DISPARITY REFINEMENT OF BUILDING EDGES USING ROBUSTLY MATCHED STRAIGHT LINES FOR STEREO MATCHING

X. Huang [1], R. Qin [1, 2, *], M. Chen [3]

[1] Department of Civil, Environmental and Geodetic Engineering, The Ohio State University, 218B Bolz Hall, 2036 Neil Avenue, Columbus, OH 43210, USA - (huang.3651, qin.324)@osu.edu
[2] Department of Electrical and Computer Engineering, The Ohio State University, 205 Dreese Labs, 2015 Neil Avenue, Columbus, OH 43210, USA - qin.324@osu.edu
[3] Lyles School of Civil Engineering, Purdue University, West Lafayette, 47907, USA - chen2422@purdue.edu

Commission I, WG I/2

KEY WORDS: Matched Straight Lines, Disparity Refinement, Support Window Definition, Edge Detection, Plane-based Adjustment

ABSTRACT:
Stereo dense matching has already been one of the dominant tools in 3D reconstruction of urban regions, due to its low cost and high flexibility in generating 3D points. However, the image-derived 3D points are often inaccurate around building edges, which limit its use in several vision tasks (e.g. building modelling). To generate 3D point clouds or digital surface models (DSM) with sharp boundaries, this paper integrates robustly matched lines for improving dense matching, and proposes a non-local disparity refinement of building edges through an iterative least squares plane adjustment approach. In our method, we first extract and match straight lines in images using epipolar constraints, then detect building edges from these straight lines by comparing matching results on both sides of straight lines, and finally we develop a non-local disparity refinement method through an iterative least squares plane adjustment constrained by matched straight lines to yield sharper and more accurate edges. Experiments conducted on both satellite and aerial data demonstrate that our proposed method is able to generate more accurate DSM with sharper object boundaries.

1. INTRODUCTION

The 3D reconstruction of urban regions is increasingly being used in several smart applications such as virtual reality, 3D modeling and navigation etc. There have been various technologies about the 3D reconstruction including light detection and ranging (LiDAR), stereo dense matching, structured light, manual reconstruction, etc. Compared with other technologies, stereo dense matching, which refers to the process of searching for pixelwise correspondences between oriented images, has a great potential in producing large-scale, dense 3D measurements of urban objects (e.g. building, road, and vegetation) with much lower cost. Current stereo dense matching methods focus on formulating the dense matching problems as the optimization of a global energy function (Hirschmuller, 2008; Klaus, 2006; Kolmogorov and Zabih, 2001; Scharstein and Szeliski, 2002), which is capable of obtaining robust matching results. However, such global optimization may produce a large amount of mismatches along building edges, appearing as irregular edges in matching results.

Normally building edges in images appear to have large intensity changes. To improve the building edges in matching results, several researches adjust edges in matching results to edges in images (also called edge refinement). Some of them refine per pixel in edges, while others refine the whole edge non-locally, therefore the edge refinement methods can be categorized into 1) local edge refinement and 2) non-local edge refinement. Local edge refinement methods (Gupta and Cho, 2010; Huang and Zhang, 2016; Lin and Liu, 2015; Park et al., 2015; Wang et al., 2013; Wu et al., 2016) adjust edge pixels in matching results to edge pixels in images. Once all the pixels are adjusted, all the edges in matching results are refined. These methods normally give a support window centered at pixels around edges, and then fully utilize the matching results of neighbor intensity-similar pixels to refine the matching result of central pixel with the basic assumption that pixels with similar intensities in the support window share the same matching results. Considering mismatches in the support window, such refinement is often proceeded by averaging the matching results in the support window (also called bilateral filter (Paris et al., 2008)) or picking median of the matching results weighting by intensity differences between the central pixel and neighbor pixels (also called weighted median filter (Lin et al., 2002)). Local refinement methods are simple and efficient in edge refinement, while they fail in the case that a plenty of mismatches exist in the supporting window. Non-local edge refinement methods (Hirschmuller, 2008; Li et al., 2015) refine the entire edge instead of single pixels. They firstly segment images and formulate the edges as the boundaries of each image segments, and then use plane function to refine the matching results in each image segments such that the matching results of these boundaries are also refined. Non-local edge refinement methods are superior in obtaining robust, sharp edges even though plenty of mismatches exist locally, while they may reduce matching accuracies in the case that the image segment does not satisfy the plane function (e.g. curved regions). In addition, current edge refinement methods (including local and non-local methods) do not consider matching results of edges, while the matching of edges is normally more robust than pixels. Hence, the edge refinement results should be better if such matched edges are utilized in the refinement.

To obtain sharp 3D building edges in the resulting DSM, this paper presumes the availability of both oriented stereo images and stereo dense matching results, and develops a new edge

* Corresponding author

refinement method that is able to detect and match straight lines in images as potential building edges, and fully utilize these matched straight lines to improve the refinement of building edges through a plane-based adjustment method. In general, our method formulates the refinement of building edges as the non-local plane-based adjustment with the basic assumption that pixels close to edges with similar intensities/colour share similar matching results, where matched straight lines are used to constraint the adjustment, termed as refinement of building edges using matched straight lines (RBESL). Our method is capable of obtaining sharp building edges in stereo matching.

The rest of the paper is organized as follows: Section 2 describes the methodology of the proposed method in detail; Section 3 shows the experimental results; and Section 4 concludes the manuscript.

2. METHODOLOGY

2.1 Overview

Given stereo images with accurate orientations, we firstly rectify them into epipolar images such that correspondences are in the same row between the two images, thereby the corresponding pixels can be represented by the column coordinate differences, termed as disparity or parallax. For a correspondence with (x_l, y_l) and (x_r, y_r) being the left and right pixel coordinate in epipolar images, the disparity between them is shown in Equation (1). Most stereo dense matching methods compute disparities for all pixels and produce a disparity map as matching result. In all our experiments, we used semi-global matching (SGM) (Hirschmuller, 2008) to produce an initial disparity map.

$$d = x_l - x_r \tag{1}$$

where d = disparity of a pixel

Given epipolar stereo images and the corresponding initial disparity map, our proposed method aims to refine disparity map with sharp building edges. The workflow of our proposed method includes: 1) **Robust matching of straight lines**: obtain matches of straight lines in the basic image (e.g. left image in this paper) by searching candidate matches of the straight lines using epipolar constraints, then using an intensity gradient orientation based operator to describe features of the straight lines and the candidate matches, and finally deciding the match with the most similar features; 2) **Non-local plane-based adjustment for disparity refinement**: adjust edges in disparity map to align straight lines in epipolar images using a plane-based adjustment constrained by matched straight lines. The workflow is shown in Figure 1.

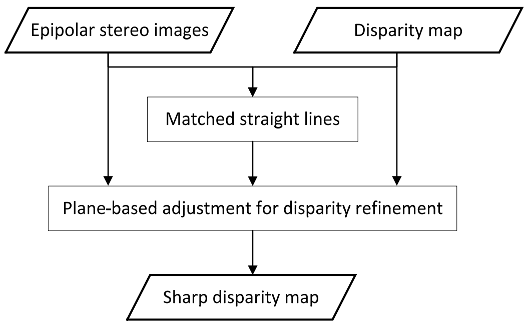


Figure 1. Workflow of proposed method

2.2 Robust Matching of Straight Lines

Matching of straight lines are normally more robust than matching results of corresponding points. Hence, the object edge in the DSM should be improved if such matched straight lines are utilized. However, due to inaccurate locations of line endpoints, fragmentation of line, etc. (Fan et al., 2010; Chen and Shao, 2013), straight line matching is still a challenge task. In this paper, we consider an epipolar constraint in the straight line matching and propose a robust straight line matching method for epipolar images. The workflow of our method is as follows.

Firstly, straight lines are detected from the input epipolar images by using the state-of-the-art line segment detector (LSD) (Von Gioi et al., 2012).

Secondly, the straight lines in the left image and the potential corresponding lines in the right image (also called candidate matches) are extracted based on epipolar constraint.
For example in Figure 2, $p1$ and $p2$ are two endpoints of the straight line L_i in the basic image (e.g. left image in this paper). e_{p1} and e_{p2} are the two corresponding epipolar lines of $p1$ and $p2$. The two points $p1'$ and $p2'$ corresponding to $p1$ and $p2$ can be found according to the disparity of $p1$ and $p2$. Two local regions R_1 and R_2 surrounding $p1'$ and $p2'$ are set to overcome the errors from the epipolar image and the initial disparity map. There are two thresholds t_e and t_d indicating the epipolar and disparity errors respectively. The straight lines in right image are potential correspondences of the straight line L_i in left images if and only if these straight lines in right images going through the two local regions R_1 and R_2 (not limited by the two endpoints of straight lines), and at least one endpoint is located in R_1 or R_2. Yellow lines in Figure 2 show the scope of potential correspondences of L_i.

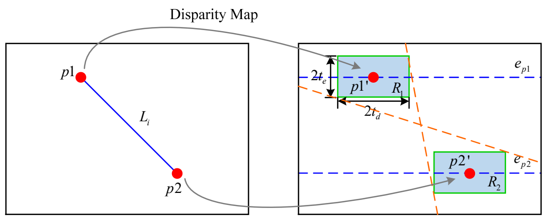


Figure 2. Candidate matches computation.

Then, line descriptors are constructed for each pair of candidate matches as follows:

1) Given a straight line L_i in the left image, the endpoints of potential corresponding line may not the correspondences of endpoints of L_i, as shown in Figure 3. L_i' is the potential corresponding line of L_i. To solve the problem of line fragmentation, L_i' is aligned horizontally to L_i by computing two intersections (denoted as q_1, q_2) between L_i' and e_{p1}, e_{p2}, and the segment q_1q_2 is used for matching computation.

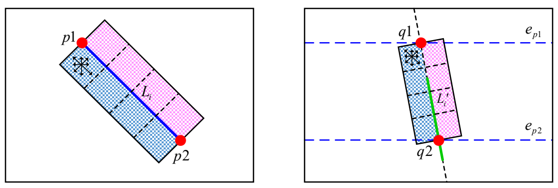


Figure 3. Line descriptor construction. The blue line is the straight line in the left image. The green line is one of potential corresponding lines in right image.

2) Rectangular support windows centred at L_i and L_i' are built, and a feature descriptor is designed to describe the features of these support windows. We firstly divide the support window into four sub-regions on each side of straight lines, as shown in Figure 3. In each sub-regions, a gradient orientation (8-orientation) histogram is computed for describing the features in such sub-region, where the direction of straight line is regarded as the orientation of each sub-region such that the orientation histogram is robust to image rotation. Considering discontinuities in disparity edges, we describe the features of each side (left side and right side) respectively by combining the 8-orientation histograms of sub-regions on the same side to form 32-dimensional feature vectors. Finally, the 32-dimensional feature vector are combined to form a 64-dimensional descriptor to describe the features of the support window, as follows:

$$Desc = \begin{bmatrix} d_{left} \\ d_{right} \end{bmatrix} \tag{2}$$

where d_{left}, d_{right} = feature vectors in the support window
Desc is a 64-dimensional descriptor

3) Finally, we compare the features of each candidate match and select the candidate match with most similar features as the final straight line matching result. The similarity of the features is measured by nearest neighbor distance ratio (NDSR) strategy, and the candidate match with the shortest distance is adopted, as shown in Equation (3).

$$Similarity = \max\{||d_{left} - d'_{left}||, ||d_{right} - d'_{right}||\} \tag{3}$$

where d'_{left}, d'_{right} = feature vectors in right images

2.3 Non-local Plane-based Adjustment for Disparity Refinement

Building edges often appear as great disparity changes in disparity map as well as great intensity changes in images. Hence, we define matched straight lines in section 2.2 as potential building edges. However, due to textures in images, not all straight lines are building edges. Hence, we firstly detect the edges from these straight lines by comparing disparities on both sides of straight lines. As the disparities of pixels around straight lines may not be accurate, we need to define rectangular region centred at each straights line as support window to guarantee enough accurate disparities involved in the disparity comparison. Only straight lines with great disparity changes (normally larger than 1 pixel) are defined as building edges. Then, we compute disparity planes for both sides of support window by plane-based adjustment, where matched straight line is used as constraints in the adjustment of the side that contains it, and finally use the disparity plane to refine disparities in both sides of the support window such that the building edges in disparity map are sharp. In general, we organize this sub-section by firstly introducing the support window of straight lines (section 2.3.1), then the edge detection (section 2.3.2) and finally the plane-based adjustment optionally constrained by matched straight lines (section 2.3.3).

2.3.1 Support Window of Straight Lines

In order to guarantee enough accurate disparities used in the disparity refinement, we define a support window centered at straight line in the basic image (e.g. left image) with its length parallel to the straight line and its width perpendicular to straight line, as shown in Figure 4. The four corners (***c***1, ***c***2, ***c***3 and ***c***4) of the support window are computed from the two endpoints (***e***, ***e'***) of the straight line and the width of the support window, as shown in Equation (4).

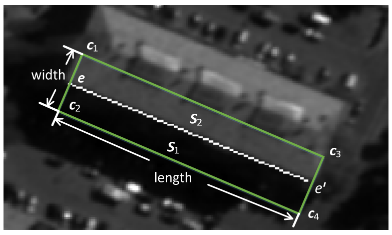


Figure 4. Support window definition. White straight line is an edge of a roof. Green rectangle is a support window centered at the straight line. ***S***1, ***S***2 are sets of pixels in both sides of the support window. (***e***, ***e'***) are endpoints of the straight line. ***c***1, ***c***2, ***c***3 and ***c***4 are four corners of the support window.

$$\begin{aligned} x_1 &= e_x - r \cdot n_x & y_1 &= e_y - r \cdot n_y \\ x_2 &= e_x + r \cdot n_x & y_2 &= e_y + r \cdot n_y \\ x_3 &= e_x' - r \cdot n_x & y_3 &= e_y' - r \cdot n_y \\ x_4 &= e_x' + r \cdot n_x & y_4 &= e_y' + r \cdot n_y \end{aligned} \tag{4}$$

where (x_i, y_i) = image coordinates of the four corners $\boldsymbol{c}_i$
(e_x, e_y), (e_x', e_y') = image coordinates ***e*** and ***e'***
r = half of the width of support window
(n_x, n_y) = an unit vector perpendicular to the straight line

Given a support window, we record pixels and their attributes (intensities or color, image coordinates and disparities) in the both sides (e.g. ***S***1 and ***S***2 in Figure 4) respectively for disparity refinement. The choice of the width of the support window is important for the final disparity enhancement result. Support window with a too small width (< 10 pixels) may not include sufficient number of pixels with accurate disparities, leading to failure of disparity refinement, while support window with a too large width (> 40 pixels) may not satisfy the plane assumption of the local surface. In our experiments, we chose width = 20 pixels as a sufficiently large number to recover erroneous edges, while not too large to violet the local plane assumption.

2.3.2 Edge Detection

Building edges in disparity map often appear as straight lines in images, while not all straight lines are building edges. Some of the straight lines are only textures in a plane, where global stereo dense matching methods (e.g. SGM (Hirschmuller, 2008), graph cuts (Kolmogorov and Zabih, 2001)) normally handle such case well, while utilizing such edges is necessary and may break the smoothness on the plane. Therefore, we tend to filter out such lines by detecting geometric edges through comparing disparities on both sides of support window. Straight lines are located by identifying those with obvious disparity changes on both sides (> 1 pixel). However, due to potential mismatches in the support window, the disparity comparison result may not be accurate. Based on the fact that the intensities of mismatches are often highly different from the one of major pixels in the support window (intensities difference larger than 10), we reduce the impact of these mismatches by firstly computing a major intensity (the intensity of major pixels) and then only considering pixels with intensities similar to the major intensity (≤10) in the edge detection.

In this paper, the major intensity is computed by selecting the median value of intensities of all pixels. However, it is possible that the intensities of edge pixels are not similar to either side of the support window, which may impact on the median values, as shown in Figure 5. Figure 5(a) is a satellite image of a roof. (enlarged region of the red shown in Figure 5(b)). We record the intensities of pixels along the green arrow in Figure 5(b) to check the intensity changes (Figure 5(c)). The intensity of upper region in (b) is 70, and the intensity of lower region is 7. However, the transitional intensities between the two dashed lines in Figure 5(c) are similar to neither 70 nor 7. These transitional intensities are actually the noises in the mid-value selection. Hence, we remove these transitional intensities before the selection. In all our experiments, we only remove intensities of pixels whose distance to the straight line is less than 2 pixels.

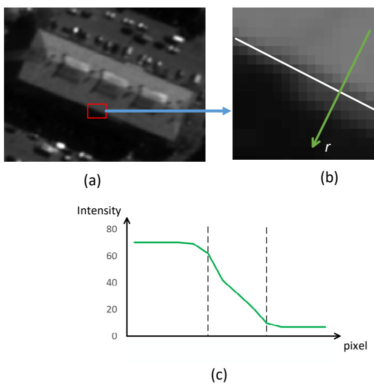


Figure 5. Intensity changes in edges. In (b), the white line is the straight line, and the green arrow is perpendicular to the straight line. In (c), the green line represent the intensities of pixels along the green arrow in (b).

Then, on each side, we only select disparities of pixels with intensities similar to the major intensity for disparity comparison, while it is possible that the pixels with similar intensities are still noises. In this paper, we combine the disparity selection and the noise elimination together, and formulate the combination as a median filter weighted by intensities of pixels in each side. The basic idea of the weighted median filter is to record more disparities with intensities similar to the major intensity (higher weights) and less disparities with intensities less similar to major intensity (lower weights), and then find the median value from all recorded disparities such that those noises can be removed, as shown in Equation (5). In this paper, we use Gaussian kernel to define the weight, which is inversely proportional to the absolute difference between the intensity of pixels and the major intensity of the side.

$$d_m(S) = \underset{p\in S}{\operatorname{med}}\{d(\boldsymbol{p})|[M \cdot w(I(\boldsymbol{p}), I(S))]\} \tag{5}$$

where S = a set of pixels in the same side of support window
$d_m(S)$ = the mid-value of disparities in S
Med = median filter operation
$\boldsymbol{p}$ = a pixel in S
$d(\boldsymbol{p})$ = the disparity of $\boldsymbol{p}$
$I(\boldsymbol{p})$ = the intensity of $\boldsymbol{p}$
$I(S)$ = the mid-value of intensities in S
$w(I(\boldsymbol{p}), I(S))$ = the weight of $\boldsymbol{p}$
M = a predefined coefficient
$[M \cdot w]$ = the number of recorded $d(\boldsymbol{p})$
[ ] = Rounding operator

In this paper, $d_m(S)$ is termed as major disparity, and $I(S)$ is termed as major intensity. After weighted median filter in Equation (5), the major disparities in both sides of support window are then compared. Only the straight line with obvious disparity changes (> 1 pixel) are selected as edges.

2.3.3 Plane-based Adjustment Optionally Constrained by Matched Straight Lines

We use plane functions (Equation (6)) to adjust the disparities in both sides of support window such that the selected edges are sharp.

$$d = ax + by + c \tag{6}$$

where d = disparity of a pixel
(x, y) is the image coordinate of a pixel
a, b, c are parameters of plane function.

Our task is to accurately compute the plane parameters by leveraging the inconsistent disparities (e.g. mismatches).

To compute the optimal plane parameters, we applied a weighted least squares method. For both sides of a building edge, normally only one side contains the edge. Hence, we deal with the two sides separately. For side with no edge, we only apply a plane-based least squares method to refine it, while for side that contains the edge, we additionally introduce a matched straight line based constraint in the least squares adjustment to improve the refinement result. Hence, we firstly compute the disparity of the straight line by averaging the disparities of the two endpoints and then judge which side contains the matched straight line by comparing the disparity of matched straight line and the major disparity of each side, as follows:

$$L\begin{cases}\in S & if\ |d_m(S) - d(L)| \le \tau \\ \notin S & otherwise\end{cases} \tag{7}$$

where L = the straight line in the basic image (e.g. left image)
$d(L)$ = disparity of L
S = a side of support window
τ = a threshold

τ is used to measure the difference between the disparity of straight line and the major disparity of S. The straight line L belongs to side S as long as the disparity difference is smaller than τ. In all our experiment, τ is set as 3 pixels.

For computation purposes in the adjustment, straight line L is formulated as a linear equation (Equation (8)). The corresponding line in the other image is defined as L'. It is worth noting that the endpoints of horizontal straight lines (parallel to image row directions) may not be correspondences. The disparities of such straight lines are unreliable. Hence, we do not consider such straight lines as the additional constraint in adjustment, and formulate the non- horizontal straight lines as linear equations with row direction as independent variable and column direction as dependent variable:

$$x = ky + h \tag{8}$$

where x = image column coordinate
y = image row coordinate
k, h = parameters of the linear equation.

The linear parameters k and h can be computed easily by the two endpoints of the straight line. After deciding which side contains the straight line, we introduce an iterative least square method with variable weights to compute optimal plane of each side. We utilize a matched straight line based constraint in the adjustment of the side containing the straight line, and do not use such constraint for the side with no straight line. Given the set of *n* pixels in one side of support window $S = \{(x_1, y_1, d_1, I(x_1, y_1)), (x_2, y_2, d_2, I(x_2, y_2)), \ldots, (x_n, y_n, d_n, I(x_n, y_n))\}$ with *x*: column coordinate, *y*: row coordinate, *d*: disparity, *I*: intensity or color, we can list a set of disparity plane functions:

$$\begin{gathered} d_1 = ax_1 + by_1 + c \\ d_2 = ax_2 + by_2 + c \\ \vdots \\ d_n = ax_n + by_n + c \end{gathered} \tag{9}$$

For side with no straight line, we only use Equation (9) to compute the disparity plane parameters, and transform the equation in a matrix form:

$$V = AX - L \tag{10}$$

where V = a vector of least square residuals
X = an 3x1 unknown vector containing $\{a, b, c\}$
A = a nx3 coefficient matrix as follows:

$$A = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ & \vdots & \\ x_n & y_n & 1 \end{pmatrix}$$

L = a nx1 constant vector as follows:

$$L = (d_1, d_2, \ldots, d_n)^{\mathrm{T}}$$

For side containing straight line, we use matched straight line based constraint in the adjustment. Combining Equation (6) and (8), we can deduce a disparity function along the straight line:

$$d = (ak + b)y + (ah + c) = my + t \tag{11}$$

where m and t are parameters of the disparity function along straight line.

Given image coordinates and disparities of endpoints of straight lines, these parameter m and t can be computed easily. Hence, we can get two new constraints based on matched straight line:

$$\begin{gathered} m = ak + b \\ t = ah + c \end{gathered} \tag{12}$$

Add Equation (12) into Equation (9), and then transform them in matrix form, and we can get new forms of A and L for side containing straight line:

$$\begin{gathered} A = \begin{pmatrix} x_1 & y_1 & 1 \\ & \vdots & \\ x_n & y_n & 1 \\ k & 1 & 0 \\ h & 0 & 1 \end{pmatrix} \\ L = (d_1, d_2, \ldots, d_n, m, t)^{\mathrm{T}} \end{gathered} \tag{13}$$

After deciding A and L, we need to evaluate the reliability of the matching result of each pixel such that we can reduce the contribution of unreliable pixels (e.g. mismatches) in the adjustment. This helps us to compute an optimal plane. In general, we give each pixel a weight that measures the reliability of this pixel. A large weight means high reliability of the pixel which must contribute more to the error function in the adjustment, and vice versa. As matched straight lines are always more reliable than pixels, we use matched straight lines as controls with high weight (Chen and Shao, 2013).

In this paper, we apply a iterative least square method to solve the plane parameters, and the computation of weights of each pixel is different in the first iteration and the later iteration of the least squares. In the initial iteration, we assume that the major disparity is reliable, and give high weights to pixels whose disparities are close to the major disparity. Based on this assumption, we define the weight of each pixel as the difference between the original disparity of this pixel and the major disparity in the initial iteration:

$$w^0(x, y) = \exp(-|d(x, y) - d_m(S)|/\sigma_d) \tag{14}$$

where $w^0(x, y)$ = an initial weight of pixel (x, y)
$d(x, y)$ = an original disparity of pixel (x, y)
$d_m(S)$ = a predominant intensity of the side S
σ_d = a smoothing factor

σ_d controls the difference between the two disparities. When the disparity of pixel is similar to the major disparity, the weight w^0 is high, and vice versa. In all our experiments, σ_d is set as 3.

After then, plane parameters can be computed by least squares method (least squares method will be described later). In the later iteration, the weight of each pixel is computed by considering the difference between the original disparity of pixel and the disparity computed from plane parameters (Equation (15)).

$$w^i(x, y) = exp(-|d^0(x, y) - d^{i-1}(x, y)|/\sigma_d') \tag{15}$$

where w^i = a weight in the i th iteration
d^0 = the original disparity of pixel
d^{i-1} = refinement results in the i – 1 th iteration
σ_d' = a disparity smoothing factor

σ_d' is adjusted adaptively according to the least squares result in the previous iteration. In this paper, we define σ_d' as the the 1.5 times of the standard deviation of the least squares results in the previous iteration.

For straight lines, the weight of them are set as 1, due to their high reliability. After deciding the weights of all pixels and straight lines, we combine all weights as a weight matrix in each iteration.

For the side with no straight line, the weight matrix is a nxn matrix as follows:

$$W = \begin{pmatrix} w(x_1, y_1) & 0 & \ldots & 0 \\ 0 & w(x_2, y_2) & \ldots & 0 \\ & & \vdots & \\ 0 & 0 & \ldots & w(x_n, y_n) \end{pmatrix} \tag{16}$$

where W is a weight matrix
w is a weight of a pixel in the support window

The choice of w is different during iteration. It is w^0 (Equation (14)) in the initial iteration and w^i (Equation (15)) in the ith iteration.

For side containing straight line, the weight of each straight line is high, and the weight matrix is a (n+2)x(n+2) matrix as follows:

$$W = \begin{pmatrix} w(x_1, y_1) & 0 & \dots & 0 \\ 0 & w(x_2, y_2) & \dots & 0 \\ & \vdots & & \\ 0 & 0 \dots & & 1\ 0 \\ 0 & 0 \dots & & 0\ 1 \end{pmatrix} \tag{17}$$

In each iteration, we use least squares method to compute the plane parameters:

$$X = (A^T W A)^{-1} (A^T W L) \tag{18}$$

Then, we use the plane parameters to compute weights of each pixel, and reassign low weights to unreliable pixel in the next iteration to further refine the parameters. The refinement proceeds iteratively until the weighted average distance between original disparity and the disparity computed by plane parameters is below a predefined threshold:

$$\begin{gathered} \overline{d} \le \tau_d \\ \overline{d} = \sum_{(x,y)\in S} w^i(x,y) \cdot |d^0(x,y) - d^i(x,y)| / n \end{gathered} \tag{19}$$

where $\overline{d}$ is the average distance
d^0 is an original disparity
d^i is refinement results in the *i*th iteration
τ_d is a predefined distance threshold

After deciding plane parameters, we use these plane parameters to refine the disparities of pixels in the same side. However, these refined disparities may not consistent with disparities beyond support window. In order to acquire smooth continuous refinement results, we develop a weighted fusion between original disparities and the refined disparities in the support window with distance from pixel to the straight line as weights, as shown in Equation (20). The weight of original disparities become high with the longer distances such that the disparities are continuous between pixels in/beyond support window.

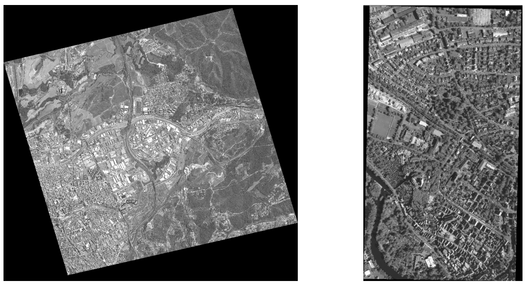
$$\begin{aligned} & d'(x,y) \\ & = w'(x,y) \cdot d^0(x,y) + (1 - w'(x,y)) \cdot d^p(x,y) \\ & w'(x,y) \\ & = \exp(-|d^0(x,y) - d^p(x,y)|/\sigma) \cdot Dis(x,y,L) \end{aligned} \tag{20}$$

where d' = the final disparity
d^p = the disparity computed by plane paraemters
w' = the weight for fusion
σ = a smooth factor
$Dis(x,y,L)$ = distance from pixel (x, y) to L

σ is used to measure the difference between d^0 and d^p.

3. EXPERIMENTS

We evaluate our proposed method on satellite images and aerial image, particular on urban scenes with dense urban objects. The ground truth data are available as LiDAR (Light detection and Ranging) based DSM. Before applying our proposed method, we rigorously rectified both image pairs into epipolar images. Then, we utilized RSP software (Qin et al., 2014) to compute initial disparity maps from these epipolar images, and applied our proposed method to obtain sharp disparity map. The originally matched disparity maps and the sharp disparity maps were then used to generate digital surface models (DSM) by triangulation. To accurately evaluate the improvement of our proposed method, we compare the results with LiDAR derived DSM. Finally, we compared the initial disparity map derived DSM (iDSM) and the sharp disparity map derived DSM (sDSM) with the LiDAR derived DSM (lDSM) to evaluate the improvement of our proposed method. Both of these two datasets are from ISPRS Benchmark, including Wordview-1 stereo images of Terressa with a GSD (ground sampling distance) of 0.5 meters (Figure 6(a)) and a stereo pair of Vaihingen aerial images with an average of 9 cm GSD (Figure 6(b)). The evaluation of satellite data is shown in section 3.1, and the evaluation of aerial data is shown in section 3.2.

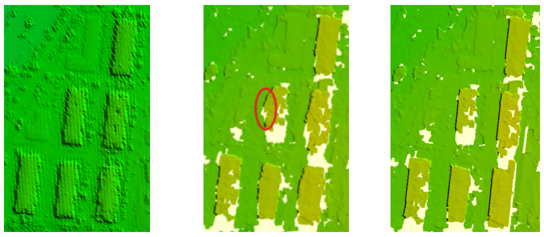

(a) Satellite dataset (b) Aerial dataset
Figure 6. The experimental dataset.

3.1 Evaluation on Satellite Data

We used WorldView-1 epipolar images in Terrassa (Figure 6(a)) and the corresponding LiDAR points provided by Singapore-ETH Centre for Global Environmental Sustainability (SEC) to test the edge refinement of our proposed method. 21454 straight lines are matched and 8346 of them are identified as edge lines. We generated iDSM and lDSM from these data, and then applied our proposed method to generate sDSM. The ground sampling distance (GSD) of iDSM and sDSM is 0.5m, and the GSD of lDSM is 1m. The sDSM and the iDSM were compared with the ground truth lDSM. To give a quantitative evaluation on our proposed method, we only considered points from support window into the accuracy measurement. In this paper, we defined the accuracy as the average of absolute differences between elevations in lDSM and iDSM/sDSM of pixels from all identified building edges. The accuracy of iDSM is only 4.83 m, while the accuracy of sDSM is improved to be 4.33 m after the refinement. The accuracy improvement shows that our proposed method can refine some mismatches around building edges, resulting in sharper and more accurate DSM.

To make a more comprehensive evaluation on our proposed method, we also selected several buildings with erroneous edges in iDSM, and then compared them with our refinement results, as shown in Figure 7. Figure 7(a-1) and (b-1) are buildings in images with very clear edges. Figure 7(a-2) and (b-2) are buildings in lDSM, while the edges are not clear, due to the low GSD of lDSM. Figure 7(a-3) and (b-3) are buildings in iDSM with overestimated edges (e.g. red circle in Figure 7(a-3)) and underestimated edges (e.g. blue circle in Figure 7(b-3)).

(a-1) image


(a-2) lDSM (a-3) iDSM (a-4) sDSM
(a) Case 1: overestimated edges

(b-1) image

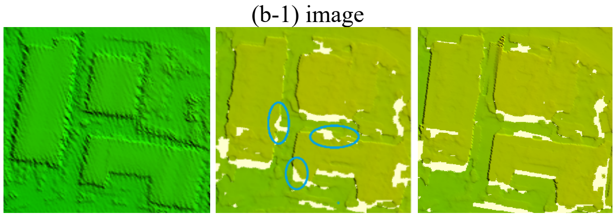

(b-2) lDSM (b-3) iDSM (b-4) sDSM
(b) Case 2: underestimated edges

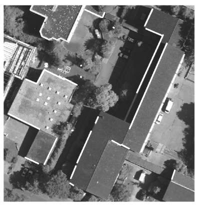
Figure 7. Examples of buildings in different DSMs from satellite images and corresponding LiDAR points. White lines in (a-1) and (b-1) are extracted straight lines. Red circle in (a-3) is over-estimated edge. Blue circle in (b-3) is underestimated edge.

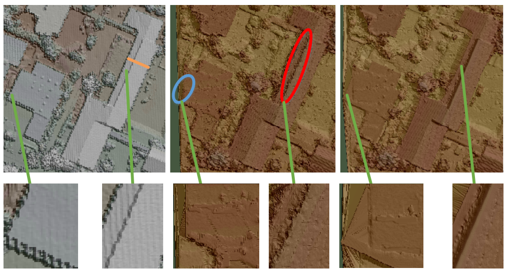
Building edges in Figure 7(a-2) and (b-2) were mosaic, due to the low resolution of lDSM or LiDAR points. The edges in Figure 7(a-3) and (b-3) are erroneous with over-estimated or underestimated cases, which are common in global matching. Our results in Figure 7(a-4) and (b-4) show that our method is capable of adjusting both over-estimated edges and underestimated edges to the straight lines in images, thus computing sharper edges than iDSM. As the GSD of sDSM is higher than lDSM, the refined building edges are even sharper than lDSM.However, not all edges in sDSM are sharp, since not all the edge lines were extracted and matched, and those edges without corresponding straight lines cannot be refined by our proposed method.

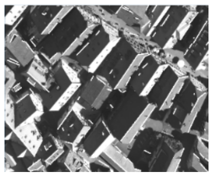
3.2 Evaluation on Aerial Data

In this experiment, we applied our method on aerial images in Vaihingen (Figure 6(b)). The images were captured by an Intergraph / ZI DMC camera. The corresponding LiDAR points were captured by a Leica ALS50 system. Both data were provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) (http://www.ifp.uni-stuttgart.de/dgpf/ DKEP-Allg.html). We generated iDSM from epipolar images and lDSM from corresponding LiDAR points, and then applied our proposed method to generate sDSM. The GSD of iDSM and sDSM is 0.087m, and the GSD of lDSM is 0.25m. 12372 straight lines were matched and 5950 straight lines are identified as edges in our method. The accuracy of iDSM is only 3.41 m, while the accuracy of sDSM is improved to be 3.17 m after the refinement, on average 0.24-0.5 meters of improvement. The accuracy improvement shows that our proposed method is also capable of sharpening building edges in aerial datasets.

Similar to experiments in section 3.1, we also illustrated visual comparisons of lDSM, iDSM and sDSM. Cases of different buildings in lDSM, iDSM and sDSM are shown in Figure 8.


(a-1) Image


(a-2) lDSM (a-3) iDSM (a-4) sDSM
(a) Case 1: different types of edges


(b-1) images

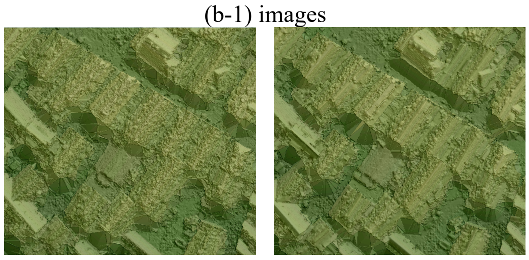

(b-2) iDSM (b-3) sDSM
(b) Case 2: building ridge

Figure 8. Examples of buildings in different DSMs from aerial images and corresponding LiDAR points. White lines in (a-1) and (b-1) are extracted straight lines. Red circle in (a-3) is an overestimated edge. Blue circle in (a-3) is an underestimated edge.

Figure 8(a) shows the case of overestimated and underestimated edges in iDSM from aerial data, where edges in lDSM (Figure 8(a-2)) are irregular due to the low GSD of lDSM. Figure 8(a-3) shows that the over-estimated edges (red circle) and underestimated edges (blue circle) are obvious in iDSM regardless of high GSD of iDSM. The sDSM in Figure 8(a-4) is sharp, which shows that our proposed method can also be applied to aerial images and compute sharp edges of buildings. We also draw the profile maps along the yellow line in (a-2) for each DSM, as shown in Figure 9.

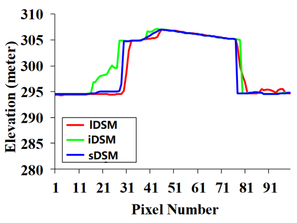


Figure 9. Profile maps of the three DSMs. The three profiles of LDSM, iDSM and sDSM are corresponding to the yellow line in Figure 8(a-2).

The yellow line in 8(a-2) is a gable roof with disparity jumps in both sides, while the profile of iDSM (green line) is continuous in the left side of the roof. The profiles of lDSM and sDSM are similar with disparity jumps in both sides. In general, sDSM is sharper, due to higher GSD.

Figure 8(b-1) shows that several ridges of gable roofs are extracted as straight lines. These building ridges are also irregular in iDSM (Figure 8(b-2)). For gable roofs, the disparities of both sides of the ridges are not consistent. Hence, our proposed method also detected these ridges as edges and refined these ridges to sharper ones. The refinement results of our proposed method is shown in Figure 8(b-3).

4. CONCLUSION

In this paper, we propose a disparity refinement method for building edges using robustly matched straight lines, to refine building edges to sharper ones. We formulate the refinement of building edges as the non-local plane-based least squares adjustment constrained by matched straight lines. The main contributions of our method include: 1) we develop a robust matching method of straight lines constrained by epipolar geometry; 2) we fully utilize the robustly matched straight lines and design a new matched straight line based constraint in the disparity refinement; and finally 3) we develop an iterative least squares method with variable weights to solve the disparity refinement robustly. Experiments on satellite and aerial images demonstrated that our proposed method is able to compute sharp building edges as well as ridges. We also observed that there were still inconsistent disparities around boundaries of support window after the weighted fusion. In our future work, we plan to address such case by considering more edge-aware filtering methods.

ACKNOWLEDGEMENTS

The study is partially supported by the ONR Basic Research Grant (Award No. N000141712928). We would like to thank the ISPRS benchmark, German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) and the Singapore-ETH Centre for Global Environmental Sustainability (SEC) for providing the WorldView-1 dataset of Terressa and Vaihingen data set.

REFERENCES

Chen M., Shao Z., 2013. Robust Affine-Invariant Line Matching for High Resolution Remote Sensing Images. *Photogrammetric Engineering & Remote Sensing,* 79(8), pp. 753-760.

Fan, B., Wu, F., Hu, Z., 2010. Line matching leveraged by point correspondences. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 390–397.

Gupta, R. K., & Cho, S. Y., 2010. A color-based approach for disparity refinement. In: International Conference on Control Automation Robotics & Vision, pp.664-667.

Hirschmuller, H., 2008. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2), pp. 328-341.

Huang, X., & Zhang, Y., 2016. An O(1) disparity refinement method for stereo matching. *Pattern Recognition*, 55, pp. 198-206.

Klaus, A., Sormann, M., Karner, K., 2006. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: Proceedings of the Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, pp. 15-18.

Kolmogorov, V., Zabih, R., 2001. Computing visual correspondence with occlusions using graph cuts. In: Proceedings of the Computer Vision, 2001. In: ICCV 2001. Proceedings. Eighth IEEE International Conference on, pp. 508-515.

Li, H., Zhang, X. G., & Sun, Z., 2015. A line-based adaptive-weight matching algorithm using loopy belief propagation. *Mathematical Problems in Engineering*, 2015(3), pp. 1-13.

Lin, C. H., & Liu, C. W., 2015. Accurate stereo matching algorithm based on cost aggregation with adaptive support weight. *Journal of Photographic Science*, 63(8), pp. 423-432.

Lin, Y., Yang, R., Gabbouj, M., & Neuvo, Y., 2002. Weighted median filters: a tutorial. *IEEE Transactions on Circuits & Systems II Analog & Digital Signal Processing*, 43(3), pp. 157-192.

Paris, S., 2008. Bilateral filtering : theory and applications. *Foundations & Trends® in Computer Graphics & Vision*, 4(1), pp. 1-74.

Park, S. H., Park, M. G., & Yoon, K. J., 2015. Confidence-based weighted median filter for effective disparity map refinement. In: International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 573-575.

Qin, R., Gruen, A., & Fraser, C., 2014. Quality Assessment of Image Matchers for DSM Generation - A Comparative Study Based on UAV Images. *Asian Conference on Remote Sensing*.

Scharstein, D. and Szeliski, R., 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3), pp. 7-42.

Von Gioi R. G., Jakubowicz J., Morel J.-M, et al., 2012. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4), pp. 722-732.

Wang, Y. C., Tung, C. P., & Chung, P. C., 2013. Efficient disparity estimation using hierarchical bilateral disparity structure based graph cut algorithm with a foreground boundary refinement mechanism. *IEEE Transactions on Circuits & Systems for Video Technology*, 23(5), pp. 784-801.

Wu, W., Li, L., & Jin, W., 2016. Disparity refinement based on segment-tree and fast weighted median filter. In: IEEE International Conference on Image Processing, pp.3449-3453.