Shallow Depth of Field effect using Stereo Cameras

Vrishabh Lakhani

Rochester Institute of Technology
Rochester, N.Y., U.S.A
val3917@rit.edu

Abstract

Depth estimation has been one of the fundamental challenges in the field of Computer Vision. There are various application in which it can be used and there has been extensive research over methods to do so in the past. Using Depth Estimation, it is possible to create an artificial shallow depth of field effect which can recreate an image which looks like it has been taken from a professional camera. Thus, in order to achieve this, there are many unique state of art methods and one of them is using Stereo Cameras. In this project, we are trying to recreate this effect, find the depth estimation from the images of the stereo camera and use it artificially create depth of field effect by taking advantage of the distance blur effect.

Keywords:
Depth estimation, Stereo Camera, Disparity Maps, Shallow Depth of Field, Computer Vision

1. Introduction

In this paper, we discuss depth estimation using stereo camera using a pipeline for stereo matching algorithm. Many application like object modelling, scene understanding and context recognition require us to recover three dimensional shape of the image. Thus, the goal of 3D shape recovery is inherently of depth estimation. The case of this paper is understanding of this pipeline which can be exploited to get a good disparity map from the stereo images.

Since there could be a lot of variation in how the stereo images are shot and what kinds of lenses and cameras they are using, we are only going to discuss disparity map generation from stereo cameras of similar specification.

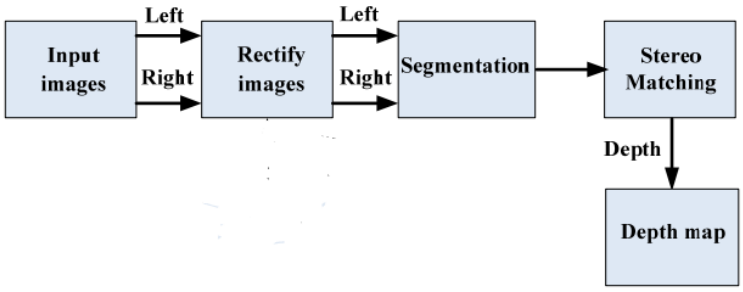

Figure 1: Disparity map generation pipeline

This disparity between stereo camera could be of different conditioning like light differences [**?**] or location differences[**?**]. Apart from the specifications, the cameras have to be on the same horizontal plane. Cameras in such a setting are called said to be in Epipolar geometry. Such sort of images have to be in the same horizontal lines and there should be no rotation between the left and right images. If there is, then there is a need for rectification of these images. Thus, in order to move forward, the first preprocessing step in our pipeline is image rectification.

After we rectify both the images, we can start image segmentation so that we can apply stereo matching algorithms over them. Segmentation divides the images into regions over which we can use create a disparity map.

The next step is the stereo matching algorithm, here we compare the pixels in the left and the right images and check for similarity. If they are eligible to be matched then they are matched and a disparity map is created from the matching pixels and the difference between the co-ordinates of the matching pixels in both the images.

After a disparity map is created, we have to apply a gradient of image blur based on how far the object is. The farther the object the greater the blur of that pixel and less blur the nearer it is to the camera's plane. Such kinds of artificial blurring is better than using deep learning approach to such problems because deep learning approach can create random artifacts while trying to predict which pixel to blur or not to blur.

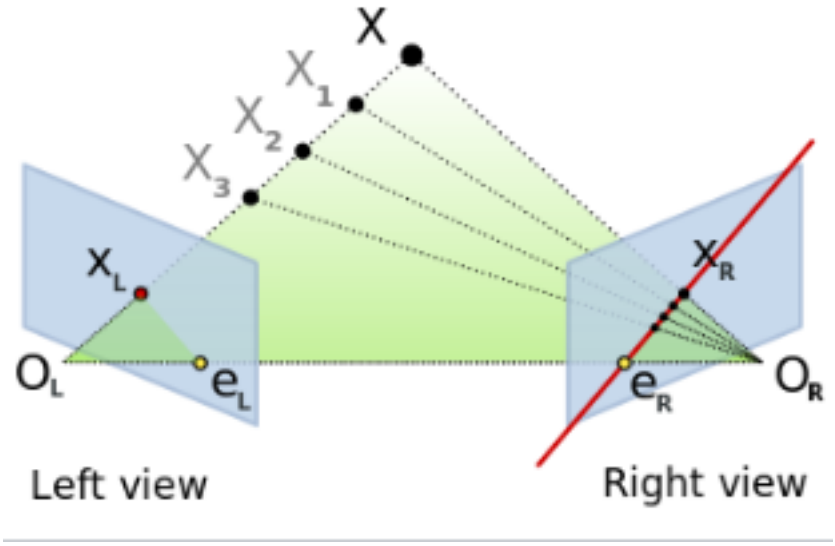

Figure 2: Epipolar line geometry

2. Camera Calibration

In order to get a good accuracy of depth estimation of the imagery, it is important to calibrate the camera based on the image context.
This helps us get an image with precise edges and corners. This gives us meaningful features in the image. In order to do so, we have to check whether the focus, aperture and other settings are correct.

3. Image Rectification

In order to simplify the disparity computation algorithm, there is a need to use epipolar geometry since we need to find the epipolar lines between two horizontally aligned images.
In real-world stereo images, there is always a room for error where the images aren't completely horizontal to each other and thus we can rectify it by using a combination of linear transformation like rotation, scaling, skewing, etc.

To automate this process in the depth estimation pipeline, we use some algorithms like SIFT which are designed for exactly such sort of application. Using such algorithms creates a pair of stereo images where the pixels lie on

the same epipolar horizontal line and thus our search spece to find the matching pixel in the pair is reduced to a one-dimensional search space instead of 2-dimensional search space which exponentially reduces the computational time of calculating the disparity map.

4. Disparity Calculation

In this section, we explain the process of disparity map calculation which is done after the images are segmented before computation and then the pixels are matched between the left and the right images. This is the main task that used in creating a depth estimation for stereo images.
In order to explain it, we need to use geometry to figure out the proportionality between the different points of the image plane. Let the distance between the same pixels on the left and the right image be d, the disparity for that pixel and x_r and x_l and the x co-ordinates of the right and left image respectively. Therefore, we see that,

$$d = x_l - x_r = f\frac{x_p + l}{z_p} - \frac{x_p - l}{z_p}$$

$$z_p = \frac{2fl}{d} = \frac{fB}{d}$$

This definition of disparity calculation using just the x-coordinates is possible only when the pixels of the images from both the cameras have the same y-coordinate, and this is true in our case since we have calibrated the cameras accordingly.
From this we can conclude that the disparity of the pixel can be computed based on the distance between the two cameras and the focal length of the lens of the camera.

5. Stereo Matching Algorithm

After the disparity for the image pixels is calculated the next step is to reconstruct the disparity map from a pixel pair from the left and the right image.
In order to do this, we need to algorithms to compute the intensity differences between each of these pixel pairs and based on that we can create an entire disparity map. Some common techniques used for this are Sum of

Squared Differences (SSD), Sum of Absolute Differences (SAD) algorithms, Correlation(C) and Normalized Cross Correlation(NCC).
The major difference between each of these algorithms is based on two metrics, the accuracy with which they generate the disparity map and the second metric being how much the computation cost of each of the algorithm is. Based on the application of the generated disparity map, we figure out which algorithm to use.

5.1. Sum of Absolute Differences

Among the disparity map generation algorithm, the simplest algorithm is the Sum of the Absolute Differences (SAD). This being the simplest algorithm, it is the fastest algorithm to compute the disparity map and can be used in application where there's a need for almost real-time generation of the disparity maps.

$$SAD(x, y, d) = \sum_{N}^{(i,j)} I_l(i, j) - I_r(i - d, j)$$

Here, i_r and I_l are the pixel intensity of the pixel from the left and the right image and summation is calculated over a window of pixels that are surrounding the given pixel. To figure out the best matching pixel, we try to minimize the SAD of the pixels from the frame. The pixel with the minimum SAD are sad to be the one which have the matching pixels.
Here, one hyperparameter that we notice is the size of the window. The size of the window gives us a chance at variability of trade-off between computation and accuracy. The bigger the window, the more accuracy the disparity estimation but higher computation cost.
Thus, in order to further optimize the process, in application where we know that label of the images, being from the right and the left camera, we calculate only one side of the pixels instead of both the sides. This results into the computation complexity reducing to half and thereby reducing the overall time that is required to compute over the entire image.

5.2. Sum of Squared Differences

Since working on images is a non-convex optimization problem, sometimes it's better to work on different norms as the distance metric and thus, we come up the Sum of Squared Differences (SSD) algorithm.

This algorithm is similar to the previous algorithm, but instead of summing up the differences in the intensity directly, we first square it and then sum the result.

$$SAD(x, y, d) = \sum_{N}^{(i,j)} (I_l(i, j) - I_r(i - d, j))^2$$

It is expected that this can be used to get a better estimation of disparity between the image pixels as compared to SSD but is computationally more expensive since we have to take the square of the intensities.

6. Distance based Gaussian blurring

After we have the depth map estimation of all the pixels in the stereo image pair, we can use this disparity map along the original image.
In an image with depth of field phenomena, we notice that the objects which are farther away have a blur which is greater than an object which is nearby. Thus, when we are creating an artificial shallow depth of field, we can take advantage of this and create this effect by blurring the pixels with higher disparity on the disparity map and lower blur over pixels with less disparity map. We can optimize this further by binning the levels of disparity maps value and creating a different blur value for each of the bins.

7. Methodology

In this section, we discuss how we ran the experiment and all the challenges we tackled while coming up with a solution to the problem.
In order to get a good dataset of stereo images, we use the Middlebury 2014 Stereo dataset. This dataset contains of 33 sub-datasets, each of which contain images taken by stereo cameras in all sorts of different lightning and camera capturing conditioning.
While trying to recreate the disparity map in our code on MATLAB, we tried two approaches, one option was to use the in-built disparityMap function and the second one was creating it using scratch. Creating it from scratch didn't have performance optimization and it ran slower than the in-built function. The in-built function also somehow managed to give more accurate results than the code which was coded from scratch. It supposedly used some advanced image rectification and smoothing algorithms in order

to get the disparity map correct. After this, we try to combine the disparity map with our original image to get a blur effect.
In doing so, we realize that instead of creating a blur for all the different levels of depth, it was a better idea to bin them into a small number of bins and apply blurring over only those bins. It still looks realistic since human eyes can't anyway understand the difference between very small differences in depth.
Thus, using the in-built function and using the binning technique we get an optimized approach which tries to balances between the performance and the accuracy of the disparity map.
While doing the block comparison, we had to apply SAD, in order to do so, we required one value for each of the pixel, this was done by using the grayscale versions of the images instead of 3 channel colored versions. Also, while coding in MATLAB, it became necessary to work in floating point and thus we had to convert our images from the default int-8 type to double precision.
Another problem we had to tackle was how to run the block matching algorithm at the edges. Since we were using the a block size of 7x7 or 10x10 over the pixels of the image, it wasn't possible to do this on the pixels on the edges like top left or top right corner since there are no pixels on the left and top of them, so for these edge cases, we had to use a template block size of 4x4. Similarly, the neighboring pixels to the corner pixels had adjusted template block size as well.
From [**?**] we understand the resulting disparity map will always have integer values and thus in order to get a more accurate answer, we would have to find some type of sub-pixel estimation of the disparity of each pixel. In order to do this, we have to consider the disparity as the root of a second degree polynomial and solve it for it's minima. On doing so, we were able to successfully get a result which was good enough to be given as the final disparity map.

8. Experimental Results

In this section,we discuss our experimental results when we try to create the image from disparity and original image. But first we need to show the results of our created disparity map based on the different hyperparameters that we can chose such as the window size while calculating (SAD), etc.
Given below is the result of different windows for the (SAD) calculation in

the figure.
After we have created the disparity map, we can create a blur effect based on the distance of the pixel from the camera plan. The resulting image with binned blurring is like this:

9. Conclusion

The paper depicts a computer vision pipeline that is used to recover disparity map based on the stereo matching algorithms used for depth estimation tasks. The algorithm is tested over a dataset which is used as a metric to test how well depth estimation algorithms perform. We realise the importance of how much hyper-tuning the parameters are important in order to create a perfect depth-map while also keeping in mind the trade-off between accuracy and performance.
We learned from this project how to utilize stereo imagery and use the additional information for tasks which are almost impossible to do with a mono-camera setting. The application of using depth-map for shallow depth of field is quite interesting as this is what happens in the potrait-mode of the latest smartphones that we see in market today. Working on this project gives hands-on experience to realise the problems which occur while trying to achieve the goal.
In the future, we plan to compare the results of depth estimation of this algorithm to a deep learning based depth estimation algorithm which is based on predicting disparity map based on the input from a mono camera setting. We also planned to test the reliability of such algorithm over stereo settings where both the camera are of different focal lengths. This is proven to be beneficial for creating shallow depth of field effect when one of the camera uses a telephoto lens.

References

1. D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., Madison, WI, USA, 2003, pp. I-I. doi: 10.1109/CVPR.2003.1211354

2. Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus Gross. 2010. Nonlinear disparity mapping for stereo-

scopic 3D. ACM Trans. Graph. 29, 4, Article 75 (July 2010), 10 pages. DOI: https://doi.org/10.1145/1778765.1778812

3. Eigen, David and Puhrsch, Christian and Fergus, Rob, 2014. Depth Map prediction from a Single Image using Multi-Scale Deep Network. NIPS 2014, Advances in Nueral Processing Systems 27, 2366-2374

4. Liu, Dongwei Nicolescu, Radu Klette, Reinhard. (2015). Bokeh Effects Based on Stereo Vision. Lecture Notes in Computer Science. 10.1007/978-3-319-23192-$1_1$7.

5. Stereo Vision Tutorial - Part I, http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/

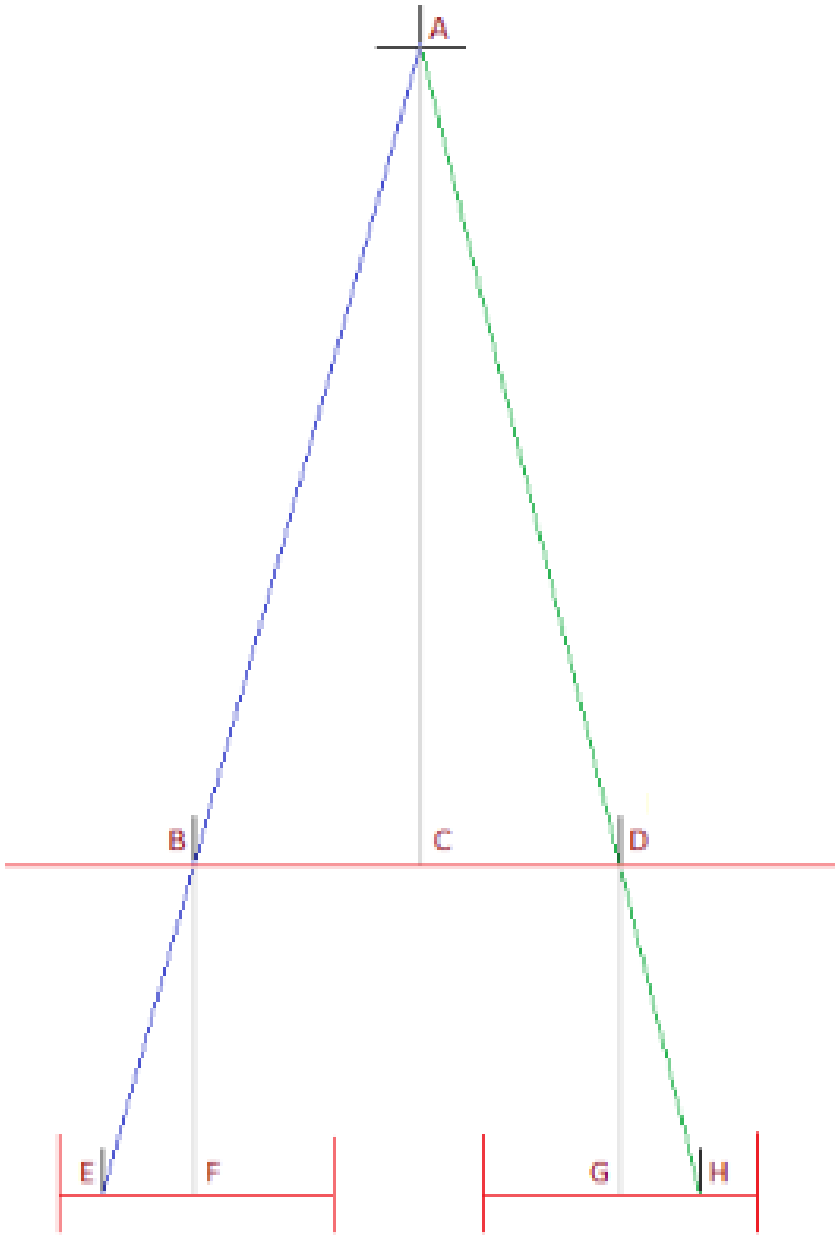

Figure 3: Disparity map geometry

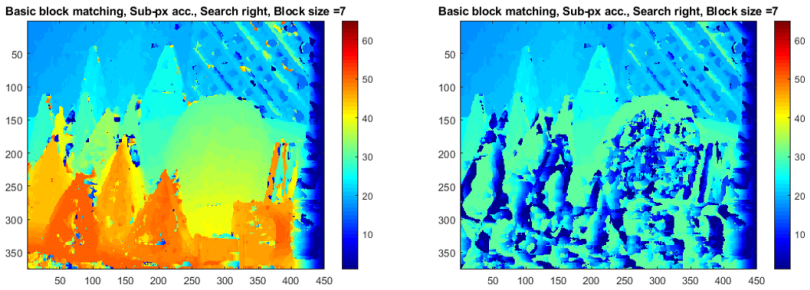

Figure 4: Disparity map geometry with a block size of 50 and 30

Figure 5: Disparity map geometry with a block size of 50