Computer Vision
Fall 2017
Final Project
Stereo Correspondence

Zhichen Guo
zguo98@gatech.edu

Description of existing methods published in recent computer vision research.

There are several methods published in recent computer vision research:

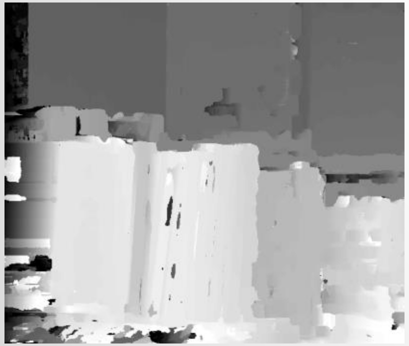
1. Moritz Menze and Andreas Geiger proposed a novel model and dataset for 3D scene flow estimation with an application to autonomous driving. Taking advantage of the fact that outdoor scenes often decompose into a small number of independently moving objects, They represent each element in the scene by its rigid motion parameters and each superpixel by a 3D plane as well as an index to the corresponding object. This minimal representation increases robustness and leads to a discrete-continuous CRF where the data term decomposes into pairwise potentials between superpixels and objects. Moreover, Their model intrinsically segments the scene into its constituting dynamic components.[1]
2. Jure Zbontar and Yann LeCun train a convolutional neural network to predict how well two image patches match and use it to compute the stereo matching cost. The cost is refined by cross-based cost aggregation and semiglobal matching, followed by a left-right consistency check to eliminate errors in the occluded regions.[2]
3. Wenjie Luo, Alexander G. Schwing and Raquel Urtasun proposed a matching network which is able to produce very accurate results in less than a second of GPU computation. They exploit a product layer which simply computes the inner product between the two representations of a Siamese architecture. They trained their network by treating the problem as multi-class classification, where the classes are all possible disparities. This allows them to get calibrated scores, which result in much better matching performance when compared to existing approaches.[3]
4. Sergey Zagoruyko and Nikos Komodakis proposed how to learn directly from image data a general similarity function for comparing image patches, which is a task of fundamental importance for many computer vision problems. To encode such a function, They opt for a CNN-based model that is trained to account for a wide variety of changes in image appearance. They explored and studied multiple neural network architectures, which are specifically adapted to this task. They showed that such an approach can significantly outperform the state-ofthe-art on several problems and benchmark datasets.[4]

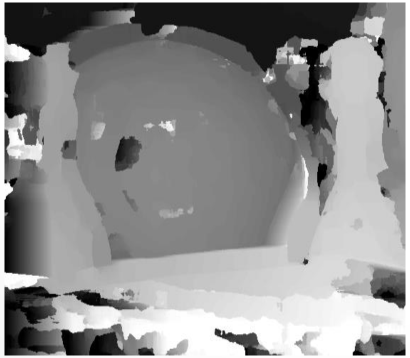
Description of the method you implemented.

The method I used is building a disparity map with SSD (the Sum of Squared Differences) which is an metric for searching for the best match along the same scan line in the left and right images.[5]

Implement the basic stereo algorithm of taking a window around every pixel in one image, and search for the best match along the same scan line in the other image. The parameters for different pairs of dataset are the scan window size, matching direction and the pixel number of test the disparity. The disparity map would contain the information of matching depth via certain direction. To group up similar components such as image pixels or image regions, a k-means segmentation methods would be used as a graph-cut techniques with the disparity maps to improve the results.[6] The classical k-means algorithm partitions a number of data points into several subsets by iteratively updating the clustering centers and the associated data points. By contrast, a weighted undirected graph is constructed in min-cut algorithms which partition the vertices of the graph into sets.

Results obtained from applying your algorithms to images or videos.

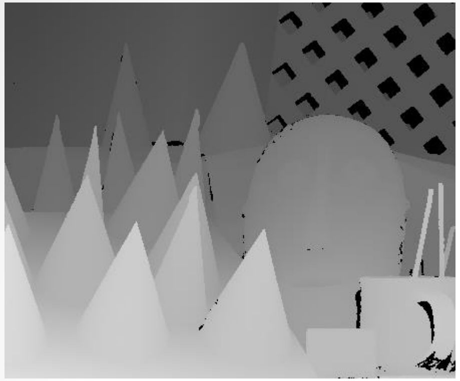

Result of Books dataset
Disparity map left to right

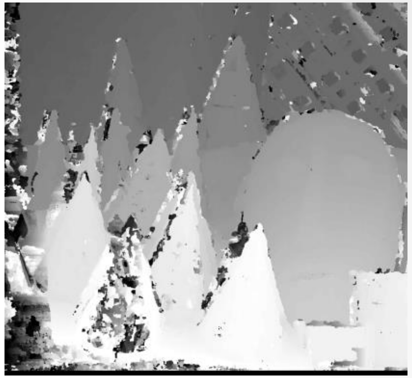

Result of Bowing dataset
Disparity map left to right

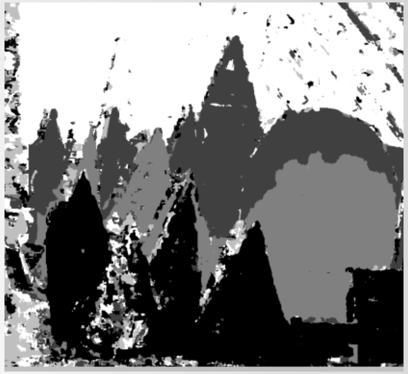
Results obtained from applying your algorithms to images or videos.

Results of Cones dataset

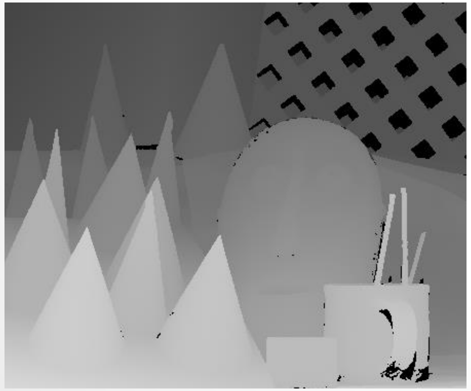
Left image

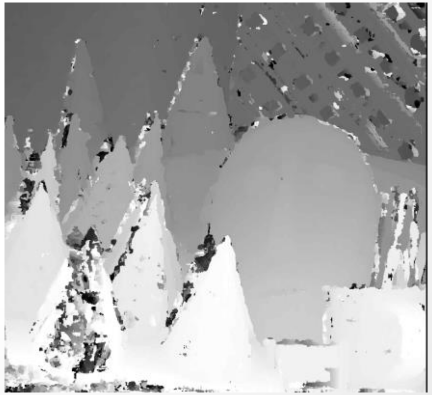

Ground truth

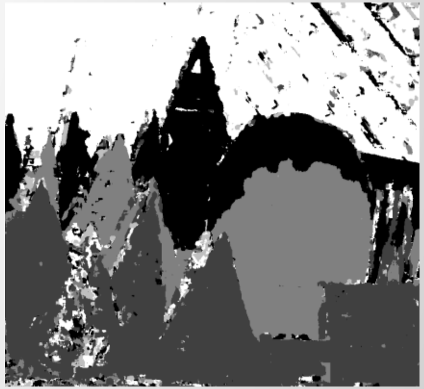

Disparity map left to right


K-mean segmentation image

Right image


Ground truth


Disparity map right to left


K-mean segmentation image

Analysis on why your method works on some images and not on others.
As shown in previous pages, my method works on cones dataset better than others.
The images with big chunk of similar pixels such as bowling dataset, lampshade and wood dataset, may not work very well on my method. It’s hard to get the difference between two images in the window.
The images with many different details such as cones dataset would work fine on my method. The difference between two images in the window is large which is easy to get a disparity map with larger contrast.
Performance statistics analysis.

Dataset	resolution	running time
Books	1390 x 1110	119 s
Bowling	1252 x 1110	105 s
Cones	450 x 375	50 s

The above data was running at 10 x 10 scan window. Increase the window size or pixel number of test the disparity would extend the running time.
The bowling dataset result is got with a 25 x 25 window size and 300 pixels to test disparity, which cost me 11 minutes and 14 seconds.
The lager images would take much longer running time. For example, the motorcycle dataset which is 2964 x 2000, cost me more than 18 minutes to run with at 10 x 10 scan window and 100 pixels to test disparity. So I tried resize the images by half or quarter before calculate the disparity maps on large dataset.
Discussion on how your results compare to the state of the art methods.
The results of the state of the art methods are very neat and clean and almost the same with the ground truth. In comparison, my disparity results are not very well, especially the bowling dataset.
My disparity results of the cone dataset is the best of all my results. And the results of the state of the art methods are better than mine.
My results of k-mean segmentation looked not good. I’d like to improve the k-mean segmentation algorithm to fix it. The graph cut part works well and the image does the segmentation as it should, the problem is the gray level may assigned randomly.
Proposals on how your methods can be improved.
There are lots of things to do to improve my methods.
My results of k-mean segmentation looked not good. That’s because I didn’t figure out how to set the grayscale base on the original pixel values. The graph cut part works well and the image does the segmentation as it should be, the problem is the gray level may assigned randomly. I’d like to fix it to improve my methods.
My disparity map has so much noise. Especially at the edge of the image and the edge of the objects in image. This part could improve using other methods to calculate the disparity, such as the features matching or block matching method which are good on detect the edge of objects between images.
My result of bowling dataset is not good. The reason may be that the images contain less detail than other datasets. It’s hard to calculate the obvious different between two images using my algorithm. Adding some noise in advance and then smooth the disparity images may improve the results. This would be the next job I’ll do.

References and citations

[1] Moritz Menze, Andreas Geiger. Conference on Computer Vision and Pattern Recognition. 2016. http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo

[2] Zbontar, Jure and LeCun, Yann. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. 2016. http://www.cvlibs.net/datasets/kitti/eval_scene_flow_detail.php?benchmark=stereo&result=92229724a5bc8ddd3a2b30406924010d6919934e

[3] Wenjie Luo, Alexander G. Schwing, Raquel Urtasun. Efficient Deep Learning for Stereo Matching. 2016. http://www.cs.toronto.edu/deepLowLevelVision/

[4] Sergey Zagoruyko, Nikos Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. 2015 https://github.com/szagoruyko/cvpr15deepcompare

[5] Image Proximity Measures. https://software.intel.com/en-us/node/504333

[6] k-means, mean-shift and normalized-cut segmentation http://cn.mathworks.com/matlabcentral/fileexchange/52698-k-means--mean-shift-and-normalized-cut-segmentation