Geometric data abstraction using B-splines for range image segmentation

Thomas Mörwald, Andreas Richtsfeld, Johann Prankl, Michael Zillich, Markus Vincze
Automation and Control Institute
Vienna University of Technology
1040 Vienna, Austria
moerwald,ari,jp,mz,mv@acin.tuwien.ac.at

Abstract—With the availability of cheap and powerful RGB-D sensors interest in 3D point cloud based methods has drastically increased. One common prerequisite of these methods is to abstract away from raw point cloud data, e.g. to planar patches, to reduce the amount of data and to handle noise and clutter. We present a novel method to abstract RGB-D sensor data to parametric surface models described by B-spline surfaces and associated boundaries. Data is first pre-segmented into smooth patches before B-spline surfaces are fitted. The best surface representations of these patches are selected in a merging procedure. Furthermore, we show how curve fitting estimates smooth boundaries and improves the given sensor information compared to hand-labelled ground truth annotation when using colour in addition to depth information. All parts of the framework are open-source and are evaluated on the object segmentation database (OSD) also available online, showing accuracy and usability of the proposed methods.

I. INTRODUCTION

An important domain of mobile robotics is perception, the extraction of meaning from sensory input, allowing autonomous robots to operate in complex environments performing user-defined tasks. Sarkar and Boyer [22] introduced a classificatory structure for perceptual organisation in computer vision, showing different levels of abstraction to bring meaning to the data under the assumption that our world is not visually chaotic, but has structure and organisation.

Representing data as parametrized models means data reduction and allows to establish a continuous model with distinctive characteristics such as its polynomial order, area, curvature and normals. Relations can be defined between higher level entities beyond raw point clouds (e.g. the finger tip of a robotic hand touching the surface of an object). Physical behaviour constrained by these relations can be incorporated [12] and even learned when visual tracking algorithms [17], [6] observe the motion of a certain object during a defined robotic manipulation [13]. A popular method to represent objects is to use shape primitives such as planes [8], boxes, cylinders, spheres and cones [16], [9] or triangulated meshes [21]. These primitives are extended by superquadrics in [3] and [2] allowing for a higher variability and generalisation. [15] use continuous surfaces to compute the geodesic distance to deal with objects with holes (e.g. a mug with handle) whereas [5] use implicit surfaces for shape estimation and grasping.


Unfortunately data from RGB-D sensors suffer from several artefacts. First, points at edges of occlusion are broken as some of them are assigned to the wrong surface (labelled with ‘a’ in Figure 2). Second, the colour of the points does not match the depth information given, especially at edges of occlusion. Ellipses labelled with ‘a’ indicate points broken at edges of occlusion whereas ‘b’ are points where the colour is not consistent with the depth. Right: Depth values are corrected using B-splines for representing surfaces and contours.

In this paper we tackle these problems and aim to solve all of them.
In this work we abstract RGB-D data with plane and B-spline models to represent segments as well as their boundaries. Compared to other approaches [1] our definition of surfaces provides a direct mapping from image into 3D world space. This allows to correct wrong and undefined values in the point-cloud leading to consistent depth and colour information. Adjusting the boundaries of the patches to the colour edges shows a considerable improvement of the segmentation, tested against hand-labelled ground-truth.

The remainder of this paper proceeds as follows. Section II describes how the range image is pre-segmented by clustering normals of locally planar patches. Introducing B-spline models of first and second order (four and nine control points) these planar patches are merged to continuous regions in Section III. In Section IV the contours of the segments are adjusted using B-spline curve fitting similar to [10] and points are reassigned accordingly. Further B-spline surfaces are fitted to the new point-cloud patches which allows for mapping 2D points from image space into 3D world space and correcting the depth data respectively. Our contributions are tested against ground-truth and the improvement is illustrated in Section V, followed by a summarizing discussion in Section VI.

II. PRE-SEGMENTATION

3D cameras, such as Microsoft’s Kinect or Asus’ Xtion provide RGB-D data, consisting of a colour image and the associated depth information for each pixel. The task of the pre-segmentation module is twofold: First the sensor characteristics are modelled and considered during surface normal calculation. Second neighbouring pixels are clustered to uniform patches without discontinuities using these normals. In our framework we exploit the relationship between 2D image space and associated depth information.

A. Normal calculation

An easy way to calculate the normals of a point-cloud is to locally fit planes to neighbouring 3D points. We use organized point-clouds as obtained from the Microsoft Kinect, where a kernel defines the neighbourhood of a certain point. There are two parameters that influence the normal calculation: the kernel radius $k_r$ and the inlier distance $d_{in}$, to account for high deviations that would distort the local plane. The former one defines the number of points used and therewith the smoothing of the normals, the latter the maximum allowed euclidean distance of the points within the window to the centre point to be considered for the normal calculation.

We estimated the spatial distribution of points from a plane with different distances and with different angles to the camera for a Microsoft Kinect. Figure 3 shows the standard deviation of the plane normals over the kernel radius $k_r$ and the distance between camera and plane point (depth). To limit the standard deviation of the normals to a certain value without loosing too much details, the kernel radius $k_r$ has to be adjusted for different depth intervals to compensate for the noise of the sensor data which increases with distance.

Figure 4 shows the mean and standard deviation of the euclidean distance between neighbouring plane points as histogram over the depth. The slope in the histogram lets us approximate the inlier distance for the normal calculation which we can adjust by the parameter $\kappa_g$:

$$d_{in}(z) = \kappa_g k_r(z)$$

B. Normal clustering

Recursive clustering of normals is controlled by the maximum allowed angle between normals $\gamma_{cl}$ and by the maximum allowed normal distance of points $d_{cl}$ to a hypothesized plane, defined by the mean of the already clustered point normals and the mean position. Since Equation (1) does not reduce the noise itself, but reflects the limit to measurability of discontinuities, the distance $z$ still influences normal clustering which we model as

$$\gamma_{cl}(z) = \epsilon_g z + \epsilon_c$$

$$d_{cl}(z) = \omega_g z + \omega_c$$

Fig. 3. Standard deviation of plane normals with respect to kernel radius $k_r$ and depth with a Microsoft Kinect sensor. (The red points show the selected kernel radius values)

Fig. 4. Histogram of the mean and standard deviation of the distance ($\Delta d$) between neighbouring plane points over the depth of the plane, measured with a Microsoft Kinect sensor. (The black line indicates the selected $\kappa_g$)
where $\epsilon_c, \omega_c$ are the offset and $\epsilon_g, \omega_g$ the slope of the function.

III. Parametrization and Model Selection

In the last section continuous patches were extracted from RGB-D data. Parametrization of these patches to certain surface models allows for a more compact representation. Two parametric models are chosen, a plane model to represent simple planar patches and B-spline surfaces allowing to represent curved surfaces of higher polynomial order. Note that B-splines could also represent planes and we could thus skip explicit plane models, but planes are more efficient in terms of data size, fitting and processing (e.g. intersections, distance).

Neighbouring patches are merged after parametrization, if a joint parametric model fits better than the two individual models. To come to a decision, model selection with a minimum description length (MDL) criterion [14] is used. The sum of savings

$$S_H = \frac{N}{A_m} - \kappa_1 S_m - \kappa_2 \sum_{k=1}^{N} (1 - p(f_k|H)),$$

for models of neighbouring patches $S_{H,i}$ and $S_{H,j}$ are compared with savings of a model fitted to a merged patch $S_{H,ij}$ and in case $S_{H,ij}$ is larger the individual patches are substituted. The number of data points explained by the hypothesis $H$ are annotated with $N$, the costs for coding different models $S_m$ (3 for planes and 9 for B-spline patches, corresponding to the number of control points needed to describe them). $p(f_k|H)$ is the probability, that a data point $f_k$ belongs to $H$, modelled with a Gaussian error model (the error is the closest distance between the data points and the surface). $A_m$ is a normalization value representing the size of merged patches and $\kappa_1$ and $\kappa_2$ are constants to weight the different terms.

IV. Model Refinement

Given the segmentation of the point-cloud as described in Section III, we aim to find a more accurate contour for each point-cloud patch such that its projection matches the colour image. If this is not the case (Figure 2) points need to be reassigned from one patch to another which requires to know if a point lies inside or outside a certain region. We use closed B-spline curves to define these regions and take advantage of fitting techniques for adjustment. Using the curve, each point in the patch is verified as inlier or otherwise excluded.

A. Curve fitting

A common way to model free-form curves and surfaces in computer-aided design (CAD), computer graphics and computer vision are B-splines and their generalisation the so-called non-uniform rational B-splines (NURBS). The reason for their popularity are the ability to represent all conic sections, i.e. circles, cylinders, ellipsoids, spheres and so forth. They are convenient to manipulate and possess useful mathematical properties, such as refinement through knot insertion, $C^{p-1}$-continuity for $p$-th order curves and the convex hull properties which we will exploit in Section IV-C.

Since the mathematical concept of B-splines would go far beyond the scope of this paper we want to give a short overview and refer the interested reader to the book by Piegl et al. [18]. A B-spline curve $c$ in $\mathbb{R}^2$ is given by

$$c(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) b_i,$$

The idea is to manipulate the B-spline curve $c : \Omega_c \rightarrow \mathbb{R}^2$ of degree $p$, by changing the values of the control points $b \in \mathbb{R}^2$. The $i$-th control point defines the B-spline curve at its region of influence determined by the basis function $N_{i,p}(\xi)$ where $\xi$ is called parameter defined on the domain $\Omega_c \subset \mathbb{R}$.

Fitting a B-spline curve of a certain degree of freedom to a set of points $p$ is the task of manipulating the control points such that the distance between the control points and the curve is minimized (Figure 5). We use a common approach for curve fitting as described in Chapter 9 of Piegl et al. [18].

B. Contour refinement

For each patch of the segmentation we compute the contour of the points in image space shown in the top left of Figure 6. For this we parse all pixels of the segmented region and mark them if at least one of the 4-connected neighbourhood pixels does not belong to the region. The marked pixels include the outer boundary, but also inner clutter points if the region is not completely dense or has holes. Hence, to get the outer ordered boundary we trace the pixel chains with an 8-connected neighbourhood kernel and select the largest chain which surrounds the inner clutter points.

To assemble the data points necessary for B-spline curve fitting we search for points that lie on edges of the image which are computed using the Canny edge detection algorithm. The top right of Figure 6 shows the updated contour.

The green points shown in the top right of Figure 6 are now used for fitting a B-spline curve. The curve is initialised by setting its control points to the average of a set of successive points of the sorted data. Afterwards the distance is minimized as shown in Figure 5. If, after convergence, the error of each point is still higher than a certain threshold ($\sim$ 2 pixels), new knots are inserted to allow for higher degree of freedom and fitting is repeated iteratively. The bottom right of Figure 6 shows the resulting curve.
Fig. 6. Adjusting the contour points to the edges of the image. Green pixels lie on image edges (black) in contrast to red pixels. Top-Left: Contour of the point-cloud patch. Top-Right: Contour updated with respect to the image edges. Bottom-Left: Colour image. Bottom-Right: Curve fitted to the updated contour.

Now that we have a more accurate representation of the region, we can identify points that lie outside of the curve. If such a point is inside the B-spline curve of another patch in its neighbourhood it is assigned to this patch. Unfortunately the depth of the newly assigned point is not known which will be treated in the following.

C. Image based surface fitting

Similar to Equation (5) a B-spline surface is given as

\[ S(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(u) M_{j,p}(v) B_{i,j} \]  

(6)

where \( N_{i,p}(u) M_{j,p}(v) \) is the tensor product of the B-spline basis functions that defines the influence of the control point \( B_{i,j} \in \mathbb{R}^3 \).

As shown on the left of Figure 7 we define the parametric domain \( \Omega_s \subset \mathbb{R}^2 \) with \( (u, v) \in \Omega_s \), of the surface \( S: \Omega_s \rightarrow \mathbb{R}^3 \) to be a subset of the image space of the camera. In more detail we define \( \Omega_s \) to be the bounding box of the control points of the B-spline curve. Together with the convex hull property of B-splines this ensures that every point inside the curve lies inside the bounding box of the control points and with Equation (6) has its corresponding point in \( \mathbb{R}^3 \). This means that these points and also the curve itself can be transformed to \( \mathbb{R}^3 \) as shown on the right of Figure 7 which is used for trimming the surface to fully model the patch in a geometric sense.

Similar to B-spline curve fitting we again minimize the distance between the 3D-points and the B-spline surface. As we require the projection of the B-spline surface to remain unchanged we constrain the control points such that they only vary in \( z \)-direction. Please look up Chapter 9 of [18] again for more details.

With this tool at hand points that do not lie within the contour of a certain patch are reassigned to neighbouring segments. Whether or not a depth value is available, it is calculated by mapping the image point to the 3D surface. Computing the surfaces and curves for all segments leads to a highly accurate, continuous, consistent and efficient representation of the scene which significantly improves the end result as we will demonstrate in the following section.

V. Experiments

We evaluate our approach using the online available Object Segmentation Database (OSD-0.2) [19], consisting of 110 table top scenes with various kinds of objects and with different complexities of scenes. The dataset provides RGB-D data with hand-annotated ground truth for all objects.

Pre-segmentation, model parametrization and model refinement is evaluated with respect to over- and under-segmentation of surface patches on objects. Since the OSD provides ground truth for segmented objects rather than continuous regions, segments from our approach are assigned to an object when more than half of it overlaps. \( N_{true} \) and \( N_{false} \) are the number of correct and incorrect segmented points and \( N_{all} \) the number of all points on an object. Over-segmentation \( F_{os} \) and under-segmentation \( F_{us} \) is then defined as:

\[ F_{os} = 1 - \frac{N_{true}}{N_{all}} \]

(7)

\[ F_{us} = \frac{N_{false}}{N_{all}} \]

(8)

For a sequence of images with multiple objects in a scene, \( N_{true} \), \( N_{false} \) and \( N_{all} \) are summed up over all frames.

A. Pre-segmentation

Adjusting the parameters for pre-segmentation is dependent on the deployed RGB-D sensor, in our case a Microsoft Kinect.
With the measurements in Fig. 3 and Fig. 4 we can infer the parameters for normal calculation shown in Fig. 3 with the chosen kernel sizes and in Fig. 4 with the chosen slope:

- with $z = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$
- $k_{\epsilon}(z) = \{2, 2, 2, 3, 4, 5, 6, 7\}$
- $\kappa_{\epsilon} = 0.01$

Equivalent to normal calculation we measure the noise of the sensor leading to the following parameters:

- $\epsilon_g = 0.07$
- $\omega_g = 0.015$

### TABLE I

<table>
<thead>
<tr>
<th>$\omega_c$</th>
<th>$\epsilon_c = 0.46$</th>
<th>$\epsilon_c = 0.52$</th>
<th>$\epsilon_c = 0.58$</th>
<th>$\epsilon_c = 0.64$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>$\alpha$: 3.89%</td>
<td>$\alpha$: 3.97%</td>
<td>$\alpha$: 3.97%</td>
<td>$\alpha$: 3.97%</td>
</tr>
<tr>
<td></td>
<td>$\omega$: 1.49%</td>
<td>$\omega$: 1.61%</td>
<td>$\omega$: 1.72%</td>
<td>$\omega$: 1.72%</td>
</tr>
<tr>
<td></td>
<td>$\kappa$: 7.695</td>
<td>$\kappa$: 6637</td>
<td>$\kappa$: 5852</td>
<td>$\kappa$: 5383</td>
</tr>
<tr>
<td>-0.002</td>
<td>$\alpha$: 3.92%</td>
<td>$\alpha$: 3.96%</td>
<td>$\alpha$: 3.96%</td>
<td>$\alpha$: 3.96%</td>
</tr>
<tr>
<td></td>
<td>$\omega$: 1.49%</td>
<td>$\omega$: 1.60%</td>
<td>$\omega$: 1.72%</td>
<td>$\omega$: 1.72%</td>
</tr>
<tr>
<td></td>
<td>$\kappa$: 7.834</td>
<td>$\kappa$: 6774</td>
<td>$\kappa$: 6017</td>
<td>$\kappa$: 5597</td>
</tr>
<tr>
<td>-0.004</td>
<td>$\alpha$: 3.90%</td>
<td>$\alpha$: 3.92%</td>
<td>$\alpha$: 3.93%</td>
<td>$\alpha$: 3.93%</td>
</tr>
<tr>
<td></td>
<td>$\omega$: 1.48%</td>
<td>$\omega$: 1.59%</td>
<td>$\omega$: 1.66%</td>
<td>$\omega$: 1.69%</td>
</tr>
<tr>
<td></td>
<td>$\kappa$: 8.175</td>
<td>$\kappa$: 7164</td>
<td>$\kappa$: 6450</td>
<td>$\kappa$: 6089</td>
</tr>
<tr>
<td>-0.006</td>
<td>$\alpha$: 3.88%</td>
<td>$\alpha$: 3.88%</td>
<td>$\alpha$: 3.90%</td>
<td>$\alpha$: 3.90%</td>
</tr>
<tr>
<td></td>
<td>$\omega$: 1.47%</td>
<td>$\omega$: 1.56%</td>
<td>$\omega$: 1.62%</td>
<td>$\omega$: 1.64%</td>
</tr>
<tr>
<td></td>
<td>$\kappa$: 9.878</td>
<td>$\kappa$: 9036</td>
<td>$\kappa$: 8423</td>
<td>$\kappa$: 8091</td>
</tr>
</tbody>
</table>

Table I shows over- and under-segmentation $\omega$ and the number of produced patches $n$ for different values of parameter $\omega_c$ and $\epsilon_c$, while keeping the other parameters constant. A significant jump of the number of produced patches can be seen in Tab. I when the parameters are set below the noise level (e.g. $\epsilon_c=0.46$ or $\omega_c=-0.006$). The improvement of over- and under-segmentation does not correlate with the number of patches produced. On the other side, when turning the parameters above the noise level the number of produced patches decreases. For the following evaluation we used $\epsilon_c = 0.58$ and $\omega_c = -0.004$, which is a trade-off between accuracy and number of produced patches.

Figure 8 shows two examples of pre-segmentation, both with two sets of parameters. While for the top-left image the parameters are optimal the right image shows additional patches on the edges of the box on the top. On the other hand the same set of parameters causes under-segmentation on the left image (blue patch) while the right image shows better pre-segmentation.

### B. Parametrization and model selection

During the parametrization and model selection procedure, described in Section III, neighbouring patches are getting merged. For the chosen $\omega_c$ and $\epsilon_c$ the number of patches decreases from 6450 to 5113 (20.73%), while over-segmentation stays constant at 3.92% and under-segmentation increases insignificantly from 1.66% to 1.71%. Reduction of patches through merging is highly dependent on the used images, because it happens usually on curved surfaces when planar pre-segmented patches get merged to a bigger B-spline surface. However, typically neighbouring patches of the same object get merged, if no discontinuities are in between.

### C. Model refinement

Model refinement, described in Section IV, improves the boundary of segmented patches using colour information. Further depth data not available is filled by mapping the point from image space to a 3D surface. This leads to the following improvement with respect to segmentation accuracy: The value for over-segmentation decreases about 36.22% (from 3.92% to 2.50%) and under-segmentation about 31.22% (from 1.71% to 1.17%).

Figure 9 shows the sequential steps of our segmentation framework. RGB-D images are pre-segmented to planar patches, which are then merged with respect to curved surfaces during model selection. The refinement step fills missing data and adjusts depth and colour edges.

### VI. Conclusion

In this work we presented a novel method for representing RGB-D images with parametric models which play an important role especially in the field of service robotics (e.g. learning physical behaviour, grasping). Several contributions are described leading to improved segmentations. A model of the sensor noise is incorporated into pre-segmentation, where planar regions are clustered together. The employed model selection method merges these using B-spline surfaces to represent curved patches. Furthermore the boundaries of the segmentation are refined using B-spline curves. Our novel formulation of B-spline surfaces, where the inverse image is defined in the image space, provides a direct mapping to 3D space. This allows to refine poor segmentation especially
when colour and depth information is not consistent. Segmenting occluded objects, where regions might not be connected, are treated in our previous work [20].

ACKNOWLEDGEMENT

The research leading to these results has received funding by the Austrian Science Foundation under grant agreement No. I513-N23 and TRP 139-N23.

REFERENCES