculated to achieve invariance to image rotation, this are the keypoints descriptors. Keypoints descriptors of both images are then fed to FLANN, that matches them identifying their nearest neighbours. In some cases, the second closest-match may be very near to the first, so the ratio between these matches distances is compared and, in case its bigger than 0.8, they get discarded [3].

The point matches are then used to calculate the homography between the two image planes, and finally, the homography is used to rectify the images.

After this, the process from section 2 is repeated, using SGBM + WLS to obtain scene depth map.

## 3.2 Results

One of the images had to be resized before computing the homography, since they had different resolutions. After applying the homography, stereo rectified images are available in Figure 10.

Figure 10: Morpheus images 0 and 1. Rectified + point matches

From here we repeat the steps from section 2 to obtain the image depth map, as shown in Figure 11.

Even though one of the images was resized, we didn't need to update the camera parameters, because we didn't use them. Homography was obtained directly from point matching.

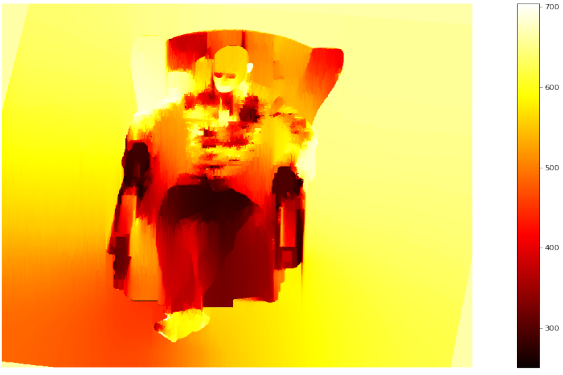


Figure 11: Image Depth with HOT color map, distance is expressed in millimeters

# 4 measuring 3D objects

There are several possible possible ways for measuring 3D objects. We could take the depth map and project it to 3D. 3D points will already be available to make distance calculations.

Since this was practically solved, another approach was chosen: using the cameras projection matrices to triangulate selected points (from both images) 3D position. In this way, we can enable interactions, allowing measuring objects with clicks.

## 4.1 Methodology

- Select a pair of points, both in the left image and the right one.
- Triangulate the points to 3D, using both cameras projection matrices
- Convert the points to euclidean coordinates and find euclidean distance between them

## 4.2 Results

Since we weren't able to find the real object size on the data set, we had to search the toy measures on the internet. "Morpheus in chair series 2 action figure", from Mcfarlane toys, has a height of 6 inches (around 15cm) , according the announcements on internet marketplaces.

Smallest parallelepiped was also difficult to measure, since we lost some points that were key as vertices of the cube in both images (e.g. farthest lower cube vertices on the left image which are aligned with chair back legs, farthest left lower cube vertex on right image)

# 5 discussion and conclusions

At section 2, the single line match based disparity map gave really bad results, as expected.

Moving to object matching by 2d blocks, results were clearly better. The size of the block was directly related to disparity 'precision'. The bigger the window, bigger the search areas, so a smaller window, in general, gave a more 'smooth' result, with better contours.


Figure 12: Morpheus images 0 and 1 with 3D measures

Uniform regions without clear patterns, like smooth single colored walls got no match and ended up with a disparity value of 0. These points ended up becoming infinite values when converted to the depth map.

When the depth map is converted to gray scale for visualization, Opencv takes the max and min value for scaling. The infinite values makes all the other ones too small, and the resulting depth map becomes a dark image. There are probably several solutions to this issue, adding 1 to each value of the disparity map seemed to be the simplest way to deal with it. Later, WLS filter took care of this and several other issues in the generated disparity map.

Bad 2.0 results were reasonable, but can be further improved with parameter tuning for each image, or with the use of more recent techniques.

At Section 3 the biggest challenge was: epipolar lines did not become parallel even after rectification. Even though this happened, point matches lines became mainly horizontal (Figure 10). Depth map got really noisy, probably because of that. Another rectification technique could be used to obtain better results (for instance, using the cameras parameters to rectify the images).

For Section 4, we got somewhere, but couldn't get very close to the correct height of the toy. The image pair had different sizes, so maybe that, along with the projection matrices, had something to do with the results imprecision.

# References

[1] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[2] Heiko Hirschmuller. Stereo processing by semi-global matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. https://core.ac.uk/download/pdf/11134866.pdf.

[3] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[4] "Dongbo Min, Sunghwan Choi, Jiangbo Lu, Bumsub Ham, Kwanghoon Sohn, and Minh N. Do". Fast global image smoothing based on weighted least squares. *IEEE*

*TRANSACTION ON IMAGE PROCESSING*, 2014. http://publish.illinois.edu/visual-modeling-and-analytics/files/2014/10/FGS-TIP.pdf.

[5] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[6] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In Xiaoyi Jiang, Joachim Hornegger, and Reinhard Koch, editors, *Pattern Recognition*, pages 31–42, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11752-2. https://www.cs.middlebury.edu/~schar/papers/datasets-gcpr2014.pdf.