Sparse Stereo Disparity Map Densification using Hierarchical Image Segmentation

Sébastien Drouyer[12], Serge Beucher[1], Michel Bilodeau[1], Maxime Moreaud[21], and Loïc Sorbier[2]

[1] Mines ParisTech, PSL Research University, CMM - Centre de morphologie mathématique, France
[2] IFP Energies nouvelles, Rond-point de l'échangeur de Solaize, BP 3, 69360 Solaize, France

Important note: this is a preliminary version of the article sent to ISMM, not the final version. The major differences are that we use an other algorithm to produce our sparse disparity maps and each region can be fit by a quadratic function. Even though results are a little bit poorer than in the final version, we make this version available as it can add some insight on the functionning of our solution.

Abstract. We describe a novel method for propagating disparity values using hierarchical segmentation by waterfall and robust regression models. High confidence disparity values obtained by state of the art stereo matching algorithms are interpolated using a coarse to fine approach. We start from a coarse segmentation of the image and try to fit each region's disparities using robust regression models. If the fit is not satisfying, the process is repeated on a finer region's segmentation. Erroneous values in the initial sparse disparity maps are generally excluded, as we use robust regressions algorithms and left-right consistency checks. Final disparity maps are therefore not only denser but can also be more accurate. The proposed method is general and independent from the sparse disparity map generation: it can therefore be used as a post-processing step for any stereo-matching algorithm.

Keywords: Stereo, hierarchical segmentation, robust regression model, waterfall, disparity map, densification

1 Introduction

One of the main research interest in computer vision is the matching of stereoscopic images, as it allows to obtain a 3d reconstruction of the observed scene. A common practice is to first rectify the pair of images [1], that is to say slightly distort them so that a point of the scene is projected on the same ordinate on both images. This rectification allows to simplify the matching process to a 1D search problem. The difference of abscissa of a point's projection between the rectified pair of images is called disparity. A disparity map estimates the disparity for each pixel of the rectified pair.

Many different algorithms evaluating these disparity maps already exist in the literature. They generally use local [2], semi-global [3] [4] or global optimization methods [10] [11] [12] that minimize matching cost of image features between two images. While some methods, especially the global ones, can produce dense disparity maps, other methods produce sparse disparity maps - where there are some pixels with undefined disparity values. This can happen for several reasons. Disparity values can be removed when a confidence measure, such as texture or uniqueness, is under a certain threshold. They can also be removed when a part of the scene is occluded, either by global and semi-global methods, or by using Left-Right Consistency (LRC) checks[22].

Global disparity evaluation methods often feature disparity propagation in order to generate a complete map [13] [14], but these propagation techniques are directly linked to the method and are not generalizable.

Apart from diffusion [15] and interpolation [16], a few general propagation methods exist in the literature. From a disparity map where only edges are matched, VMP [17] diffuses disparity values by using predefined masks and a voting process. FGS [18] has been used by the OpenCV library [19] for propagating LRC checked disparity values using an edge-preserving diffusion model.

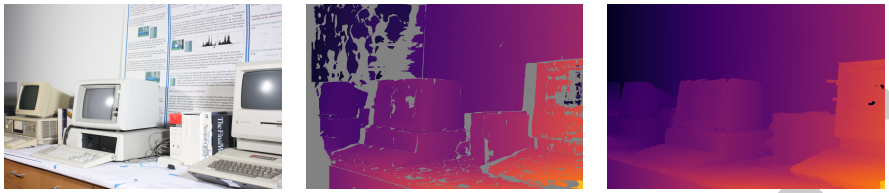
This work proposes a new densification method that is able to produce a denser and more accurate disparity map from an initial sparse disparity map. We called our method TDSR for Top Down Segmented Regression. From a coarse segmentation of the image, we try to fit each region's disparities using robust regression models. If the fit is not satisfying, the process is repeated on a finer region's segmentation.

This coarse to fine approach allows the algorithm to rely on general purpose parameters. Since each region is processed separately, the algorithm can be easily parallelized to improve efficiency. As we use linear regression models, it is possible to get not only the disparity values but also normals in the disparity space, which can be convenient for multi-view stereo algorithms and object recognition. Finally, the proposed method is general and independent from the sparse disparity map generation: it can therefore be used as a post-processing step for any stereo-matching algorithm.

2 Top Down Segmented Regression method

2.1 General concept

We have two rectified stereo images (left and right) depicting the same scene (see example in figure 1a). We also have a sparse disparity map between the left and the right images obtained using an existing stereo matching algorithm (see figure 1b). Large areas of the disparity map can be undefined (gray pixels), and defined areas can contain errors. Furthermore, we know the camera matrix: if a point is defined in the sparse disparity map we can know its 3D relative position in the scene, and vice versa. From these information, we want to obtain a complete and accurate disparity map (see figure 1c).



(a) Left image (b) Sparse disparity map (c) Objective

Fig. 1: **Objective : sparse disparity map to complete one**. Adirondack pair of the Middlebury dataset[23].

We will suppose that the scene is a collection of different **objects**. Each object's surface is composed of one or multiple **simple shapes**. Those simple shapes can be planes, curves, tubes or spheres: all the 3D points inside a shape can be modeled using a bivariate quadratic polynomial, that we will call **model**.

Our method will therefore try to separate all the simple shapes in the reference (left) image. If each region is a simple shape, we can obtain a 3D point cloud estimate of the corresponding region from the sparse disparity map (by converting each disparity value to a 3d point) and modelize it by a bivariate quadratic polynomial. The model can then be used to correct existing disparity values and fill undefined values in the region, by converting back each 3d point of the model to a disparity value. However, this is not automatically feasible by a segmentation algorithm, so we will instead rely on hierarchical segmentation.

The hierarchical segmentation of an image can be represented by a **partition tree**. The concept is very similar to the binary partition tree[29] presented by Salembier and Garrido, except that a node can have more than two children. The top node represents the whole image, its children represent the most important regions of the image according to the algorithm used, their children represent secondary regions, and so on until a node can't be separated into more sub regions.

An adapted segmentation can't be obtained by simply choosing a specific level of the partition tree: nodes at the lower levels will produce a very coarse segmentation of the image, nodes at the higher levels will produce an over-segmentation. At the middle, some simples shapes are over-segmented and others are merged with neighboring shapes.

We will therefore proceed the following way. We start from the top node, and try to modelize its associated 3D point cloud. If the modelization is satisfying or if the node doesn't have any children, we stop here, associate the model to the node, and delete all the node's children. If it is not satisfying, we retrieve the node's children, and apply the same process individually to each node.

If we combine all the leafs of the resulting tree, we can obtain a **modeled segmentation** of the image, where each region is associated to a model. Some post-processing are done on this modeled segmentation that we will describe in section 2.6. It can then be converted to a complete disparity map.

In order to concretize this general concept, we need to define the following:

– How do we obtain the sparse disparity map S?
– How do we construct the partition tree H?
– How do we modelize a region?
– What are the conditions defining a satisfying modelization?

2.2 The sparse disparity map S

In order to compute a complete disparity map, we first need an initial sparse disparity map. The quality of our complete disparity map is highly correlated to the quality of the initial disparity map, so choosing an accurate algorithm at this stage is crucial.

We chose to compute this disparity map between the left and right image using the MGM[4] algorithm as it evaluates high quality disparity maps. Chosen parameters are shown in table 1. Except for the window size that we increased for greater robustness, we used the ones proposed in [5]. Census[20] is used as the distance function because it is robust to change in illumination and exposure. Sub-pixel precision is obtained using the V-fit algorithm[21]. Inconsistent disparities are removed using the Left-Right Consistency (LRC) check[22] with threshold set to 1.

Parameter	Value
P1	8
P2	32
Distance function	CENSUS
Window size	9x9
Sub-pixel refinement method	V-fit
Number of propagation directions	8

Table 1: MGM chosen parameters.

2.3 The partition tree H

Obtaining an adapted partition tree for our images is not a straightforward task.

First, the assumption of having simple shapes separated at least at the leafs of the partition tree can be difficult to attain on some images. Segmentation, whether it provides hierarchical information or not, relies on the image gradient. An object occluding another one with similar color can, in the image, have indistinguishable borders with very low or even null gradient.

Secondly, the partition tree must divide the image in a progressive manner. If it does not - for instance a simple shape is merged with another one at a level and divided into twenty parts at the next level of the tree - it can cause our approach to poorly perform in many ways. Since the input disparity map is sparse, it is possible that some regions at the over-segmented level would contain no disparity values making them impossible to modelize. Since regions are much smaller and contain fewer disparity points, the modelization would be more sensitive to errors in the sparse disparity map.

We took these two problems into account when constructing our partition tree. This partition tree will be obtained from the image gradient and defined markers, so we address these issues when computing them.

A first problem that can occur is to have a border with a very progressive change of color between two regions. If we compute a morphological gradient [8] of size 1 in those areas, we will have a very low gradient around those borders, preventing an accurate segmentation. There is a change of color that is occurring, but we need to look at a larger scale. However, we don't want to simply use a morphological gradient with a larger size since we might lose out on small details. That is why we used the multi-scale gradient presented in [6]. We computed this gradient from scale 1 to scale 6.

For markers, we initially used the h-minima transformation of the gradient, with $h = 5$. A second problem is that, when two adjacent regions have similar colors, the gradient can be less than h at some parts of the border between them. This effect is known as gradient leakage [30]. A way to detect these leakage is that markers will get thinner at these locations. Therefore, we want to encourage a split when the markers get thinner. That is why we applied on these markers the adaptive erosion presented in [7] with $\alpha = 0.25$.

We then compute the valued watershed segmentation from the gradient and markers, and apply on it the enhanced waterfalls operator discussed in [9], as it better preserves the important contours than the standard waterfall algorithm [8]. The result is a grayscale image S_h. If $S_h(x, y) = n$ and $n \neq 0$, then the pixel is a border between two or more regions at the level n or higher of the hierarchical segmentation. The higher the level, the coarser the segmentation. See figures 2a 2b 2c.

We finally transform this grayscale image S_h to our partition tree H. The transformation process is illustrated in figures 2d 2e 2f. First, we define $S_h^n = S_h \geq n$ with $n > 0$: it represents the segmentation at the level n. Let's define $N = \max(S_h)$. We then compute for all $n \in [1, N]$, $L_h^n = L(\overline{S_h^n}) + p$, where $\overline{S_h^n}$ is the complement of S_h^n, L is the label transform and $p = 0$ if $n = N$ and $p = \max(L_h^{n+1})$ if $n < N$. Each label in L_h^n can be seen as a node identifier in H. The children of the root node are L_h^N. Due to the very nature of S_h, if $L_h^n(x, y) = L_h^n(x', y')$ then $L_h^{n+1}(x, y) = L_h^{n+1}(x', y')$. A region R is represented by a single label in the level n of the hierarchical segmentation: there will therefore be a single region R_p in the level $n + 1$ that contain the pixels of the region R. We consider that R_p is associated to the parent node of the node associated with the region R.

2.4 How is a region modelized

When we process each node of the partition tree, we want to obtain a model that best reflects the real geometry of the region. For doing that, we have an estimated 3D point cloud extracted and converted from the sparse disparity map which contain errors and bias. We need to answer two questions. First, do we take all the 3D points into account or do we select a specific subset ? Then, how do we compute the model from this subset ?

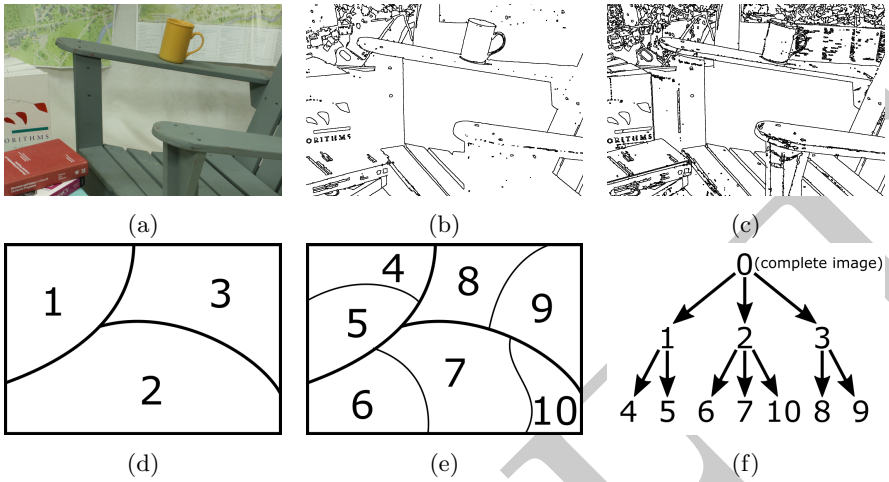


Fig. 2: **Waterfall and hierarchy tree.** (a) Extract of the left image of the Adirondack pair in the Middlebury dataset. (b) Top level of the enhanced waterfall segmentation of (a). (c) Intermediate level. (d) Example of a top level of waterfall segmentation, where each region has been labeled. (e) Example of a lower level of waterfall segmentation, labeled. (f) Hierarchy tree constructed from (d) and (e).

Points selection

A common issue is the well-known fattening effect[24]. For computing each point's disparity, stereo algorithms generally match blocks of pixels [2] [3] [4] instead a single pixels for more robustness. However, around strong edges, the center of the block inherits the disparity of the more contrasted pixels in the block, hence the fattening effect. The worst consequences of this effect take place around occlusions: near the border separating two objects, pixels on the occluded object will inherit the disparity of pixels on the occluding object. Such an issue can seriously impede the modelization process.

Since disparities around borders are not reliable, we decided not to take them into account when computing the model. For doing that, we first separate the region A into two separate regions: A_b being the border of A and A_i being the inside of A. More specifically $A_i = \epsilon(A)$, $A_b = A \setminus A_i$, $\epsilon(A)$ being the morphological erosion of A.

We want here to select all the pixels in A_i whose matching blocks don't intersect with the border. Therefore the retained pixels $\hat{A}_i$ belong to $\hat{A}_i = \epsilon_{\frac{B}{2}}(A)$, where $\epsilon_{\frac{B}{2}}$ is the morphological erosion of size $\lceil \frac{B}{2} \rceil$ (on a square grid) and B is the matching block size.

We will keep all pixels in A_b in the subset where the models will be fitted, as $\hat{A}_i$ might contain an insufficient number of disparity values for an accurate fitting. For instance, a region might have a completely textureless inside resulting

in an absence of disparity values. Though there might be some outliers in the borders, due notably to occlusions, we are relying on the model computation method we will describe later to filter them out.

As a result, we will modelize only the 3D points from the sparse disparity map in the region $\hat{A} = (\hat{A}_i \cup A_b)$.

Points modelization

Now that we have selected the 3D points, we can compute our model. There are however some issues that need to be addressed.

First, we have seen previously that some points from A_b can contain outliers which disparity have been contaminated by a nearby object. An other issue is the presence of small areas with disparity values very different from the ground truth [28]. The problem generally appears on areas where there is very little or repetitive texture, as the stereo matching algorithms generally fail on these areas.

We will therefore use RANSAC[26] to perform a robust regression on these 3D points: if the proportion of outliers isn't too large, they should be automatically excluded from the model evaluation process thanks to RANSAC. First, we will perform a linear RANSAC regression: we try to fit $z = A+Bx+Cy$. If the model is satisfying (see section 2.5), we stop here and the linear equation is affected as the model of the region. Otherwise, we perform a quadratic RANSAC regression: we try to fit $z = A+Bx+Cy+Dx^2+Exy+Fy^2$. If the fitting score (see section 2.5) is better than the linear regression, we choose the quadratic equation as the model of the region. However, we impose a minimum number of points (30) for using the quadratic RANSAC regression as we don't want to create a too complex model from limited data.

If there are no points in the sparse disparity map, we can't compute any model: we will affect to the region an undefined model.

2.5 What are the conditions defining a satisfying modelization

We have constructed a model from a set of 3D points, and now we want to know if the model is satisfying. By satisfying, we mean that we want a large proportion of points that are close to the model. First, we convert the model to disparity values thanks to the camera matrix. For each point i of the set, we know therefore its value in the sparse disparity map d_i and we also know its disparity value according to the model $\widetilde{d_i}$. A point is an outlier if $|d_i - \widetilde{d_i}| > t_d$, t_d being a custom threshold (we chose $t_d = 2$). We define the fitting score s as the percentage of points in the set that are not outliers. A model is satisfying if $s > t_s$, with $t_s = 90\%$.

2.6 Post-processing

We have at this stage a modeled segmentation S_m of our image. To further improve the results obtained, some post-processing are done on S_m. We will note D the disparity map obtained from S_m.

The first problem is that probably some regions have an undefined model because of the absence of points in the sparse disparity map. We have to affect a valid model to these regions, and to do that we will use the modeled segmentation's concept to our advantage. Indeed, these regions are surrounded by regions associated with a model: the main idea is to fill the undefined region with the most probable model in the adjacent regions.

However, since each undefined region might cover multiple simple shapes, we will cut these regions the following way. First, we create a label map ℓ_1 labeling all regions with an undefined model. We create a second label map ℓ_2 labeling all the regions of the watershed segmentation computed in section 2.3. We then compute ℓ_f such as $\ell_f(x) = \ell_f(y) \iff ((\ell_1(x) = \ell_1(y)) \wedge (\ell_2(x) = \ell_2(y)))$. Each region labeled in ℓ_f will then be processed independently.

For each label l and associated area A_l, we define its external border as $B_l = (\delta A_l) - A_l$, δA_l being the morphological dilation of A_l of size 1. We list all different models in $S_m(B_l)$. We want to choose the model that creates the largest consensus across adjacent regions separated to the current region by a low gradient. For doing that, we create a list l_p of all positions where $g(B_l) < \min(g(B_l)) + t_g$, g being the gradient of the left image obtained following the process explained in section 2.3. t_g is a threshold to be defined (we chose 10). We affect to $S_m(A_l)$ the model M that maximize the number of points p in l_p such as $(M(p) - D(p)) < t_d$, t_d being the threshold used in section 2.5.

The labels are processed from the one that contain the lowest proportion of undefined models in B_l to the one that contain the highest.

Once S_m and D are filled, we compute S'_m and D' using the same process than for S_m and D but by inverting the left and right images. We remove values in D that are inconsistent according to the LRC check with threshold set to 1. We also remove values at the same positions in S_m. We then fill the values using the same process as described earlier.

3 Results

We benchmarked our TDSR method using the well-known Middlebury dataset[23]. The dataset contains several stereo-images pairs with their associated disparity map ground truth. An executable is also provided allowing to quantitatively compare disparity maps to their ground truth according to various criteria. We chose the following two commonly used criteria in the Middlebury comparative table: "Bad 2.0", which computes the percentage of disparity values farther than 2.0 from their associated ground truth, and "Avg. error", that computes the average absolute difference between the disparity values and their associated ground truth. These criteria have been separately computed on all values and

only on values where no occlusion is occurring. They were computed on full resolution (F).

We compared the results of our TDSR method with three interpolations techniques directly applied on the sparse disparity map: nearest, linear, and cubic.

We also compared our results with the Weighted Least Squares disparity map post filtering method of the OpenCV library [19]. This method uses the Fast Global Smoothing [18] algorithm for diffusing high confidence disparity values using an edge preserving scheme. However, this method is parametric, bringing additional challenges to the comparison. There are three main parameters: "depthDiscontinuityRadius" (or "depthDR"), defining the size of low-confidence regions around depth discontinuities, "lambda", defining the amount of regularization, and "sigma", defining how sensitive the filtering process is to the source image edges. We computed three complete maps: one with the default parameters, one where the parameters have been optimized to minimize "Bad 2.0", and one where the parameters have been optimized to minimize "Avg. error". The parameters have been optimized using a grid search. Tested parameters are shown in table 2.

Parameter	**Tested**	**Default**	**"Bad 2.0" optimized**	**"Avg. error" optimized**
depthDR	1, 3, 5, 7, 10	5	3	3
lambda (×1000)	0, 0.1, 0.3, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16, 20	8	0.1	14
sigma (/10)	1, 2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20, 30	10	10	12.5

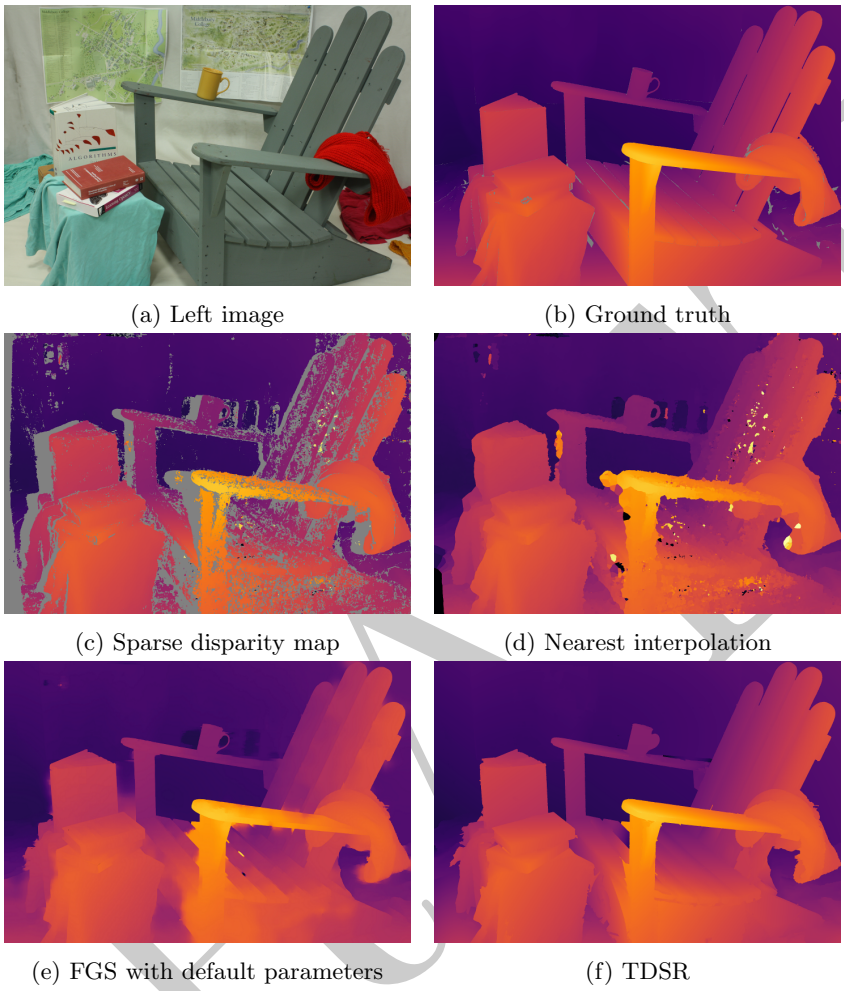
Table 2: Tested parameters for FGS.

Produced disparity maps can be qualitatively compared on the "Adirondack" pair of the Middlebury dataset in figure 3. They can be quantitatively compared on the whole training set in table 4.

Compared to other stereo methods in the Middlebury benchmark, on the training set, using the "Avg. error" criterion on all pixels, our solution currently ranks 8th over 55 compared to all other methods, and 1st over 13 compared to methods operating on the same resolution (full resolution). The results are therefore comparable with state of the art methods.

Interpolation methods are very sensitive to the noise in the sparse disparity map and they don't work well on occlusions. Depending on the chosen parameters, the Fast Global Smoothing method can either produce a less noisy disparity map but with the risk of having the disparity of some objects contaminated by neighboring ones (high "lambda", optimized for "Avg. error"), or more noise but less contamination (low "lambda", optimized for "Bad 2.0"). Default parameters give a good tradeoff between both effects.

Qualitatively, compared to the interpolation and FGS methods, our approach appears robust to noise and seems to globally manage occlusions. However, some



(a) Left image (b) Ground truth

(c) Sparse disparity map (d) Nearest interpolation

(e) FGS with default parameters (f) TDSR

Fig. 3: **Image and disparity map produced for the Adirondack pair of the Middlebury dataset.**

Method	**Bad 2.0 (no occ)**	**Avg. error (no occ)**	**Bad 2.0 (all)**	**Avg. error (all)**
Nearest interpolation	19.7	6.09	27.4	12.5
Linear interpolation	20.5	5.89	29.9	13.0
Cubic interpolation	20.5	5.98	29.8	12.8
FGS (Optimized on Bad 2.0)	20.0	5.87	26.7	11.8
FGS (Default)	25.4	5.62	31.4	9.22
FGS (Optimized on Avg. error)	31.3	5.4	37.2	8.69
TDSR	***18.2***	***4.63***	***23.3***	***7.20***
→ Improvement to best interp.	*8%*	*21%*	*15%*	*42%*
→ Improvement to best FGS	*9%*	*14%*	*13%*	*17%*

Fig. 4: **Quantitative comparison between interpolation, FGS, and our proposed TDSR method.** *all*: Taking all points into account. *no occ*: Taking only non-occluded points into account.

occlusions, notably near the armrest in the foreground (see figure 3f), are not well evaluated due to lack of data. Some regions can also have their disparity contaminated by nearby objects with similar color.

Quantitatively, compared to the interpolation and FGS methods, whether we choose to only look at non occluded pixels or to look at all the pixels, our method produces better results according to all chosen criteria. It is important to note that our algorithm's parameters were manually fixed and not optimized for minimizing any of the two chosen criteria, though we optimized the parameters of the FGS algorithm for each criterion and compared our algorithm's results to the best cases of the interpolation and FGS methods.

4 Conclusion

We have presented the Top Down Segmented Regression (TDSR) algorithm that allowed us to densify noisy sparse disparity maps. Our method generated promising results. TDSR is more robust to noise and better preserves the edges than interpolation or diffusion algorithms. The quality of the produced complete disparity map is comparable with state of the art methods.

Our approach is also completely independent from the sparse disparity map generation, so it can be used in complement to any stereo matching algorithm as a post-processing step.

In fact, TDSR could also be adapted to densify other sparse spatial data or to refine dense but noisy data. Dense disparity maps refinement, depth map super-resolution or semantic segmentation post-processing are potential applications that would require very little changes on the proposed approach.

References

1. Ayache N., Hansen C.: Rectification of images for binocular and trinocular stereovision. RR-0860, INRIA. 1988. <inria-00075694>
2. Konolige K.: Small vision systems: hardware and implementation. Eighth International Symposium on Robotics Research. pp. 111–116, 1997
3. Hirschmüller H.: Stereo Processing by Semiglobal Matching and Mutual Information, IEEE Transactions on pattern analysis and machine intelligence, Vol. 30, No. 2, February 2008
4. Facciolo G., Franchis C., Meinhardt, E.: A Significantly More Global Matching for Stereovision. BMVA Press. BMVC 2015, 2015, Swansea, United Kingdom. Proceedings of the British Machine Vision Conference 2015, 2015, <10.5244/C.29.90>.
5. Facciolo G., Franchis C., Meinhardt, E.: A Significantly More Global Matching for Stereovision - Supplementary Material. BMVA Press. BMVC 2015, 2015, Swansea, United Kingdom. Proceedings of the British Machine Vision Conference 2015, 2015, <10.5244/C.29.90>.
6. Bricola J-C., Bilodeau M., Beucher S.: A multi-scale and morphological gradient preserving contrast. 14th International Congress for Stereology and Image Analysis, Jul 2015, Liège, Belgium.
7. Bricola J-C., Bilodeau M., Beucher S.: A top-down methodology to depth map estimation controlled by morphological segmentation. 2014.

8. Beucher S.: Segmentation d'Images et Morphologie Mathématique. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 1990.
9. Beucher S.: Towards a unification of waterfalls, standard and P algorithms. `http://cmm.ensmp.fr/~beucher/publi/Unified_Segmentation.pdf`. 2013.
10. Kolmogorov V.: Graph Based Algorithms for Scene Reconstruction from Two or More Views. PhD thesis, Cornell University, 2003
11. Yáng Q., Wang L., Yang R., Stewénius H., Nistér D.: Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. CVPR, June 2006.
12. Bricola J-C., Bilodeau M., Beucher S.: Morphological processing of stereoscopic image superimpositions for disparity map estimation. 2016.
13. Alvarez L., Deriche R., Sánchez J., Weickert J.: Dense disparity map estimation respecting image discontinuities: a PDE and scale-space based approach. Journal of Visual Communication and Image Representation 13 (1) (2002) 3–21.
14. Scharstein D., Szeliski R.: Stereo matching with non-linear diffusion. International Journal of Computer Vision 28 (2) (1998) 155–174
15. Weickert J.: Anisotropic diffusion in image processing. Ph.D. Thesis, Universität Kaiserslautern, 1998.
16. Ralli J., Pelayo F., Díaz J.: Increasing efficiency in disparity calculation. BVAI2007, vol. 4729, 2007, pp. 298–307.
17. Ralli J. et al.: A method for sparse disparity densification using voting mask propagation, J. Vis. Commun. (2009) doi:10.1016/j.jvcir.2009.08.005
18. Min D., Choi S., Lu J., Ham B., Sohn K., Do, M.: Fast Global Image Smoothing Based on Weighted Least Squares. IEEE Trans. Image Processing 23(12): 5638-5653 (2014)
19. Itseez. Open source computer vision library. https://github.com/itseez/opencv.
20. Zabih R., Woodfill J.: Non-parametric local transforms for computing visual correspondence. In Eklundh, J.O., ed.: Computer Vision – ECCV '94. Volume 801 of LNCS. Springer, Berlin (1994) 151–158
21. Haller I., Nedevschi S.: Design of interpolation functions for subpixelaccuracy stereo-vision systems. IEEE Trans. on Image Proc., 21(2):889–898, 2012
22. Fua P.: A parallel stereo algorithm that produces dense depth maps and preserves image features. Machine Vision and Applications, 6(1):35–49, December 1993.
23. Scharstein D., Hirschmüller H., Kitajima Y., Krathwohl G., Nesic N., Wang X., and Westling P.: High-resolution stereo datasets with subpixel-accurate ground truth. In German Conference on Pattern Recognition (GCPR 2014), Münster, Germany, September 2014.
24. Kanade T., Okutomi M.: A stereo matching algorithm with an adaptive window: Theory and experiment. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(9):920–932, 1994.
25. Anandan P.: A Computational Framework and an Algorithm for the Measurement of Visual Motion. Int. Joun. Comp. Vis. vol 2, 1989, pp. 283-310.
26. Fischler M., Bolles R.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM. 24 (6), 1981, pp. 381–395.
27. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
28. Morovec K., Harvey R., Bangham J.: Improving stereo performance in regions of low texture. In Proc. BMVC-98, pages 822–831, Southampton, UK, 1998.

29. Salembier P., Luis Garrido: Binary partition tree as an efficient representation for filtering, segmentation and information retrieval IEEE Transactions on Image Processin 9(4), 2000, pp 561-576.
30. Vachier C., Meyer F. (2005). The viscous watershed transform. Journal of Mathematical Imaging and Vision, 22(2-3):251–267.