Adaptive Non-Local Means for Cost Aggregation in a Local Disparity Estimation Algorithm

Pedersen , Casper; Nasrollahi, Kamal; Moeslund, Thomas B.

Published in:
Proceedings. 22nd International Conference on Pattern Recognition, ICPR 2014

DOI (link to publication from Publisher):
10.1109/ICPR.2014.422

Publication date:
2014

Document Version
Accepted author manuscript, peer reviewed version

Adaptive Non-Local Means for Cost Aggregation in a Local Disparity Estimation Algorithm

Casper Pedersen, Kamal Nasrollahi, and Thomas B. Moeslund
Visual Analysis of People Laboratory, Aalborg University
Sofiendalsvej 11, 9200 Aalborg, Denmark
Email:kn@create.aau.dk

***Abstract*—The overall method used for determining disparity in a stereo setup is a widely recognized framework consisting of four steps of cost space computation, cost aggregation, disparity selection, and post-processing. In this paper a cost aggregation approach for a typical local disparity estimation method is introduced. The method introduced is built on top of an existing method called Adaptive Support-Weight using this known framework. The introduced method improves Adaptive Support-Weight method by utilizing a larger amount of data inspired by the method of Non-Local Means. The extra data is handled in a way that tries to preserve the location of depth discontinuities in the final disparity map. Experimental results on Middlebury benchmark database show that the proposed method suffers from less artifacts compared to state-of-the-art disparity estimation methods.**

I. Introduction

The task of disparity estimation has been widely researched for a number of years. It is in almost every case presented as a task of finding corresponding pixels between two images. These two images are typically taken by two cameras in a stereo setup where the cameras are aligned horizontally. The task of finding the disparity of a pixel in one image can thereby give the depth of what that pixel is conveying in the scene indirectly if the baseline between the cameras is known. This is utilized in many applications.

There are two main approaches that are used for finding per pixel correspondence between two images. There are global approaches which try to determine the disparity of more than one pixel at a time. This is seen in a lot of works where the focus is on an accurate disparity map, for example in [1], [4], [11]. In [1] dynamic programming is used for each horizontal line in one of the images to determine every pixel's disparity in this line, based on a number of constraints. In [11] graph trees and in [4] segment-trees, which are graph trees that represent group of pixels with similar characteristics, are used for non-local pixel correspondences. These global approaches are often computational heavy because they like [1] involve dynamic programming, graph cut algorithms on so forth. Additionally these global approaches are often not well adapted to a parallel programming scheme which we try to utilize in this paper. This is done as we see a general shift towards more heterogeneous computing where the vast processing power of the GPU is utilized. This is done to move algorithms with a lot of similar instructions closer to real-time.

The other main approach is local approaches which try to determine the disparity of each pixel individually, like [10], [9], [2], [6]. These are typically not as computational heavy as the global approaches as they use a simpler scheme. In two recent works using the GPU, [10] and [9], we see how a local approach is divided into four steps of 1. Cost space calculation 2. Cost aggregation 3. Disparity selection 4. Post-processing. This is a general framework which we also will use. We will go more into depth with what happens in the different steps in the Section II. As presented in [10] such a local approach is well adapted to run on the GPU boosting computational speed of such an algorithm immensely.

The most important step of this four step framework, and where local approaches differ most from each other, is the aggregation step. A lot of works like [8] use support windows also called kernel windows to do this aggregation. In [3] a number of different disparity estimation algorithms are surveyed including [8] of Adaptive Support-Weight which our algorithm takes basis in. The various algorithms in [3] all use different kernel windows for the aggregation step and different methods for weighting pixel inside these kernel windows.

The reason for these different kernel windows are that there are a number of things that makes the task of determining disparity by finding corresponding pixels difficult. Some of the most influential factors are for example image noise, uniform regions in the images and the perspective distortion between the images in a stereo setup.

The perspective distortion is an important factor that we cannot control. Between the two images in a stereo setup, parallax will enforce this perspective distortion between surfaces that do not have the same depth. This means that when using a kernel window in Cost Aggregation, when determining a pixels disparity, we only want the kernel window to cover pixels of the same depth or disparity since we then will eliminate the perspective distortion of the parallax.

This point is tried to be enforced in various ways, in [3] by for example adapting the kernel window size and shape. But in [8] the Adaptive Support-Weight does it by weighting the pixels in a kernel window of a fixed size based on what is called the gestalt principles. These principles weights pixels in the kernel window based on similarity and proximity. This is done based on the thesis that pixel that a similar in color and close to each other are probable to be belonging to the same surface. So instead of finding a kernel window that only covers pixels of the same surface, this method uses a kernel window and then tries to only take the pixels of the same surface into account in this kernel window. This method produces very good results especially by the means of local approaches. The fact that it is a local approach means that it can be very well

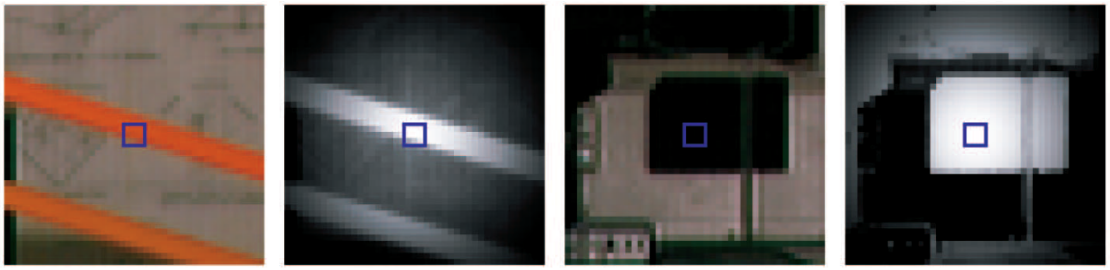

Fig. 1. Parts of two images included by kernel windows and the resulting weights. For the weight images (the second and fourth) brightness corresponds to a larger weight. The blue squares indicate the pixel under consideration [8].

parallelized which we will do in this paper to compare it to our novel approach. Our approach tries to make Adaptive Support-Weight more accurate and robust towards image noise. This is done by taking more data into account in the weighting process which is done for the kernel windows, inspired by the approach of Non-local Means presented in [5]. Furthermore we will show that this is not straight forward and care must be taken to ensure the preservation of edges or depth discontinuities in the image.

The rest of the paper is as follows: first we will in Section II give a short explanation of the Adaptive Support-Weight algorithm before we in Section III introduce our approach called Adaptive Non-Local Means. In Section IV we will discuss some experimental results, and finally conclude the paper in Section V.

II. Adaptive Support-Weight

In [8] the Adaptive Support-Weight algorithm for the Cost Aggregation step in a local disparity estimation method was presented. As mentioned it tries to obey the gestalt principles which are the visual cues of similarity and proximity. These cues are indicators that pixels belong to the same surface. This is desirable for the pixels we take into consideration when doing the Cost Aggregation.

Cost Aggregation is the second step of the framework used here for the local disparity estimation. The first was as mentioned in the previous section Cost Space Calculation which is measuring similarity between the pixels in the reference and target image. This is in [8] done by the Sum of Absolute Differences between the r,g and b channel of the pixel in the reference image and the possible corresponding pixels in the target image. The two images are rectified so correspondence is only in the horizontal direction.

Cost Aggregation uses a set of rectified images and produces a Cost Space which is a 3D volume with the dimensions of the images and a third dimension which represents the disparity range that we want to search. A given entry in the Cost Space corresponds to a pixel location in the references image and a pixel location of a potential disparity in the target image. The kernel windows are what are used to filter each entry in the Cost Space in the step of Cost Aggregation. They take neighboring pixels into account when determining the similarity between pixels. And it is here we as mentioned only want to take pixels of the same surface into account due to the perspective distortion. Another way to phrase it is that we only want to give a high weight to pixels in the kernel windows which are similar in color to the pixels we are considering (the center pixel of the kernel windows) and are close to the pixels we are considering. The pixels we are considering are of course those that correspond to the entry in the Cost Space we are filtering for, a pixel in the reference image and a potential corresponding pixel in the target image.

Mathematically Cost Aggregation can be expressed as:

$$E(p,p_d) = \frac{\sum_{q\in N_p, q_d \in N_{p_d}} w(p,q)w(p_d,q_d)e(q,q_d)}{\sum_{q\in N_p, q_d \in N_{p_d}} w(p,q)w(p_d,q_d)} \quad (1)$$

Where p is the pixel under consideration in the reference image, q is the current pixel in the kernel window of the reference image which we sum over for the entire kernel window which is noted N_p, p_d, q_d and N_{p_d} is the same just for the other image and offset by the disparity, meaning p_d is $(x+d, y)$. $e(q, q_d)$ is the original entry in the Cost Space and $E(p, p_d)$ is of course the filtered Cost Space which is the output from the Cost Aggregation.

This computation is of course done for every x,y and d (disparity). The function w is the weighting function which is done based on the gestalt principles. The calculation of the weight of one pixel in a kernel window can be expressed as:

$$w(p,q) = exp(-(\frac{\Delta c_{pq}}{\gamma_c}) + \frac{\Delta g_{pq}}{\gamma_p})) \quad (2)$$

Where p is the pixel under consideration (center pixel in kernel window), q is the pixel in the kernel window we want to find a weight for, Δc_{pq} is the similarity between pixel p and q measured by the Euclidean distance between the two pixel colors in the CIELab color space, Δg_{pq} is the proximity between pixel p and q measured as the Euclidean distance in pixel coordinates, and γ_c and γ_p are constants.

A visualization of this weighting principle is seen in Figure 1 where the blue box indicates the pixel we are considering in the kernel window which is the image. The second image shows the weighting in this kernel window. Bright gray is a large weight. It is seen how pixel of similar color and close to the center pixel are brighter in the grey-scale images indicating a high weight.

After the Cost Aggregation is done, the third step of the local approach framework is to choose a disparity for each pixel in the reference image. This is done by a simple Winner-Takes-All approach where for each pixel the disparity is chosen as the potential corresponding pixel in the target image with

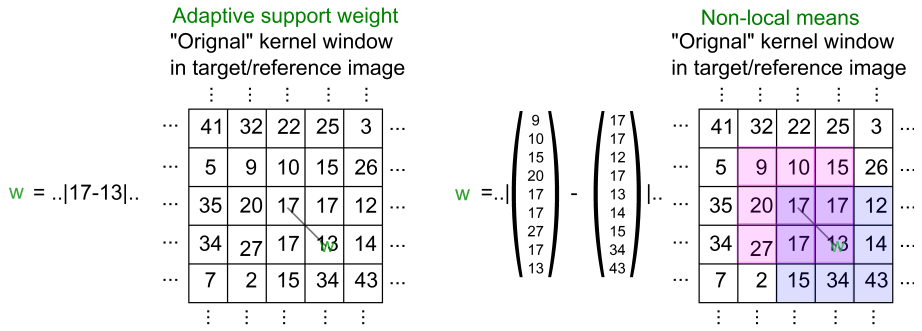

Fig. 2. The process of finding a weight for one pixel in one of the original kernel windows with both the Adaptive support weight algorithm form the last section and the introduced adaptive NLM algorithm.

the lowest Cost Space value or similarity measure. No post-processing is done for the Adaptive Support-Weight algorithm in [8], though it could possibly be improved by some post-processing. We will in our approach also omit this step and concentrate on the Cost Aggregation.

III. Adaptive Non-Local Means for Cost Aggregation

We will divide the explanation of our algorithm into two steps expanding the Adaptive Support-Weight algorithm. First we will adapt theory of Non-Local Means to the algorithm explained in the previous section, and then we will alter this new approach to preserve edge locations in the produced disparity map.

A. Non-Local Means in Cost Aggregation

Our approach uses the same framework as Adaptive Support-Weight. The first step of the Cost Space Calculation is done in the same way where we simply calculate a Cost Space of Sum of Absolute Differences also called a SAD volume. The Cost Aggregation takes as mentioned basis in the approach explained in the previous section. But it is inspired by the presentation of Non-Local Means as presented in [5]. Non-Local Means resembles the method used in Adaptive Support-Weight for aggregation in terms of using a kernel window of fixed size and computing weights based on placement and radiometric features of the pixels. But instead of determining the similarity based on comparing just two pixels, Non-Local Means (NLM) uses more data to give a more reliable result. We adapt this concept which is typically used in image filtering or as in [5] for Super Resolution to the Cost Aggregation scheme.

In Adaptive support weight we used equation 2 for finding the weight of one pixel in one kernel window. In equation 3 we introduce an altered version of equation 2 which we use in our NLM exspansion of the algorithm:

$$w(p,q) = exp(-(\frac{\|R_p - R_q\|_2}{\gamma_c}) + \frac{\Delta g_{pq}}{\gamma_p})) \qquad (3)$$

Where R_q and R_p are vectors containing values of the pixels in an area around pixel q and p, CIELab values. This area is a kernel window. We now have a process of finding weights for two kernel windows where we also use two kernel windows for each of these. We therefore choose to call the kernel windows we are finding weights for the original kernel windows which

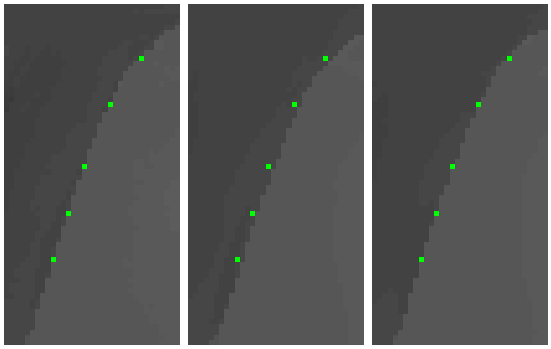

Fig. 3. Left) Close up of an edge in the disparity map produced by the Adaptive Support-Weight algorithm with a kernel window size of 35x35. The edge from the texture image is preserved and a number of pixels laying right on the bright side of the edge is marked by green. Mid) Close up of the same edge in the disparity map produced by the NLM algorithm with an original kernel window size of 35x35 and a 7x7 kernel window for finding weights. Right) Close up of the same edge in the disparity map produced by the Adaptive NLM algorithm with an original kernel window size of 35x35 and a 7x7 kernel window for finding weights. The pixels with the same coordinates are highlighted in all the three images.

are the ones we use to filter the SAD volume. This means that the process of finding the weights for one of the original kernel windows can be depicted as in Figure 2. The example is for the gray scale case.

For comparison, the process of finding one weight in one of the original kernel windows is shown for both algorithms. We see how only the pixel we are finding the weight for and the pixel under consideration is used in Adaptive Support-Weights. But in our NLM approach two vectors are extracted by two kernel windows around the pixels. When this has been done for every pixel in the original kernel windows the result is having a weight for each pixel in the two original kernel windows and filtering the entry in the SAD volume, excactly like i Adaptive Support-Weight. If the size of the kernel windows we use to extract the vectors is set to zero in radius, the NLM approach actually reduces to the Adaptive Support-Weights algorithm.

This algorithm is of course not as computational fast as Adaptive Support-Weights since it must do these extra reads to extract a vector instead of just one read per pixel to obtain its value. But the gain should be that the weight computation process also becomes much more reliable since more data in taken into account. It will not be as vulnerable to noise and

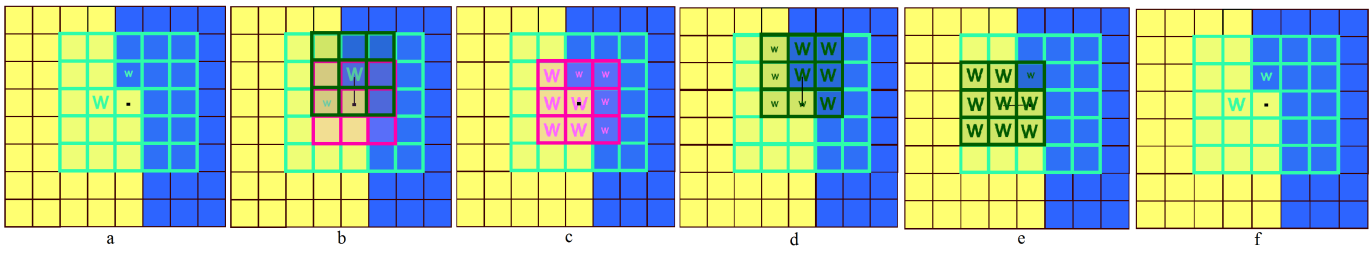

Fig. 4. a) Example of weights obeying the gestalt principles in one of the original kernel windows, b) Weights of original kernel window using the NLM. NLM uses two additional kernel windows to extract vectors of pixel color which are subtracted in part of finding the weight, c) Weights in the kernel window centered at the pixel under consideration. Pixels on the same side of the edge as the center pixel has a large weight, d) Weights in the kernel window centered at the pixel we are currently finding the weight for in the original turquoise kernel window. The center pixel is on the opposite surface than the pixel under consideration, e) Weights in the kernel window centered at pixel we find the weight for in the original turquoise kernel window. Pixels on the same side of the edge as the center pixel has a large weight, and f) The two resulting weights in the original kernel window from the center pixels of Figure (d) and (e). Obtained by weighting the subtracted pixels in the kernel windows.

thereby more accurate in noisy regions.

Figure 3(left) shows a zoom of a disparity map computed with Adaptive Support-Weight, and Figure 3(mid) shows the same region computed with the NLM approach. The green pixels are to indicate where the edge is in the original image. We see comparing Figure 3(left) and Figure 3(mid) that the NLM apporach can in some cases have problems around depth discontinuities or edges in the image. This can be explained in theory by the extra data being considered when calculating the weights. A pixel right on either side of an image edge should have high weights in its original kernel window corresponding to that side. This will obey the gestalt principles by only considering pixels of the same color and thereby of the same surface. An example of this can be seen in Figure 4(a).

We see how the weight of the pixel on the same side of the edge as the pixel under consideration is high, indicated by the large ”W”, in one of the original kernel windows shown in turquoise. This is of course due to its similarity in color to the pixel under consideration which is indicated by the dot. This is opposed to the pixel with the small ”w”. It has a low weight because it lies on the other side of the edge and thereby has a different color. It is this principle that Adaptive Support-Weight tries to make use of. But in the NLM approach we use more pixels when finding the weights. This makes the weights more accurate and consequently the disparity estimation more accurate but in Figure 4(b) we see what can go wrong close to an edge.

Pixels can obtain a wrong weighting with regards to the gestalt principles. This is because pixels from the other surface on the other side of the edge are taken into consideration. In Figure 4(b) it can be seen how the kernel window for the pixel under consideration, the pink one, contains four blue pixels even though it itself lies on the yellow surface. This means that the pixel we want to find a weight for, the center of the green kernel window, obtains a large weight indicated by the large ”W”. This is because the two kernel windows we subtract to find the weight, the pink and green, are very similar. We also see the pixel under consideration is not similar to pixels on the same surface as itself, i.e. the pixel with the small ”w” which would have a kernel window that only contains one blue pixel (this kernel window is not shown in Figure 4(b)). These wrong weightings with regard to the gestalt principles for the original kernel window can result in the pixel being misclassified and getting a disparity that results in it to lie on the wrong side of the image edge.

To improve the novel NLM method further we therefore introduce Adaptive NLM. In this new algorithm we try to resolve the above mentioned problem around edges by also utilizing the gestalt principles when subtracting the two kernel windows when finding the weights with equation 3.

1) Adaptive NLM: With NLM we have introduced using kernel windows when determining the weights. We want the data of these kernel windows, in the example of Figure 4(b) the pink and green ones, to obey the gestalt principles as well. Like the original turquoise kernel window does in Figure 4(a) and NLM fails to do in Figure 4(b). Pixels inside the two kernel windows will be weighted with regards to their similarity in color and spatially closeness to center pixel of the kernel window. The center pixel of the two being the pixel we want to find a weight for in the original kernel window (center of green kernel window in Figure 4(b)), and the pixel under consideration (the center of the pink in Figure 4(b)).

One must not be confused by the fact that we are now using weights of two kernel windows to determine the weights of one of the original kernel windows. The process is of course done for every pixel in both original kernel windows, the one in the target image and the one in the reference image. By weighting pixels in the kernel windows of the NLM approach, we solve the problem at edges described earlier.

It is here shown in different steps. In Figure 4(c) we see that the pixels inside the kernel window of the pixel under consideration are weighted high if they are of similar color, i.e. on the same side of the edge [1].

The same is the case for the kernel window belonging to the pixel we want to find the weight for in the original kernel window, as seen in Figure 4(d). By examining both Figure 4(c) and Figure 4(d), we notice that out of the pixels that are to be subtracted only the centers share a large weight and the two centers are not of the same color. This results in a low weight in the original kernel window when subtracting the two kernel windows as can be seen in Figure 4(a).

[1]The weighting of the spatial proximity is not taken into account when illustrating the weights as it is not the important point here

Figure 4(e) shows the same as Figure 4(d) but for a different pixel inside the original kernel window. Here some of the pixels that have a large weight are to be subtracted from pixels which also have a large weight from Figure 4(c). This will result in a large weight in the original kernel window since the pixels sharing large weights in the kernel windows are of the same color.

The results explained when subtracting the windows of Figure 4(d) and Figure 4(e) from the kernel window in Figure 4(c) and using equation 3 can be seen in Figure 4(f).

We see that because of the weighting inside the kernel windows the final weighting of the original kernel window stays true to the gestalt principles like in Figure 4(a) and unlike in Figure 4(b). It is shown directly in Figure 3(right) where we see the same region of a disparity map as we saw in Figure 3(left) and Figure 3(mid) but computed with the Adaptive NLM approach. We see that the edge of the disparity map lies at the green pixels as was the case in Figure 3(left). The new weight determination process can be described mathematically as:

$$w_{\text{ANLM}}(p,q) = exp(-(\frac{\|R_p - R_q\|'_2}{\gamma_c}) + \frac{\Delta g_{pq}}{\gamma_p}))$$
$$\|R_p - R_q\|'_2 = \frac{\sqrt{\sum_{p' \in R_p, q' \in R_q} w_{\text{asw}}(p,p') w_{\text{asw}}(q,q')(p'-q')^2}}{\sum_{p' \in R_p, q' \in R_q} w_{\text{asw}}(p,p') w_{\text{asw}}(q,q')} \quad (4)$$

Where w_{asw} is the weighting from Adaptive Support-Weight in equation 2, p' and q' are the pixels inside the kernel windows used for finding weights. The rest of the entities have been introduced previously.

This new weighting can now be used directly in equation 1 which was the filtering of the SAD volume with what we have called the original kernel windows but with new weights.

IV. Experimental Results

We will just look at results comparing our Adaptive NLM method to what it took basis in, the Adaptive Support-Weight algorithm. This is to show that making a robust weight computation will improve the final disparity map. The image pair which was tested on is seen in Figure 5 (a) and (b). The left image is chosen as the reference image so the results shown in this section is disparity maps of this image.

We have simply executed the two algorithms on the image pair with parallel implementation which has been done. We have utilized the OpenCL framework to execute the algorithms on the GPU. An algorithm like Adaptive NLM would not be relevant to execute as a CPU implementation since number of pixel reads is so big that it would take in the matter of hours to produce a disparity map with the algorithm given a reasonable image resolution. Using the power of modern GPU and tools like OpenCL is what is starting to make algorithms like Adaptive NLM relevant.

In Figure 5(c) we see the disparity map computed by Adaptive Support-Weight of [8] with a kernel window size of 35x35. We see that with this kernel window size there are some artifacts which a the black areas where the disparity is not rightly determined. However, if we look at the result of the Adaptive NLM in figure 5(d) computed with the same original kernel window size and with a 5x5 kernel for the weight computations in Cost Aggregation these areas of misclassified disparity are reduced.

This is of course due to the fact that weighting process is more robust due to the extra data used resulting in a better obedience with regards to the gestalt principles. The same point is seen comparing Figure 5(e) and Figure 5(f) where a larger original kernel window size is used. Because of this the number of faulty disparity are lower. But we see the improvement again in Figure 5(f) which has a lower number of artifacts.

Finally, we show the results of comparing the proposed Adaptive NLM algorithm against two more state-of-the-art aggregation methods on Middlebury benchmark database [7]. These two methods are taken from [11] and [6]. There results are shown in Figure 6. It can be seen that the results obtained by our proposed algorithm suffers from less artifacts compared to these two methods as well as the method of [8] (Figure 5).

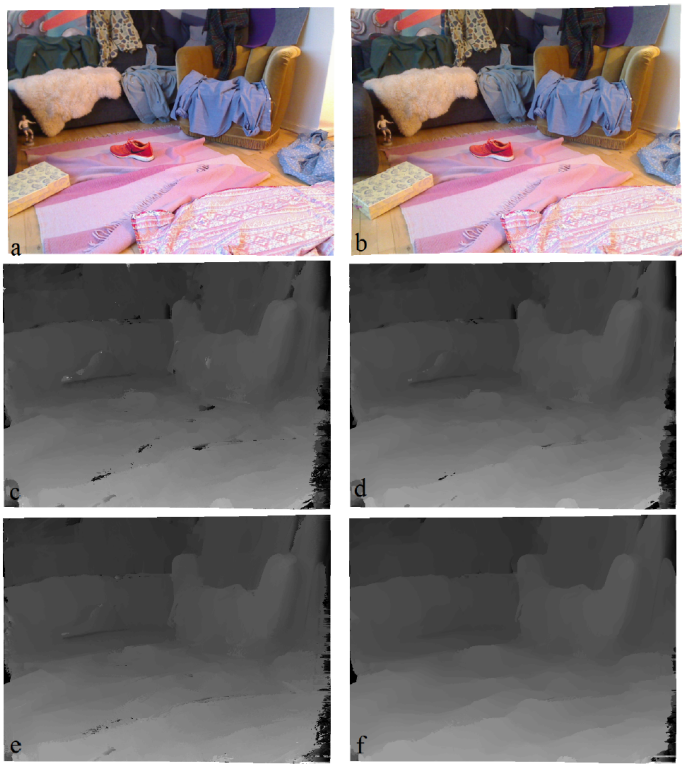

Fig. 5. a-b) The horizontal image pair we will do Depth map Estimation testing on, c) Disparity map computed by Adaptive Support-Weight (Kernel window size 35x35) [8], d) Disparity map computed by Adaptive NLM (Original kernel window size 35x35 and weight kernel window size 5x5), e) Disparity map computed by Adaptive Support-Weight (Kernel window size 75x75) [8], f) Disparity map computed by Adaptive NLM (Original kernel window size 75x75 and weight kernel window size 5x5).

V. Conclusion

Inspired by the method of non-local means this paper has introduced a method for Cost Aggregation for disparity estimation in a stereo setup. It has been shown that using a kernel window to extract the data which is used to calculate

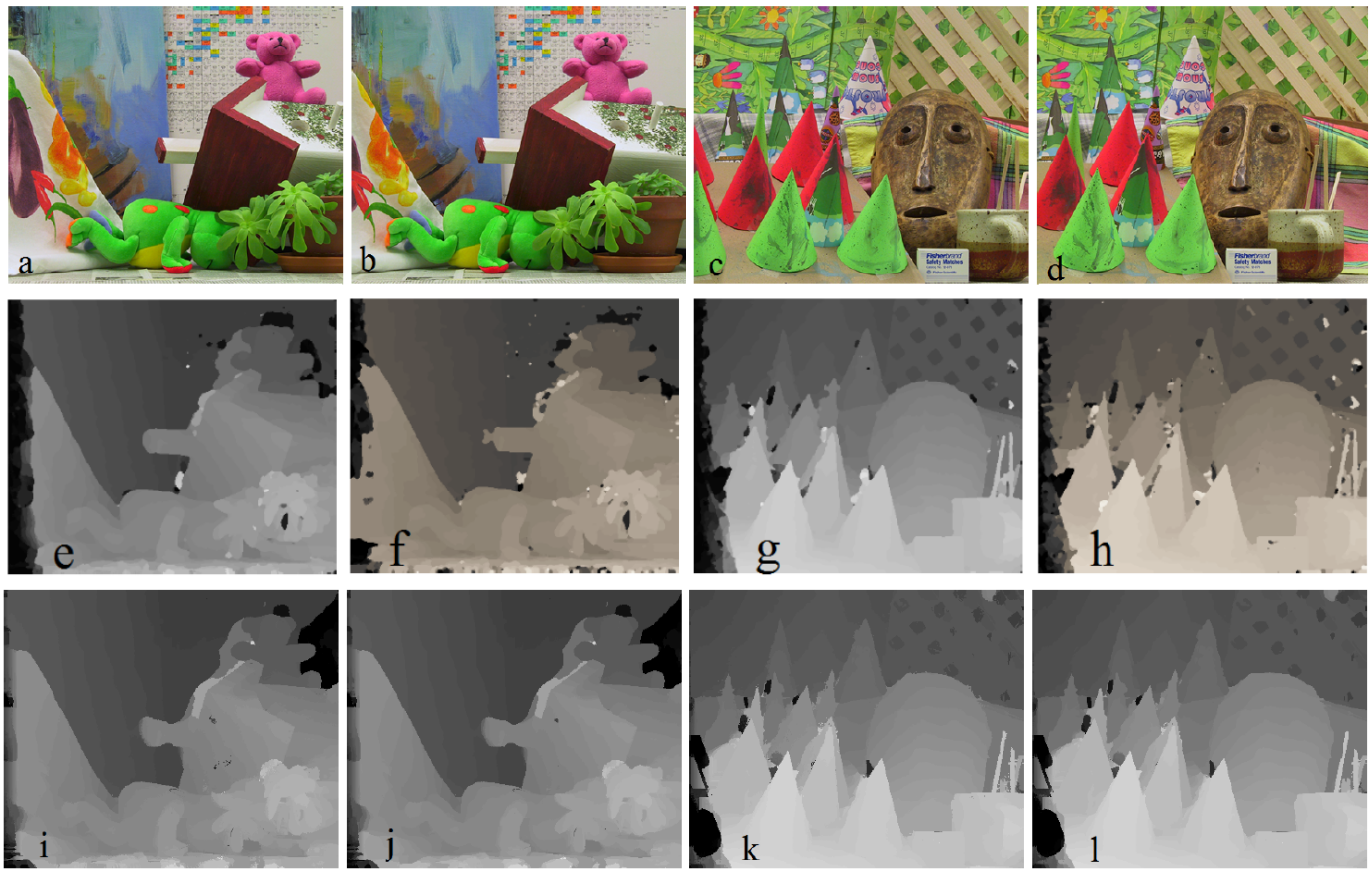

Fig. 6. Comparing the results of the proposed algorithm against two more state-of-the-art methods using two sample pair images from Middlebury benchmark database [7]: a-b) pairs used for calculating the disparity of the first sample, c-d) pairs used for calculating the disparity of the second sample, and disparity maps obtained by: a local guided image filter based cost aggregation taken from [6] (e and g), a non-local cost aggregation method taken from [11] (f and h), the proposed algorithm using a 35x35 kernel (i and j), and the proposed algorithm using a 75x75 kernel (k and l). The results of (e)-(h) are taken from [11].

the weights of the kernel window used for Cost Aggregation is preferable over just using single pixels. It has also been shown that this data must also be selected on the basis of the gestalt principles to ensure that pixels do not obtain the wrong disparity around edges. Experimental results on Middlebury benchmark database show that the proposed method suffers from less artifcats compared to state-of-the-art disparity estimation methods.

ACKNOWLEDGMENT

The authors would like to thank Thorbjoern Vynne from Solution57 ApS for his support towards this work.

REFERENCES

[1] Antonio Criminisi, Jamie Shotton, Andrew Blake, Carsten Rother, and Philip H.S. Torr, *Efficient dense-stereo and novel-view synthesis for gaze manipulation in one-to-one teleconferencing*, International Conference on Computer Vision, 9th IEEE, 2003.

[2] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, *Linear stereo matching*, International Conference on Computer Vision,13th IEEE, 1708-1715, 2011.

[3] Minglun Gong, Ruigang Yang, Liang Wang, and Mingwei Gong, *A performance study on different cost aggregation approaches used in real-time stereo matching*, International Journal of Computer Vision, 75(2): 283-296, 2007.

[4] Xing Mei, Xun Sun, Weiming Dong, Haitao Wang, and Xiaopeng Zhang, *Segment-Tree based Cost Aggregation for Stereo Matching*, Computer Vision and Pattern Recognition, IEEE Conference on, 313-320, 2013.

[5] Matan Protter, Michael Elad, Hiroyuki Taleda, and Peyman Milanfar, *Generalizing the non-local means to super-resolution reconstruction*, Image Processing, IEEE Transactions, 18(1): 36-51, 2009.

[6] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, *Fast cost-volume filtering for visual correspondence and beyond*, Computer Vision and Pattern Recognition, IEEE Conference on, 3017-3024, 2011.

[7] D. Scharstein and R. Szeliski, *Middlebury stereo evaluation*, http://vision.middlebury.edu/stereo/eval/

[8] Kuk-Jin Yoon and In So Kweon, *Adaptive support-weight approach for correspondence search*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 28(4): 650-656, 2006.

[9] Giovanni Visentini, and Amit Gupta, *Depth estimation using Open Compute Language (OpenCL)*, International Conference on Latest Computational Technologies, 2012.

[10] Christian Weigel, and Niklas Treutner, *Flexible OpenCL accelerated disparity estimation for video communication applications*, 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 1-4, 2011.

[11] Q. Yang, *A non-local cost aggregation method for stereo matching*, Computer Vision and Pattern Recognition, IEEE Conference on, 1402-1409, 2012.