# Final Project
# Stereo Correspondence

Chenguang Ji
CS6476 Spring 2021
cji62@gatech.edu

***Abstract*—Stereo correspondence has been one of the most researched area in computer vision. In this report, we will introduce two methods for obtaining the disparity map from an image pair: sum squared difference (SSD) method and tree-structure based dynamic programming (DP) method. We will discuss the algorithm for both methods and compare our results with ground truth qualitatively, and quantitatively using the metrics we develop. At the end, we will compare the pros and cons of these methods.**

***Index Terms*—stereo correspondence, sum squared difference, dynamic programming**

## I. Discussion of existing methods

Based on how the disparities are assigned to each pixel, the algorithms can be classified as local methods and global methods.[1]

Local methods use the information of each pixel, and its neighboring region to determine disparities. One commonly used local method is block matching algorithm, which tries to find a region in the target image that resembles a corresponding region in the template image. Usually the search is conducted in the epipolar line. We can define the similarity of two image patches by cross correlation, normalized cross correlation or sum squared difference (SSD). The SSD method is fast and effective, and we will discuss it later.

Another class of local methods is gradient-based methods, or optical flow methods. Zhou and Boulanger developed a radiometric invariant stereo matching algorithm, which takes both the view independent and view dependent colors to compute the relative gradient.[2]

Block matching and gradient-based methods are sensitive to regions with uniform texture. Venkateswar and Chellappa use a feature-matching algorithm to solve this problem. They use hierarchical features consisting of lines, vertices, edges and surfaces. Higher level features (surfaces) are matched first and used to constrain the matches at lower levels.[3]

Major drawbacks of local methods include discontinuity due to occlusions and uniform texture. Global methods impose non-local constraints to improve accuracy of the result. In global methods, the problems are formulated in terms of global energy minimization. Both graph cut and dynamic programming (DP) are used to optimize energy.

1D dynamic programming optimize energy along scanlines.[4] With each scanline optimized separately to obtain smooth disparity along each row. And the global energy minimum is achieved as the sum of respective scanlines. However, as the disparities are only estimated in the horizontal direction, artifacts are noticeable along vertical direction.

Dynamic programming on 2D gives true global optimization. Veksler applies dynamic programming to a tree structure, with nodes being each pixel and edges being the 2D image grid with great importance.[5] Bleyer and Gelautz combines two semi-1D dynamic programming result to obtain a 2D optimization.[6] The Bleyer and Gelautz method is used in this report and will be discussed in detail later.

Hong and Chen use graph cut method to divide reference image into non-overlapping segments and assign the corresponding disparity plane to each segment.[7] Boykov, Veksler and Zabin improve the algorithm efficiency by using $\alpha$-$\beta$ swap and $\alpha$-expansion.[8]

## II. Discussion of Algorithms implemented

The first method implemented is SSD method. For each pixel p(x,y) in the template image I, we take a window of size k centered at p. And then we compare this window to windows with the same size, centered at p'(x+d,y) in the target image I', where d $\in$ D, with D denotes the set of possible disparity values. SSD is defined as the sum of squared element-wise difference between these two windows. d with the smallest SSD value is assigned as disparity of p.

With input image of size w $\times$ h, this method takes O(D$\times$w$\times$h). This can be reduced to O(w$\times$h) by pre-computing a set of difference images between target image and template image shifted by d (d $\in$ D), and then convolve the squared difference images with a box filter.[9]

The second method implemented is simple tree DP method.[6] Still, let I be the template image and I' be the target image, and D denotes the set of possible disparity values. We are trying to find for each pixel p $\in$ I, the disparity value dp $\in$ D. The energy function is defined as

$$E(D) = \sum_{p \in I} m(p, dp) + \sum_{(p,q) \in X} s(dp, dq) \qquad (1)$$

m(p,dp) is the data term, which computes the intensity difference between p(x,y) and p'(x+dp,y). s(dp,dq) is the smoothness term, it defines the smoothness of p to its 4-connected neighbor q in I. s(dp,dq) is defined in equation (2) and (3).

**Algorithm 1:** SSD method

```
for d in range(0, D) do
    Calculate difference images between I' and I shifted by d
    Calculate squared difference images
    Convolve squared difference image with a box filter of size k to obtain SSD images
end
for p(x,y) in I do
    Find the SSD image with the smallest value at location (x,y)
    The corresponding d for that image is the disparity at p(x,y)
end
```

$$s(dp, dq) = \begin{cases} 0 & \text{if dp = dq} \\ P1 & \text{if |dp-dq| = 1} \\ P2 & \text{otherwise} \end{cases} \quad (2)$$

$$P2 = \begin{cases} P3{\cdot}\text{P'2} & \text{if } |\text{Ip-Iq}| \leq \text{T} \\ P'2 & \text{otherwise} \end{cases} \quad (3)$$

In this method, two types of simple trees are constructed, horizontal tree and vertical tree. As is shown in Fig. 1, the tree is rooted on p whose disparity is to be computed. We will discuss the horizontal tree first, and vertical tree can be processed accordingly. The horizontal tree includes all horizontal scanlines and one verticle scanline, which enforces the smoothness along both directions.

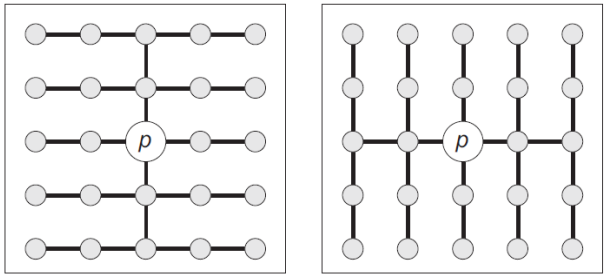


Fig. 1. a: Horizontal tree. b: Vertical tree.

First, we will compute the optimal energy along horizontal scanlines. The energy is computed recursively by function l().

$$l(p,d) = m(p,d) + min_{i \in D}((s(d,i) + l(q,i))) \quad (4)$$

In equation (4), q is the sibling of p. As is shown in Fig. 2. p can have two different siblings, one on the left and the other on the right. These two scenarios are called forward pass (F) and backward pass (B), respectively. The optimal energy alone horizontal lines (C) is derived by combining F and B according to equation (5).

$$C(p,d) = F(p,d) + B(p,d) - m(p,d) \quad (5)$$

The next step is to compute the optimal costs of p from all horizontal scanlines. As is shown in Fig. 3, this can also be computed from both forward (downward) and backward (upward) directions. Similar to equation (5), the optimal energy for p is defined as

$$l'(p,d) = C(p,d) + \sum_q min_{i \in D}((s(d,i) + l'(q,i))) \quad (6)$$

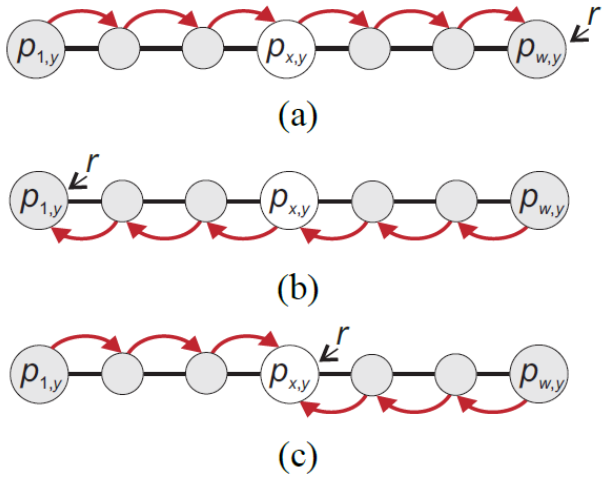


Fig. 2. Computation of optimal costs of a scanline. a: Forward pass. b: Backward pass. c: Optimal costs for p are derived by combining forward and backward passes.

After the optimal costs H for p from horizontal trees is obtained. We can calculate the optimal costs V for p from vertical trees using the same steps discussed above. To combine horizontal tree with vertical tree, we first calculate V, and then integrate V to the expression of m(p,d) in equation (4)

$$m'(p,d) = m(p,d) + \lambda(V(p,d) - min_{i \in D} V(p,i)) \quad (7)$$

Here $\lambda$ is a parameter to adjust the weight of vertical tree on the final disparity value. This term serves as a penalty when d calculated from V is not an ideal disparity value along vertical directions. By this method, for each pixel, we are determining the disparity globally, as we utilize both horizontal and vertical trees, which extend to the whole image.

Finally, we determine the disparity value by searching for d which gives the smallest overall energy.

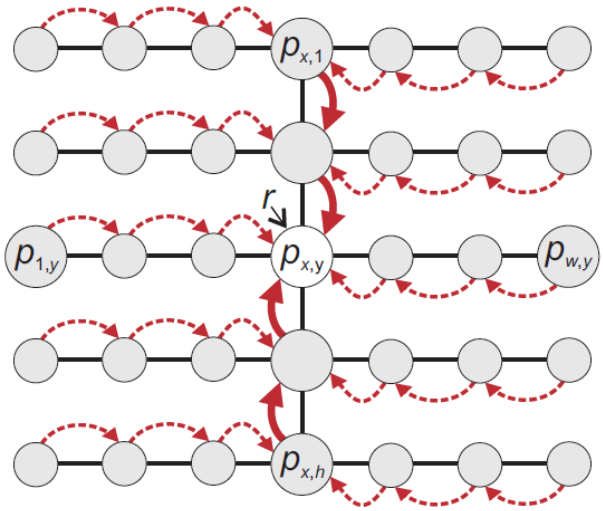


Fig. 3. Compute optimal cost from all horizontal lines.

With input image of size w × h, we need to compute several (w × h × D) vector. From equation (4), this takes a complexity of O(w × h × $D^2$). To improve this, we further simplify equation (4) to equation (8), which has a complexity of O(w × h × D)

$$l(p,d) = m(p,d) + min(l(q,d), l(q,d-1)+P1, l(q,d+1)+P1, \min_{i \in D} l(q,i) + P2) \quad (8)$$

## III. Comparison Metrics

For each image pair, we compute 3 disparity maps: (1) Left view to right view using SSD method. (2) Right view to left view using SSD method. (3) Left view to right view using DP method. Besides visual differences and artifacts, (1) and (3) are compared with ground truth using two comparison metrics. One is histogram correlation (**C**). It is defined as the correlation of histograms of two images,[10,11] in the range of 0-1, while 0 represents the lowest similarity and 1 represents the highest similarity. Histogram gives us the intensity distribution of an image. In our case, it is a good way to tell if a disparity map contains similar overall pixel shift compared to ground truth.

The second metrics is normalized averaged difference (**Z**), which is defined by the average of absolute disparity difference of all pixels, normalized by the bound of disparity values D. For example, if disparity map A and ground truth B has an averaged difference of 3, while the bound of disparity is 30, **Z** will be 0.1. This normalization is to offset the scale difference introduced by image scale and disparity scale. This metrics is a pixel-wise comparison of two disparity maps, a smaller **Z** value means a smaller disparity error.

## IV. Results and Discussion

### A. Jadeplant

For the Jadeplant inputs, both SSD and DP methods produce good results, but there are distinct differences. SSD results have more detailed features as well as more noises. DP result is smoother, especially in the featureless background, however, details are lost, especially where there are occlusions and overlaps. In the black box regions, there are large color blocks compared to ground truth. **C** and **Z** are 0.60 and 0.29 for SSD method, 0.51 and 0.21 for DP method. It shows that SSD method performs better in overall disparity distribution, which is not surprising since it looks very close to the ground truth. DP method performs better in terms of averaged error, because the noises in SSD result introduce large error when we compare it to the ground truth pixel-wise.

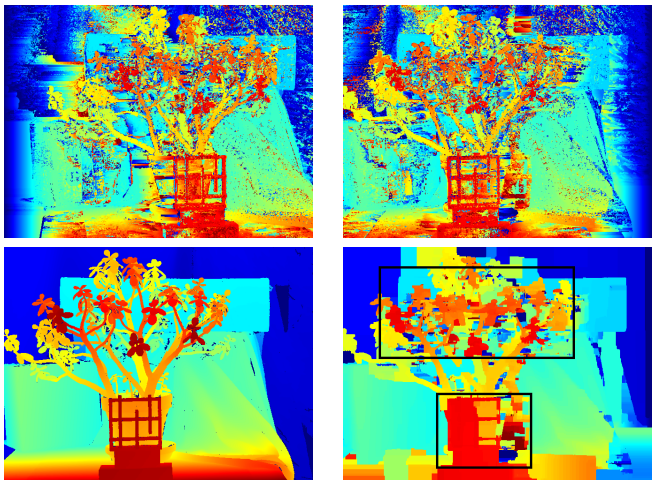

Fig. 4. Jadeplant image set. Top left : left view, SSD. Top right: right view, SSD. Bottom left: ground truth. Bottom right: left view, DP. Differences are marked with black boxes.

### B. Newkuba

Similar to Jadeplant inputs, SSD results are more noisy, while DP result lose more detailed features. What different is, in this inputs set, the DP result matches well with ground truth (expect it does not show the tripod and lamp rack), while the SSD results are almost unrecognizable. **C** and **Z** are 0.50 and 0.35 for SSD, 0.75 and 0.18 for DP. DP result outperforms SSD result in both evaluation metrics, with high pixel-wise accuracy as well as overall discrepancy compared to ground truth.

### C. Classroom2

The Classroom2 image set differ from the other two in which it almost has no fine features, but large featureless regions. The SSD results have noises in the whole image, while DP result shows good match to ground truth, except region in the middle and close to the edges. **C** and **Z** are 0.42 and 0.58 for SSD, 0.77 and 0.18 for DP. This is the input set where two methods show largest performance difference.

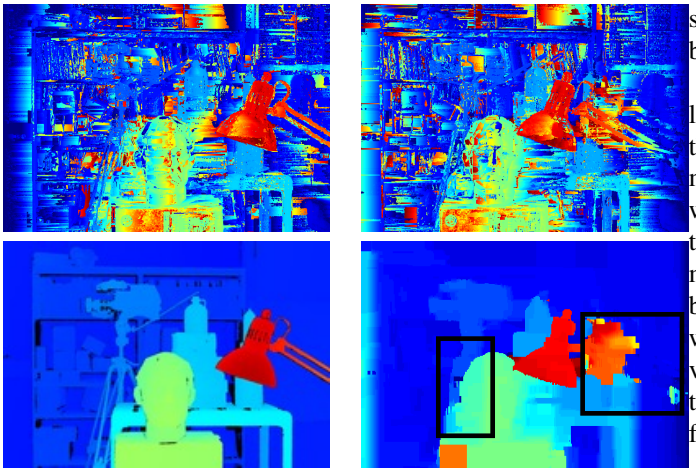

Fig. 5. Newkuba image set. Top left : left view, SSD. Top right: right view, SSD. Bottom left: ground truth. Bottom right: left view, DP. Differences are marked with black boxes.

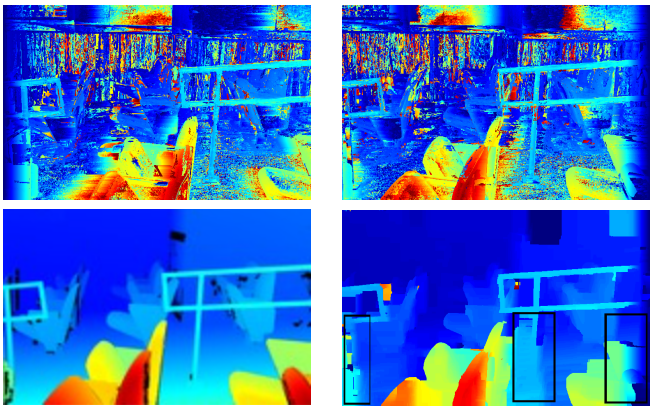

Fig. 6. Classroom2 image set. Top left : left view, SSD. Top right: right view, SSD. Bottom left: ground truth. Bottom right: left view, DP. Differences are marked with black boxes.

## V. Comparison of Methods and Issues

As is discussed in the last section, SSD method preserves more detailed features, but introduces more noises, while DP method provides better smoothness, but failed to show some fine features. In general, this is because SSD method is a local block matching method, while DP is a global method.

For SSD method, we assign disparity based on local similarity. It is hard to get perfect pixel-to-pixel match by only comparing block intensity, especially when there are large featureless regions. What makes this worse is, the disparity value for each pixel is independent, so we can have neighboring pixels with out-of-order disparity values. There are advantages of SSD method. It is fast O(w $\times$ h) and easy to implement, and the above mentioned drawbacks can be improved by tuning the SSD window size.

For DP method, the disparity of each pixel is correlated with not only its horizontal and vertical neighbors, but every other pixel in the image through propagation of dynamic programming, so the smoothness is constrained. Also, this propagation improves its ability to handle occlusions. Although we lose some fine features, the overall performance of DP method is better than SSD method.

The three inputs set are chosen as examples of (1) Featureless foreground. (2) Featureless background. (3) Featureless in the whole scene. As expected, DP method outperforms SSD method in featureless regions. And it also works surprisingly well when the fine features are in the background (Newkuba). I think this is because the smoothness term "blurs" the disparity map by enforcing continuity. When fine features are in the background, it does not matter because disparities are smaller when the objects are further (so they are also closer in value). But when the fine features are in the foreground, the smoothness term competes with data term, making small features unrecognizable.

To further improve the results, we can tune the parameters (k for SSD method, and P1,P2',P3, T and $\lambda$ for DP method). We can also compare both left to right and right to left results to identify occluded pixels, and overwriting those pixels with smooth and meaningful values. [6]

## References


[1] Bebeselea-Sterp, E. , Brad, R. and Brad, R., A Comparative Study of Stereovision Algorithms. International Journal of Advanced Computer Science and Applications, 2017.

[2] Zhou, X. and Boulanger, P., Radiometric invariant stereo matching based on relative gradients, 19th IEEE International Conference on Image Processing, 2012.

[3] Venkateswar, V., Chellappa, R., Hierarchical stereo and motion correspondence using feature groupings. Int J Comput Vision 15, 245–269, 1995.

[4] Harlyn Baker, H. and Thomas O., B., Depth from edge and intensity based stereo. In Proceedings of the 7th international joint conference on Artificial intelligence, 1981

[5] Veksler, O., Stereo correspondence by dynamic programming on a tree, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[6] Bleyer, M. and Gelautz, M., Simple but Effective Tree Structures for Dynamic Programming-Based Stereo Matching.Int Conf Comput Vis Theory and Appl. 2008.

[7] Hong, Li. and Chen, G., Segment-based stereo matching using graph cuts, Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004.

[8] Y. Boykov, O. Veksler and R. Zabih, Fast approximate energy minimization via graph cuts, in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001.

[9] Piazza post @544, https://piazza.com/class/kjliq19wrwi2rj?cid=544, Accessed: Apr. 20, 2021

[10] OpenCV, histogram comparison, https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html, Accessed: Apr. 20, 2021

[11] OpenCV, histograms, https://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html, Accessed: Apr. 20, 2021