Robotics and Autonomous Systems 76 (2016) 141–151

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Modeling connected regions in arbitrary planar point clouds by robust B-spline approximation

Thomas Mörwald [a,*], Jonathan Balzer [b], Markus Vincze [a]

[a] Vienna University of Technology, Austria

[b] University of California, Los Angeles, USA

HIGHLIGHTS

- The *Asymmetric Distance* removes outliers inside the curve and finds the concave hull.
- Estimates the unknown required degrees of freedom by *Error-Adaptive Knot Insertion*.
- Handle deep and narrow concavities by *Concavity Filling*.
- High robustness, and compression rates up to a factor of 300 are reported.
- Fully integrated into PCL and created a tutorial.[1]

ARTICLE INFO

Article history:
Available online 7 December 2015

Keywords:
B-spline
Reconstruction
Curve fitting
Boundary fitting
Concave hull
Surface trimming

ABSTRACT

This paper presents an algorithm for robustly approximating the boundary of a domain, latent in a planar set of scattered points, by a B-spline curve. The algorithm is characterized by three key features: First, we propose a distance measure, called the *Asymmetric Distance* (AD), which allows for handling outliers inside the curve and finding the outer boundary or concave hull by specifying very natural parameters like smoothness and accuracy. Second, we provide a solution to the problem of unknown required degrees of freedom by *Error-Adaptive Knot Insertion* (EAKI). During the iterations of our re-weighted least-squares formulation, we check for regions of high error on the curve and locally increase the degrees of freedom if necessary. Third, we present a method to handle deep and narrow concavities, called *Concavity Filling* (CF). The curve is examined for areas of large distances to the closest data points. In these regions, we explicitly strap the curve to internal points to force it to bend inwards and fill the concavity. Compared with the state of the art, our method shows fundamental improvement in terms of robustness and applicability to real-world data. For 3D reconstruction of organized and unorganized point clouds, prevalent in robotic RGBD perception, we achieve higher robustness compared to state-of-the-art methods and compression rates up to a factor of 300. We have integrated our code into the Point Cloud Library (PCL) and created a tutorial that guides through the steps of the algorithm (see footnote 1).

1. Introduction

1.1. Motivation

While B-spline curves have been acknowledged as a multi-purpose tool in computer graphics and geometric modeling for decades, their potential seems to remain partially unexplored in computer vision and robotics perception. One of their earliest accounts in this realm is given in the seminal work by Kass et al. on *Active Contours* [1]. "Continuous" representations are critical to the active contour model as they endow a disconnected set of points with a topology. From the topology emerges a meaningful notion of distance between points, enabling the definition of differential operators. Indeed, an important ingredient in any numerical active contour scheme – be it for segmentation, tracking and reconstruction or similar purposes – is stable estimation of inner-geometric properties of the evolving curve, such as tangent field and curvature. Roughly two decades later, with the advent of RGBD sensors, such as Microsoft's Kinect, a surge of interest in geometric vision has opened new opportunities: RGBD sensors are capable of producing dense depth maps at a rate of 30 frames

* Corresponding author.
E-mail address: moerwald@acin.tuwien.ac.at (T. Mörwald).

[1] http://pointclouds.org/documentation/tutorials/bspline_fitting.php.

http://dx.doi.org/10.1016/j.robot.2015.11.006
0921-8890/

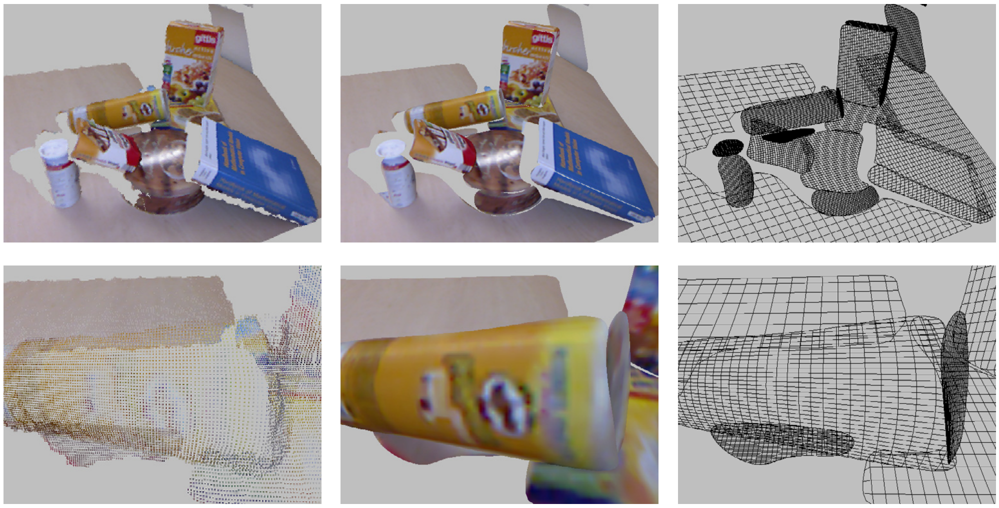


Fig. 1. Reconstruction of an organized segmented [3] point cloud with B-spline curves and surfaces (OSD dataset [4]). The parametric domain of the surface and the curve is a sub-set of the image space. As a consequence, direct evaluation of the physical coordinates, and thus texture mapping becomes a trivial assignment operation. The wire-frame meshes show the B-spline surfaces trimmed by the B-spline curves evaluated using our approach. (From left to right: point cloud, textured meshes, meshed and trimmed B-spline surfaces. Top: full view. Bottom: close-up view.)

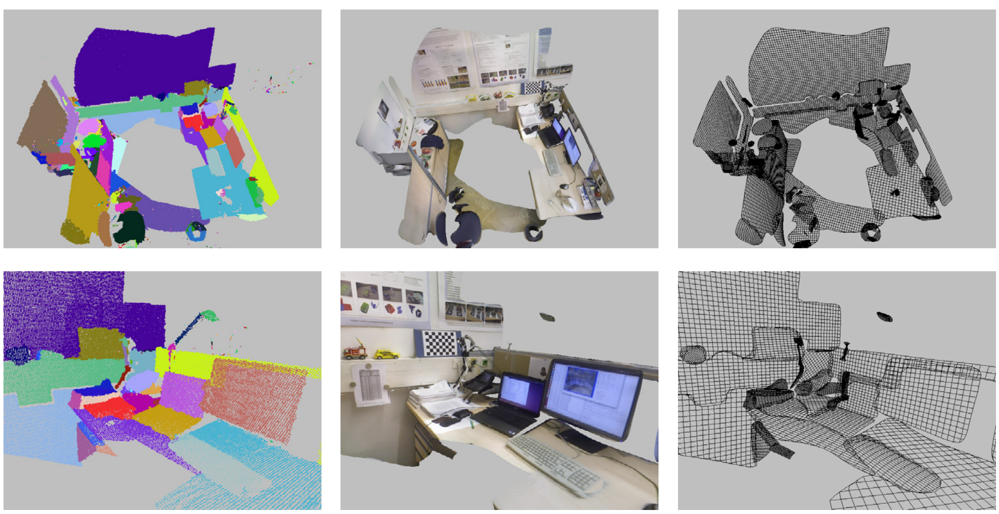


Fig. 2. Reconstruction of an unorganized segmented point cloud with B-spline curves and surfaces (*Office 1* in Table 1). The parametric domains of the curves and surfaces lie in the respective eigenspaces of the point cloud segments. From left to right: point cloud, textured meshes, meshed and trimmed B-spline surfaces. Top: full view. Bottom: close-up view.

per second. While this redundancy is desirable to some extent, it also poses challenges in terms of memory consumption. Thanks to their unmatched approximation power, spline surfaces have the ability to compress point-based models by orders of magnitudes without compromising accuracy and additionally provide visually appealing representations, cf. Figs. 1–3. In this examples, curves are needed to trim excess material along occlusion boundaries. As recently argued in [2], parametric models hold great promise in solving visual inference problems by *isogeometric finite-elements analysis*, where a common spline basis is used for geometric modeling and the analysis of partial differential equations (PDEs). B-spline curves in particular may prove useful for application in boundary-elements analysis, which converts certain types of PDEs on planar domains to integral equations on the respective boundary and thus offers significant reduction of computational complexity.

Despite their relevance, spline curves remain rather exotic entities within the field of computer vision and point cloud processing. This can largely be attributed to the difficulties of their construction from point sets, which naturally arise from the regular lattice of digital images. Data is often incomplete or suffers from heavy noise and clutter. In low-level vision and image processing, the points typically coincide with the regular image lattice, whose natural topology they inherit as in Fig. 1. This is no longer the case in applications such as the one shown in Figs. 2 and 3 where – as a consequence of projection – data is scattered to subpixel locations and the neighborhood relations are not easily accessible any more. Meanwhile, it seems there is still no generic approach that can deal with all of the aforementioned problems without resorting to substantial manual user input or non-generic preprocessing.

Our goal is to infer the geometry of some simply connected domain, whose boundary interpolates most of a given point cloud, exhibiting most or all of the properties discussed above. The outcome shall have the form of a parametric B-spline curve. Owing to the non-linearity of the problem at hand, curve-fitting methods are usually iterative in nature. While this does require an initial guess, our method is largely independent from it. The only assumption we make is that the domain enclosed by the

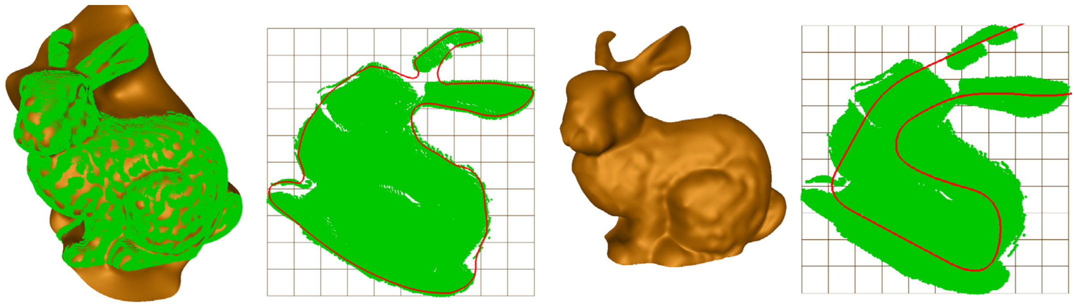


Fig. 3. *Stanford bunny*: Reconstruction of a single view from the range sensor, by fitting a B-spline surface to an unorganized point cloud (left) and a B-spline curve on the parametric domain of the surface using our approach (mid-left), which is used for trimming the surface (mid-right). The method of [5] is not designed to correctly approximate the contour (right). The data was taken from the Stanford 3D Scanning Repository.

initial curve contains the vast majority[2] of the unknown domain as a subset. Other authors have described a series of complications while evolving the starting curve towards the best approximation:

- *Overshooting*: The evolving curve is attracted by spurious data in the interior of the true solution. As a remedy, we suggest a change in the instantaneous[3] Riemannian metric of the plane. Our metric distinguishes itself from those described in [6,5,7], because it exhibits asymmetry w.r.t. the trace of the curve. Our *Asymmetric Distance* (AD) is formally introduced in Section 2.1.
- *Concavities*: While dealing with the overshooting problem, care has to be taken not to confuse clutter in the interior with deep and narrow concavities, which are actually part of the true shape. To our best knowledge, no curve fitting method exists which can recover complex shapes and deep and narrow concavities without manual initialization or an additional initialization algorithm like quad-tree cell partition in [5] or the chord-length parametrization method [8,9]. We address the issue in Section 2.3 by a generic technique which we dub *Concavity Filling* (CF).
- *Undersampling*: Only in the presence of the true solution, correspondence between points on the solution curve and the data points becomes apparent. This correspondence, however, is critical to choosing a resolution of the approximant, which is appropriate for the given data. As shown in Section 2.2, our *Error-Adaptive Knot Insertion* (EAKI) approach automatically adjusts the number of degrees of freedoms (NDOF) of the B-spline curve to facilitate the spectral properties of the measured point cloud. This leads to a locally optimal resolution and allows for a trivial initialization.

The remainder of this paper is organized as follows. A discussion of further related work is found in Section 1.2 followed by a more detailed description of three established methods in the beginning of Section 2. Further we will shortly discuss the problems of fitting B-splines and introduce our solutions AD, EAKI and CF. Section 3 presents an exhaustive experimental validation of our method in comparison with those deemed current state of the art.

1.2. Other related work

Our approach extends [5] by modifying the squared distance proposed therein. Through the data-dependent metric, our approach naturally relates to the literature on robust statistics, in particular, the Gauss–Newton algorithm for re-weighted least-squares (RWLS) problems. The subtle difference is that we circumvent classification customarily performed in RWLS methods to separate inliers from outliers. Classification requires robust estimates of the data variance, e.g. by the median of absolute deviations. The latter generalizes to dimensions greater than one only with severe technical difficulties which is why we avoid it in the present paper. Furthermore, we overcome the problem of specifying the degrees of freedom manually and increase the NDOF during optimization, similar to [10]. It is conceptually much simpler and thus easier to implement. Implicit B-spline models are proposed in [11–13] to infer the zero set of a bivariate tensor-product B/T-spline function. Fitting B-spline curves to point clouds in the presence of obstacles is introduced in [14,15], where an optimization problem subject to an inequality constraint is solved. Hu et al. [16] present a method that takes advantage of both algebraic and geometric distance minimization and therefore avoids additional constraints. Often it is necessary to modify an existing curve fitting method to apply to a problem with certain characteristics, such as noise, outliers, unknown NDOF and so forth [17–19]. This paper builds on our previous work on using B-spline surfaces for image segmentation [3,4,20] and on exploiting the implicit smoothing properties of B-spline curves for boundary refinement of closed regions [21].

2. Approach

2.1. Asymmetric weighting

Assume we are given a set $\{\mathbf{p}_i\} \subset \mathbb{R}^2$, $i = 1, 2, \ldots, m$, of m unorganized, scattered points in the plane with considerable non-uniformly distributed noise and heavy clutter. The task is to find the continuous planar curve that best approximates the data. A common way to represent continuous curves is by B-splines, often used in computer graphics, CAD/CAM, computer vision, and image processing. According to [22, Ch. 3.2], a B-spline curve is defined as

$$\mathbf{c}(t) = \sum_{j=0}^{n} \varphi_j(t)\mathbf{b}_j \tag{1}$$

where φ_j are the *basis functions*, $\mathbf{b}_j$ are called *control points*, and t is the curve parameter from a compact real domain or its periodic continuation. The periodicity carries over to the trace of all curves considered in this paper. The properties, and in particular the support, of the basis functions φ_i of polynomial degree p is uniquely determined by the value of the *knot vector* $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_k, \ldots, \xi_{n+p+1}) \in \mathbb{R}^{n+p+1}$. We denote differentiations w.r.t. the curve parameter by apostrophes. Hence, $\mathbf{c}'(t)$ is the *tangent vector* at t, and $\mathbf{c}''(t)$ the *curvature vector*. The best approximation of the point cloud $\{\mathbf{p}_i\}$ is characterized by a (global) minimizer of the objective function

$$f(\mathbf{b}_j) := \sum_{i=1}^{m} e_i^2 + f_s(\mathbf{b}_j) \tag{2}$$

[2] But – as seen in Fig. 8 – not necessarily all.

[3] We call the metric instantaneous because it depends on the shape of the curve at the current iteration.

Fig. 4. Reconstruction of an organized point cloud subject to heavy noise and clutter. From left to right: point cloud, textured meshes, meshed and trimmed B-spline surfaces.

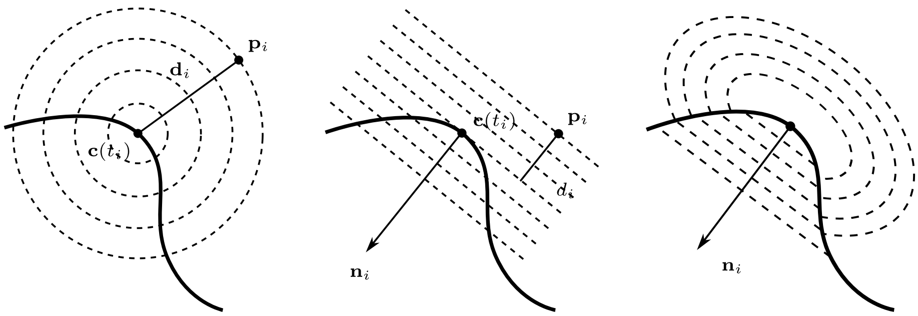


Fig. 5. From left to right: iso-value curves for PD, TD and SD.

w.r.t. the control points $\mathbf{b}_j$, where e_i is an error term describing the distance between the data points and the curve. Let $\mathbf{d}_i$ be the vector pointing from $\mathbf{c}(t_i)$ to $\mathbf{p}_i$. The *footpoint* t_i, with $\mathbf{c}(t_i)$ being the closest point to $\mathbf{p}_i$, can be determined by Newton's method. The *Point Distance* (PD) is defined as

$$e_{PD,i} := \|\mathbf{d}_i\|. \tag{3}$$

The second term of Eq. (2) is necessary to obtain a regular, i.e., visually satisfying solution, which for our method is defined as

$$f_s(\mathbf{b}_j) = \int w_s \|\mathbf{c}''(t)\|^2 dt. \tag{4}$$

Here, $w_s \in \mathbb{R}_{\geq 0}$ is a scalar weighting factor.

Blake and Isard [7] observed that utilizing the following *Tangent Distance* (TD) in place of the PD leads to significantly faster convergence:

$$e_{TD,i} := \mathbf{d}_i^T \mathbf{n}_i, \tag{5}$$

where $\mathbf{n}_i$ is the unit normal vector. Note that we pick the orientation of the curve that makes all normal vectors point outwards. The drawback of the TD, as shown in [5], is that the method is less robust with respect to local minima. Therefore Wang et al. [5] introduced the *Squared Distance* (SD) term to benefit from both, the robustness of the PD and fast convergence of the TD:

$$e_{SD,i}^2 := \begin{cases} \dfrac{d_i}{d_i - \rho_i} \left(\mathbf{d}_i^T \mathbf{t}_i\right)^2 + \left(\mathbf{d}_i^T \mathbf{n}_i\right)^2 & \text{if } d_i < 0, \\ \left(\mathbf{d}_i^T \mathbf{n}_i\right)^2 & \text{if } 0 \leq d_i < \rho_i, \end{cases} \tag{6}$$

where $\mathbf{t}_i = \mathbf{c}'(t_i)$ is the tangent vector at t_i. Here, $\rho_i = \|\mathbf{c}''(t_i)\|$, and $d_i = \mathbf{d}_i^T \mathbf{n}_i$ is the signed distance between $\mathbf{p}_i$ and the curve, i.e., $d_i < 0$ if $\mathbf{p}_i$ is on the opposite side of $\mathbf{n}_i$ and $d_i \geq 0$ if they are on the same side. Fig. 5 illustrates the three distance measures PD, TD and SD. For a more detailed discussion let us refer to the paper of Wang et al. [5]. Our experiments suggest the superior performance of the SD. For the remainder of this paper, we thus choose $e_i = e_{SD,i}$ unless stated otherwise.

As depicted in Fig. 4, segmentations are often subject to heavy clutter and outliers at the boundary as well as inside. Figs. 2 and 3 shows examples where the point cloud is not organized and a boundary other than the convex hull cannot be defined properly. To this end, we propose to augment the original distance by a scalar function w_a which weights points inside the boundary less heavy than points outside:

$$w_a(d) := \begin{cases} e^{-\frac{d^2}{\sigma^2}} & \text{if } d < 0 \\ 1 & \text{if } d \geq 0 \end{cases} \tag{7}$$

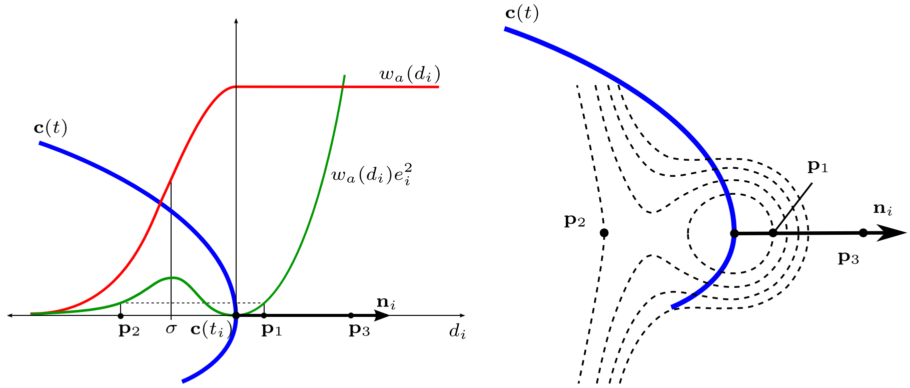
where σ defines the width of the transition of the weighting function with respect to the signed distance. The lack of symmetry is illustrated in Fig. 6. Eq. (2) then translates into

$$f(\mathbf{b}_j) = \frac{1}{2} \sum_{i=1}^{m} w_a(d_i) e_i^2 + f_s(\mathbf{b}_j). \tag{8}$$

During optimization, the curve is forced towards the outer boundary points, while points in the interior are largely ignored. This applies also to points which are part of a concavity. But fortunately, the half bell-shaped function w_a iteratively closes the gap between the curve and the data points. This is different to most other approaches like [22,7,5], where all points are treated equally. Fig. 7 demonstrates the effectiveness of the weight function.

2.2. Error-adaptive knot insertion (EAKI)

In real world applications, the number of degrees of freedom the boundary approximation should have is usually unknown. Initialization typically requires user interaction. Automatic estimation schemes such as mentioned in Section 1.2 are rarely generic enough to handle a sufficiently large class of problems. In [10], knots are inserted or removed depending on the distance between neighboring knots. In our opinion, this is quite counter-intuitive, since segments where a low resolution of the curve already fits large parts of the contour do not require refinement. Our methods *automatically* adapts the NDOF by iteratively inserting knots to the B-spline curve at points where the error is above the accuracy specified by the user. More precisely, at each step of the fitting iteration, we measure the distance from every curve point $\mathbf{c}(\bar{\xi}_k)$ to the closest point of the point cloud, where $\bar{\xi}_k := (\xi_k + \xi_{k+1})/2$, $k = 1, \ldots, n+p$, are the midpoints of two adjacent elements of the



(a) Asymmetric distance function. (b) Iso-value curves of $w_a(d_i)e_i^2$ for PD.

Fig. 6. *Asymmetric Distance*: Weighting function $w_a(d_i)$ (red) and the resulting asymmetric distance (AD) term $w_a(d_i)e_i^2$ (green) for fitting the point $\mathbf{p}_i$ attached to the footpoint of the B-spline curve $\mathbf{c}(t)$ (blue). Two opposing points $\mathbf{p}_1$ and $\mathbf{p}_2$ lie on the same iso-value curve even when the Euclidean distance of $\mathbf{p}_2$ is higher. In other words, a point $\mathbf{p}_3$ outside the curve, with the same Euclidean distance from $\mathbf{c}(t_i)$ as a point $\mathbf{p}_2$ inside, causes a much higher error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

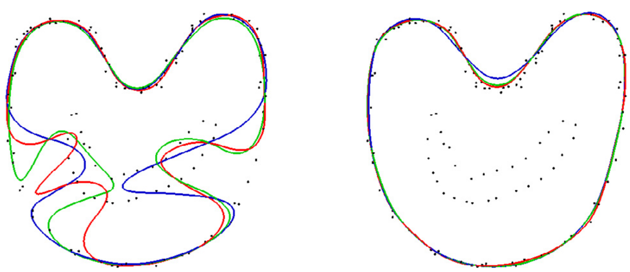


Fig. 7. (left) Typical problem when adding some clutter inside the boundary of a dataset such as Fig. 10 of [5]. (right) Solving the problem using the asymmetric distance (AD) of our approach. Red: point distance [22]. Green: tangent distance [7]. Blue: squared distance [5]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

knot vector $\boldsymbol{\xi}$. If the distance exceeds the accuracy specified by ε_a, a new knot is inserted at $\bar{\xi}_k$. The fitting approach procedure makes the knot distribution non-uniform; it particularly tends to increase the resolution in knot spans of high curvature. This behavior is illustrated in Fig. 8.

2.3. Concavity Filling (CF)

Noise at the boundary and sharp turns may erroneously lead the fitting process to stagnate because the AD term de-weights data points inside the boundary too severely (Fig. 9). Increasing σ might be a remedy in some cases but causes problems when neighboring boundaries are close to each other.

Instead we iterate through the midpoints $\bar{\xi}_k$ of the knot spans (see Section 2.2) and for each $\mathbf{c}(\bar{\xi}_k)$, we strap it to its nearest neighbor within the set of data points $\{\mathbf{p}_i\}$. The key idea is that thereby, the nearest neighbor of $\mathbf{c}(\bar{\xi}_k)$ is not understood in the sense of the Euclidean metric but a curve-dependent, instantaneous distance function $d_c : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$. With slight abuse of earlier notation, we set $\mathbf{d}_k = \mathbf{p} - \mathbf{c}(\bar{\xi}_k)$ and $\mathbf{n}_k = \mathbf{n}(\bar{\xi}_k)$. Now we are in the position to define

$$d_{c,k}(\mathbf{p}) := \begin{cases} 0, & \text{if } \|\mathbf{d}_k\| = 0, \\ \infty, & \text{if } \mathbf{n}_k^T\mathbf{d}_k \geq 0,\ \|\mathbf{d}_k\| \neq 0, \\ \dfrac{\|\mathbf{d}_k\|^2}{|\mathbf{n}_k^T\mathbf{d}_k|}, & \text{if } \mathbf{n}_k^T\mathbf{d}_k < 0,\ \|\mathbf{d}_k\| \neq 0. \end{cases} \tag{9}$$

The shape of the level sets of a typical $d_{c,k}$ are shown in Fig. 10(a). Strapping the evolving curve to the obtained nearest neighbors is achieved by enhancing Eq. (8) with the following energy:

$$f_c(\mathbf{b}_j) := \sum_{k=1}^{n+p} w_c \|\mathbf{p}_{c,k}^* - \mathbf{c}(\bar{\xi}_k)\|^2 \tag{10}$$

where $w_c \in \mathbb{R}_{\geq 0}$ and

$$\mathbf{p}_{c,k}^* = \arg\min_{\mathbf{p} \in \{p_i\}} d_{c,k}(\mathbf{p}). \tag{11}$$

As we are minimizing Eq. (10), the curve is strapped to data points $\mathbf{p}_i$ behind the sharp turn. At the same time, CF ignores curve elements respectively knot spans that are already interpolating, as indicated in Fig. 10(b). Once the curve at $\mathbf{c}(\bar{\xi}_k)$ is close enough to some $\mathbf{p}_i$, i.e. $d_{c,k}(\mathbf{p}_i) < \sigma$, the iteration will be governed again by the influence of the AD term. Combining Eqs. (4), (8), and (10), we finally obtain the objective function

$$\begin{aligned} f(\mathbf{b}_j) = & \sum_{i=1}^{m} w_a(d_i)e_i^2 + \int w_s \|\mathbf{c}''(t)\|^2 dt \\ & + \sum_{k=1}^{n+p} w_c \|\mathbf{p}_{c,k}^* - \mathbf{c}(\bar{\xi}_k)\|^2. \end{aligned} \tag{12}$$

3. Experimental evaluation

3.1. Implementation

We have created a C++ implementation of our curve fitting algorithm based on the *openNURBS* library.[4] All of our code is available within the *Surface* module of the Point Cloud Library. A tutorial guiding through the steps of creating the Stanford bunny example can be accessed at the PCL tutorial pages.[5] An outline of our approach is shown in Algorithm 1. For initialization we simply calculate the bounding circle of the point cloud and set the four initial control points of the closed periodic B-spline curve to lie on this circle while being shifts of $\pi/4$ apart (see Fig. 8 left).

Suppose that the two nearest-neighbors problems have been solved so that the obtained footpoints t_i and strapping points $\mathbf{p}_{c,k}^*$ are known and fixed. In that case, minimization of Eq. (12) reduces to a weighted linear least-squares problem. We assemble

4 http://www.rhino3d.com/opennurbs.

5 http://pointclouds.org/documentation/tutorials/bspline_fitting.php.

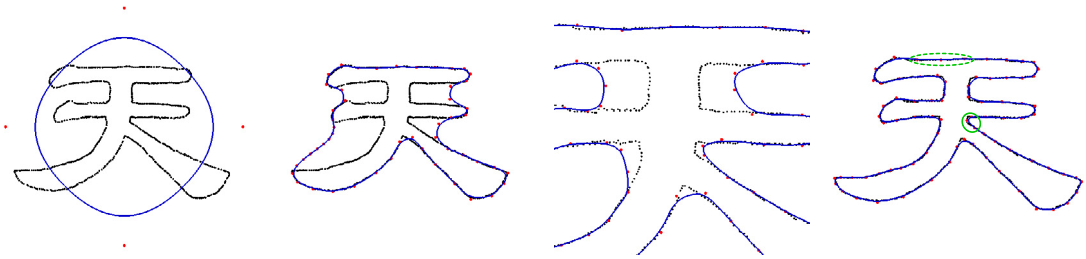


Fig. 8. *EAKI*: Control points (red) are iteratively inserted, automatically adapting the curve (blue) to the required degrees of freedom of the outline of the Chinese character *tian*. From left to right: Initial curve, 10, 15 and 30 iterations with 4, 61, 73 and 82 control points respectively. Note the simple initialization and the iterative increase of the NDOF by knot insertion, while the AD term fastens the curve to the data points. The green ellipses highlight the resolution of control points at regions of high curvature (solid line) and low curvature (dashed line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

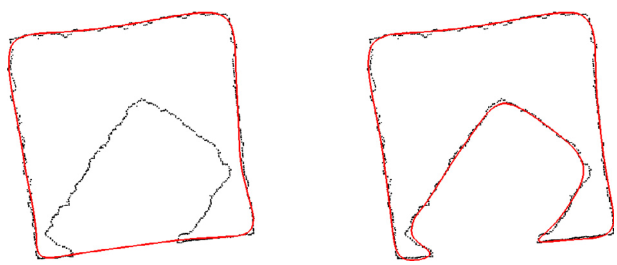


Fig. 9. Handling of sharp turns. Without (left) and with (right) concavity filling (CF).

Algorithm 1 Curve fitting algorithm.

initialization
while !terminated **do** ▷ Termination

  for all points $\mathbf{p}_i$ **do** ▷ Parametrization
    find footpoint t_i for $\mathbf{p}_i$ using Newton's method
  end for

  for all midpoints $\bar{\xi}_k$ of spans in $\boldsymbol{\xi}$ **do** ▷ CF Eq. (11)
    find closest point $\mathbf{p}^*_{c,k}$ of $\mathbf{c}(\bar{\xi}_k)$
  end for

  $\min_{\mathbf{b}_j} f(\mathbf{b}_j)$ ▷ Eq. (16)

  for all midpoints $\bar{\xi}_k$ of spans in $\boldsymbol{\xi}$ **do** ▷ EAKI
    find closest point $\mathbf{p}_i$ of $\mathbf{c}(\bar{\xi}_k)$
    if $\|\mathbf{p}_i - \mathbf{c}(\bar{\xi}_k)\|^2 > \varepsilon_a$ **then**
      insert new knot at $\bar{\xi}_k$
    end if
  end for

end while

the corresponding residual vectors as follows: Denote by $\boldsymbol{\Phi} \in \mathbb{R}^{m\times n}$ the matrix that contains the values of the n basis functions b_j, one row for each footpoint t_i. The matrix $\mathbf{B} \in \mathbb{R}^{n\times 3}$ shall gather the unknown control points, also one per row. Analogously, at each iteration, we need to assemble matrices $\mathbf{P} \in \mathbb{R}^{m\times 3}$, $\mathbf{N} \in \mathbb{R}^{m\times 3}$, and $\mathbf{T} \in \mathbb{R}^{m\times 3}$ respectively from the data points $\mathbf{p}_i$, normals $\mathbf{n}_i$, and tangent vectors $\mathbf{t}_i$. The (unweighted) residual of the data term for the PD is then given as

$$\mathbf{r} = \boldsymbol{\Phi}\mathbf{B} - \mathbf{P}. \tag{13}$$

The residual vectors arising from TD and SD are constructed in a similar manner. The second term of Eq. (12) responsible for smoothing is applied not to the curve directly, but instead to the control points. This is a reasonable approximation for Eq. (4), since the curvature vector $\mathbf{c}''(t)$ results from the arrangement of the control points as they are fitted to the data points. With the second-order centered-differences operator $\mathbf{R} = ((r_{ij}))$,

$$r_{ij} = \begin{cases} -1, & \text{if } i = j \\ \frac{1}{2}, & \text{if } i = j+1 \text{ or } i = j-1 \\ 0 & \text{else,} \end{cases} \tag{14}$$

the approximate discrete curvature residual becomes $\mathbf{r}_s = \mathbf{RB}$. The residual of the CF term in Eq. (12)

$$\mathbf{r}_c = \boldsymbol{\Phi}_c\mathbf{B} - \mathbf{P}_c \tag{15}$$

is derived analogously to Eq. (15) but $\boldsymbol{\Phi}_c \in \mathbb{R}^{(n+p)\times n}$ is now constructed from the values of the basis functions at the knot span centers $\bar{\xi}_k$, and $\mathbf{P}_c \in \mathbb{R}^{n+p} \times 3$ stacks the strapping points $\mathbf{p}^*_{c,k}$ defined in Eq. (11). From Eqs. (13), (15) and $\mathbf{r}_s = \mathbf{RB}$ combined with their respective weights, we obtain the following overdetermined sparse linear system

$$\begin{pmatrix} \mathrm{diag}(w_a(d_i))\boldsymbol{\Phi} \\ w_s\mathbf{R} \\ w_c\boldsymbol{\Phi}_c \end{pmatrix} \mathbf{B} = \begin{pmatrix} \mathrm{diag}(w_a(d_i))\mathbf{P} \\ \mathbf{0} \\ w_c\mathbf{P}_c \end{pmatrix}. \tag{16}$$

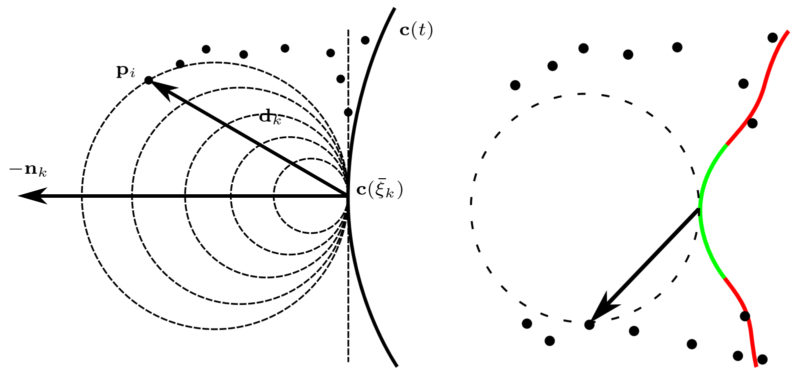
The solution of the Gaussian normal equation of (16) coincides with the solution of the original least-squares problem. In Algorithm 1 the optimization loop terminates if the incremental change of control points falls below a certain threshold.

3.2. Results

We evaluate the performance of our method with respect to three indicators: First, we compare it to state-of-the-art methods based on PD, TD and SD. Second, we demonstrate the robustness with respect to noise and point density by creating an artificial example with varying point distribution and keeping the parameters for curve fitting fixed. Third, we provide a quantitative evaluation reporting numbers on accuracy, speed and compression rate when applying our algorithm to the task of 3D reconstruction of point clouds.

Comparison. We qualitatively compare to the established methods based on PD, TD, and SD. A quantitative comparison is all but impossible since the former are not designed for treating clutter inside the boundary and rely on proper initialization schemes to obtain good solutions. However, as we want to point out the significance of our approach in real world scenarios, we show cases where the bare PD, TD, and SD are insufficient unless combined with the proposed asymmetric weighting function. Fig. 7 shows how outliers force the curve to evade the true boundary. This kind of noise is typical for segmented point clouds from Kinect-like sensors caused by illumination highlights, reflections and slanted surfaces, as shown in Fig. 4.

Robustness. We are using the same set of parameters for a certain type of point data (e.g. unorganized point clouds) and do not need



(a) Iso-value curves of d_c.

(b) Affected curve elements.

Fig. 10. *Concavity Filling*: (a) Iso-value curves for finding the closest point to the curve at point $\mathbf{c}(\bar{\xi}_k)$ w.r.t. the outward pointing normal vector $\mathbf{n}_k$. Note that with the distance metric $d_{c,k}$, the point $\mathbf{p}_i$ is closer than the point right above $\mathbf{c}(\bar{\xi}_k)$ and therefore used for fitting the curve along the knot span containing $\bar{\xi}_k$. (b) Only curve elements with no support from data points (green) are strapped to their closest point w.r.t. Eq. (10). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

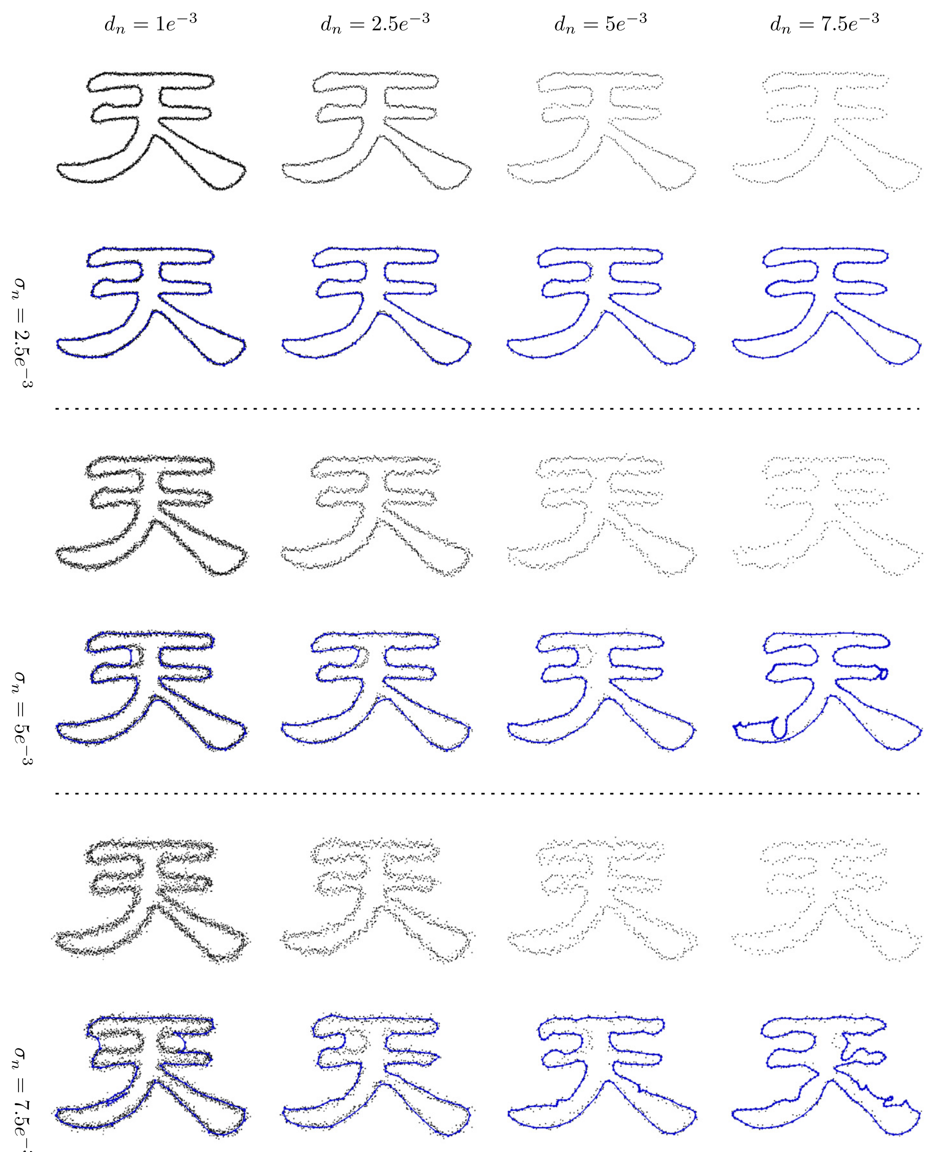


Fig. 11. Robustness when fitting the Chinese character *tian*, without changing the parameters but the distribution of the data points. Point distance d_n increases from left to right, noise increases from top to bottom. The resulting curve is shown in blue after 40 iterations. ($\sigma = 0.0002$, $\varepsilon_a = 0.015$, $w_s = 0.5$, $w_c = 1.0$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

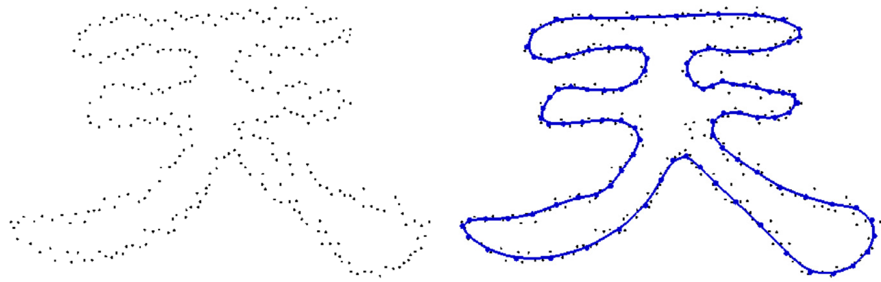


Fig. 12. Fitting of the difficult case in the lower-right of Fig. 11 ($\sigma_n = 7.5e^{-3}$, $d_n = 7.5e^{-3}$) with changed parameters ($\sigma = 0.0002$, $\varepsilon_a = 0.017$, $w_s = 0.5$, $w_c = 1.0$, 40 iterations).

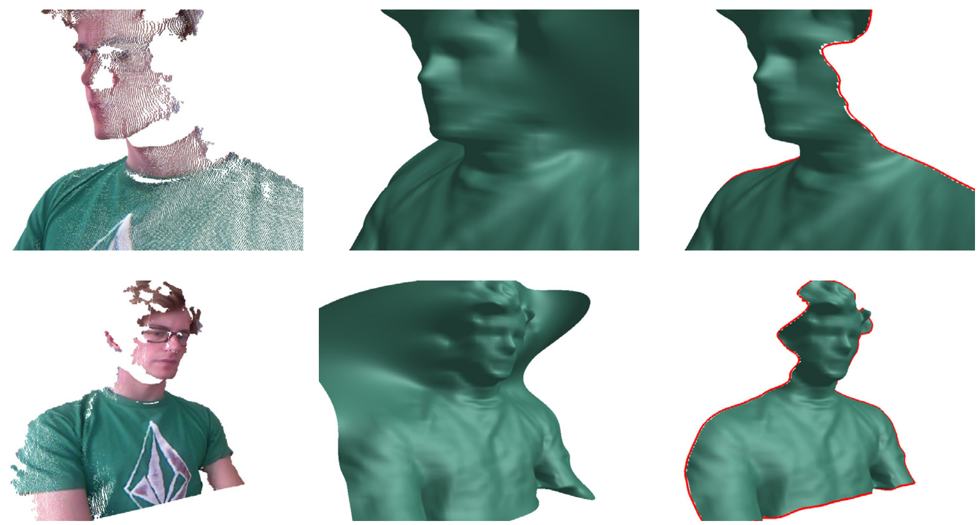


Fig. 13. Reconstruction: The curve is fitted to a point cloud (left) in the parametric domain of a B-spline surface (middle) which allows for trimming it (right).

Table 1
Evaluation results for point cloud reconstruction. From left to right: number of data points (#Pts), number of control points (#CP), mean error (ME) in milimeter or pixels (marked with *), computation time (t) in seconds and compression rate (CR). Index c indicates curve- and s surface measures.

	#Pts.	#Seg	#CP$_c$	#CP$_s$	ME$_c$	ME$_s$	t_c	t_s	CR
Office 1	285k	77	1254	693	7.1	6.4	7.5	0.6	130
Kitchen 1	594k	166	2628	1494	8.5	8.1	14.3	1.2	127
Kitchen 2	742k	184	3045	1656	8.6	8.2	18.4	1.5	140
Living 1	985k	448	6074	4032	9.8	10.1	37.4	3.8	85
NYU 935	219k	31	675	279	3.7*	8.5	14.5	0.4	209
NYU 1314	217k	20	409	180	3.2*	7.3	8.3	0.4	334
NYU 1315	201k	32	542	288	3.8*	8.5	11.9	0.4	218
OSD 51	182k	18	393	162	0.6*	1.8	7.9	0.4	299
OSD 59	180k	45	842	408	0.6*	1.5	15.1	0.4	130
OSD 61	178k	45	812	405	0.6*	1.6	12.7	0.4	131

to adjust them for each capture. To get a better impression of the variance the algorithm can handle with one set of parameters we have created a toy example, where we fit a curve to the outline of the Chinese character *tian*. Fig. 11 shows the results with changing point density and noise. Our method fails if the parameter for accuracy for EAKI ε_a is lower then the distance between some adjacent points which follows from the fact that fitting a concave boundary is an ill-posed problem. In other words EAKI and CF treat the gaps between points as concavities and try to fill them. However, changing the parameter ε_a changes this behavior and leads to a satisfying solution (Fig. 12).

Efficiency for 3D reconstruction. To demonstrate the efficiency in terms of run-time and data compression, we evaluated several single-view RGBD shots (organized point clouds) as well as full 3D scans of rooms (unorganized point clouds). We segmented the point cloud into piecewise smooth regions. For each of the segments, we compute the eigenspace and define the planar domain for the B-spline curve as the subspace spanned by the eigenvectors with the two highest eigenvalues. All points of the segment are projected onto this plane and a bounding curve is computed using our algorithm. For an accurate surface representation, we fit a B-spline surface to these points, with its parametric domain being the bounding box enclosing the projected points. Trimming and triangulation within the eigenspace becomes a trivial operation by uniformly triangulating the bounding box and subsequently cutting off vertices outside the B-spline curve (see Fig. 13). Table 1 shows mean errors, computation times and compression rates with respect to the raw 3D point cloud. Please note, that the numbers listed depend on the choice of parameters, i.e. by raising the parameter for accuracy, the error rates but also the compression rates drop, since more control points are required. Consequently the computation time increases, since more iterations are needed until convergence but also due to the higher number of parameters

Fig. 14. Reconstruction of organized segmented point clouds with B-spline curves and surfaces (NYU dataset [23]).

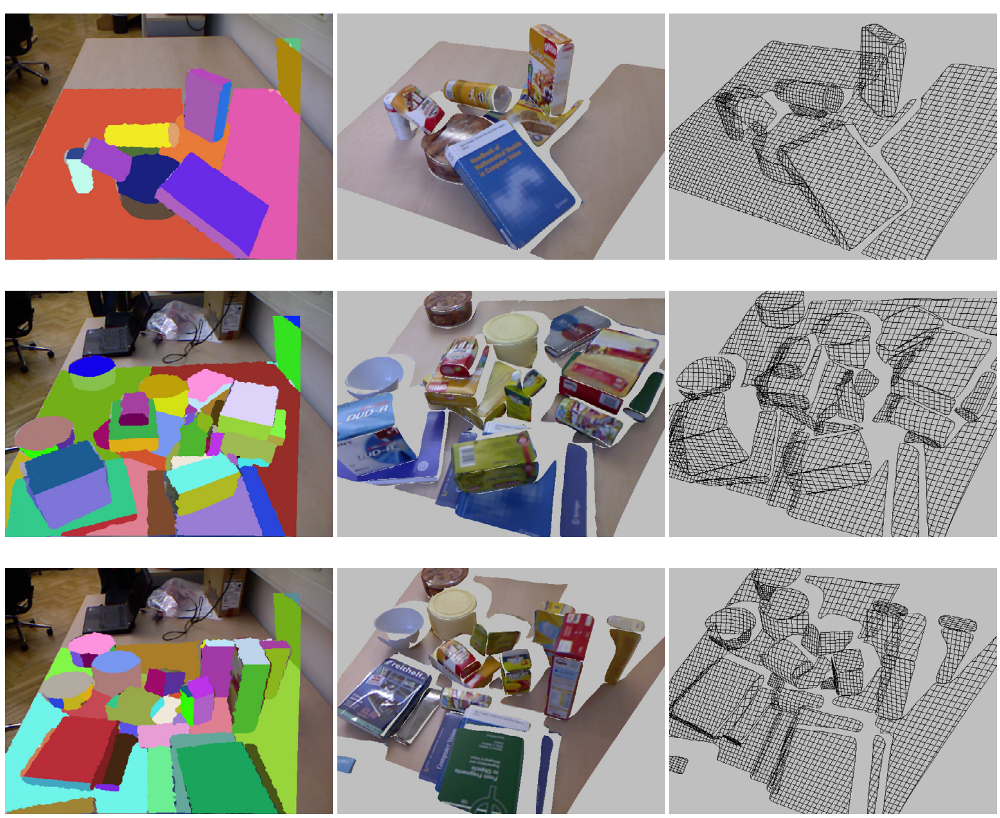


Fig. 15. Reconstruction of organized segmented point clouds with B-spline curves and surfaces (OSD dataset [4]).

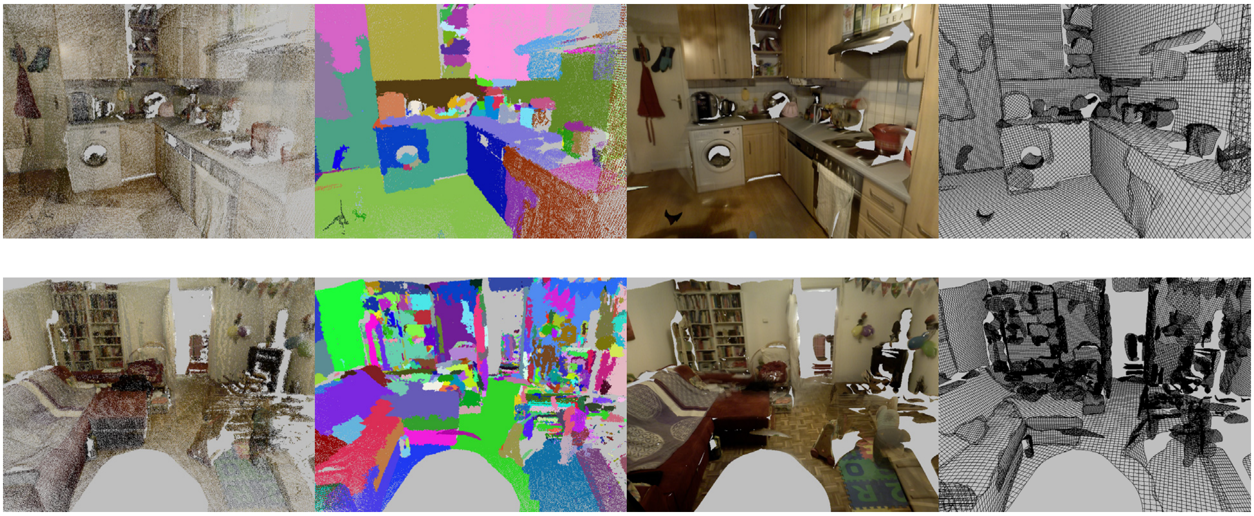


Fig. 16. Reconstruction of unorganized segmented point clouds with B-spline curves and surfaces.

(i.e. control points). Model files (.obj), images and videos of the reconstructed scenes are available on our webpage.[6] The scenes are shown in Figs. 14–16.

3.3. Discussion

The parameters available are defined such that they correspond to the characteristics of the data (*smoothness* w_s, *accuracy* ε_a, *concavity* w_c and *transition width* σ). As demonstrated at hand of the Berkeley data, one set of parameters is often sufficient to fit all the curves shown in the results. When the goal is to find the concave hull (which is generally not well-defined, cf. Fig. 3), we need those parameters to define which distance between points is treated as concavity (w_c), how much noise should be smoothed out (w_s), and finally which points should be treated as inside (σ) respectively as belonging to the contour.

One failure case of the proposed method is when point clouds consist of more than one cluster, i.e., the number of simply-connected components must be known beforehand. We are working on a method to detect self-intersections of the B-spline which would indicate the underlying topology. As stated in [24], splines are not immediately suitable for representing discontinuities. Still, one would be able to detect the locations of these discontinuities and reduce the continuity of the spline by knot insertion followed by a readjustment by our method.

Let us remark that although we have only shown experiments with closed curves, our approach would also work with non-periodic B-splines on open curves. In this case, the terms *inside* and *outside* simply translate to *left* and *right* w.r.t. the B-spline direction.

4. Conclusion

In this work, we have proposed an algorithm that robustly detects and approximates the contour of planar point clouds with B-spline curves. Due to the asymmetric distance term, our method is robust against clutter inside the actual contour. Governed by the contour, concavity filling forces the curve to bend inwards if appropriate. Our mechanism for error adaptive knot insertion provides additional degrees of freedom when filling concavities and at sharp corners and consequently increases the overall accuracy. We currently investigate how to estimate the required parameters from point cloud statistics for a more convenient or even fully automatic usage. Another direction we would like to follow is to transfer our method from curves to surfaces. For local knot insertion (EAKI), we would like to exploit the characteristics of T-splines [25], to fit them to 3D point clouds.

[6] http://users.acin.tuwien.ac.at/tmoerwald/?site=6.

References

[1] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, Int. J. Comput. Vis. 1 (1988) 321–331.

[2] J. Balzer, T. Mörwald, Isogeometric finite-elements methods and variational reconstruction tasks in vision—A perfect match, in: IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1624–1631.

[3] A. Aldoma, T. Mörwald, J. Prankl, M. Vincze, Segmentation of depth data in piece-wise smooth parametric surfaces, in: Computer Vision Winter Workshop, CVWW, 2015.

[4] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, M. Vincze, Segmentation of unknown objects in indoor environments, in: IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, 2012.

[5] W. Wang, H. Pottmann, Y. Liu, Fitting B-spline curves to point clouds by curvature-based squared distance minimization, ACM Trans. Graph. 25 (2006) 214–238.

[6] H. Pottmann, M. Hofer, Visualization and Mathematics III, Springer, 2003, pp. 223–244. (Chapter geometry of the squared distance function to curves and surfaces).

[7] A. Blake, M. Isard, Active Contours—The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion, Springer, 1998.

[8] H. Yang, W. Wang, J. Sun, Control point adjustment for B-spline curve approximation, Comput.-Aided Des. 36 (2004) 639–652.

[9] H. Park, Choosing nodes and knots in closed B-spline curve interpolation to point data, Comput.-Aided Des. 35 (2000) 123.

[10] H. Delingette, J. Montagnat, New algorithms for controlling active contours shape and topology, in: European Conference on Computer Vision, 2000, pp. 381–395.

[11] Z. Yang, J. Deng, F. Chen, Fitting unorganized point clouds with active implicit B-spline curves, Vis. Comput. 21 (2005) 831–839.

[12] H. Yang, B. Jüttler, Evolution of T-spline level sets for meshing non-uniformly sampled and incomplete data, Vis. Comput.: Int. J. Comput. Graph. 24 (2008) 435–448.

[13] R. Feichtinger, M. Fuchs, B. Jüttler, O. Scherzer, H. Yang, Dual evolution of planar parametric spline curves and T-spline level sets, Comput.-Aided Des. 40 (2008) 13–24.

[14] S. Flöry, Fitting curves and surfaces to point clouds in the presence of obstacles, Comput. Aided Geom. Design 26 (2009) 192–202.

[15] S. Flöry, M. Hofer, Constrained curve fitting on manifolds, Comput.-Aided Des. 40 (2008) 25–34.

[16] M. Hu, J. Feng, J. Zheng, An additional branch free algebraic B-spline curve fitting method, Vis. Comput. 26 (2010) 801–811.

[17] O. Grove, From CT to NURBS: Contour Fitting with B-spline curves, Comput. Aided Des. Appl. 8 (2011) 3–21.

[18] X. Zhaoa, C. Zhang, B. Yang, P. Li, Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation, Comput.-Aided Des. 43 (2011) 598–604.

[19] L. Barbieri, F. Bruno, M. Muzzupappa, J. Pernot, Constrained fitting of B-spline curves based on the force density method, in: International Conference on Innovative Methods in Product Design, 2011.

[20] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, M. Vincze, Learning of perceptual grouping for object segmentation on RGB-D data, J. Vis. Commun. Image Represent. 25 (2014) 64–73.
[21] T. Mörwald, A. Richtsfeld, J. Prankl, M. Zillich, M. Vincze, Geometric data abstraction using B-splines for range image segmentation, in: IEEE International Conference on Robotics and Automation, 2013.
[22] L. Piegl, W. Tiller, The NURBS Book, in: Monographs in Visual Communication, Springer, 1996.
[23] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: ECCV, 2012.
[24] K. Siddiqi, B. Kimia, C. Shu, Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution, Graph. Models Image Process. 59 (1997) 278–301.
[25] T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, ACM Trans. Graph. 22 (2003) 477–484.

Thomas Mörwald received a Ph.D. (Dr.Techn.) in Electrical Engineering at the Vision for Robotics group at the Vienna University of Technology (TUW) in 2013 and a M.Sc. (Dipl.-Ing.) in Mechatronics at the Johannes Kepler University in Linz, 2008. He was part of the CogX project [FP7/2007–2013], the InModo project [FFG/836490] and is now within the FLOBOT project [H2020- ICT-2014-1/645376]. He accomplished an internship in the USA (University of California, Los Angeles, 2013), Saudi Arabia (King Abdullah University of Technology, 2011), England (University of Birmingham, 2010), USA (Purdue University, 2006) and Brazil (Pontificia Universidade de Minas Gerais, 2005). His research interests are in the field of 3D object tracking, surface reconstruction using B- and T-splines, GPU computing, computer graphics and visualization.

Jonathan Balzer holds a doctorate in Computer Science (Dr. Rer. Nat.), a Masters Degree in Applied Mathematics (Dipl.-Math. Techn.), in addition to a Mechanical Engineering degree (Dipl.-Ing.), all from the Karlsruhe Institute of Technology. He has been a researcher at the Karlsruhe Institute of Technology, the Technical University of Vienna, King Abdullah University of Science and Technology (KAUST), and the University of California, Los Angeles (UCLA). He is currently the CTO and co-founder of Vathos LLC based in Dusseldorf, Germany.

Markus Vincze received his diploma in Mechanical Engineering from Technical University Wien (TUW) in 1988 and a M.Sc. from Rensselaer Polytechnic Institute, USA, 1990. He finished his Ph.D. at TUW in 1993. With a grant from the Austrian Academy of Sciences he worked at HelpMate Robotics Inc. and at the Vision Laboratory of Gregory Hager at Yale University. In 2004, he obtained his habilitation in robotics. Presently he leads the ''Vision for Robotics'' laboratory at TUW with the vision to make robots see. V4R regularly coordinates EU (e.g., ActIPret, robots@home, HOBBIT, Squirrel) and national research projects (e.g, vision@home) and contributes to research (e.g., CogX, STRANDS, ALOOF) and innovation projects (e.g., Redux, FloBot). With Gregory Hager he edited a book on Robust Vision for IEEE and is (co-)author of 42 peer reviewed journal articles and over 300 reviewed other publications. He was the program chair of ICRA 2013 in Karlsruhe. Markus' special interests are cognitive computer vision techniques for robotics solutions situated in real-world environments and especially homes.