AE640 Autonomous Navigation (2018/19)

PROJECT REPORT

A Stereo Matching Algorithm with an Adaptive Window

Submitted to Prof. Mangal Kothari
on 18th April 2019

Group Members :
Alok Ranjan (160085)
Mahesh Saboo (160595)
Vasu Bansal (160776)

Our project is based on '*A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*' by *Takeo Kanade* and *Masatoshi Okutomi.*
Link for the same is :
https://www.researchgate.net/publication/224749504_A_stereo_matching_algorithm_with_an_adaptive_window_Theory_and_experiment

Source code and more results for our implementation can be found at :
https://github.com/mssaboo/Stereo-Matching

Presentation can be found at :
http://bit.ly/AE640ProjectPPT

Summary of the paper : The common problem involved in stereo matching by computing correlation or sum of squared differences (SSD) is that of selecting an appropriate window size. It must be large enough, so that enough intensity variations are covered for reliable matching. If it's too large, then it may have projective distortions. If window size is too small it will give poor disparity estimate as it will not be able to cover intensity variations because the signal or the intensity variation to noise ratio is low.

Large window size may cover a region in which the depth of scene points varies and the position of maximum correlation or minimum SSD may not represent correct matching due to this reason. *Thus a window size must be selected adaptively, based on local variations of intensity and disparity.*

The paper by Takeo Kanade, presents a method to appropriately select window size by *evaluating local variations of intensity and the disparity*. A statistical model which enables one to assess how the variations in intensity and disparity within a window affects the uncertainty of disparity estimate at the centre point of window is used. This enables us to devise a method which searches for a window that estimates disparity with the least uncertainty for each pixel of the image. The method controls both the size and the shape (rectangle) of the window.

This adaptive-window method is embedded in an iterative stereo-matching algorithm. It starts with in initial estimate of the disparity map. Then the algorithm iteratively updates the disparity estimate for each point by choosing the size and shape of a window until it converges.

Iterative Stereo Algorithm with an Adaptive Window :

1. Start with and initial disparity estimate $d_0(x, y)$.
2. For each point (x, y), we want to choose a window that provides the estimate of disparity increment having the lowest uncertainty. For the chosen window, calculate the disparity increment(Δd) first and update the disparity estimate by $d_{i+1} = d_i + \Delta\text{d}$.

   Now we need to select a window which would be rectangular and its height and width can be controlled independently giving least disparity uncertainty.

(a) Place a small 3 x 3 window centered at the pixel, and compute the uncertainty.

(b) Expand the window by one pixel in one direction, e.g., to the right x+, for trial, and compute the uncertainty for the expanded window. If the expansion increases the uncertainty, the direction is prohibited from further expansions. Repeat the same process for each of the four directions z+, x--, y+, and y- (excluding the already prohibited ones).

(c) Compare the uncertainties for all the directions tried and choose the direction which produces the minimum uncertainty.

(d) Expand the window by one pixel in the chosen direction.

(e) Iterate steps (b) to (d) until all directions become prohibited from expansion or until the window size reaches a limit that is previously set.

Conclusion of paper : The paper has presented an *iterative stereo matching* algorithm using an adaptive window. The algorithm selects a window adaptively for each pixel so that it produces the disparity estimate having the least uncertainty. By evaluating both the intensity and the disparity both the intensity and the disparity variations within a window, one can compute both the disparity estimate and its uncertainty which can be used for selecting the locally adaptive window.

An important feature of the algorithm is that it is completely local and does not include any global optimization. Also. the algorithm does not uses any post-processing or smoothing. The smooth surfaces are recovered as smooth while sharp disparity edges are retained.

The experimental results have demonstrated a clear advantage of the algorithm over algorithms with a fixed-size window both on synthetic and on real images.

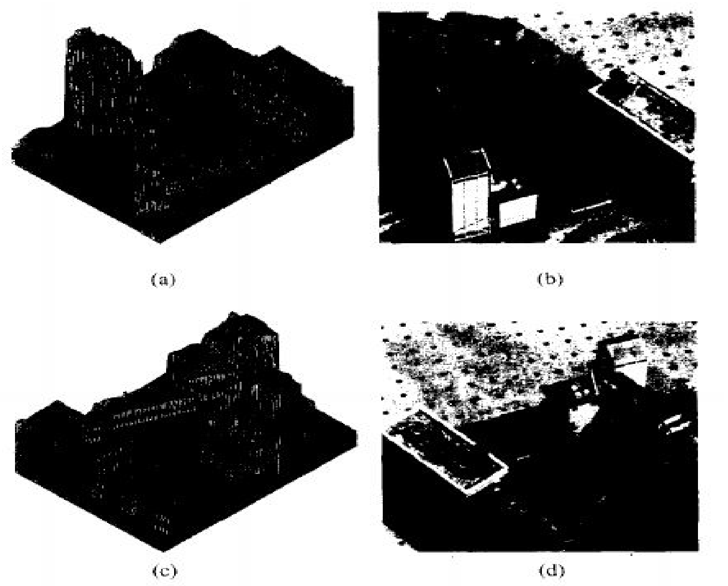


Fig. 17. Isometric plots of the computed disparity map and their corresponding actual view: (a), (b) Isometric plot and corresponding view from the lower left corner; (c), (d) Isometric plot and corresponding view from the upper right corner.

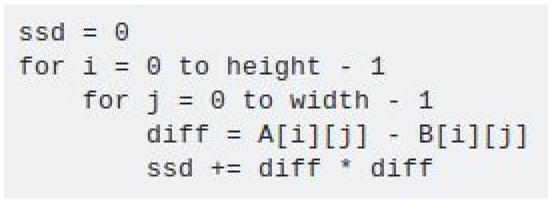
Our Implementation : First we initialised our node. We give our parameters which are our left and right images giving suitable path.

Then we create image publisher object using in-built image_transport library.
Next we read the image and convert it into a ROS message. This step we done twice for both left and right image.

We convert our left and right CV images into grayscale images.
Next we define two matrices, for each image and one more for disparity which is initialized to zero.

We set our several parameters which are minimum and maximum disparity and half-block size. We define our disparity range as the difference of maximum disparity and minimum disparity.

The pseudo-code for calculating SSD is as shown :



```
ssd = 0
for i = 0 to height - 1
    for j = 0 to width - 1
        diff = A[i][j] - B[i][j]
        ssd += diff * diff
```

The general idea is that for matching images the SSD will be small. If you're trying to match two images, where one image is translated by some amount, then you would typically do a brute force approach where you calculate the SSD over a range of x, y displacements and then identify the minimum SSD value, which should then correspond to the best alignment offset.

SSD is generally only used due to its simplicity.

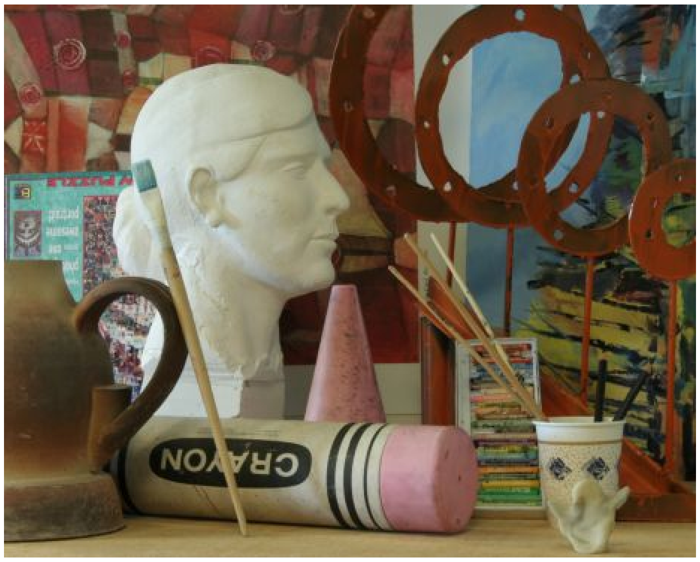
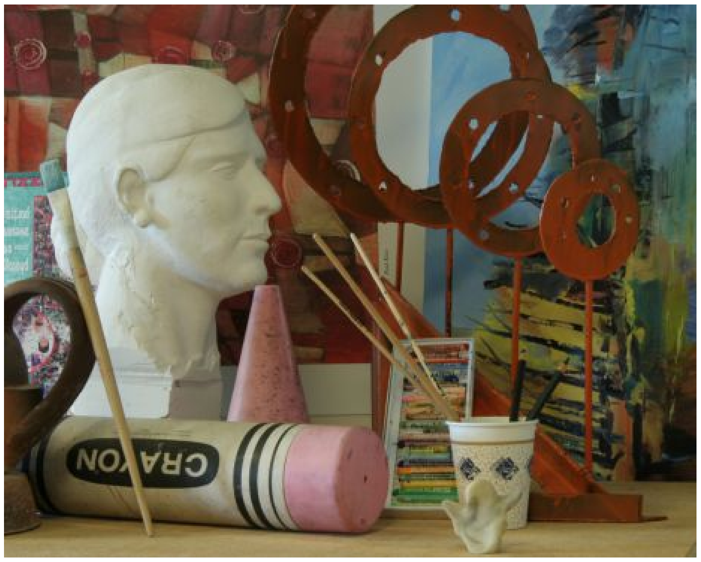
We calculate SSD on values of RGB in corresponding pixels in left and right images and then store it in matrix of SSD value if it is less than the previous value and also update the disparity matrix with the current range of disparity. We then normalise our matrix of disparity values.

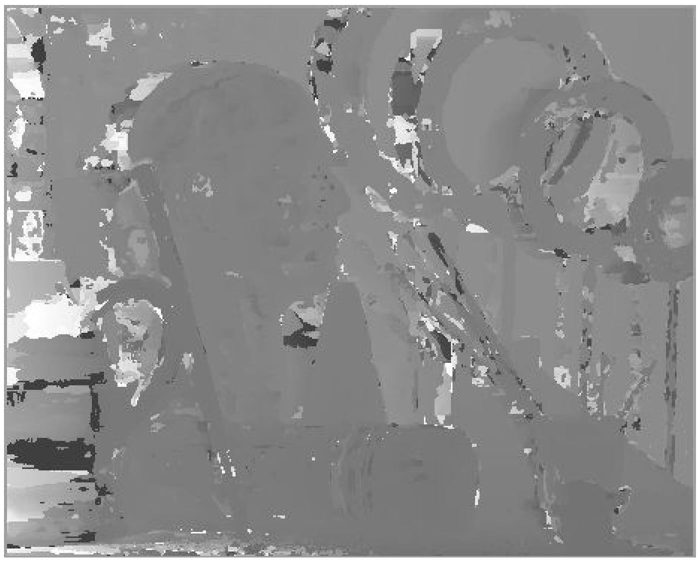
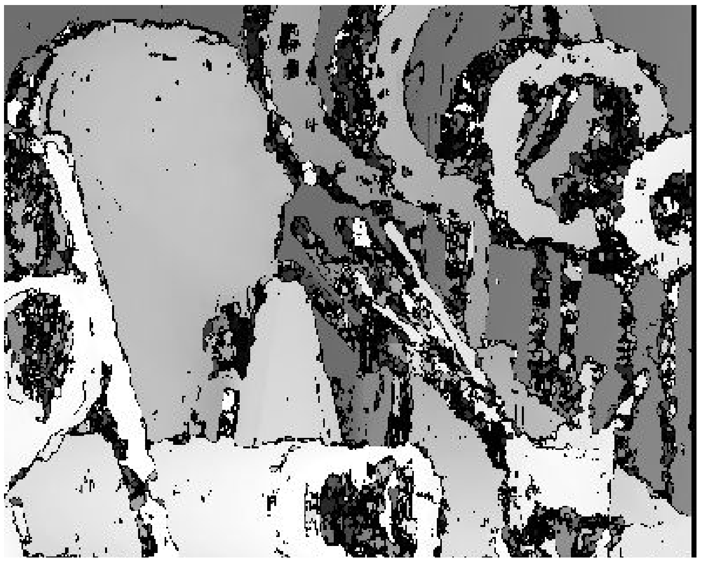
This matrix was converted into image using OpenCV and then was published as ROS message. This was the implementation of SSD in our code for getting disparity matrix.
We also used the inbuilt OpenCV library to get disparity map and compared the results.

Image	Topic
Left image	/image_l

Right image	/image_r
Disparity using SSD	/image_disp_ssd
Disparity using inbuilt function	/image_disp_ib

Results:

	
Left	Right

	
Disparity Using SSD	Disparity Using OpenCv Function