Point Cloud Library - Trimble Code Sprint Final Report

Thomas Mörwald

September 21, 2012

1 Best-fit feature estimation

Based on sensor characteristics (noise, biases, range-dependent errors) and data-collection geometry (sensor location and orientation relative to the objects in the point cloud), estimate best-fit for common geometry types (e.g. surfaces, spheres, cylinders, etc.). Identify all points likely to be part of the object, including potentially identifying outliers that are likely part of the object, but not used in the best-fit solution.

2 Theoretical Background

A common way to represent geometry in computer graphics, CAD and computer vision are B-spline curves and surfaces as well as their abstraction to non-uniform rational B-splines (NURBS). They are heavily used in a wide field of applications including entertainment industry, mechanical-, electrical and medical engineering, architecture and computer vision. In the fundamental book of [3] the concept of B-spline curves and surfaces as well as NURBS are described. Simple methods like *least-squares-minimization* are employed for approximating point-clouds using B-splines. More advanced techniques redefine the distance measure between data points and the surface leading to a more robust and faster fitting procedure [1], [5]. As we will see later on B-spline surfaces are defined as the tensor product of their basis functions, which means that their boundaries are four-sided with the edges having the same order and degree of freedom as the surface itself. Looking at a disk, where the surface is planar (i.e. of 1st order) and the edge is a circle (i.e. of 2nd order), it becomes obvious that the edges have to be modelled separately.

2.1 B-spline Curves

Since the mathematical concept of B-splines would go far beyond the scope of this report we want to refer the interested reader to the book of Piegl et al. [3]. We start from the definition of B-spline curves:

$$\mathbf{c}(\xi) = \sum_{i=0}^{m} N_i(\xi)\mathbf{b}_i \tag{1}$$

To change the shape of the B-spline curve $\mathbf{c} : \Omega_c \to \mathbb{R}^2$ of degree p, the control points $\mathbf{b} \in \mathbb{R}^2$ are manipulated. $N_{i,p}(\xi)$ are the basis function defining the region of influence of $\mathbf{b}_i$. $\xi \in \Omega_c \subset \mathbb{R}$ is the parameter of the curve.

Fitting a B-spline curve to a set of data points $\mathbf{p}$ is the task of finding values for the control points that minimize the distance between $\mathbf{p}$ and $\mathbf{c}(\xi)$ as shown in Figure 1.

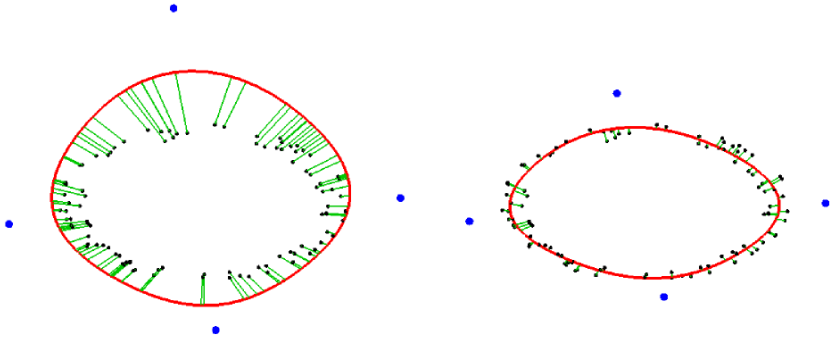

Figure 1: The distance (green) between the points (black) and the closed B-spline curve (red) is minimized by manipulating the control points (blue).

The measure for the distance is not necessarily the Euclidean distance between $\mathbf{p}$ and its closest point on the surface, which is also known as *point-distance-minimization* (PDM). Another possibility is to use *tangent-distance-minimization* (TDM) as in [1] or *squared-distance-minimization* (SDM) described in [5].

2.2 B-spline Surfaces

A B-spline surface $\mathbf{S} : \Omega_s \to \mathbb{R}^3$ of degree p is defined by using the tensor product of the basis functions.

$$\mathbf{S}(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(u) M_{j,p}(v) \mathbf{B}_{i,j} \tag{2}$$

where the parameters (u, v) are defined on the domain $\Omega_s \subset \mathbb{R}^2$. $\mathbf{B}_{i,j}$ is a grid of control points influencing the surface with the respective basis functions $N_{i,p}(u)M_{j,p}(v)$. The fitting approaches mentioned in Section 2.1 are equivalently applied as shown in Figure 2.

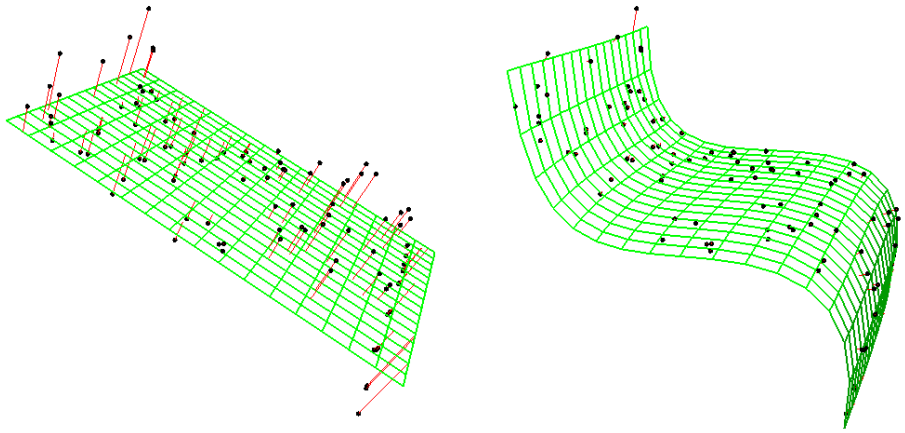

Figure 2: The distance (red) between the points (black) and the B-spline surface (green) is minimized by manipulating the control points.

2.3 Trimmed Surfaces

One possibility to get rid of the four-sided shape of a B-spline surface is to trim away areas that lie outside a certain region. Such a trimming region $\Omega_t \subset \Omega_s$ can be defined on the parametric domain using B-spline curves.

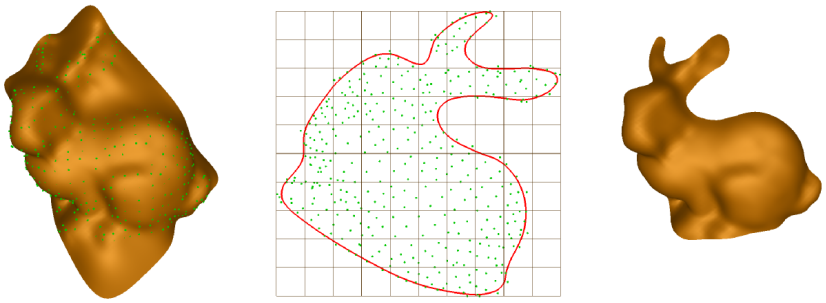

Figure 3: Trimming a B-spline surface (brown) using a B-spline curve (red). Left: The surface fitted to the data points (green). Middle: The curve modelling the outer contour defines a region Ω_t within Ω_s (grid). Right: The trimmed B-spline surface.

Figure 3 shows a sparse point-cloud of the Stanford bunny which is fitted and trimmed using B-spline curves and surfaces. A method on how to fit a B-spline curve on a manifold such that the curve encloses the points is described in [2].

3 Implementation

The code for fitting B-spline curves and surface was implemented in the sub-architecture *pcl/surface* within a sparate namespace *pcl:on_nurbs*. The concepts of B-splines and NURBS were taken from the free and open-source library *OpenNURBS*[1] where the classes *ON_NurbsCurve* and *ON_NurbsSurface* were exploited. The source code of OpenNURBS resides in

- *pcl/surface/openNURBS*

3.1 Curves

Considering the three distance measures PDM, TDM and SDM as mentioned in Section 2.1 we have implemented the algorithms for curve fitting for 2D point-clouds in the classes:

- *pcl::on_nurbs::FittingCurve2dPDM*
- *pcl::on_nurbs::FittingCurve2dTDM*
- *pcl::on_nurbs::FittingCurve2dSDM*

The curves are typically initialised using principal-component-analysis (PCA), where the curve is initialised in the plane formed by the two biggest eigenvectors of the data. The eigenvalues are used to define the radius of the initialised control points from the mean. Although smoothness is inherent in B-splines an

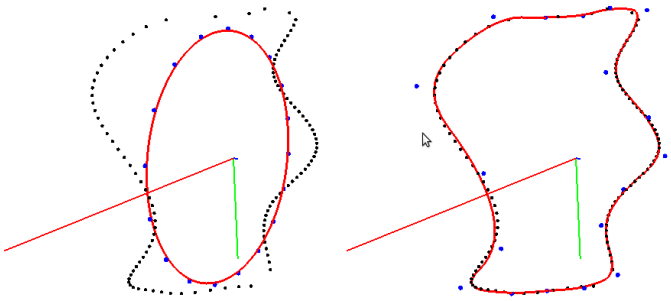

Figure 4: Left: The control-points (blue) of the curve (red) are initialised using the eigenvectors of the PCA (coordinate frame), calculated from the points to be fitted (black). Right: The 3D curve after fitting.

additional parameter can be defined by a non-discrete regularisation term for problems of high degree of freedom (i.e. high number of knots / control-points). The closest points on the curve are calculated using Newton's method.

Equivalent to 2D curve fitting the algorithms for 3D are implemented in the class

[1] http://www.opennurbs.org

- *pcl::on_nurbs::FittingCurve*

where 3D data and curves are processed. In contrast to the 2D cases TDM and SDM have not been implemented so far. Examples on fitting curves to a set of points are located in the files:

- examples/surface/example_nurbs_fitting_curve3d.cpp
- examples/surface/example_nurbs_fitting_curve.cpp

We have experimented with the distance measures PDM, TDM and SDM experiencing similar behaviour as described in [5], where SDM turns out to be the most robust way to fit curves. Unfortunately this comes to the cost of extra computational time and storage since the tangents and normals have to be evaluated and the basis functions for every dimension need to be defined separately.

3.2 Surfaces

The algorithms for surface fitting are implemented similar to curves. But instead of TDM and SDM we introduced our own method. In SDM a mixture of tangential-distance and point-distance is used depending on the local curvature of the curve at the closest point of a data point. In our method we allow the user to define the influence of tangent- and point-distance manually, leading to a lower computational load while convergence is faster than PDM but still robust enough. Unfortunately the required storage for fitting is as high as needed for SDM. Again the surface is initialised using PCA.

An example for surface fitting and trimming is located in:

- examples/surface/example_nurbs_surface.cpp

Trimmed Surfaces

To define a region $\Omega_t \subset \Omega_s$ we are using B-spline curves that reside in the parametric domain of the surface. The direction of rotation of the closed curve defines whether the area inside or outside is trimmed. Figure 5 shows trimming of a surface with a curve first defined clockwise and then counter-clockwise.

3.3 Triangulation

Triangulation of NURBS curves and surfaces is quite straight forward. The parameter space Ω_c and Ω_s respectively is uniformly sampled. A bit more effort has to be made for trimmed surfaces.

As for non trimmed surfaces a triangulated plane of a certain resolution in u and v direction is created, covering the whole domain Ω_s. Then each vertex has to be tested if it lies inside or outside the curve. Therefore the cross product of the vector of a vertex to its closest point on the curve with the tangent vector at the closest point is checked whether it is positive or negative with respect to

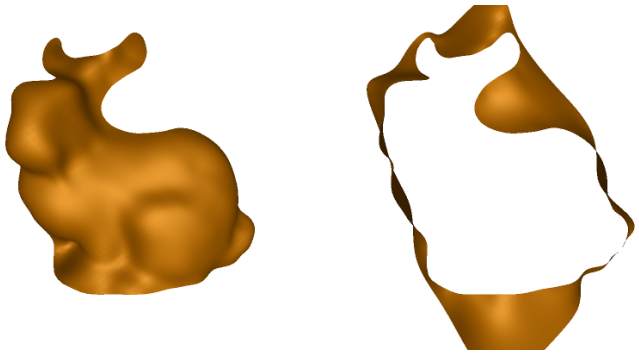

Figure 5: Trimming the surface fitted to the bunny example with curves of the same shape. Left: Direction of rotation of the curve defined clockwise. Right: Direction of rotation of the curve defined counter-clockwise.

z direction. The z direction is defined by the coordinate frame of the surface given by the PCA (or initialised by the user).

Knowing which vertices are inside the curve we trim triangles where no vertex is inside and fully add triangles where all of them are inside. For those triangles having inliers and outliers, the outlying vertices are moved to the trimming curve and the triangle is also added to the mesh. The implementation of triangulation is located in the class

- *pcl::on_nurbs::Triangulation*

3.4 Solving

Linearisation

The mathematical problem of fitting a B-spline curve or surface to a point-cloud can be described as linear system. Equation (1) can be reformulated as

$$\mathbf{f}_c = \mathbf{A}_c \mathbf{b}_c \tag{3}$$

where $\mathbf{f}_c$ is the vector of points to be fitted. $\mathbf{A}_c$ is a matrix containing the basis function, which multiplied by the vector of control-points $\mathbf{b}_c$ results in the curve at the respective parameter ξ. Minimizing Equation (3) in the PDM sense leads to minimizing

$$||\mathbf{A}_c \mathbf{b}_c - \mathbf{f}_c||^2 \tag{4}$$

Equivalent to curve fitting Equation (2) leads to

$$\mathbf{f}_s = \mathbf{A}_s \mathbf{b}_s \tag{5}$$

and

$$||\mathbf{A}_s \mathbf{b}_s - \mathbf{f}_s||^2 \tag{6}$$

Solver

Such a system can be solved by using standard solvers such as

- *Eigen::JacobiSVD*
- *SuiteSparse::umfpack_di_solve*[2]

where the first is a dense and the second a sparse solver which enormously speeds up the computation time. The choice of the solver can be made in the *BUILD* configurations by enabling or disabling the flag *USE_UMFPACK*.

4 Future Work

Considering the huge amount of literature on B-spline fitting a lot of modern algorithms could be implemented in future. However, there are two main issues that remain open and are desired by the community. The first one is to be able to fit a closed surface equivalent to closed curves and second to generalize the fitting approach to NURBS.

Impact

Since the PCL community has undergone an immense growth during the last year, the *pcl::on_nurbs* subarchitecture has already been used during development and developer and researchers from all over the world are frequently asking questions about the usage.

In our research group we are interested mainly on object reconstruction and segmentation where we already succeeded pushing the state-of-the art [4]. We hope that also others benefit from this contribution to PCL.

Acknowledgement

The work leading to these results has been achieved during the PCL Trimble Code Sprint, funded by Trimble.

References

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

[2] S Flory and M Hofer. Constrained curve fitting on manifolds. *Computer-Aided Design*, 40(1):25–34, 2008.

[2] http://www.cise.ufl.edu/research/sparse/SuiteSparse/

[3] Les Piegl and Wayne Tiller. *The NURBS book*. Monographs in visual communication. Springer, 1996.

[4] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of Unknown Objects in Indoor Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[5] Wenping Wang, Helmut Pottmann, and Yang Liu. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics*, 25(2):214–238, 2006.