5 Laurie Marceau
Stereo camera and depth map

This section covers the stereo vision. This task explains the calibration of the camera, with the intrinsic and extrinsic parameters, as well as the disparity map and the top view of the depth map. To cover the whole process, issues encountered are discussed, as well as their proposed solution.

5.1 Camera calibration

Reconstructing the 3D geometry of a scene requires at least two images taken by two separate cameras at the same time. For the 3D reconstruction to be accurate, two types of calibration need to be done, the intrinsic and extrinsic calibration of the stereo camera. This function is implemented in the Cameracalibrator.cpp file.

The intrinsic calibration is the inner parameters of one camera, which includes distortion, focal length and optical center. The method to get those parameters is by using a known structure that has many identifiable points, like a chessboard. It is possible to compute the relative location and orientation of the camera, at each image, by viewing the chessboard from different angles, as shown on the next figure. When using the calibration function, it is important to verify if the size of the board is accurate (default is 9,6), and the square size is adequate (default is 2.5 cm).

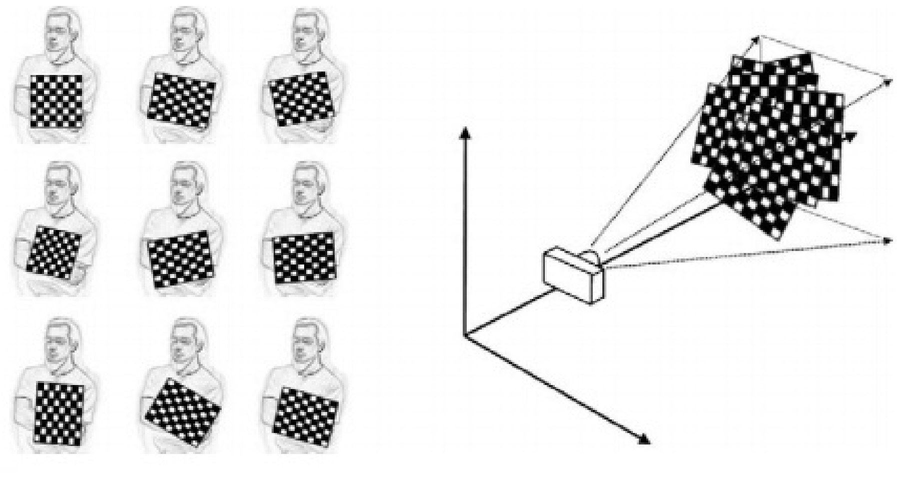

Figure 5: Intrinsic calibration

Once the intrinsic parameters are known, the extrinsic parameters must be determined. Using two images taken from the two cameras at the same time, the rotation and position of the cameras are calculated. The points in the two different images are matched. That gives us a rectified image in which both projected image planes are reconstructed to lie on the same plane. In short, with this transformation, both cameras are treated as if they were in the same plane and vertically aligned.

When the images are rectified, the geometry of the system is simplified, which makes the calculations required to get the 3D information simpler. The resulting geometry is called epipolar standard geometry, as shown on the next figure.

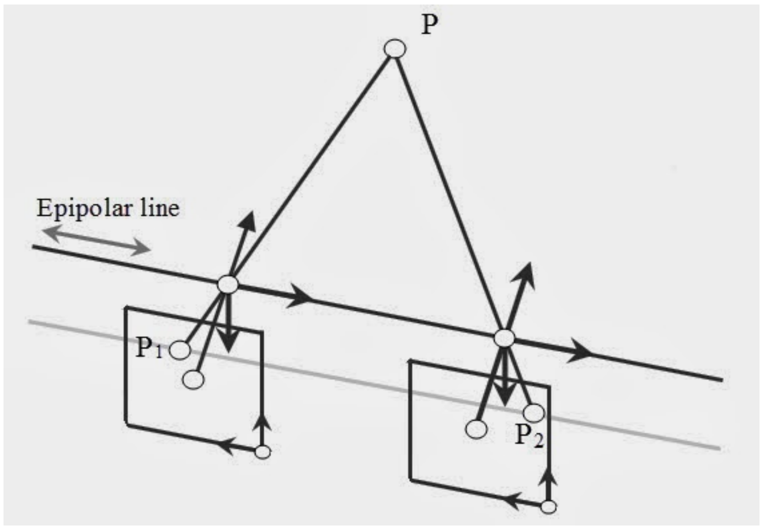

Figure 6: Extrinsic calibration

In this program, the intrinsic and extrinsic calibrations are done at the same time, with the same images. Those images are saved in the "calibration" folder, with the intrinsic and extrinsic files. There is no need to do the calibration again, unless a new camera is used. If you do the calibration, the reprojection error can be used to determine if the calibration went as planned.

5.2 Disparity map

A disparity map is the difference in location of an object, in two images of the same scene, taken at the same time. Disparity map can be compared to the human vision, with our left and right eye. The brain uses the disparity to calculate depth information from two dimensional images. The term disparity was originally used to define the 2D vector of equivalent characteristics between the left and right eyes. Disparity is inversely proportional to depth, and can be used to do a mapping of three-dimensional positions. This function is implemented in the Depthsubstraction.cpp file.

Since the rectified images were calculated earlier, we can get depth information from them in an intuitive way. Below are an image and some simple mathematical formulas which prove that intuition is correct.

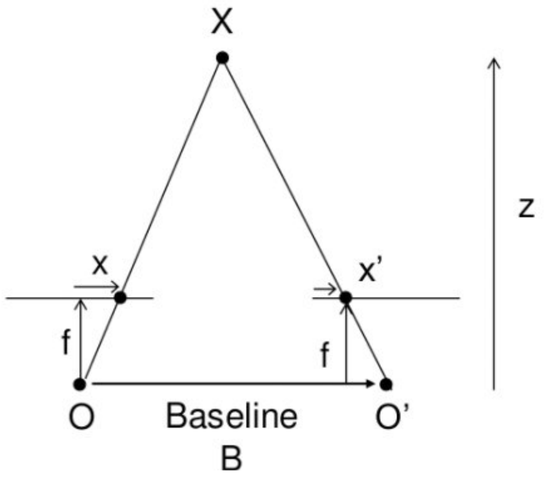

Figure 7: Depth calculation

Here, z = f * b/d, where z = depth, f = focal length, b = distance between cameras, and d = the disparity or distance between corresponding points. The disparity is defined as x' - x, the distance between projected points on the image plane.

OpenCV conveniently provides two algorithms to calculate the disparity map: stereo block matching (BM) and stereo semi-global block matching (SGBM). As per some research and testing, the BM algorithm was defined as the fastest algorithm between the two. Since the car can go to 80 km/h, a fast algorithm is really necessary, even though some accuracy is lost when using the BM algorithm. The default method is set in the main.cpp. If the SGBM method is used, it is necessary to change the parameters in the matchingmethod.hpp file. The default parameters are determined for the BM method, and won't work with the SGBM method. It is also possible to change the used method through the trackbar when using the program.

The next figure contains the first result of the rectified image (left) and its disparity map (right). The preliminary results were not reliable, and contaminated with a high degree of noise. By adjusting the values of the following parameters in the matchingmethod.hpp, better result can be achieved.

- preFilterCap – Truncation value for the prefiltered image pixels.
- SADWindowSize – Matched block size. It must be an odd number >=1.
- minDisparity – Minimum possible disparity value.
- numDisparities – Maximum disparity minus minimum disparity. This parameter must be divisible by 16.
- textureTreshold - Limit on the SAD window response such that no match is considered whose response is below this parameter.

- uniquenessRatio - Margin in percentage by which the best (minimum) computed cost function value should “win” the second best value to consider the found match correct. Normally, a value within the 5-15 range is good enough.
- speckleWindowSize – Maximum size of smooth disparity regions to consider their noise speckles and invalidate.
- speckleRange – Maximum disparity variation within each connected component.
- disp12MaxDiff – Maximum allowed difference (in integer pixel units) in the left-right disparity check.

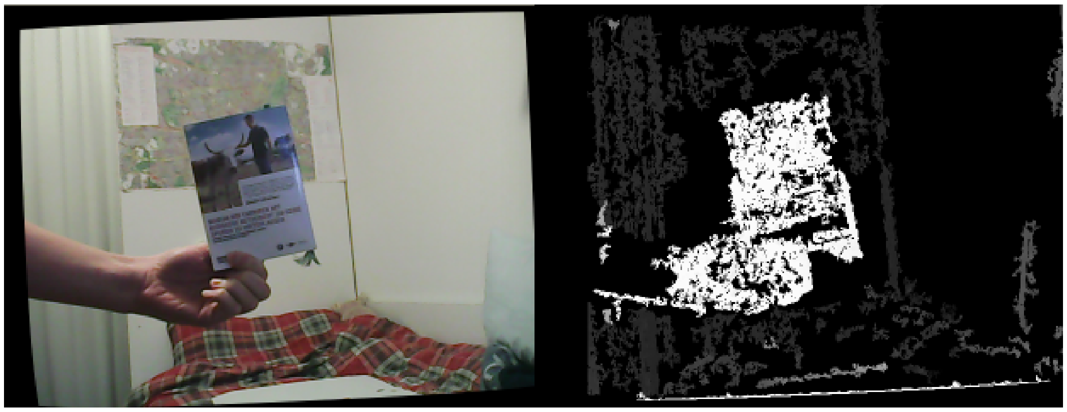
Figure 8: Initial disparity map

As easily deduced, the brighter the pixel, the closer it is located. Black pixels mean that no matching between the right and the left rectified images was possible. The next figure shows the disparity map after adjustment of the parameters.

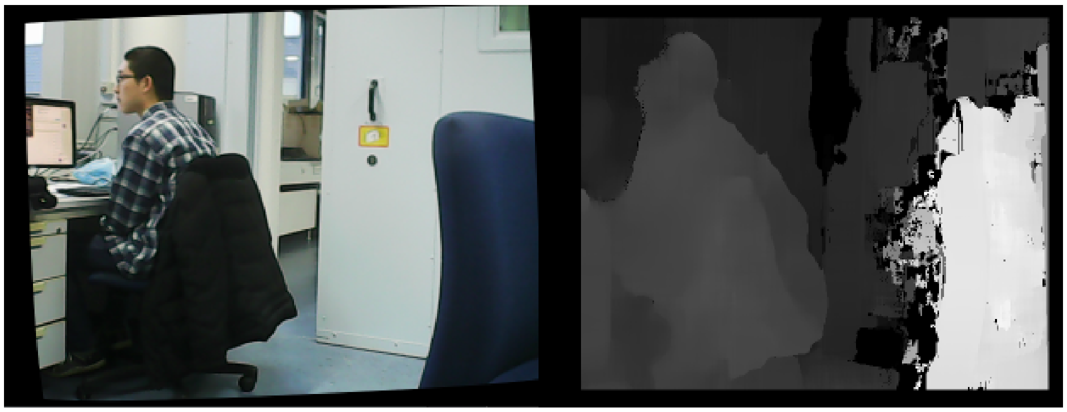
Figure 9: Final disparity map

Even after adjustment of the parameters, a lot of noise is still visible around the chair in the last image. However, it is safe to say that the noise is minimal compared to the initial disparity map. The best range for this algorithm, with this particular stereo camera, is from one meter to approximately five meters. Closer than one meter, the occlusion is too big, and the distance cannot be determined.

5.3 Top view of depth map

Since the car needs to decide where to go, a two-dimensional top view of the distance seen by the stereo camera is provided. The top view of the depth map is used to indicate where the obstacles are. This function is implemented in the Depthsubstraction.cpp file. The following figure show the theoretical top view of the depth map.

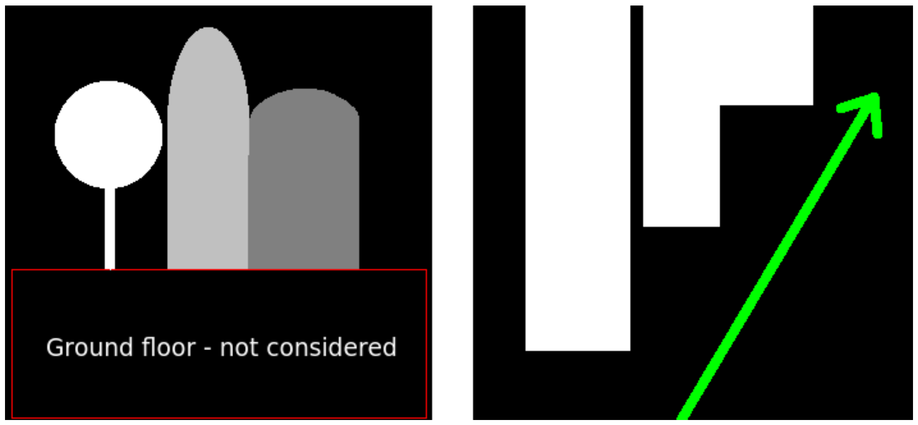

Figure 10: Theoretical top view of the depth map

The figure shows three different objects at different depths, and the obtained result on the right. The green arrow represent where the car would go in this situation. Since the camera on the actual car shows a good portion of the floor, the lower part of the image is not considered. Otherwise, the ground would be seen as an object. After some consideration, the top third of the image was also not considered, as no objects appear there, except the roof or the sky.

The following figure shows the final result of the top view. In this picture, we can clearly see the three different people in the depth map, and the corresponding depth result in the top view. Some noise was initially visible in the top view, but it was reduced after the implementation of a moving average algorithm. The algorithm was used to equalize the data in the array of the top view.

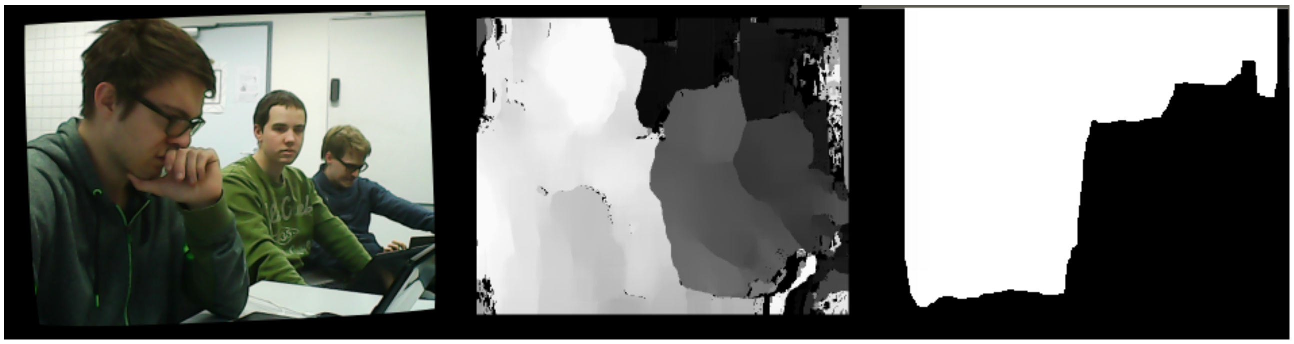

Figure 11: Result top view of the depth map

In short, the top view is calculated from the middle part of the depth map. The present strategy used, is an average of the pixel intensity of each column. The result where concluding, so other methods where not tested. Another good method would to localize in each column the brightest pixel, and it's intensity to determine the depth of this particular column.

5.4 Issues encountered and solutions

A lot of noise was present through the whole process, first with the disparity map, and then with the top view. For the disparity map, it was attempted to reduce the noise with some OpenCV filter, like a Gaussian blur, or dilatation. Unfortunately, the image processing was slowing the image stream and/or was not very effective, so the filtering was removed. For the top view, as the average intensity of the pixels was giving some unwanted fluctuation between the column values, a moving average algorithm was implemented with good result.

Also, since the ground and the ceiling (or sky) were present in the footage of the car, it was decided to discard them to evaluate the top view. The reduction in the amount of pixels to analyze speeds up the process, and makes it also more reliable, since the ground and the sky are certainly not objects.

Another problem encountered is the lighting. There is a lot of fluctuation if some windows are present, or if the car is moving outside. No further testing was made with this problem, but it needs investigation since light could disturb the view of the car, and make him collide into an object.

The BM algorithm might be the fastest one implemented in OpenCV, but it requires some testing since the car can go to really high speed. If the object detection is not fast enough, this could result in collision of the car into objects.

The last problem is the resolution of the camera. Since it is not particularly high, the objects located far away are not differentiated from the background in the disparity map. Therefore, the obstacles can only be seen when they are really close (1 to 5 meters) to the stereo camera. A proposed solution would be to have cameras with better resolutions.