Volume xx (200y), Number z, pp. 1–14

Extracting Sharp Features from RGB-D Images

Y-P. Cao[†1] and T. Ju[2] and J. Xu[1] and S-M. Hu[1]

[1]Tsinghua University, China
[2]Washington University in St. Louis, USA

Abstract

Sharp edges are important shape features and their extraction has been extensively studied both on point clouds and surfaces. We consider the problem of extracting sharp edges from a sparse set of color-and-depth (RGB-D) images. The noise-ridden depth measurements are challenging for existing feature extraction methods that work solely in the geometric domain (e.g., points or meshes). By utilizing both color and depth information, we propose a novel feature extraction method that produces much cleaner and more coherent feature lines. We make two technical contributions. First, we show that intensity edges can augment the depth map to improve normal estimation and feature localization from a single RGB-D image. Second, we designed a novel algorithm for consolidating feature points obtained from multiple RGB-D images. By utilizing normals and ridge/valley types associated with the feature points, our algorithm is effective in suppressing noise without smearing nearby features.

1. Introduction

Sharp features carry important structural and semantic information of a surface. They are useful in a variety of downstream geometric processing tasks, including segmentation [dGGDV11, CYW15], simplification [KKK06], stylization [XCJ*09, PKG03], and deformation [GSMCO09]. Man-made objects, in particular, can be well described by the network of sharp features alone (together with surface normals along them) [MZL*09]. As a result, extensive research has been conducted on robust detection of sharp features on discrete surfaces as well as unorganized point clouds (see next section for a brief review).

Driven by their low prices and compact sizes, consumer-level color-and-depth (or RGB-D) sensors, such as Microsoft Kinect, are becoming increasingly popular in 3D scanning and modeling [CLH15]. With the advance in registration and reconstruction algorithms, highly detailed geometry can be recreated from continuous RGB-D video streams (e.g., [NIH*11]). However, given only a few RGB-D images (e.g., Figure 1 (a)), it remains difficult to reconstruct a clean and feature-preserving point cloud or surface. Such difficulty, in turn, hinders the detection of sharp features using current geometry-based methods (e.g., Figure 1 (d,e)).

We propose a novel method for extracting sharp features from RGB-D data. Unlike existing methods that are solely based on geometry, we exploit the complementary information in both color and depth channels to produce less noisy and more coherent features. Sharp features are located in regions that exhibit abrupt changes in surface normals. Due to the inherent noise and incompleteness in the depth map, it is difficult to robustly estimate surface normals from the depth points alone, not to mention the variation of normals. On the other hand, on a diffuse and texture-less surface in a shadow-free environment, color changes are directly correlated with normal variations. Motivated by these observations, for each RGB-D image, we combine prominent 2D edges in the color channel with 3D depth points to more accurately localize feature points (Figure 1 (b)). The combination of the two channels also allows our method to work even for textured surfaces (e.g., the floor tiles in Figure 1 (a)) and complex lighting conditions (e.g., shadow on the floor). If multiple RGB-D images capturing an object from different views are available, we merge feature points obtained from each RGB-D image to form a more complete set of feature curves (Figure 1 (c)).

Contributions To the best of our knowledge, our method is the first sharp feature extraction method that harvests both color and depth information in multiple RGB-D images. Besides proposing a two-stage framework (Figure 1 (b,c)), we make the following technical contributions at each stage:

1. *Feature extraction from single RGB-D image (Section 3):* We show that edges in the color channel can augment the depth map to improve both the estimation of normals and localization of feature points.
2. *Feature merging from multiple RGB-D images (Section 4):* We design a method for consolidating feature points equipped with surface normals and feature types. Thanks to a novel definition of feature affinity, our method can handle large amount of noise without smearing nearby features.

† caoyanpei@gmail.com

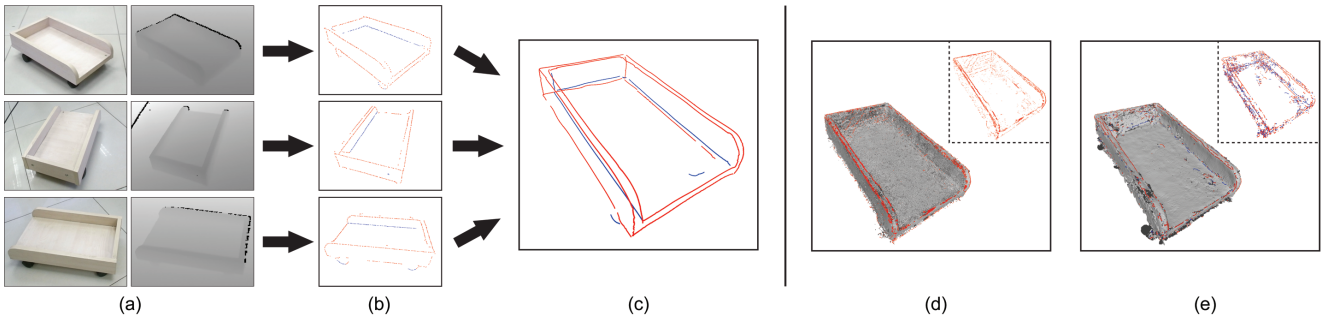

Figure 1: *Work flow of our method (a,b,c) and comparison with existing geometry-based feature detection methods (d,e). Given an input sequence of 10 RGB-D images of the Cart captured by Kinect (a, showing three images), our method first extracts 3D feature points from each image (b, red for ridge points and blue for valley points), then consolidates feature points obtained from all images and forms feature lines (c). By exploiting both intensity and depth information, our result is significantly less noisy and more coherent than features detected directly on the registered point cloud (d, using [MOG09]) or on the reconstructed surface (e, using [YBS05]). The point cloud in (d) is registered by [HKH*10] and has been filtered by Edge Aware Resampling [HWG*13]. The surface in (e) is reconstructed using a feature-preserving method [OGG09] from the filtered point cloud.*

2. Related works

In the following, we briefly review previous works relevant to our goal (sharp feature extraction) and technical contributions (RGB-D feature analysis and point consolidation).

Sharp feature extraction Sharp features on a surface are closely related to ridges and valleys, which are the loci where the maximum (or minimum) curvature is maximal (or minimal) along the maximal curvature directions. Given a discrete mesh representation, extracting ridges and valleys relies on the estimation of differential properties such as normals and curvatures, which can be obtained by globally [OBS04] or locally [HG01, YBS05, KK06] fitting a polynomial or employing discrete differential operators [HPW05, YBYS07]. Besides ridges and valleys, other types of line features on surfaces have also been proposed that consider either local geometry [KST08] or in conjunction with other aspects such as view direction [DFRS03] or perceptual saliency [WB01].

If the input is an unorganized point cloud, methods have been proposed to extract features directly from the points without the need for surface reconstruction. Without a mesh structure, several methods [GWM01, PKG03, DVVR07] obtain shape information within a spatial neighborhood of a point using principal component analysis (PCA). PCA provides the directions and magnitude of principal shape variation, from which decision can be made regarding whether the point is a feature and, if so, the type of the feature (e.g., a sharp edge or a corner). Continuous feature lines are then formed by graph-based methods, such as constructing and pruning a minimal spanning tree. Beside PCA, local shape information can also be learned using the geometry of Voronoi cells [MOG09], clustering estimated point normals [WHH10], or fitting truncated Fourier series [AMMK13]. While these methods use existing points as candidates of feature points, others (like us) create new feature locations. Daniels et al. [DHOS07] project particles towards the intersections of estimated planes around the feature, while Lee and Bo [LB15] first fit smooth developable surfaces to the points and then construct feature curves as the surface intersections.

The majority of the mentioned feature extraction methods works by examining the local geometry, which can make them sensitive to high-frequency noise. To improve the quality of results on noisy inputs, one can first filter the point cloud or surface to suppress noise while retaining the salient features. To this end, various feature-preserving filtering methods have been proposed. For point clouds, Lipman et al. [LCOLTE07] introduced the locally optimal projector (LOP) that projects particles towards local L_1 median of the points. Variants of the method were proposed to achieve a more even distribution [HLZ*09] or better preserve sharp features [HWG*13]. Recently Sun et al. [SSW15] proposes a feature-aware denoising method based on L_0 minimization that maximizes sparsity. For meshes, impressive feature-preserving filtering results have been achieved using bilateral filtering [FDCO03, ZFAT11], mean curvature flow [HP04], and L_0 minimization [HS13]. Piece-wise smooth surfaces with well-defined sharp features can also be directly reconstructed from noisy point clouds using methods such as moving-least squares [FCOS05], kernel regression [OGG09], or L_1 minimization [ASGCO10].

Geometric features from RGB-D Due to depth distortion, extracting geometric features directly from RGB-D images is difficult and has received much less attention. Some methods only detect contours that lie at depth discontinuities (also called jump edges, and occluding/occluded edges) [LPVV11, ZK15]. We know of only two methods for detecting sharp geometric features from a single RGB-D image [SLG13, CTC13], which we shall discuss in more details.

Schafer et al. [SLG13] apply 2D Canny edge detector to both depth and intensity channels captured by Time-of-Flight (ToF) cameras. Intensity edges capturing non-geometric features, such as shadow and texture, are pruned by thresholding the variation in depth or in the gradient of depth. Due to the inherent noise in the depth channel and its gradient, it is difficult for this method to produce clean and complete set of intensity edges corresponding to sharp features (see comparison in Figure 5). Another important drawback of this method is that it does not produce 3D locations of feature points.

Working only in the depth channel, Choi et al. [CTC13] select a subset of the depth points whose local neighborhood exhibit large variation in locally estimated normals. However, these normals, and hence the resulting feature points, are sensitive to noise in the depth channel. Furthermore, as depth points rarely lie exactly on a sharp feature, the output is often made up of thick bands of points around the feature (see comparison in Figure 7).

The combination of color and depth channels has been used for segmentation [CLW*14], primitive detection [HM15], and object recognition [KBK15, LJHW15] from RGB-D images. However, none of these methods specifically harvests the complementary information of color and depth for extracting sharp features. On the other hand, the features produced by our method can potentially augment these methods particularly when the inputs contain feature-rich objects.

Point cloud consolidation To combine feature points extracted from multiple images, we draw inspirations from existing methods for consolidating noisy point clouds onto either surfaces [LCOLTE07, HLZ*09, HWG*13, SSW15] or curves [CTO*10, TZCO09,HWCO*13]. Common to these methods is minimizing an energy that penalizes deviation of points from its neighbors as well as non-uniform distribution. Our consolidation of feature points adds to this line of research by additionally considering normals and ridge/valley type that are available to us at the feature points in defining the deviation term, with the goal of better differentiating nearby features.

3. Computing feature points from one RGB-D image

Extracting sharp features directly from the depth channel is challenging due to the significant amount of noise and possible missing data. Our idea is to exploit the additional information provided by the color channel. Our motivating observation is that, in an ideal setting where a diffuse, texture-less surface is lit with ambient lighting, variation in surface normal is directly captured by the change in color values. Since the color channel typically has less distortion than the depth channel, strong edges in the color channel would offer a more direct and accurate indication of (the projection of) sharp features. Although edges in the color channel may also correspond to non-geometric features (e.g., texture, shadow, etc.) in more practical settings, we may utilize the depth information to differentiate such edges from those edges capturing sharp features.

Our method for single-image feature extraction proceeds in four steps, as shown in Figure 2. We start by extracting strong edges from the color channel, which we call *intensity edges* (see (a)). As we expect the surface to be continuous and smooth away from these intensity edges, we can robustly estimate Hermite samples (i.e., 3D positions and normals) from the depth channel within neighborhoods *bounded* by the intensity edges (see (b)). These Hermite samples, in turn, can prune intensity edges that do not correspond to geometric features (see (c)). Finally, the 2D intensity edges are combined with 3D Hermite samples to solve for the 3D feature point locations (see (d)).

We next detail each step in order. Our primary contributions lie in the second step, where we use 2D intensity edges for robustly estimating 3D depths and normals (Section 3.2), and in the last step,

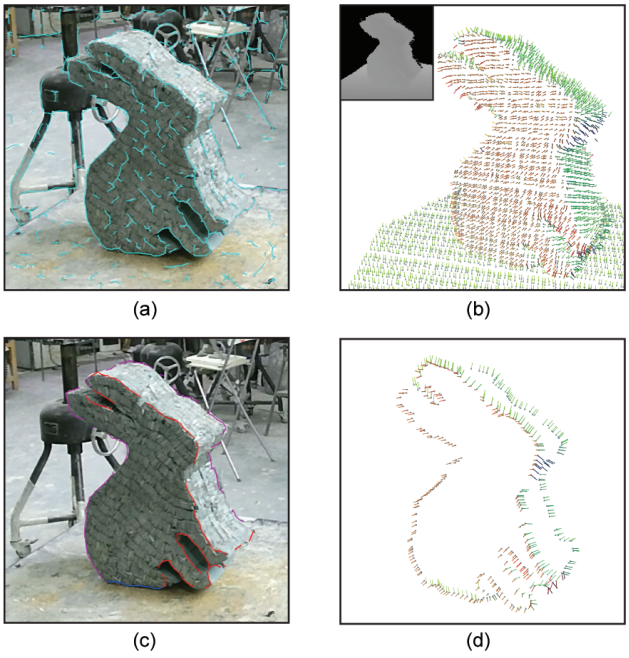

Figure 2: *The four steps in computing features points from a single RGB-D image. (a): 2D intensity edges (cyan) extracted from the color channel. (b): 3D Hermite samples estimated from the depth channel (the depth map is shown in the insert). (c): Pruned 2D intensity edges, classified into ridges (red), valleys (blue), and ridge-contours (purple). (d): Computed 3D feature points with normals. All normals are colored by their directions.*

where we introduce a least-square formulation of feature points that combine both 2D and 3D constraints (Section 3.4).

3.1. Extracting 2D intensity edges

Our computation is performed on a 2D grid structure instead of directly on the pixels, for several reasons. First, the depth channel usually has lower-resolution than the color channel. Second, due to depth errors such as noise, missing data, and misalignment with the color channel, the effective resolution of the depth channel is even lower. We therefore consider a grid where each grid square contains $k \times k$ pixels. The value of k should be large enough to accommodate errors in depth values but small enough to minimize smearing of close-by features. We found that $k = 5$ offers a good balance in our experiments (see Figure 3).

We use the grid-based edge detector of [Lin98] to extract intensity edges. We first compute the intensities at pixels by converting the RGB color to YCrCb color and taking the Y channel (luminance). The detector requires second and third order intensity derivatives at the grid points. To reliably obtain these derivatives, we first perform a Gaussian filtering with a kernel size of 25 pixels over the intensity image (the intensity values are obtained from the RGB values using luminosity). A grid edge $\overline{xy}$ connecting two grid points x,y is considered to *cross* the intensity edge if (1) the second order derivatives in the gradient directions have different signs at x and y, and (2) the sum of

the third order derivatives in the gradient directions at x and y is smaller than a threshold (we use -50 in our experiments). If $\overline{xy}$ crosses the intensity edge, we estimate the crossing point on $\overline{xy}$ (which we call *intensity edge point*) by linear interpolation of the second order derivatives at x and y [Lin98]. Two intensity edge points on a grid square are then connected to form an *intensity edge segment*. See Figure 3 for an illustration.

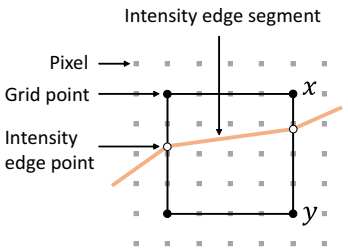

Figure 3: *Grid structure*

3.2. Computing 3D Hermite samples

We next estimate a Hermite sample in 3D for each grid point on the 2D grid structure mentioned above. The estimated positions and normals play critical roles in both pruning non-geometric intensity edges (Section 3.3) and computing the 3D location of feature points (Section 3.4).

A natural way to estimate Hermite samples is to fit planes in a neighborhood of depth points. The key question is the choice of neighborhood. While a small neighborhood is not enough to filter out the noise in the depth points, a large neighborhood may "round off" the positions and normals near a sharp feature. Our solution is to use a *smart* neighborhood that is delimited by the extracted intensity edges. Since these 2D edges contain projections of sharp geometric features and depth discontinuities, restricting plane-fitting to one side of an intensity edge permits the use of larger neighborhood sizes without rounding off the sharp features. This is similar in principle to bi-lateral filtering, except that we use edges in the color channel as barriers for filtering the depth channel.

Specifically, for each 2D grid point x, we gather all depth points whose corresponding pixels fall within a certain distance (we use 10 pixels) from x, and remove those whose straight-line connection with x intersects with any intensity edge segment. We then estimate a plane from the remaining depth points using principal component analysis (PCA). Finally, we obtain a Hermite sample $\{p_x, n_x\}$ where p_x is the intersection of the plane and the view ray through x and n_x is the plane normal oriented towards the camera (i.e., the outward surface normal). We assume that the camera is calibrated and the view ray can be directly obtained from device-specific parameters.

We demonstrate the advantage of our smart neighborhoods in Figure 4 on three regions of the Cart example (second to fourth rows). Compared with small (5 pixel radius) and large (10 pixel radius) neighborhoods that do not use intensity edges, smart neighborhoods results in less noisy Hermite samples that better respect the sharp features and depth discontinuities. Our method can also deal with small amounts of missing data (e.g., box A) and misalignment between the color and depth channels (e.g., box B).

Robust estimation of Hermite samples is important for the remaining steps in our work flow. As shown in Figure 4 (last two rows), Hermite samples estimated by our smart neighborhood result in more complete and coherent 2D intensity features after pruning and cleaner 3D feature points in the end.

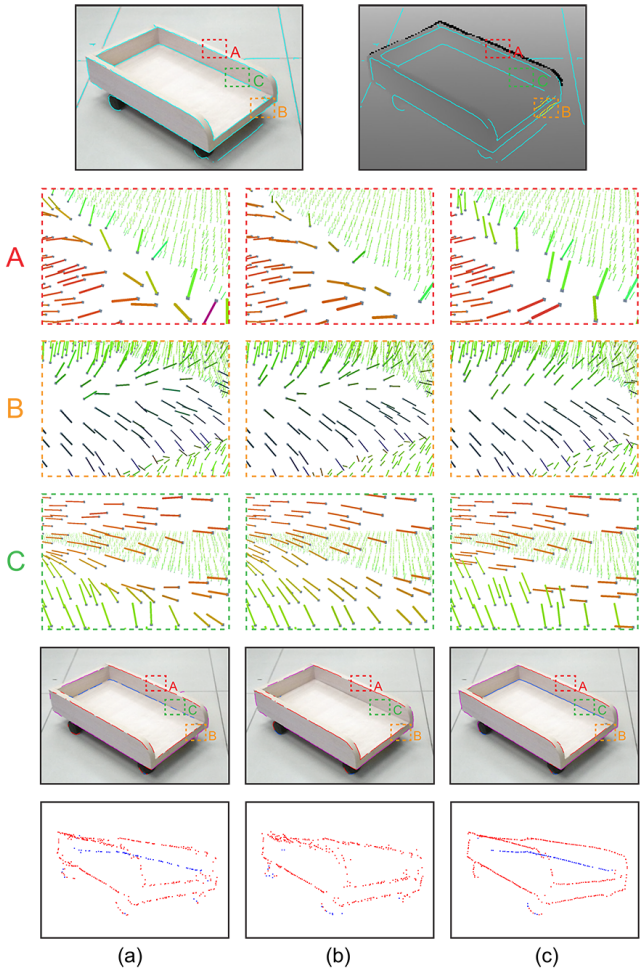

Figure 4: *Comparison of Hermite samples (2nd to 4th rows) estimated from a single RGB-D image of Cart (top row, showing intensity edges in cyan) using small (a), large (b) and smart (c) neighborhoods, and comparison of the pruned intensity edges (5th row) and 3D feature points (6th row) using the Hermite samples. The Hermite samples are shown for three outlined boxes (A,B,C) on the Cart, which contain missing depth information (box A) and misalignment between color and depth (box B; yellow lines in the depth map indicate strong edges in the depth channel).*

3.3. Pruning and classifying 2D intensity edges

An intensity edge may correspond to a sharp feature in two scenarios: either the surface on both sides of the feature is visible or only one side of the surface is visible. In the first scenario, the depth measurements are continuous across the intensity edge but the normals vary significantly. In the second scenario, the intensity edge coincides with depth discontinuity, but it may also correspond to the silhouette of a smooth surface. In the latter case, the normal on

the side of the intensity edge with smaller depth is nearly orthogonal to the view direction.

Based on these observations, we use a multi-step check to determine whether an intensity edge point r on a grid edge $\overline{xy}$ (see Figure 3) corresponds to a sharp feature. Let p_x, n_x and p_y, n_y be the Hermite samples for x and y, and u be the average of the view directions at x and y. First, following [CTC13], we consider r as on a depth discontinuity if the ratio $\|(p_x - p_y)\cdot u\| / \max(\|p_x \cdot u\|, \|p_y \cdot u\|)$ is greater than some constant ε_1. Unlike absolute difference in depth values, this ratio tolerates more depth noise for points further away from the camera, where the depth values are less accurate. If r is on a depth discontinuity, and suppose p_y is closer to the camera, we say r correspond to a sharp feature if $\|n_y \cdot u\|$ is greater than some constant ε_2. If r is not on a depth discontinuity, it corresponds to a sharp feature if $n_x \cdot n_y$ is smaller than some constant ε_3. We set $\varepsilon_1 = 0.025, \varepsilon_2 = 0.1, \varepsilon_3 = 0.7$ in our experiments.

After pruning all intensity edge points that do not correspond to sharp features, we can easily determine the feature types of the remaining points. Since valley lines cannot appear as contours, we label all intensity edge points on depth discontinuities as *ridge-contour* type. The remaining intensity edge points are classified into ridge or valley by comparing the orientation of cross-product of the normals associated with the Hermite samples and the orientation of the 2D intensity edge segments.

Using robust Hermite samples, our method obtains 2D intensity edges that more faithfully capture sharp features than methods that rely on raw depth values, such as [SLG13]. The method of [SLG13] combines depth discontinuity with those intensity edges pruned by thresholding the variation of depths and depth gradients. Figure 5 compares our result (a) with their results under two different settings of their pruning thresholds (b,c). While the lower thresholds leave behind many spurious edges, the higher thresholds remove some edges corresponding to sharp features. In contrast, our method produces cleaner edges that capture the major sharp features, and we further compute 3D feature points (see next section).

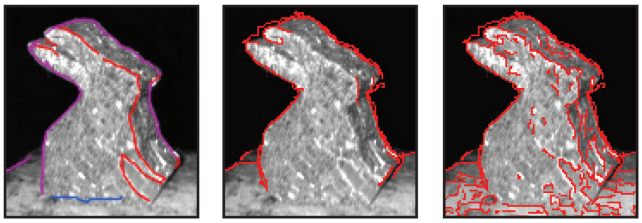

Figure 5: *Comparing pruned 2D intensity edges produced by our method (a) with the result of [SLG13] at two different threshold settings (b,c). As [SLG13] works on IR images, we feed both methods with the same IR image and depth map of the Bunny.*

3.4. Computing 3D feature points

By now we have collected a rich set of constraints about sharp features, both in 3D (Hermite samples, Figure 2 (b)) and over the 2D image plane (pruned and classified intensity edges, Figure 2 (c)). In this final step, we combine these constraints to compute the 3D location of feature points.

We start by exploring local information, namely Hermite samples and intensity edge points within a single grid square, to compute initial positions of the feature points. We then improve the coherence of feature points along a feature curve using iterative filtering. Both steps are solved by minimizing simple quadratic functions that combine 2D and 3D constraints. Finally, we associate each feature point with feature labels (e.g., ridge or valley) and surface normals. These additional data will be utilized in the next stage of our method for robust consolidation of multiple sets of feature points.

Computing initial locations Our objective function for computing the feature locations is guided by two principles. On one hand, the feature point should be close to the planes defined by the Hermite samples. On the other hand, the projection of the feature point onto the image plane should not stray too far from the intensity edges. Consider a grid square (see Figure 3) that has one or more intensity edge points of the same feature type on its edges. We seek a 3D feature location q that minimizes the following quadratic energy,

$$\alpha \sum_{r \in R} (Mq - r)^2 + \sum_{\{p,n\} \in H} ((q-p)\cdot n)^2. \qquad (1)$$

The first term measures the sum of 2D distances from the projection of q (where M is the camera projection matrix) to the set R of all 2D intensity edge points on the square, and the second term measures the sum of 3D distances from q to the planes defined by the set H of 3D Hermite samples for the four grid points of the square (see picture). If the intensity edge points on this square are of ridge-contour type, H is restricted to those grid points that are on the occluding side of the intensity edge segment. The balancing weight, α, is set to 1.5 in our experiments.

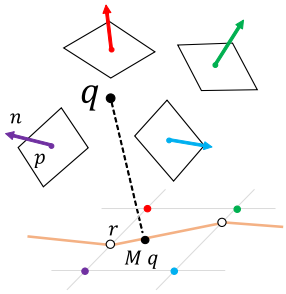

Figure 6: *Notations in Equation 1.*

The importance of using both 2D and 3D constraints is demonstrated in Figure 7 (a,b). Without the 2D term (i.e., setting $\alpha = 0$), the projection of the minimizer of Equation 1 can stray far from the intensity edges in the view plane, and the resulting 3D feature points easily fail to form clean lines (see green boxes in (a)). This is largely due to the imperfect Hermite samples. Adding the 2D term pulls the features points closer to the visible feature in the view plane, producing much more coherent lines in 3D (see green boxes in (b)).

Iterative filtering The initial locations of feature points are computed using local Hermite and intensity information. While points along the same feature curve have a coherent 2D projection (due to the intensity edge constraints), their depth may not be smooth along the curve (see Figure 7 (b) right).

A natural solution to create smoother feature curves would be solving Equation 1 using Hermite samples in a large neighborhood. However, if we treat all Hermite samples in the expanded

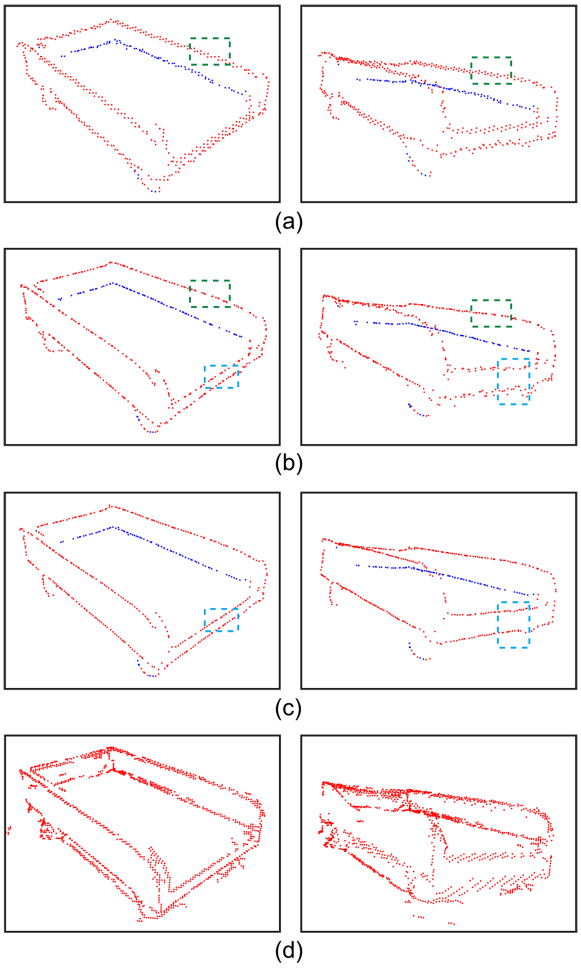

Figure 7: *Compare feature points computed with only 3D constraints (a) (minimizing Equation 1 with $\alpha = 0$), with both 2D and 3D constraints (b) (minimizing Equation 1 with $\alpha = 1.5$), and after iterative smoothing (c) (minimizing Equation 3), shown in the original view direction (left) and from a different angle (right). Green and blue boxes respectively highlight areas of differences between (a,b) and between (b,c). The last row compares with the feature points produced by [CTC13], which only considered the depth values.*

neighborhood equally, the minimizing location could be adversely affected by surface geometry far away from the sharp features. This may lead to distortion of the shape of features, particularly those on curved surfaces.

To minimize feature distortion while improving fairness, our idea is to weigh the contribution of each Hermite sample in the expanded neighborhood by how much it agrees with the initial feature location. The agreement considers both the position p and the normal n of a Hermite sample: it is higher if the initial feature location, q, is closer to p and if q is closer to the plane defined by p and n. Such weighting effectively *adapts* the neighborhood of Hermite samples to the planarity of the local geometry, because less samples will contribute to the objective function for more curved surfaces.

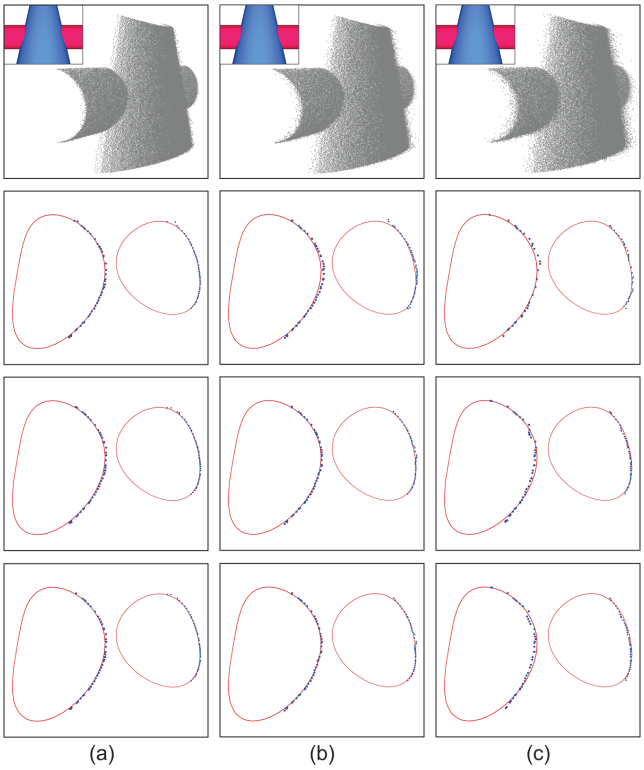

Figure 8: *Evaluating feature points on a synthetic example at three different levels of noisiness, low (a), medium (b), and high (c). Top: Input image and depth map. Second row: Initial feature points. Last two rows: Updated feature points after 3 and 6 iterations of filtering. The red curves indicate the actual intersections between the cone and the cylinder.*

Specifically, we use the Mahalanobis distance to evaluate the disagreement between a point q and a Hermite sample $\{p,n\}$ [LZT*08],

$$h(q,\{p,n\}) = \|q-p+\alpha_h((q-p)\cdot n)n\|, \tag{2}$$

with $\alpha_h = 2$. We build an expanded set of Hermite samples H^+ by tracing a fixed number (e.g., 8) of intensity edge segments from the current grid square and collecting the Hermite samples for the grid points in the traced squares. Given the initial feature location q, we compute an updated feature location q' by minimizing a modified quadratic energy involving weighted contributions of Hermite samples,

$$\alpha \sum_{r\in R}(Mq'-r)^2 + \sum_{\{p,n\}\in H^+} \exp\left(\frac{-h(q,\{p,n\})^2}{\sigma_h}\right)((q'-p)\cdot n)^2 \tag{3}$$

After updating the feature location, its agreement with the Hermite samples changes, and this process can be iterated (i.e., setting $q = q'$ and solving Equation 3 again). In our experiments, we found that the process converges after just a few iterations, and hence we ran three iterations for all our test data. We use $\sigma_h = 0.00125L$ where L is the largest dimension of the scene bounding box.

By considering both 2D and 3D constraints, our iterative filtering

diminishes the jaggedness of the initial feature points while keeping their 2D projects close to the intensity edges (see Figure 7 (c)). We further evaluate the effect of iterative filtering on curved features using a synthetic example in Figure 8. We take two intersecting curved surfaces, a cylinder and a cone, and feed the algorithm with a noise-free RGB image coupled with a noisy depth map (we tested three levels of noisiness). Observe that iterative filtering maintains the shape of the curve at low noise levels, and introduces only minor distortion at high noise levels.

Feature labels and normals As a final step, we label each 3D feature point as on a ridge, valley or ridge-contour based on the type of the intensity edge points in the corresponding 2D grid square. Each feature point is also associated with one or two normals, depending on its label. A ridge or valley point has two normals, each being the weighted average of the normals of the Hermite samples on one side of the ridge or valley (using the same weights as in Equation 3). For a ridge-contour point, we only consider Hermite samples on the occluding side of the ridge, and hence only one normal is obtained. To facilitate merging of multiple sets of feature points in the next stage, we shall treat a ridge-contour point simply as a ridge point equipped with a single normal. Examples of the feature points and their normals are shown in Figure 2 (d).

4. Merging feature points from multiple RGB-D images

Feature points extracted from a single RGB-D image rarely can capture all sharp features in the scene. Also, noise in the image and mismatch between color and depth channels can introduce errors in the locations, normals, and labels of the feature points. Given a sequence of RGB-D images capturing the object from different angles, our goal is to produce a more complete set of feature points while correcting the errors that appear in individual images. Note that a dense RGB-D video is not required; we only need an image set large enough such that each visible sharp feature appears as intensity edges in multiple images.

We start by aligning depth maps in successive images using standard RGB-D registration algorithms. In our experiments, we use a ICP-based rigid-body alignment that utilizes both the color (SIFT) and depth information [HKH*10]. It is not surprising that applying the computed transformations to the feature points does not create a clean and complete feature set (Figure 9 (a)). The noise comes from both errors in the transformations and, more importantly, errors in the feature point locations extracted from each image.

To improve the quality of this initially aligned set of feature points, we first apply the classical ICP alignment to the feature points alone while utilizing the feature normals to define the distance metric. This second alignment offers marginal improvement over initial one, and the majority of noise remains (Figure 9 (b)). In the next (and the core) step, we perform an iterative, non-rigid consolidation of feature points to clean and continuous lines (Figure 9 (c,d,e)). A key component of consolidation is a definition of feature affinity that prevents merging of nearby features during consolidation.

The two steps (rigid-body ICP and non-rigid consolidation) are explained below. We assume the input is a sequence of feature point sets that have been aligned using any RGB-D registration methods. Each feature point f is represented as a 4-tuple $\{p_f, N_f, t_f, u_f\}$ that stores its position (p_f), normal set (N_f), type (t_f, either ridge or valley), and view direction (u_f). For ridge points with a single normal, the view direction serves as the surrogate for the "invisible" normal on the other side of the ridge (see Section 4.2).

4.1. Feature-based ICP

We start by performing ICP alignment on the successive images using only the feature points. Let F, F' be feature sets extracted from two RGB-D images (after being transformed by the initial alignment). For more robust results, we extend the point-to-plane metric [SHT09] to form a *point-to-feature* metric that measures the *sum* of distance from one feature point $f \in F$ to *both* planes associated with its nearest feature point $f' \in F'$. Specifically, we seek a rigid transformation T that minimizes the quadratic energy:

$$\sum_{f \in F} \left(\sum_{n \in N_{f'}} \left(((T p_f - p_{f'}) \cdot n)^2 \right) + \beta (T p_f - p_{f'})^2 \right) \qquad (4)$$

The second term is designed to prevent drifting of points along the feature (we use $\beta = 0.5$). For efficiency, we adopt a linear approximation of rigid transformations [lL04] so that the minimization can be solved using standard linear least-squares.

4.2. Consolidating feature points

To handle the large amount of noise present after rigid-body alignment, we need to move individual feature points so that points along the same feature form thin and smooth lines. A key challenge towards this goal is how to suppress the noise without merging nearby features (e.g., the two parallel edges on the front of the cart as seen in Figure 9 (b) top). To address this, we first present a definition of feature affinity that can distinguish points lying on spatially close but geometrically distinct features. We then utilizes the affinity measure in an iterative algorithm that consolidates features points with high affinity.

Feature affinity Given a pair of features points, we wish to assess the likelihood that they lie on the same feature. A straight-forward definition would be based on the point-to-feature distances as defined in the ICP alignment step; a high affinity is given for pairs where each feature point lies close to the planes associated with the other feature point. However, such definition would not be able to distinguish features with different *surface orientations*. This is illustrated in 2D in Figure 10, where we sample point pairs $\{f, g\}$ at sharp features in three different scenarios. Since the locations and associated planes of $\{f, g\}$ are the same in all three cases, the two feature points would have the same affinity. However, except in the first case, the two points clearly lie on different features. Note that the two parallel edges in front of the cart in Figure 9 (b) is similar to the second scenario.

To disambiguate features with dissimilar surface orientations, we consider both the normal directions and the ridge/valley label associated with a feature point. Intuitively, the affinity between features f, g should be high if all three criteria are met:

1. p_f (resp. p_g) lies close to both p_g (resp. p_f) and the planes defined by $\{p_g, N_g\}$ (resp. p_f, N_f).

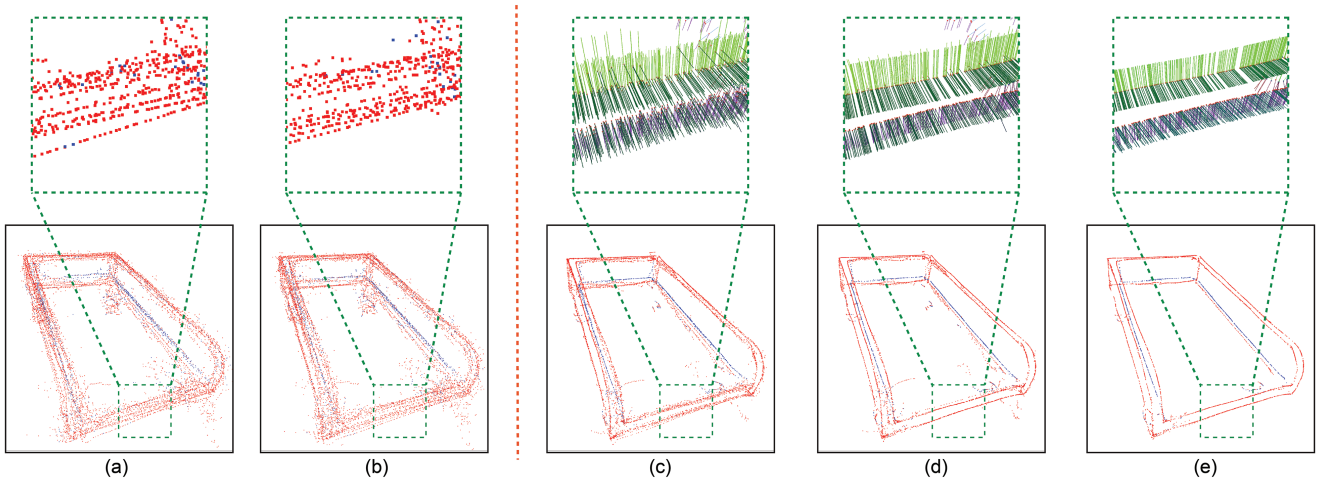

Figure 9: *Work flow of merging feature point sets. (a): Feature points obtained from 10 RGB-D images that are aligned using a standard RGB-D registration method. (b): Result after applying our feature-based rigid-body ICP alignment. (c,d,e): Results after first, second, and third iterations of our feature consolidation algorithm. Feature points are colored by type (red: ridge; blue: valley) and their normals are colored by directions.*

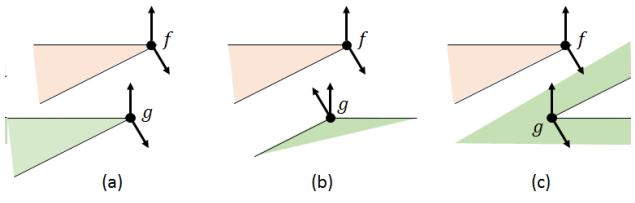

Figure 10: *A pair of feature points $\{f,g\}$ in three different scenarios. The shaded region indicates the interior of the surface. Note that the point-to-feature distance between f and g are the same in all three cases. However, only the pair in (a) have matching normal directions and feature labels (ridge), which indicates that f and g are likely to belong to the same feature.*

2. The normals N_f, N_g have similar orientations.
3. Both features have the same label, i.e., $t_f = t_g$.

To see how these criteria work on the examples in Figure 10, note that one of the normals in N_f is opposite to one in N_g in scenario (b), which fails the second criteria, and labels t_f, t_g in (c) are ridge and valley respectively, which fails the third criteria.

More formally, we capture the first criteria by the distance function:

$$d_1(f,g) = \frac{1}{|N_g|}\sum_{n\in N_g} h(p_f,\{p_g,n\}) + \frac{1}{|N_f|}\sum_{n\in N_f} h(p_g,\{p_f,n\}) \tag{5}$$

where $|\cdot|$ denotes cardinality and h is the Mahalanobis distance defined in Equation 2.

To evaluate the second criteria, we define the augmented normal set V_f for a feature point f as N_f if $|N_f| = 2$ and $N_f \cup \{u_f\}$ otherwise. In the latter case, we use the view direction u_f as a guesstimate of the invisible normal on the other side of the ridge. Let $V_f = \{v_f^1, v_f^2\}$. There are two ways to match up the normal sets associated with feature points f,g, and we take the smaller matching difference among the two:

$$d_2(f,g) = \min(m(v_f^1,v_g^1) + m(v_f^2,v_g^2), m(v_f^1,v_g^2) + m(v_f^2,v_g^1)). \tag{6}$$

Here m measures the difference between two vectors,

$$m(n_1,n_2) = \begin{cases} 1 - n_1 \cdot n_2, & \text{if } n_1 \neq u_f \text{ and } n_2 \neq u_g \\ 0, & \text{else if } n_1 \cdot n_2 > 0 \\ 2, & \text{otherwise.} \end{cases} \tag{7}$$

To combine these criteria together, we make the assumption that the feature points are already roughly aligned, and hence proximity (i.e., criteria 1) should dominate the overall affinity. As a result, we defined the affinity between feature points f,g as

$$w(f,g) = \begin{cases} \exp\left(\frac{-d_1(f,g)^2}{\exp(-d_2(f,g)^2/\sigma_2)\sigma_1}\right), & \text{if } t_f = t_g \\ 0, & \text{else} \end{cases} \tag{8}$$

Note that the orientation difference, d_2, acts as a scalar weight (greater than 1) of proximity, d_1. We use $\sigma_1 = 0.01$, $\sigma_2 = 0.0625$ in our experiments.

Iterative consolidation Our algorithm iteratively pulls each feature point towards the planes associated with neighboring high-affinity features points. Since both our affinity measure and the consolidation objective (presented below) depend on the normals of a feature point as well as its position, we adjust both positions and normals in this process. In each iteration, we first update the positions of feature points given their current normals, and then adjust the normals with the positions fixed.

To update the position of a feature point f, we consider all feature points S_f in a spherical neighborhood centered at p_f with radius r. We set $r = 0.05L$ where L is the largest dimension of the scene bounding box. We seek a new location of p_f, noted as p, that

minimizes the following quadratic energy:

$$\sum_{g\in S_f}\left(w(f,g)\sum_{n\in N_g}((p-p_g)\cdot n)^2\right)+\frac{\gamma}{\rho_f}(p-p_f)^2 \qquad (9)$$

The first term favors locations close to the planes associated with features points in S_f, while the second term prevents drifting. Weighting the second term by the point density ρ_f locally around f also has the effect of uniformly distributing points along the feature: a higher value of ρ_f would allow more drifting and hence leads to lower density. Following previous works [HLZ*09], we define $\rho_f = 1+\sum_{g\in S_f}\exp(-\|p_f-p_g\|^2/\sigma_\rho)$. We use $\gamma = 8$ and $\sigma_\rho = 0.15$.

After updating the positions of all feature points, we replace the normals at a feature point f by the affinity-weighted average of normals at feature points in the neighborhood S_f. For each $g \in S_f$, we consider one of the two matchings between augmented normal sets V_f, V_g that realizes the smaller difference $d_2(f,g)$ (Equation 6). Each vector in V_g (if it is not the view direction u_g) then contributes to the average with weight $w(f,g)$.

Example and comparison As an example, Figure 9 (c,d,e) show the results of successive iterations of consolidation on the rigidly-aligned points in (b). In just three iterations, features points form clearly distinguishable lines with little residue noise. Thanks to our affinity measure, close-by features (e.g., the two edges in the close-up view) are well separated.

We compare the results with two state-of-art point cloud consolidation methods, WLOP (weighted locally optimal projector) [HLZ*09] and L_1 skeletonization [HWCO*13], in Figure 11 (a,b). We ran both methods on the same point cloud (Figure 9 (b)) as our consolidation method. Observe that WLOP, designed for points representing a surface, is not effective in contracting the points to lines and the result still has a significant amount of noise. L_1 skeletonization, designed for points forming tubular shapes, incorrectly merges many nearby features.

Our consolidation is performed following the feature-based ICP step. Even though feature-based ICP offers small improvements over the initially aligned point clouds (e.g., Figure 9 (a)), we have found that such improvements can be critical to the success of consolidation. Without feature-based ICP, the point cloud can become too cluttered, and our consolidation method may either produce redundant line features or fail to produce clean lines. An example is shown in Figure 11 (d).

Choice of parameters We take a closer look at two key parameters and justify our choice of their values by showing results under alternative values (Figure 12).

We first consider σ_2 in Equation 8, which determines the contribution of normal mismatch (d_2) to the overall affinity between two feature points. Setting it too low makes the affinity sensitive to slight variation in normal directions, and hence feature points obtained from different views representing the same feature may not be merged (Figure 12 (a)). On the other hand, setting σ_2 too high diminishes the role of normal matching in the affinity, and feature points that lie on different features with disparate normal directions could be incorrectly merged (Figure 12 (b)). Our default

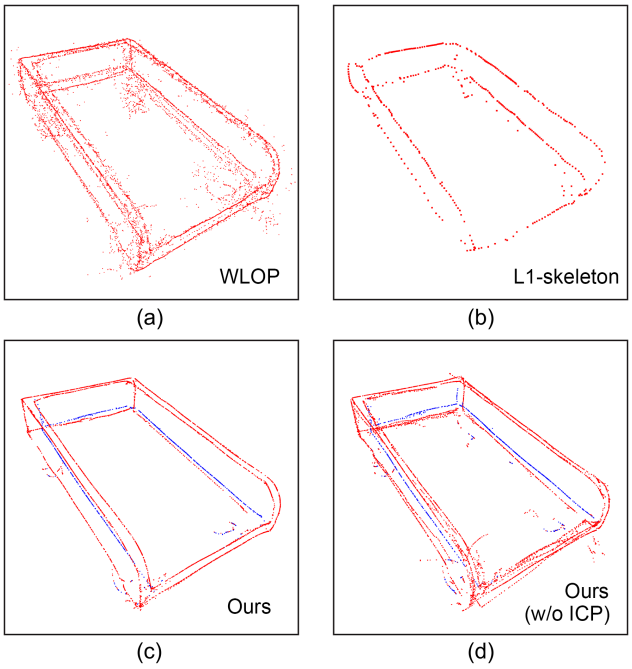

Figure 11: *(a,b,c): Consolidation results using WLOP [HLZ*09], L_1 skeletonization [HWCO*13], and our method on the same feature point cloud (Figure 9 (b)). (d): Result of our consolidation method run directly on the point cloud before feature-based ICP (Figure 9 (a)).*

choice ($\sigma_2 = 0.0625$) trades off tolerance of normal variation with discrimination of different features (see Figure 9 (e)).

Next, we consider the radius r that defines the neighborhood S_f of features points when formulating the objective function in Equation 9. A very small radius may prevent merging of feature points that belong to the same feature but do not locate in close proximity (Figure 12 (c)). On the other hand, a very large radius could lead to broken features due to aggregation of feature points (Figure 12 (d)). Our choice ($r = 0.05L$) offers a balance between maintaining the connectivity of feature curves and tolerating misalignment of feature points.

5. Forming lines

Given the consolidated feature points, we can form continuous polylines that approximate the corresponding sharp feature. We adopt the graph-based approach commonly used for connecting feature points on point clouds [GWM01, PKG03].

We first construct a weighted graph by connecting a feature point f to every feature point g in the neighborhood S_f (as defined in the previous section) with an edge whose weight equals the affinity $w(f,g)$ (Equation 8). We then remove edges with weights below a threshold (0.7 in our implementation). For each connected component in the remainder of the graph, we reset each edge weight to be its Euclidean length, and follow the method in [GWM01] to compute a *minimum spanning pattern* (MSP) that contains cycles with more edges than a user-specified constant. As suggested in the

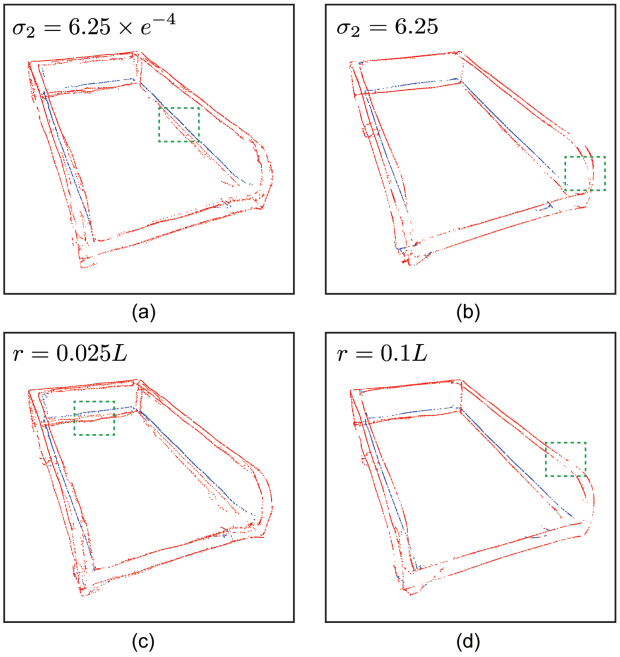

Figure 12: *Consolidation results using different choices of* σ_2 *(in Equation 8) and r (for defining the neighborhood* S_f *in Equation 9, as fraction of the largest dimension L of the scene bounding box). See discussion in text and compare the result generated using our default setting (*$\sigma_2 = 0.0625, r = 0.05L$*) in Figure 9 (e).*

paper, we set the constant to be $\sqrt{K}/2$ where K is the total number of points. Finally, branches shorter than a given threshold (10 in our implementation) are removed from the MSP. Result of line extraction on the consolidated features in Figure 9 (e) is shown in Figure 1 (c).

6. Experiment results

We tested our method on a variety of synthetic inputs as well as real data collected by Microsoft Kinect sensor. Each real data set consists of 8 to 13 RGB-D images. Besides those already mentioned, we show a few additional data ranging from individual objects to complete scenes.

Parameters All our experiments use the same set of parameter values as discussed in the text. We evaluate in Figure 13 two parameters $\varepsilon_1, \varepsilon_3$ for pruning the intensity edges (Section 3.3) that could strongly affect our results. While lower ε_1 results in more ridge-contour intensity edges (i.e., lying at depth discontinuity), higher ε_3 results in more ridge or valley intensity edges. Despite the variation in both feature points extracted from a single RGB-D image (see top row) and the consolidated feature curves from multiple images (see bottom row), the prominent sharp features of the object are captured by our method in every parameter setting.

Single objects We demonstrate our method on two complex sculptures in Figure 14. As in Figure 1, observe that our method produces much more meaningful and contiguous sharp features than geometry-based methods that work on the registered point clouds or reconstructed surfaces.

Our method is robust to the change in distance between the camera and the object. In the Cabinet example of Figure 15, the 10 input RGB-D images (of which 6 are shown) are collected at distances ranging from 0.5 meters to 3 meters away from the main object. Note that the algorithm nicely consolidates feature points lying on fine features, such as edges of the books, from both near and distant views. Also notice that the 3D features points extracted from each image are not affected by the texture of the books or the cabinet.

Scenes Our method can be applied to extract sharp features in an entire scene. We used two synthetic scene data sets in [CZK15], which are RGB-D video streams capturing realistically rendered rooms, as well as a benchmark data captured using Kinect v2 [WMS16]. We ran our method on a selected subset of frames in two videos and tried with different number of frames (see Figure 16). Note that our method produces clean and coherent feature curves using as few as 5 frames in each example, and that increasing the number of input frames results in more complete feature set. In contrast, even though a highly detailed surface can be reconstructed from the video stream using dense reconstruction [CZK15], extracting meaningful sharp features from such surfaces using geometry-based methods (e.g., [YBS05]) is still challenging.

Performance All computations were performed on a laptop PC with 8G RAM. Computing feature points from a single RGB-D image (at resolution 640 by 480) takes around 2 seconds. Merging features from multiple images takes between 15 seconds (e.g., the Bunny) to 70 seconds (e.g., the Shell), with the majority of time spent on the non-rigid consolidation step (feature-based ICP takes around 10 seconds in all examples).

7. Conclusion and discussion

We proposed a method for extracting sharp features (ridges and valleys) directly from a set of RGB-D images without the need for surface reconstruction. We start by extracting features from a single image that harvest both intensity edges and depth values. Features from multiple images are combined in a novel consolidation process designed for points equipped with normals and ridge/valley labels.

Limitations Our method is designed for sharp features that appear as strong edges in the color channel. Hence it cannot robustly detect rounded ridges or valleys. It will also fail under low-light conditions where intensity edges become unreliable. While our single-frame feature point extraction method can tolerate small amount of data error, such as missing depth values, incorrect depth values, and misalignment between depth and color channels, more severe errors can cause our method to fail. Typical failures include missing features (e.g., along the thin bevel of the Cart in Figure 4 (c)), incorrect labelling of features (e.g., ridges identified as valleys), and incorrect feature locations (e.g., Figure 8 (c)). These errors, as well as errors in the initial RGB-D registration, can all lead to inaccurate features after consolidation. In addition to improving the

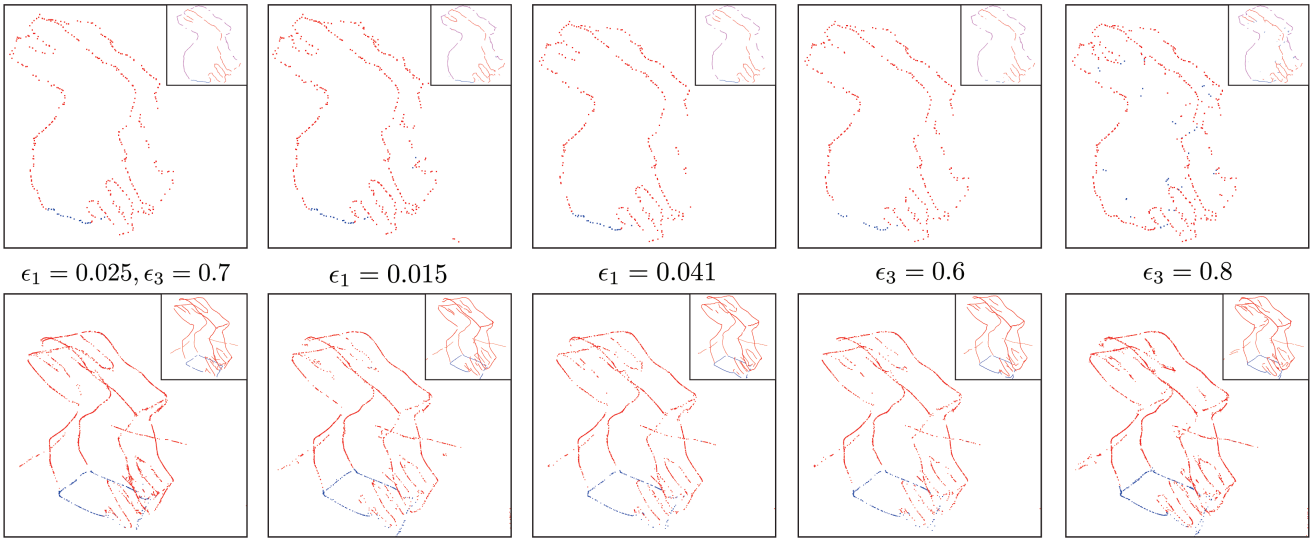

Figure 13: *Top row: pruned intensity edges (in inserts) and computed feature points from a single RGB-D image of Bunny (see Figure 2) under different parameter settings (the left-most image uses the default setting). Bottom row: the consolidated points and final curve network (in inserts) for each parameter setting from a collection of RGB-D images.*

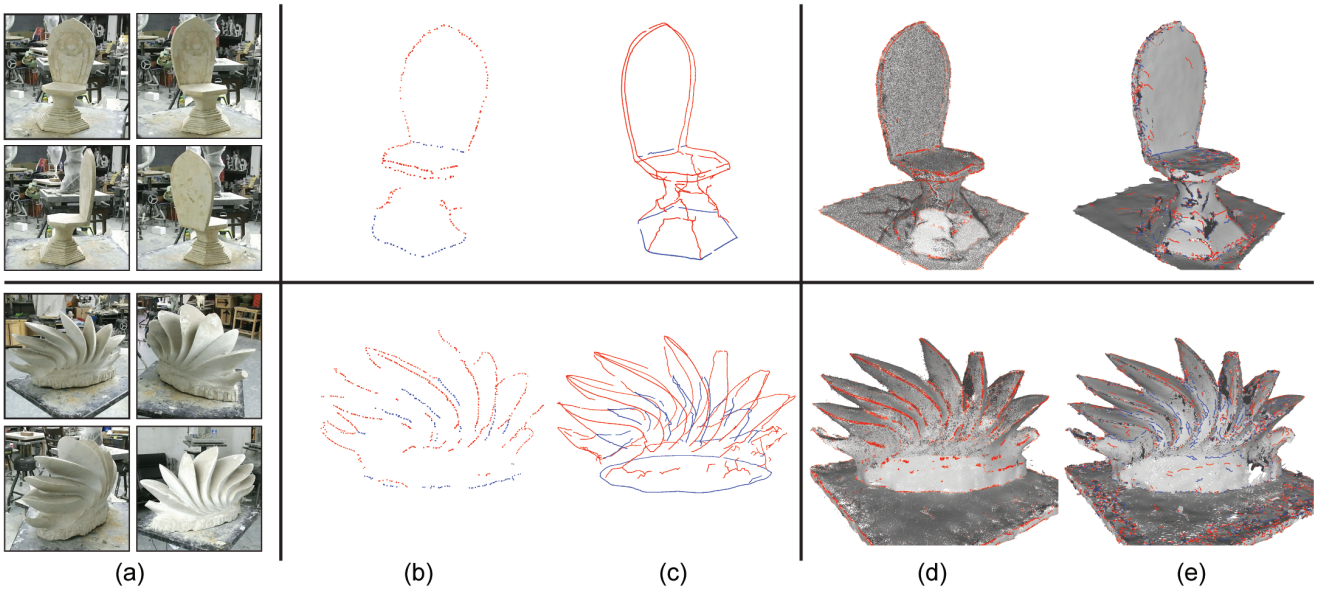

Figure 14: *Feature extraction on two sculptures (top and bottom rows). (a): Input RGB-D images. (b): Feature points extracted from the first image. (c): Connected feature curves computed from all images. (d): Feature points detected by [MOG09] on the registered depth maps. (e): Ridge and valleys extracted by [YBS05] on the reconstructed surfaces. For (d,e), we use [HKH*10] for registering depth maps, [HWG*13] for filtering the registered point cloud, and [OGG09] for surface reconstruction from the filtered point cloud.*

robustness of our method, we would also like to explore the ability to recover sharp corners in addition to edges and, eventually, a complete feature network.

Applications Sharp features can be potentially useful in a variety of tasks involving RGB-D scans, such as segmentation [CLW*14], primitive detection [HM15], and object recognition [KBK15].

We are particularly interested in applying these sharp features to model man-made objects directly from RGB-D images. Since man-made shapes can be well described by a network of sharp features together with the surface normals [MZL*09], our feature extraction method is an ideal starting point for modeling. As a proof-of-concept, we show in Figure 17 abstract surface models of Cart and Bunny created by (1) interactively connecting the feature curves produced by our method into complete wireframes and (2) surfacing the wireframes using the method of [ZZCJ13]. In the future,

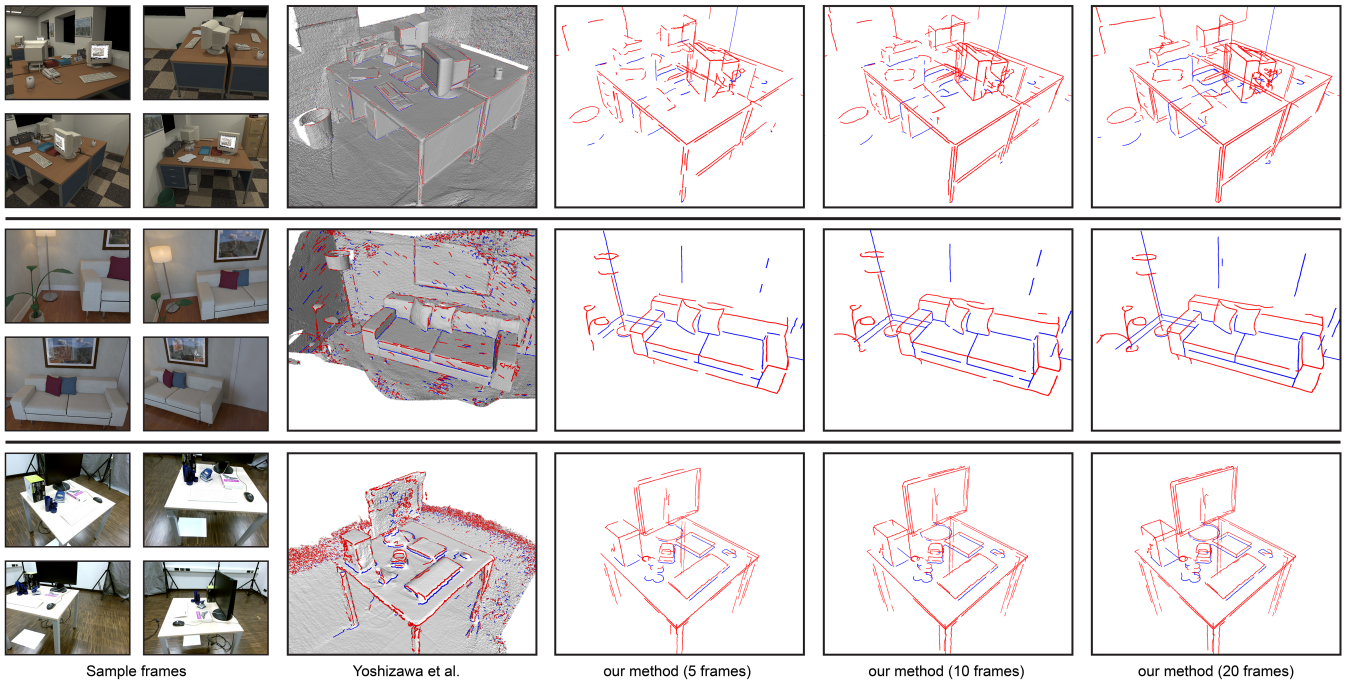

Figure 16: *Far-left: Selected frames from RGB-D video streams of synthetic (top and middle) and real (bottom) scenes. Mid-left: Features extracted using a mesh-based method [YBS05] on dense reconstruction [CZK15] from all frames in each video. Right: features produced by our method on only a subset of frames in each video.*

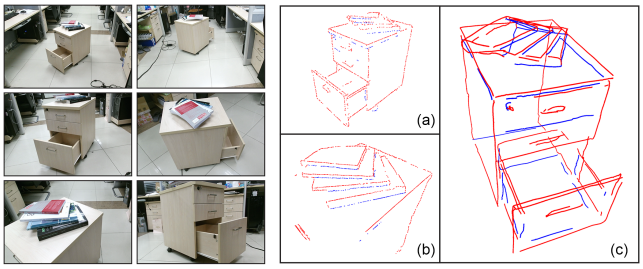

Figure 15: *RGB-D images capturing a cabinet with books at varying distances (left), the feature points extracted from two images (a,b), and the connected feature curves computed from all images (c).*

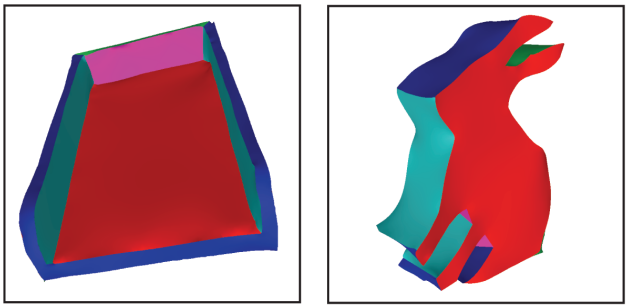

Figure 17: *Abstract surfaces created from extracted sharp features.*

we would like to explore more automated means for completing the wireframe from the sharp feature curves as well as better surfacing algorithms that take into account of the normals along the sharp features and the depth points.

Acknowledgement We thank the anonymous reviewers for their constructive comments. The work is supported in part by NSF (grants IIS-0846072, IIS-1319573), the Natural Science Foundation of China (Project Number 61521002, 61120106007), a Research Grant of Beijing Higher Institution Engineering Research Center, and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

[AMMK13] ALTANTSETSEG E., MURAKI Y., MATSUYAMA K., KONNO K.: Feature line extraction from unorganized noisy point clouds using truncated fourier series. *The Visual Computer 29*, 6-8 (2013), 617–626.

[ASGCO10] AVRON H., SHARF A., GREIF C., COHEN-OR D.: ℓ1sparsee reconstruction of sharp point set surfaces. *ACM Trans. Graph. 29*, 5 (Nov. 2010), 135:1–135:12.

[CLH15] CHEN K., LAI Y.-K., HU S.-M.: 3d indoor scene modeling from rgb-d data: a survey. *Computational Visual Media 1*, 4 (2015), 267–278.

[CLW*14] CHEN K., LAI Y.-K., WU Y.-X., MARTIN R., HU S.-M.: Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Transactions on Graphics 33*, 6 (2014).

[CTC13] CHOI C., TREVOR A. J. B., CHRISTENSEN H. I.: RGB-D edge detection and edge-based registration. In *2013 IEEE/RSJ Inter-*

national Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013 (2013), pp. 1568–1575.

[CTO*10] CAO J., TAGLIASACCHI A., OLSON M., ZHANG H., SU Z.: Point cloud skeletons via laplacian based contraction. In *Shape Modeling International Conference (SMI), 2010* (2010), pp. 187–197.

[CYW15] CAO Y., YAN D., WONKA P.: Patch layout generation by detecting feature networks. *Computers & Graphics 46* (2015), 275–282.

[CZK15] CHOI S., ZHOU Q.-Y., KOLTUN V.: Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (July 2003), 848–855.

[dGGDV11] DE GOES F., GOLDENSTEIN S., DESBRUN M., VELHO L.: Exoskeleton: Curve network abstraction for 3d shapes. *Computers & Graphics 35*, 1 (2011), 112–121.

[DHOS07] DANIELS J. I., HA L. K., OCHOTTA T., SILVA C. T.: Robust smooth feature extraction from point clouds. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007* (2007), SMI '07, pp. 123–136.

[DVVR07] DEMARSIN K., VANDERSTRAETEN D., VOLODINE T., ROOSE D.: Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Comput. Aided Des. 39*, 4 (Apr. 2007), 276–283.

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C. T.: Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3 (July 2005), 544–552.

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (July 2003), 950–953.

[GSMCO09] GAL R., SORKINE O., MITRA N. J., COHEN-OR D.: iwires: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics (Siggraph) 28*, 3 (2009), #33, 1–10.

[GWM01] GUMHOLD S., WANG X., MACLEOD R.: Feature extraction from point clouds. In *In Proceedings of the 10 th International Meshing Roundtable* (2001), pp. 293–305.

[HG01] HUBELI A., GROSS M.: Multiresolution feature extraction for unstructured meshes. In *Proceedings of the Conference on Visualization '01* (2001), VIS '01, pp. 287–294.

[HKH*10] HENRY P., KRAININ M., HERBST E., REN X., FOX D.: RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. of the International Symposium on Experimental Robotics (ISER)* (2010).

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 176:1–176:7.

[HM15] HENNESSEY J. W., MITRA N. J.: An image degradation model for depth-augmented image editing. *Comput. Graph. Forum 34*, 5 (2015), 191–199.

[HP04] HILDEBRANDT K., POLTHIER K.: Anisotropic filtering of nonlinear surface features. *Comput. Graph. Forum 23*, 3 (2004), 391–400.

[HPW05] HILDEBRANDT K., POLTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *Proceedings of the Third Eurographics Symposium on Geometry Processing* (2005), SGP '05.

[HS13] HE L., SCHAEFER S.: Mesh denoising via l0 minimization. *ACM Trans. Graph. 32*, 4 (July 2013), 64:1–64:8.

[HWCO*13] HUANG H., WU S., COHEN-OR D., GONG M., ZHANG H., LI G., CHEN B.: L1-medial skeleton of point cloud. *ACM Trans. Graph. 32*, 4 (July 2013), 65:1–65:8.

[HWG*13] HUANG H., WU S., GONG M., COHEN-OR D., ASCHER U., ZHANG H. R.: Edge-aware point set resampling. *ACM Trans. Graph. 32*, 1 (Feb. 2013), 9:1–9:12.

[KBK15] KIFORENKO L., BUCH A. G., KRÜGER N.: Object detection using a combination of multiple 3d feature descriptors. In *Computer Vision Systems - 10th International Conference, ICVS 2015, Copenhagen, Denmark, July 6-9, 2015, Proceedings* (2015), pp. 343–353.

[KK06] KIM S.-K., KIM C.-H.: Finding ridges and valleys in a discrete surface using a modified mls approximation. *Comput. Aided Des. 38*, 2 (Feb. 2006), 173–180.

[KKK06] KIM S.-K., KIM S.-J., KIM C.-H.: Extraction of ridges-valleys for feature-preserving simplification of polygonal models. In *Computational Science - ICCS 2006*, vol. 3992 of *Lecture Notes in Computer Science*. 2006, pp. 279–286.

[KST08] KOLOMENKIN M., SHIMSHONI I., TAL A.: Demarcating curves for shape illustration. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 157:1–157:9.

[LB15] LEE K. W., BO P.: Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering* (2015), –.

[LCOLTE07] LIPMAN Y., COHEN-OR D., LEVIN D., TAL-EZER H.: Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. 26*, 3 (July 2007).

[Lin98] LINDEBERG T.: Edge detection and ridge detection with automatic scale selection. *Int. J. Comput. Vision 30*, 2 (Nov. 1998), 117–156.

[LJHW15] LV X., JIANG S.-Q., HERRANZ L., WANG S.: Rgb-d handheld object recognition based on heterogeneous feature fusion. *Journal of Computer Science and Technology 30*, 2 (2015), 340–352.

[lL04] LIM LOW K.: *Linear least-squares optimization for point-toplane ICP surface registration*. Tech. rep., UNC, 2004.

[LPVV11] LEJEUNE A., PIÉRARD S., VAN DROOGENBROECK M., VERLY J.: A new jump edge detection method for 3D cameras. In *International Conference on 3D Imaging (IC3D)* (December 2011).

[LZT*08] LEHTINEN J., ZWICKER M., TURQUIN E., KONTKANEN J., DURAND F., SILLION F., AILA T.: A meshless hierarchical representation for light transport. *ACM Trans. Graph. 27*, 3 (2008).

[MOG09] MÉRIGOT Q., OVSJANIKOV M., GUIBAS L.: Robust voronoi-based curvature and feature estimation. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009), SPM '09, pp. 1–12.

[MZL*09] MEHRA R., ZHOU Q., LONG J., SHEFFER A., GOOCH A., MITRA N. J.: Abstraction of man-made shapes. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 137:1–137:10.

[NIH*11] NEWCOMBE R. A., IZADI S., HILLIGES O., MOLYNEAUX D., KIM D., DAVISON A. J., KOHLI P., SHOTTON J., HODGES S., FITZGIBBON A.: Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR* (October 2011), IEEE, pp. 127–136.

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 609–612.

[OGG09] OZTIRELI C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum 28*, 2 (2009).

[PKG03] PAULY M., KEISER R., GROSS M. H.: Multi-scale feature extraction on point-sampled surfaces. *Comput. Graph. Forum 22*, 3 (2003), 281–290.

[SHT09] SEGAL A., HAEHNEL D., THRUN S.: Generalized-icp. In *Proceedings of Robotics: Science and Systems* (Seattle, USA, June 2009).

[SLG13] SCHÄFER H., LENZEN F., GARBE C.: Depth and intensity based edge detection in time-of-flight images. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2013 International Conference on* (2013), pp. 111–118. 1.

[SSW15] SUN Y., SCHAEFER S., WANG W.: Denoising point sets via l 0 minimization. *Comput. Aided Geom. Des. 35*, C (May 2015), 2–15.

[TZCO09] TAGLIASACCHI A., ZHANG H., COHEN-OR D.: Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph. 28*, 3 (July 2009), 71:1–71:9.

[WB01] WATANABE K., BELYAEV A. G.: Detection of salient curvature features on polygonal surfaces. *Comput. Graph. Forum 20*, 3 (2001), 385–392.

[WHH10] WEBER C., HAHMANN S., HAGEN H.: Sharp feature detection in point clouds. In *Proceedings of the 2010 Shape Modeling International Conference* (2010), SMI '10, pp. 175–186.

[WMS16] WASENMULLER O., MEYER M., STRICKER D.: CoRBS: Comprehensive rgb-d benchmark for slam using kinect v2. In *IEEE Winter Conference on Applications of Computer Vision (WACV)* (March 2016), IEEE. URL: `http://corbs.dfki.uni-kl.de/`.

[XCJ*09] XU K., COHEN-OR D., JU T., LIU L., ZHANG H., ZHOU S., XIONG Y.: Feature-aligned shape texturing. *ACM Trans. Graph. 28*, 5 (2009), 108:1–108:7.

[YBS05] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: Fast and robust detection of crest lines on meshes. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (2005), SPM '05, pp. 227–232.

[YBYS07] YOSHIZAWA S., BELYAEV A. G., YOKOTA H., SEIDEL H.: Fast and faithful geometric algorithm for detecting crest lines on meshes. In *Proceedings of Pacific Graphics 2007, Hawaii, USA* (2007), pp. 231–237.

[ZFAT11] ZHENG Y., FU H., AU O.-C., TAI C.-L.: Bilateral normal filtering for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1521–1530.

[ZK15] ZHOU Q.-Y., KOLTUN V.: Depth camera tracking with contour cues. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[ZZCJ13] ZHUANG Y., ZOU M., CARR N., JU T.: A general and efficient method for finding cycles in 3d curve networks. *ACM Trans. Graph. 32*, 6 (2013), 180:1–180:10.