Large Occlusion Completion Using Normal Maps

Engin Tola, Andrea Fossati, Christoph Strecha, Pascal Fua

Computer Vision Laboratory
École Polytechnique Fédérale de Lausanne
Switzerland

Abstract. Dense disparity maps can be computed from wide-baseline stereo pairs but will inevitably contain large areas where the depth cannot be estimated accurately because the pixels are seen in one view only. A traditional approach to this problem is to introduce a global optimization scheme to fill-in the missing information by enforcing spatial-consistency, which usually means introducing a geometric regularization term that promotes smoothness.
The world, however, is not necessarily smooth and we argue that a better approach is to monocularly estimate the surface normals and to use them to supply the required constraints. We will show that, even though the estimates are very rough, we nevertheless obtain more accurate depth-maps than by enforcing smoothness. Furthermore, this can be done effectively by solving large but sparse linear systems.

1 Introduction

Though dense short-baseline stereo matching is well understood [1, 2], its wide baseline counterpart is much more challenging due to large perspective distortions and increased occluded areas. It is nevertheless worth addressing because it can yield more accurate depth estimates while requiring fewer images to reconstruct a complete scene.

It has been shown that replacing traditional correlation windows by SIFT-like descriptors such as DAISY [3], which can be efficiently computed at every point in the image, yields more effective dense matching and therefore better depth maps for widely separated images. This, however, does not address the occlusion issue, that is, of pixels that are seen in one image only and to which it is difficult to assign a reliable depth value. A traditional approach to solving this problem is to rely on a global optimization scheme to fill the resulting holes in the depth map by enforcing spatial consistency, which usually means introducing a geometric regularization term that promotes smoothness.

The world, however, is not always smooth, especially in urban environments, and this approach often results in over-smoothing. In this paper, we show that we can improve upon this situation by estimating the normals in the occluded areas and using these estimates to more accurately fill-in the holes in our depth maps. More specifically, even though single-view estimation of surface normals is known to be difficult, as evidenced by the fact that shape-from-texture remains an open problem in uncontrolled environments, we will show that it is possible to train classifiers to categorize the normal direction as left or right, up or down, and frontal or non-frontal. Given such categories, we formulate constraints on the normals and solve a large but sparse linear system involving these constraints along with those imposed by depth estimates in non-occluded areas.

We will show both qualitatively and quantitatively that the resulting depth maps are much more accurate than those obtained by simply imposing smoothness on several urban scenes for which the ground truth data are available.

2 Related Work and Approach

For short-baseline stereo techniques, occlusions are a concern but one that usually affects only a small fraction of the resulting disparity map. A number of recent algorithms simultaneously solve for depth and occlusion estimation using energy minimization approaches [4, 5] and fill-in the disparity maps using values from the background. Others rely on image segmentation and assign a depth value to individual segments [6, 7]. Yet another approach is to use a texture-based algorithm to propagate depth and texture together from the boundaries of occluded areas, while taking edge information into account [8]. These approaches, however, assume that the occluded content spatially occupies a small region, and hence imposing a form of smoothness together with a texture-gradient like constraint is able to cope with the problem. This assumption, however, is not necessarily true when the occlusions start occupying larger regions.

The approach we advocate here to address these issues is to estimate the surface normals in the occluded areas from the local texture and the changes in spatial frequencies that orientation changes produce. In that sense, our approach is closely related to the traditional shape-from-texture problem, which has been thoroughly researched over the years [9–13]. These methods start from the assumption that the properties of the textures are statistically homogeneous in the 3D space, and that observed distortions result from the perspective projection and depend on the orientation of the surface normal. To quantify this effect, they often rely on the fact that an instance of the texture has been observed from a known viewpoint and explicitly compute differential geometric parameters using tools such as the wavelet transform [14], the Fourier transform [9], or Gabor filters [15].

For an algorithm designed to work in a wide range of situations, however, the basic assumption that underpins these approaches is far too restrictive. Real-world textures are rarely homogeneous enough to make them amenable to such techniques, even if one limits oneself to built-up environments. However, even in such an environment, regularities exist and can be exploited. For example, most urban areas contain structures with very similar, even if not homogeneous, texture contents such as brick walls, windows, and regularly-shaped pavements. Therefore, we propose to train classifiers from generic outdoor textures of buildings to estimate normal cues for a generic *urban* wide-baseline stereo algorithm. This is generic in the sense that it does not require any information about the current, scene apart from the fact that it contains building-like structures. To the best of our knowledge, this is the first learning-based approach to surface normal inference. It is in the same spirit as approaches that use learning for depth estimation from single images [16] but is designed for true metric reconstruction.

Normal estimation from a single image is a difficult and potentially ill-posed problem. To achieve as much robustness as possible to varying textures, noise, and rapid depth changes, we first train classifiers to handle simple sub-problems, such as deciding whether the surface normal orientation is *left or right*, *up or down* and *frontal or not*. The results are then used to constrain a global reconstruction algorithm. In practice, even though the output of the classifiers are only coarse surface normal estimates, they are sufficient to yield much improved reconstructions.

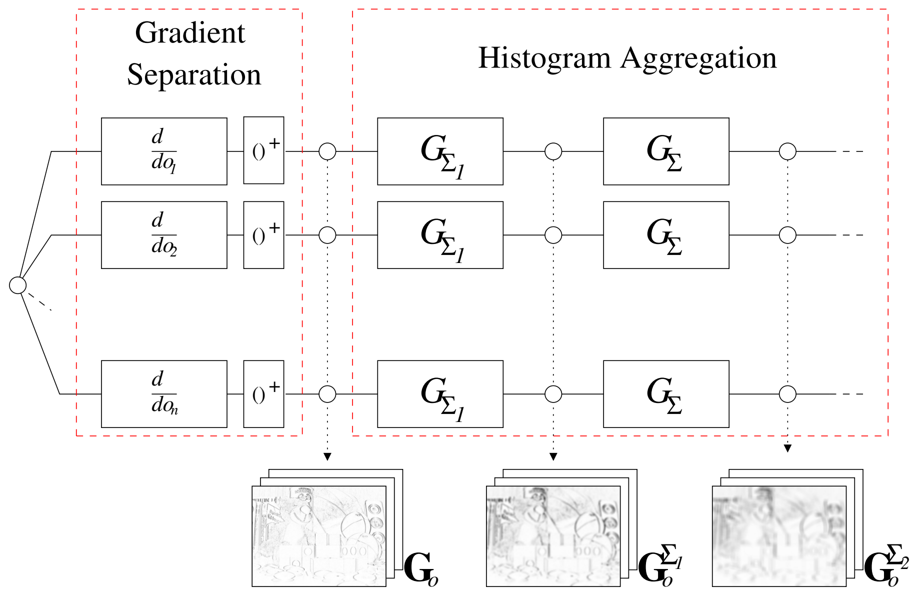


Fig. 1. DAISY descriptor. It is computed by first separating the gradients into different layers according to their orientation and then using Gaussian convolutions to perform the aggregation over different sized windows efficiently.

3 Constraining the Normals

As discussed above, in textured areas, spatial frequencies change with surface orientation. There are many ways to capture this effect, using for example the Fourier or Wavelet transforms [9, 14]. In this work, we rely on gradient histograms, which are also sensitive to orientation changes and can be computed locally and fast at every pixel to form DAISY descriptors [3]. We made this choice because DAISY has also proved very good at matching points across widely separated images, and we can also use it to establish correspondences in the non-occluded areas.

DAISY is a dense descriptor composed of concatenated gradient histograms. They are computed by first separating gradients into different layers and aggregating gradient magnitudes at different orientations, as shown in Figure 1. Once the aggregation is done, a histogram with a certain aggregation window size is computed by reading the values of the pixels from the appropriate Gaussian convolved layers.

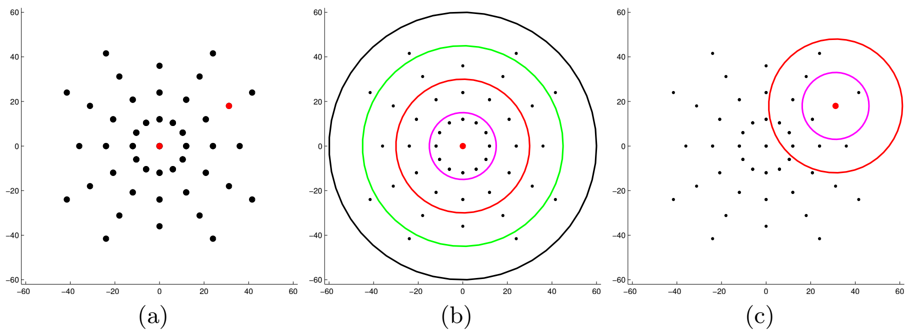


Fig. 2. Feature extraction grid. We compute features around the patch at discretized locations. **(a)** Grid locations. **(b)** Each circle represents a possible window size for histogram computation. **(c)** Possible aggregation sizes for a grid point near the edge are shown. Larger window sizes are not allowed at grid points near the edge in order to limit the patch size.

We use this same procedure to compute our features to estimate normal cues. To be specific, our features are the values of the bins of the gradient histograms computed as explained above on a dense grid as shown in Figure 2 to pass a richer set of possibilities to the classifier.

Given the features computed at points where the surface normals are known, we train a binary classifier with the Adaboost [17] learning procedure using two types of weak classifiers. The first weak classifier type is a simple decision rule of the form

$$c_i = \begin{cases} 1 \text{ if } f_p \geq f_q \\ -1 \text{ otherwise} \end{cases}, \tag{1}$$

where f_p is the p^{th} feature and f_q is the q^{th} feature. Both of them are chosen randomly and the comparison direction is optimized over the training data.

The second weak classifier type is a level-2 decision tree structured as

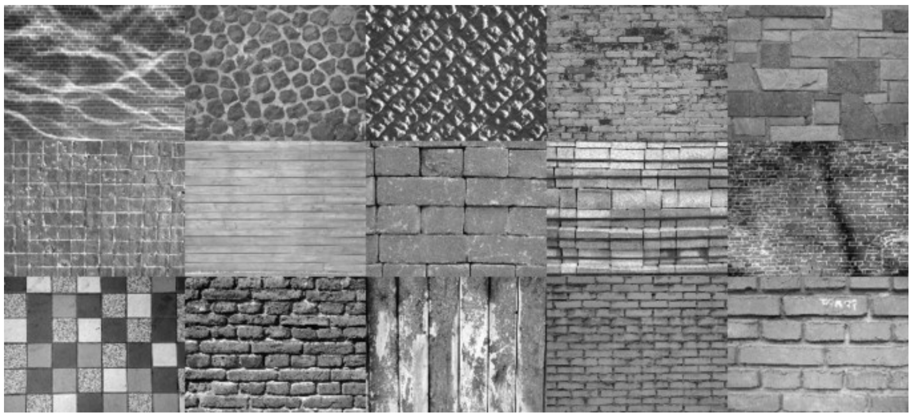


Fig. 3. Training Textures. Some of the textures used for training normal classifiers.

$$c_i = \begin{cases} 1 \text{ if } f_p \geq \sigma_p \ \& \ f_q \geq \sigma_q \\ 1 \text{ if } f_p < \sigma_p \ \& \ f_q \geq \sigma_r \\ -1 \text{ otherwise} \end{cases}, \tag{2}$$

where the decision thresholds σ's and the comparison directions are again optimized over the training data.

During the training phase, we randomly generate and train 1000 weak classifiers of both types at every iteration of the Adaboost procedure and then pick the one that minimizes the weighted error of the samples. The learned strong classifier is then of the form

$$\mathbb{C} = \sum_{k=1}^{K} w_k c_k, \tag{3}$$

where K is the number of weak classifiers and w_k are the weights associated with them.

Training is performed on a set of generic building textures, some of which are shown in Fig. 3. We extract patches from these textures and warp them according to a surface normal. For example, in order to train the *left-right* classifier, we used patches warped to the left and right of the plane parallel to the fronto-parallel normal. Once the warping is done, these patches are used as positive and negative examples for training.

We compute features from 8-bin gradient histograms within a 60 pixel radius of the patch center. We use 4 aggregation windows of sizes 15, 30, 45 and 60. Since we enforce an upper limit to the maximum area where a feature can be computed, the size of the feature windows becomes smaller as we approach the patch boundary.

As said earlier, we train 3 such classifiers for the normal cue estimation: *left-right* (C_{lr}), *up-down* (C_{ud}), and *frontal-non-frontal* (C_{fnf}), using the same

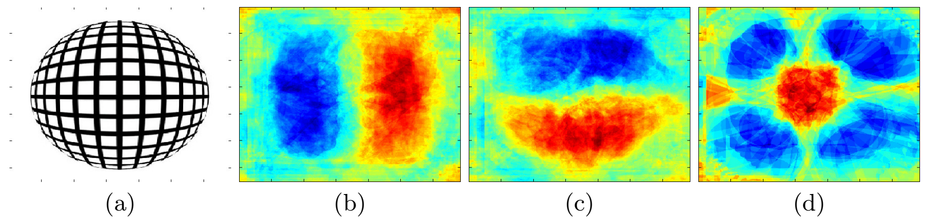


Fig. 4. Classification of a sphere. We test the trained classifiers on an image of a checkerboard pattern overlayed on a sphere **(a)**. We show the score response of each classifier. **(b)** Result of the left-right classifier. **(c)** Result of the up-down classifier. **(d)** Result of the frontal-nonfrontal classifier.

feature space. After training, we evaluate the performance of the classifiers on an image of a checkerboard pattern overlaid on a sphere, shown in Figure 4.

We also show the performance of the classifier on a real scene, for which we know the true normal values, in Figure 5. All the pixels that are correctly classified are marked with a color and we use different colors to distinguish among correct classes. If the classifier is uncertain, i.e. gives a low response to either classes, we do not mark the pixel.

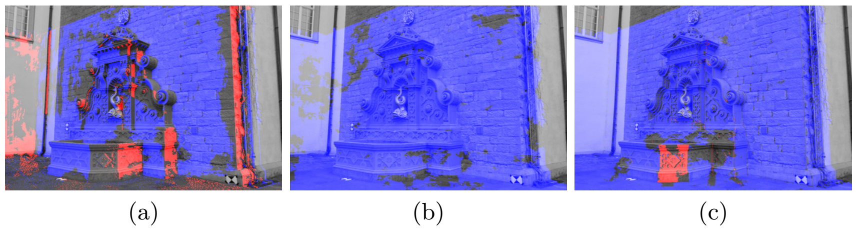


Fig. 5. Normal Estimation Performance. We show the performance of the normal estimation on an example image. We mark the pixel with a color if it is correctly classified. **(a)** shows the correctly classified left(blue) and right(red) pixels. **(b)** shows the correctly classified up(blue) and down(red) pixels. **(c)** shows the correctly classified frontal(red) and non-frontal(blue) pixels.

4 Optimization

In this section, we explain how we combine the output of the classifiers and the depth estimates computed from the stereo algorithm. We formulate the optimization problem as the minimization of an energy function E, which takes into account both the constraints that derive from knowing some depth values, and those provided by our normal estimation algorithm. They are expressed in terms of two energy terms, E_D and E_N respectively, and we take E to be:

$$E(d) = E_D(d) + E_N(d) \tag{4}$$

with

$$E_D(d) = \sum_{i,j \in I} V_{i,j}(d_{i,j} - \overline{d}_{i,j})^2, \tag{5}$$

where $V_{i,j}$ is an indicator function defined as

$$V_{i,j} = \begin{cases} 1 \text{ if } \overline{d}_{i,j} \text{ is provided} \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

and

$$\begin{aligned} E_N(d) = \sum_{i,j \in I} w^x_{i,j}(\frac{d_{i+1,j} - d_{i-1,j}}{2} - \overline{d}^x_{i,j})^2 + \\ \sum_{i,j \in I} w^y_{i,j}(\frac{d_{i,j+1} - d_{i,j-1}}{2} - \overline{d}^y_{i,j})^2 \end{aligned} , \tag{7}$$

with $\overline{d}^x_{i,j}$ and $\overline{d}^y_{i,j}$ representing the estimated x and y-gradient of the depth map at pixel (i, j), and $w^x_{i,j}$, $w^x_{i,j}$ their relative weights.

The depth estimate d for the scene will then be computed as

$$d = \operatorname*{argmin}_d E(d), \tag{8}$$

by imposing

$$\frac{\partial E(d)}{\partial d} = 0. \tag{9}$$

We formulate this problem as a sparse linear system $Ad = b$ and solve it using Cholesky factorization as described in [18].

The depth gradients in Equation 7 are computed from the normal cues at each pixel location by combining the results of our normal classifiers. In addition to this cue, these gradients can also be computed imposing smoothness and, where available, normal estimates obtained from stereo data. The gradient estimates used in the optimization will then be composed by a weighted average of such terms:

$$\overline{d}^x_{i,j} = \frac{\lambda_S d^x_S + \lambda_{N^x(i,j)} d^x_{N(i,j)}}{\lambda_S + \lambda_{N^x(i,j)}}, \tag{10}$$

$$\overline{d}^y_{i,j} = \frac{\lambda_S d^y_S + \lambda_{N^y(i,j)} d^y_{N(i,j)}}{\lambda_S + \lambda_{N^y(i,j)}}. \tag{11}$$

The λ terms represent the influence of the different factors in the computation of the estimated gradients, and they are also used to compute the relative weights of the gradients in the energy function $E(d)$ as:

$$w^x_{i,j} = \lambda_S + \lambda_{N^x(i,j)}, \tag{12}$$

$$w^y_{i,j} = \lambda_S + \lambda_{N^y(i,j)}. \tag{13}$$

The value of λ_S is kept constant in all the experiments shown in the paper and the values of $\lambda_{N^x(i,j)}$ and $\lambda_{N^y(i,j)}$ are proportional to the score of the classifiers. Moreover, the values of d^x_S and d^y_S indicate the smoothness component and are, therefore, set to zero. Finally the terms $d^x_{N(i,j)}$ and $d^y_{N(i,j)}$ are obtained from the normal estimation process. The Frontal-Non-frontal classifier, C_{fnf}, works as a detector about the degree of the deviation from being frontal which is equivalent to the slant of the patch. We coarsely set the value of the slant σ by comparing the classifier response to an uncertainty threshold, T_{fnf}:

$$\sigma_{i,j} = \begin{cases} 0 \text{ if } |C_{fnf}| < T_{fnf} \\ 0 \text{ if } C_{fnf} > T_{fnf} \\ \pi/4 \text{ if } C_{fnf} < -T_{fnf} \end{cases}. \tag{14}$$

In uncertain cases, such as when $|C_{fnf}| < T_{fnf}$, we assign 0 to the slant and do no worse than smoothness.

The other two classifiers C_{lr} and C_{ud} provide information about the tilt angle τ and we similarly set it in a coarse way by comparing the classifier responses to uncertainty thresholds, T_{lr} and T_{ud}:

$$\tau_{i,j} = \begin{cases} 0 \text{ if } |C_{lr}| < T_{lr} \;\&\; |C_{ud}| < T_{ud} \\ 3\pi/4 \text{ if } C_{lr} > T_{lr} \;\&\; C_{ud} > T_{ud} \\ -3\pi/4 \text{ if } C_{lr} > T_{lr} \;\&\; C_{ud} < -T_{ud} \\ \pi/4 \text{ if } C_{lr} < -T_{lr} \;\&\; C_{ud} > T_{ud} \\ -\pi/4 \text{ if } C_{lr} < -T_{lr} \;\&\; C_{ud} < -T_{ud} \\ \pi/2 \text{ if } |C_{lr}| < T_{lr} \;\&\; C_{ud} > T_{ud} \\ -\pi/2 \text{ if } |C_{lr}| < T_{lr} \;\&\; C_{ud} < -T_{ud} \\ 0 \text{ if } C_{lr} < -T_{lr} \;\&\; |C_{ud}| < T_{ud} \\ \pi \text{ if } C_{lr} > T_{lr} \;\&\; |C_{ud}| < T_{ud} \end{cases}. \tag{15}$$

This is a coarse 1-out-of-8 possible direction selection, but nonetheless gives enough information to improve the quality of the results. Again, in an uncertain situation, we set the tilt angle τ to 0. This procedure of normal selection lets us use normal cues where our classifier successfully retrieves them, and falls back to simple smoothness when it fails.

The relation between the slant-tilt representation and the normal direction is given as

$$n_{i,j} = \begin{bmatrix} sin(\sigma_{i,j}) * cos(\tau_{i,j}) \\ sin(\sigma_{i,j}) * sin(\tau_{i,j}) \\ cos(\sigma_{i,j}) \end{bmatrix} \tag{16}$$

Once the normal, $n_{i,j}$, is decided for a pixel, we compute implied depth gradient for this pixel using:

$$d^x_{N(i,j)} = d(i,j) \frac{n^T_{i,j} K^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{n^T_{i,j} K^{-1} \begin{bmatrix} i-1 \\ j \\ 1 \end{bmatrix}}, \tag{17}$$

$$d^y_{N(i,j)} = d(i,j) \frac{n^T_{i,j} K^{-1} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}{n^T_{i,j} K^{-1} \begin{bmatrix} i \\ j-1 \\ 1 \end{bmatrix}}. \tag{18}$$

Since this formulation depends on a depth estimate, $d(i,j)$, we first solve the system using smoothness only and then initialize the gradient depth estimates from the smooth solution. In practice, we run the optimization algorithm for a few iterations and re-estimate $d^x_{N(i,j)}$ and $d^y_{N(i,j)}$ at every iteration from the current solution.

5 Experiments

To evaluate the effectiveness of including the surface normals in the reconstruction algorithm, we first performed synthetic experiments by introducing artificial occlusions in scenes for which high quality laser scans are available [19]. We compared our results against standard smoothness constraints, which is intrinsically not suitable to recover piecewise smooth surfaces.

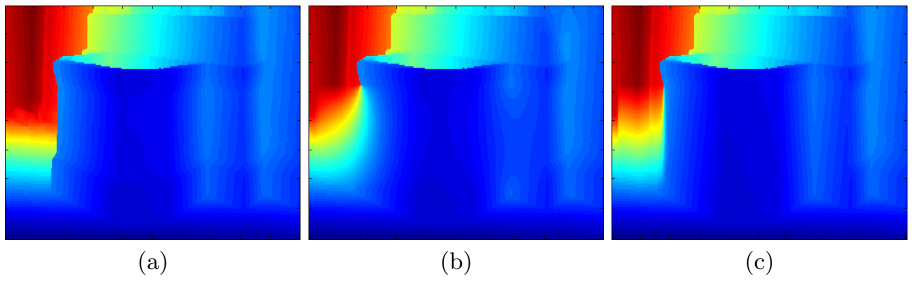


Fig. 6. Zoom-In on Fountain image. A zoomed in version of the recovered lower-left occlusion in Fig. 9 **(a)** The original depth map. **(b)** Recovered depth map with smoothness. **(c)** Recovered depth map using normal constraints.

In Figure 9 and Figure 10, we artificially occluded the black rectangular regions, treated the depths outside them as being known, and ran our regularization algorithm with and without surface normal constraints. We display the

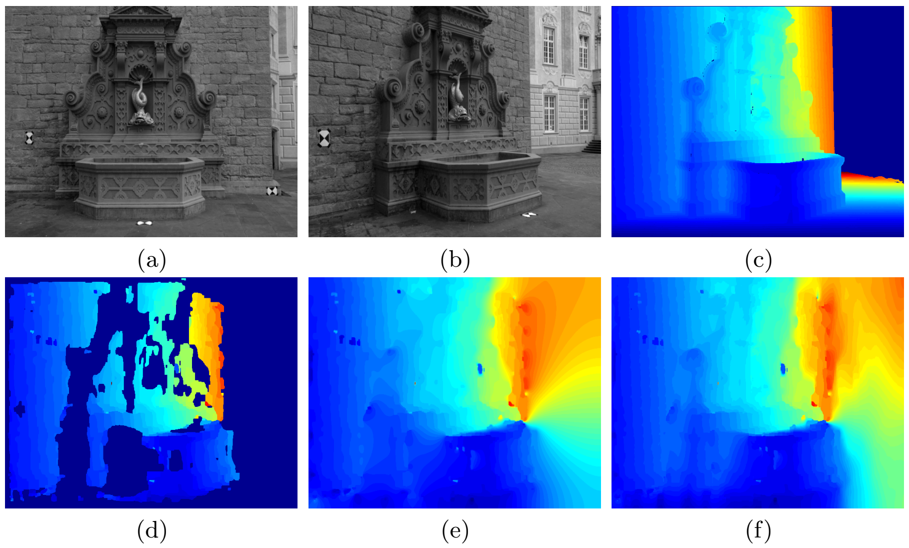


Fig. 7. Occlusion recovery with stereo . **(a-b)** Input images. **(c)** Ground truth depth map. **(d)** Output of the stereo algorithm. **(e)** Depth map recovered using only smoothness constraints. **(f)** Depth map recovered using normal cue based constraints

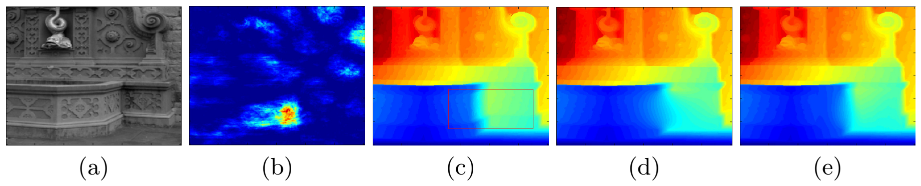
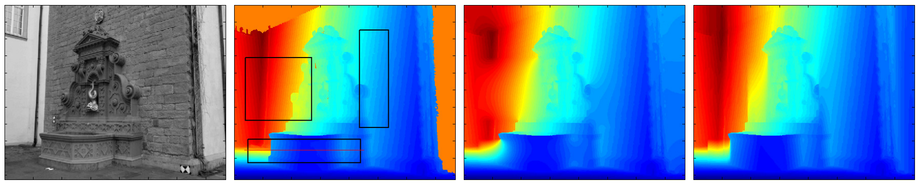
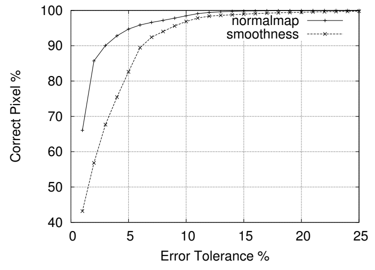


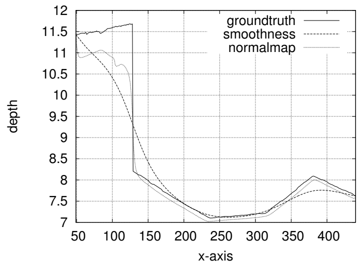
Fig. 8. Reconstruction Zoom-In. In this experiment, we occluded some part of a ground truth depth map and tried to recover it using only smoothness and normals. **(a)** shows the original image and **(b)** is the estimated score for the left-right classifier, with blue indicating left and red indicating right. **(c)** is the original depth map with a red rectangle showing the occluded region. The result of using smoothness is shown in **(d)**, and the result of using the constraints of **(b)** is shown in **(e)**. It is evident from this figure that even using this coarse normal cue is enough to generate a better depth map than simple smoothness.



(a) (b) (c) (d)



(e)



(f)

Fig. 9. Occlusion Recovery for the Fountain dataset. (a) Original image **(b)** Ground truth depth map with occluded area borders shown in black. **(c)** Depth map recovered by imposing smoothness. **(d)** Depth map recovered by imposing surface normal based regularization. **(e)** Percentage of pixels within a certain distance from the ground truth. **(f)** True and recovered depth along the red line in the lowermost occluded rectangle.

recovered depth maps and the error curves for both approaches. Error curves are computed in occluded regions only, and they show the percentage of pixels whose depth estimates are within a given threshold, expressed as a fraction of scene's depth range. Inspecting the estimated depths along scan lines in occluded regions, such as the ones of Figures 9**(f)** and 10**(f)** shows that our normal cues definitely bring the results closer to the ground truth than the smoothness constraints.

In Figures 6 and 8, we show close-up views of regions where smoothness fails to recover the actual structure and estimated surface normals do better.

Next, in Figure 11, we used a wide baseline stereo algorithm [3] to estimate depth and occlusion maps and fill in the occlusions using either smoothness or surface-normal based regularization for different image pairs. Deviations from the ground truth are also plotted. Smoothness constraints are indiscriminately smooth in every direction while the surface-normals are better behaved. As can be noticed, the positive effects of our approach are more evident when the baseline is wider, and therefore the occlusions larger, since it does not affect pixels for which the depth has been retrieved through the stereo algorithm. In Figure 7 we perform the same experiment on a different image pair.

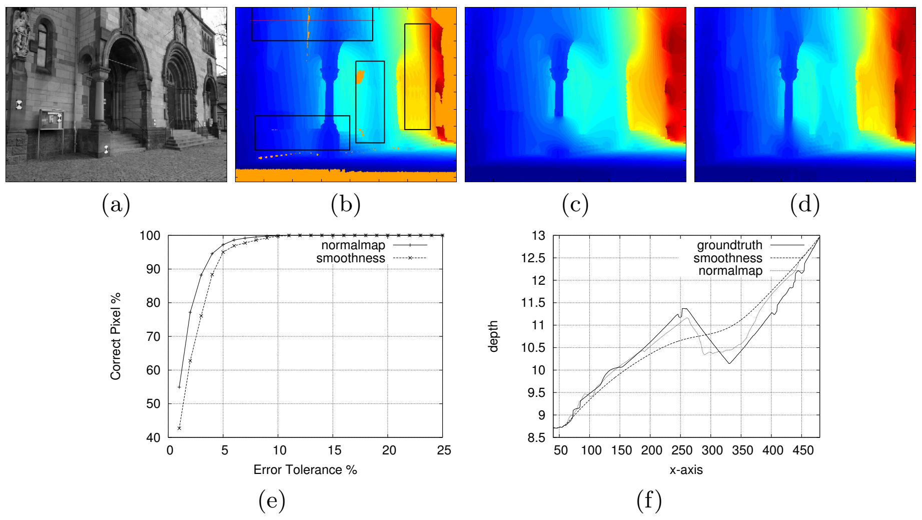


Fig. 10. Occlusion Recovery for the HerzJesu dataset. (a) Original image **(b)** Ground truth depth map with occluded area borders shown in black. **(c)** Depth map recovered by imposing smoothness. **(d)** Depth map recovered by imposing surface normal based regularization. **(e)** Percentage of pixels within a certain distance from the ground truth. **(f)** True and recovered depth along the red line in the lowermost occluded rectangle.

6 Conclusion

In this paper we proposed a solution to recover depth in large areas which are occluded due to the intrinsic nature of wide baseline stereo. Our approach depends on estimating coarse normal cues using an Adaboost based binary classifier. Although our normal estimators do not extend to any generic texture, they work well for common building textures. We showed that using normal cues improves the quality of the recovered depth maps quantitatively and qualitatively, especially if the scene contains non-smooth, piecewise-smooth structures.

One possible shortcoming of our method is its two step approach: first computing depth and occlusion estimates and then filling in the occlusions. We believe that a joint solution of the stereo and normal cue steps should improve the overall performance as both steps are highly correlated and we will pursue this idea in future.

References

1. Brown, M., Burschka, D., Hager, G.: Advances in Computational Stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003) 993–1008
2. Scharstein, D., Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. Computer Vision and Image Understanding **47** (2002) 7–42

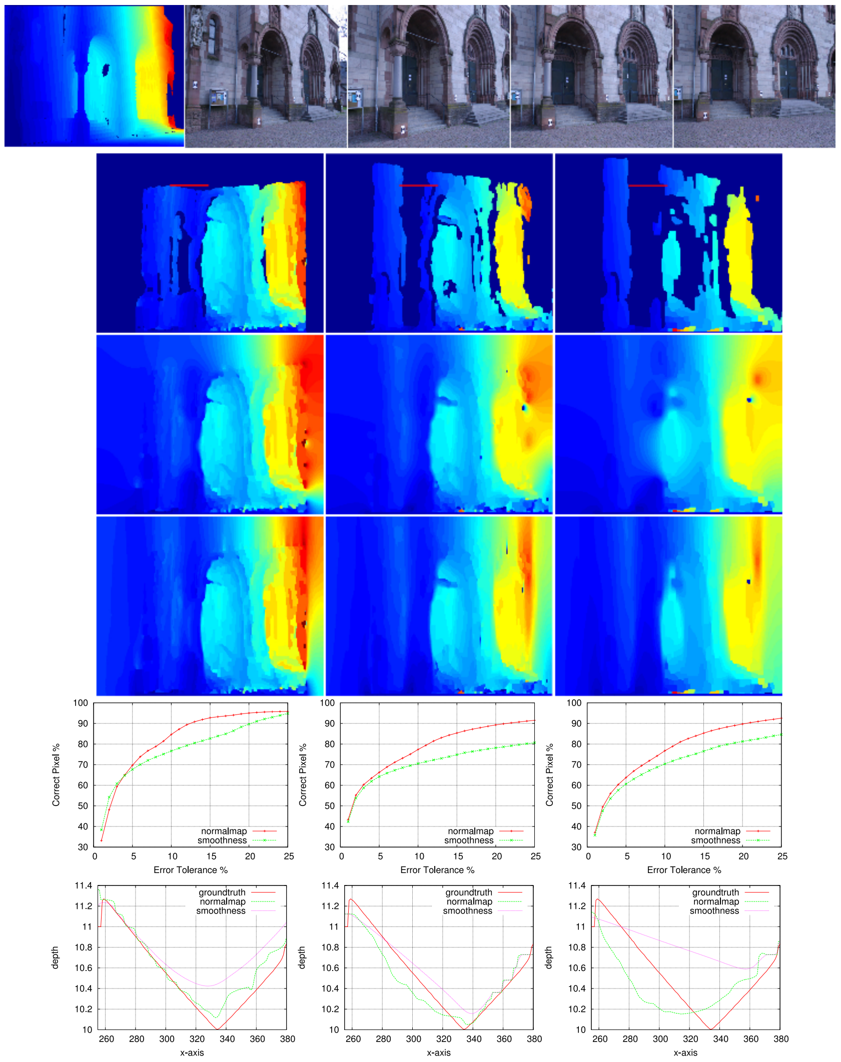


Fig. 11. Results on stereo with normal cue regularization. We try to recover the depth map given in the **first row** using the first image and other images one by one. **Second Row.** Estimated depth and occlusion maps using the wide baseline stereo algorithm given in [3]. **Third Row.** Depth maps recovered using smoothness constraint. **Fourth Row.** Depth maps recovered using the normal cue based constraint. **Fifth Row.** Error curves of two approaches expressed as a percentage of scene's depth range. **Sixth Row.** True and recovered depth along the red line in the **third row**.

3. Tola, E., Lepetit, V., Fua, P.: A Fast Local Descriptor for Dense Matching. In: Conference on Computer Vision and Pattern Recognition. (2008)
4. Deng, Y., Yang, Q., Lin, X., Tang, X.: A Symmetric Patch-Based Correspondence Model for Occlusion Handling. In: International Conference on Computer Vision. (2005) 1316–1322
5. Sun, J., Li, Y., Kang, S., Shum, H.Y.: Symmetric Stereo Matching for Occlusion Handling. In: Conference on Computer Vision and Pattern Recognition. (2005) 399–406
6. Wang, L., Jin, H., Yang, R., Gong, M.: Stereoscopic Inpainting: Joint Color and Depth Completion from Stereo Images. In: Conference on Computer Vision and Pattern Recognition. (2008)
7. Klaus, A., Sormann, M., Karner, K.: Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. In: International Conference on Pattern Recognition. (2006) 15–18
8. Torres-Méndez, L., Dudek, G.: Reconstruction of 3D Models from Intensity Images and Partial Depth. In: American Association for Artificial Intelligence Conference. (2004) 476–481
9. Bajcsy, R., Lieberman, L.: Texture Gradient as a Depth Cue. Computer Graphics and Image Processing **5** (1976) 52–67
10. Aloimonos, Y., Swain, M.: Shape from Texture. In: International Joint Conference on Artificial Intelligence. (1985) 926–931
11. Malik, J., Rosenholtz, R.: Computing Local Surface Orientation and Shape from Texture Forcurved Surfaces. Computer Vision and Image Understanding **23** (1997) 149–168
12. Gårding, J.: Shape from Texture for Smooth Curved Surfaces. In: European Conference on Computer Vision. (1992) 630–638
13. Lobay, A., Forsyth, D.: Shape from Texture Without Boundaries. Computer Vision and Image Understanding **67** (2006) 71–91
14. Clerc, M., Mallat, S.: Shape from Texture Through Deformation. International Conference on Computer Vision **1** (1999) 405
15. Super, B., Bovik, A.: Shape from Texture Using Local Spectral Moments. IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1995) 333–343
16. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3D Scene Structure from a Single Still Image. IEEE Transactions on Pattern Analysis and Machine Intelligence **31** (2009) 824–840
17. Freund, Y., Schapire, R.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In: European Conference on Computational Learning Theory. (1995) 23–37
18. Davis, T.A.: Direct Methods for Sparse Linear Systems. SIAM, Philadelphia (2006) Part of the SIAM Book Series on Fundamentals of Algorithms.
19. Strecha, C.: Multi-view evaluation - http://cvlab.epfl.ch/data (2008)