Check the website for the newest version of this document and codes. This exercise has recently been updated, so it is likely that corrections and clarifications will be done. This is version 1.0.

Depth estimation from stereo cameras

Introduction

When looking out of the side window of a moving car, the distant scenery seems to move slowly while the lamp posts flash by at a high speed. This effect is called parallax, and it can be exploited to extract geometrical information from a scene. From multiple captures of the same scene from different viewpoints, it is possible to estimate the distance of the objects, i.e. the *depth* of the scene. By tracking the displacement of points between the alternate images, the distance of those points from the camera can be determined. Different disparities between the points in the two images of a stereo pair are the result of *parallax*. When a point in the scene is projected onto the image planes of two horizontally displaced cameras, the closer the point is to the camera baseline, the more difference is observed in its relative location on the image planes. Stereo matching aims to identify the corresponding points and retrieve their displacement to reconstruct the geometry of the scene as a depth map.

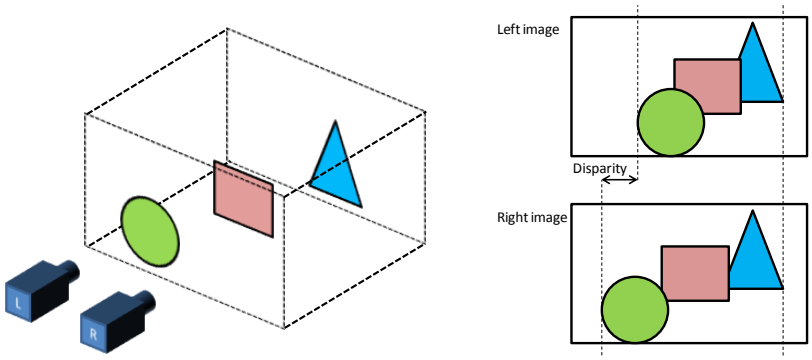


Figure 1.1 Disparity in a stereo pair. The green circle is closest to the camera, so its location in the image changes the most when the camera is displaced, while the distant blue triangle does not seem to be moving at all.

Stereo matching has traditionally been used in machine vision e.g. to make machines aware of the surrounding environment in different applications. More recently it has been adapted to produce and transmit content for 3D televisions, especially for multiview displays where it can save significant bandwidth compared to sending all required views separately. It also allows scaling of 3D content for different sizes and types of displays. As a passive method, stereo matching does not have to rely on explicitly transmitted and recorded signals such as infrared or lasers, which experience significant problems when dealing with outdoor scenes or moving objects, respectively. However, stereo matching does have its own drawbacks, the discovery of which is a part of this assignment.

Middlebury stereo database

The Middlebury stereo database has a collection of stereo pairs for use in development of stereo matching algorithms. Some of the Middlebury images are the de facto standard in comparing the results when new algorithms are proposed. They also rank matching algorithms on the quality of the produced depth maps.

Task: Find the Middlebury stereo database online and from the 2006 stereo datasets, download *Cloth1*, *Plastic* and two stereo pairs of your choice for your experiments. You can also pick pairs from other years. To limit the time required to process the images, use the link labeled "*2 views: ThirdSize"* to get subsampled images. **Note that the values in the ground truth disparity maps of this size should be divided by 3 to correspond to actual disparity**. Unzip the files to your working directory and familiarize yourself with the contents of the resulting folders. Don't modify the structure so your script is compatible with the reference environment.

File	view1.png	view5.png	disp1.png	disp5.png
Content	Left image	Right image	Left disparity	Right disparity

Question: What data sets did you choose? Pick any of the high ranking matching algorithms in the top 20 or so and check the publication behind it. What kind of similarity metric (i.e. how does the algorithm determine, which points correspond) the algorithm uses? Cite the publication.

Disparity map quality

The quality of estimated disparity is commonly measured in bad pixels – the percentage of pixels in the estimated disparity that are different from the ground truth. Small errors can be allowed by requiring the estimate to be within a given threshold.

Task: Implement a function that counts those pixels in the estimated disparity that are different from the ground truth and returns a percentage of the different pixels in the whole image. You can consider a pixel wrong if its value is off by more than 1 in either direction. Check that your function works correctly: use it to compare `magic(5)` against `magic(5)'`, the result should be 0.4800.

Disparity estimation

The whole concept of stereo matching is based on finding correspondences between the input images. In this exercise, correspondence between two points is determined by inspecting the $n \times n$ pixel neighborhood *N* around both points. The pairing that has the lowest *sum of absolute differences*,

$$SAD = \sum_{\boldsymbol{x} \in N} |L(\boldsymbol{x}) - R(\boldsymbol{x})|,$$

is selected as a corresponding point pair. In practice, a matching block is located for each pixel in an image. The relative difference in the location of the points on the image planes is the *disparity* of that point. Due to the assumption of being constrained into a 1-dimensional search space, these disparities can be represented as a 2D disparity map which is the same size as the image. Disparity of a point is closely related to the depth of the point, as can be seen in Figure 1.1. This is essentially a block matching scheme familiar from video compression (a.k.a. motion estimation), although in this application the search space can be constrained to be horizontal only (with certain assumptions). The matching block size is one of the most important parameters that affect the outcome of the estimation. Smaller blocks can match finer detail, but are more prone to errors, while large blocks are more robust, but destroy detail. In this assignment, a symmetric, square block of radius *r* has $(2r + 1)^2$ pixels.

Task: Implement a Matlab-script that estimates disparity from the Middlebury images with different block radii using the provided function `estimateDepth_AD`. Start from *r=0* and go high enough that you can be fairly certain the quality will not improve anymore. Measure the quality of the reconstructed map with the bad pixels -function you implemented. Make a single plot that contains a graph for each stereo pair depicting the quality metric as a function of block radius.

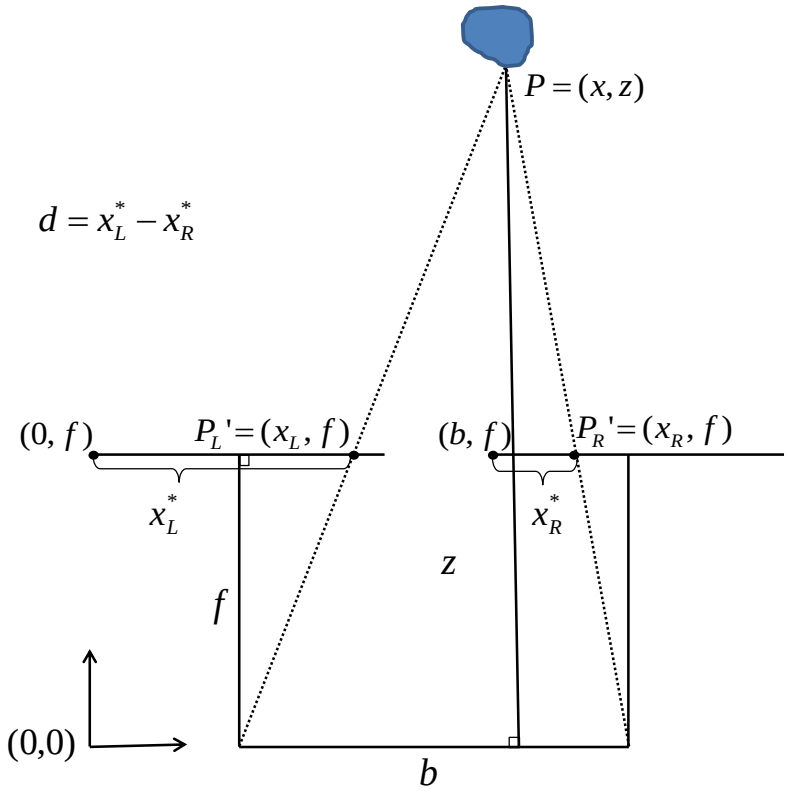
Question: Which block size is the best for each image? Is there a correlation between the type of content and the best block size? Attach the best estimate of each image to the report. Take a look at the output parameter `cost` from the depth estimation function. What kind of problem areas could you identify based on that information? On the other hand, what problems cannot be determined from it?

Depth from disparity

While the disparity map estimated from the stereo pair already distinguishes between objects at different distances, disparity is not the same as depth. The relation depends on the camera configuration.

Task: Solve *z* from Figure 2.1 by exploiting same shaped triangles. Baseline *b*, focal length *f* and image space coordinates x_L^* and x_R^* are known.

Question: Which triangles did you use? What is the formula?



2.1 Point *P* projected ($P'_L$and P'_R) to two image planes with parallel optical axes. The cameras are separated by baseline *b* and both have focal length *f*. There are same shaped triangles in the diagram that can be used to solve for *z* if disparity *d* (difference of image space coordinates x_L^* and x_R^*) is known.

Real data sets

The Middlebury data sets have high quality images captured in a controlled setting and with carefully calibrated equipment. In practice, the task becomes more difficult. Camera alignment even on physically connected cameras is never perfect, and there may be issues with different properties of the cameras. The camera used for this assignment is a Fujifilm FinePix REAL 3D W1 stereo camera. It outputs MPO image files, which are not understood by Matlab, so for further processing, they will be converted to a more convenient format using e.g. *mposplit* (available in the lab, also at http://cstein.kings.cam.ac.uk/~chris/mposplit/index.html).

Task: Contact the assistant by email and schedule a time for capturing your own stereo images using the equipment of the 3D Media Laboratory in TC409. Bring a USB-stick or equivalent for taking the pictures with you.

Rectification

For efficient stereo matching, the concept of *epipolar lines* is essential. The horizontal translation in the camera positions between the stereo pair should only create differences in the horizontal direction. This allows for the matching to happen only in one direction, along an epipolar line, greatly reducing the search space. However, this is rarely the case in a realistic capture setting. Camera misalignment makes it impossible to reliably search only on the epipolar lines. This is why a software rectification step is performed after capture. In short, the misalignment of the image planes can be corrected by rotating them in a 3-dimensional space. The rotation matrices doing this can be computed from the images themselves or from specifically captured calibration images. While the topic of calibration and rectification would be enough for a laboratory assignment of its own, in this exercise, the rotation matrices and the function to apply them are provided for the stereo equipment found in the lab.

Task: Load the rotation matrices using `load('fuji_rect.mat');`
Apply the matrices to your self-taken images using the `rectify` function. Make the stereo pair to approximately match the size of the Middlebury images by either cropping or resizing. Apply the depth estimation function on several image pairs and try to find such radii that are good compromises between loosing detail and getting matching errors. Visually estimate the correct maximum disparity by looking at `imshow(cat(1, L, R), [])`. Note that the disparity range should be equal or more than the disparity of the foremost object, but not less. Attach the best depth map you are able to generate to the report.

Question: How does the subjective quality of the depth maps compare to the ones estimated from Middlebury data? Are there any distinctive problems with some type of content? If (and when) any especially problematic images or sections appear, also include an example of that.

3D model

A basic 3D-model can be created from a set of points in 3D space. In a depth map, a 3D-point *(x,y,z)* is represented as the value *z* in the pixel coordinates *(x, y)*. A typical representation of a 3D model is a mesh consisting of a set of connected triangles, which Matlab can automatically create between given points. In addition to the geometry of the scene, the model can be improved by adding color information, i.e. *texture* to it by defining the color of the model at each point, i.e. *vertex*. The colors of the scene are naturally available in the source stereo pair from which the depth was generated from.

Task: Create a 3D model using the `surface` tool by estimating disparity from a self taken stereo pair. Apply the disparity to depth conversion using the formula you determined from Figure 1.1. The parameters *b* and *f* are included in the calibration file for the camera. Convert the disparity estimate from pixels to meters by multiplying with the given *pixelsize.* You can apply filtering (`medfilt2` etc.) to the depth map to get rid of artifacts and select just a particularly good quality section the estimated depth. Invert the depth map by subtracting it from the maximum depth value. Hide the grid lines using the `'EdgeColor'`-parameter as they are too thick by default and they cover the real texture. Rotate the model so that the structure is visible and attach an image of it to the report.

Question: What is the drawback of a geometric model created just from depth information, i.e. what information is missing?

Summary of tasks

1. Download images From Middlebury
2. Make a function for evaluating the quality (% of bad pixels)
3. Estimate depth from the Middlebury images using different radii
4. Solve relation between depth and disparity
5. Schedule a time for capturing stereo images
6. Apply rectification to the captured images
7. Estimate depth from the rectified images
8. Construct a 3D mesh
9. Make a report including the plots, images and answers specified in the detailed instructions

Summary of deliverables

1. The report
2. Matlab function that computes the percentage of mismatched pixels
3. For the Middlebury data set, a Matlab script that generates the graph for bad pixels as a function of block size when executed in the directory that contains the extracted Middlebury directories. Don't copy&paste code, put input data into arrays and use for-loops.
4. Matlab script that applies rectification, estimates the depth, and generates a 3D model of a stereo pair you took yourself during the laboratory visit. Simply use the parameters you find to provide best results, no need to include iterations like in section 3.
5. The self-taken stereo pair used for section 4.

Hints

- Work consistently in floating point format – images load as uint8 by default (try executing `uint8(0)-uint8(1)` and you will get the idea)
- Remember the scaling coefficient in Middlebury ground truths
- You can easily iterate through different data sets by saving the names of the sets in a cell array:
  ```
  datasets{1} = 'Cloth1';
  datasets{2} = 'Plastic';
  ```
  Then indexing it with a loop variable `i`
  ```
  L = imread([datasets{i} '\view1.png']);
  ```
- Check that you are computing the quality metric by comparing the estimate to the correct ground truth – the depth maps are different from different viewpoints
- The `surface` tool takes floating point RGB in [0, 1] as its texture input. Adjust the image format to match this.
- Matlab matrix indexing starts from upper left corner while the convention for 3D graphics is to start from lower right. Adjust `surface` input accordingly (`flipdim`)
- You can rotate the 3D model accurately by issuing the command `camorbit(30, 10, 'direction', [0 1 0]);` Rotating the model by mouse is quite inconvenient
- You can use *Edit->Copy figure* to copy and paste a figure directly from Matlab into Microsoft Word as vector graphics. If using LaTeX, saving figures in *.eps* is a good choice.
- The Matlab functions are provided as P-files (instead of the more familiar M-file). There are other courses where the task is to create such functions, thus these are obfuscated to avoid directly distributing model answers for those assignments