User Guidance and Automatic Completion for Generating Planar B-Rep Models

Dorian Rohner, Maximilian Sand, and Dominik Henrich

Lehrstuhl für Angewandte Informatik III (Robotik und Eingebettete Systeme),
Universität Bayreuth, D-95440 Bayreuth, Germany
`dorian.rohner@uni-bayreuth.de`

Abstract. A representation of the world as a 3D model is a common necessity in robotics and automation. In previous work, we developed a concept to generate boundary representation (B-Rep) models from multiple point clouds using a hand-held depth-camera and to register them without a prior known pose. During the online reconstruction, properties of the sensor and the system (like noise) lead to small holes in the B-Rep. To prevent tedious post-processing, holes should be closed during the reconstruction. Our goal is to automatically close identified holes. However, not every hole can be closed automatically, as it may be unreasonable. For this case we develop a visual indication for the user, so he can close the hole by recording another depth image. In an experimental validation, we conclude the usefulness of the addition to the system.

Keywords: Reconstruction, Volumetric models, Visualization, CAD

1 Introduction

To generate a 3D representation of the surrounding world is a necessary skill needed in robotics, automation, and their applications. For example, it is useful for the reconstruction of a factory layout or of an robot cell [12]. This representation ranges from point clouds to more complex models, e.g. boundary representation (B-Rep) models. In our previous work [3], a method to generate planar B-Reps from a single point cloud was presented. Multiple point clouds can be incorporated in real-time, providing a valid B-Rep model in each time-step. By using the known pose of the camera (e.g. in [3] a robotic manipulator was used) two B-Rep models can be merged. The B-Rep is handled by an half-edge data-structure. The requirement of a known pose can be lifted by utilizing the method from our work [4], by using the angles between three linear independent faces to generate a possible rigid body transformation between two B-Reps. By combining these approaches, it is possible to generate a B-Rep as representation of the world by using a depth sensor, without a known camera pose. This enables the use of hand-held cameras, which leads to applications like object and robot cell reconstruction. The camera is moved by a user through the scene. Every point cloud is reconstructed into a B-Rep and merged into the world representation. However, the resulting model is not necessarily complete. One

possible reason may be occlusion, as a result of a complex scene. A similar aspect are missing points due to absorbed or reflected light. Another problem is that a scene can not be reconstructed from too noisy data. A last possible source for errors is the user, e.g. when he misses parts of the scene. There are two methods for dealing with the problem of incomplete models: On the one hand, sufficient small holes inside the representation should be closed automatically to enable a comfortable usage of the system. On the other hand, it is necessary to direct the user during the recording if the holes can not be closed automatically.

This paper consists of the following parts: The next section presents the state of the art how to handle incomplete representations. We present our approach in Section 3. First, we determine attributes to classify holes and afterwards we suggest a method to automatically close them and to generate user guidance for holes, which can not be closed automatically. The approach is validated in Section 4. In the last section, we discuss our contribution and future work.

2 State of the Art

The most simplest way to give user guidance is a visualization of the model [5]. The model is typically rendered from the user’s view. This visualization helps the user detect missing parts. A more advanced method is visual feedback to notify the user of missing parts. This can be done by colouring the backsides of the model, as they are only visible, when a hole exists and the user can look inside the model [1]. Finally, user guidance is possible by suggesting a camera pose, from which the hole can be closed. This problem is known as *Next-Best-View* in the context of robotics [6, 7]. However, a precise pose is unnecessary as it is impossible for a human to take exactly this pose. Additionally, in our domain of live reconstruction there is no set of possible views, and the human operator may have additional knowledge of the scene. Therefore, it is only necessary to identify holes and give a hint from which direction to look. It is not necessary to give a quality measure for each pose or to propose a reasonable one, as all holes should be closed in the end and the selection can be left to the human. Overall it is at least necessary to give a visualization to ensure a simple and intuitive live reconstruction. This is based on the Hand-Eye-Control System presented in [2], which reasons, that a human reacts in such a way, that the difference between the current state and a goal state is reduced [1].

The other problem mentioned are small holes. For this incompleteness user guidance can be generated, but it is not always helpful, especially when the holes exist because of occlusion or properties of the sensor. Therefore it is necessary to automatically close these holes. This problem is known as *Model Repair* [8, 9]: In general, these methods can be split into procedures based on point clouds or on meshes. The mentioned methods are typically based on the surface or on volumetric grids. In this case we will focus on the mesh-based methods, due to the B-Rep as the underlying structure. The surface-based methods follow different approaches. One possible way is to split the hole in smaller, easier to repair ones [16]. Another way is based on a triangulation of the hole, with the goal of

Fig. 1: Examples for holes with different orientation and different cardinality k . The blue half-edges are the hole, the red one the transition-half-edges.

Fig. 2: Visualization of the used variables for an outer hole with $k = 3$. There are two separated transition-half-edges and one twin-transition-half-edge, the hole-half-edges (blue), the transition-half-edges (red), transition-vertices (red-dotted), and the intersection-point of the three face's planes (green).

minimizing the amount of necessary triangles [17]. A last group of methods is based on a implicit surface representation and trying to utilize additional knowledge resp. properties of the object [13–15]. The volumetric approach is based on using a volumetric representation, with the idea of expanding the area around the hole, and therefore closing the model. These methods are usually more reasonable for more complex holes. In contrast to these methods, we are using the B-Rep data structure, therefore a direct application of these methods is not possible. Methods especially for B-Reps use features like edges and vertices, but are also based on a tessellation [10, 11]. Additionally, our goal is to use the hole closing during the live reconstruction, so there may not be enough information to close the hole with the before mentioned methods.

3 Our Approach

3.1 Definition and Handling of holes

Based on B-Rep models, it is easy to determine incomplete parts within a B-Rep due to the half-edge data-structure by testing, whether every half-edge has a twin-edge. Therefore, a *hole* is a closed loop of half-edges without twin-edges. A hole can be described by its *cardinality* k , which is the number of adjacent faces of the hole. Using this definition, the external boundary of the B-Rep is also recognized as a hole. This leads to the definition of the *orientation*: A hole is called *inner* hole if it can be closed by enlarging every face, which is adjacent to the hole. Otherwise, the hole is called *outer*. Examples for holes with different attributes are visible in Figure 1. A last definition are the so called *transition-half-edges* T_i (red edges in Figure 1). These are twin-edges which start (or end)

in a vertex on the boundary of a hole, but are not part of the boundary itself. However, they still have to follow the boundary of the corresponding face. Based on this definition, we call the vertices, which are part of the hole and a transition-edge starts/ends in them, *transition-vertices* (red-dotted circles in Figure 1).

For further formalisms later on, we define the following variables: The involved faces f_i with $i = 1, \dots, k$ of a hole have the normal $\mathbf{n}_i \in \mathbb{R}^3$. It should be mentioned that one physical face may have multiple identifiers following this definition, due to the fact, that the cardinality may count one face multiple times. The transition-vertex, in which the hole-half-edges switch from face f_l to f_i is called t_i with the position $\mathbf{T}_i \in \mathbb{R}^3$. The direction of the transition-half-edge, which ends in t_i is called $\mathbf{h}_i$. Based on this definitions, the vector from a transition-vertex t_i to the next one t_j is called $\mathbf{v}_{ij} = \mathbf{T}_j - \mathbf{T}_i$. For the special case of $k = 3$, we call the intersection point of the planes defining the three faces as $\mathbf{S} \in \mathbb{R}^3$, and the vector from one transition-vertex t_i to the intersection point $\mathbf{S}$ as $\mathbf{s}_i = \mathbf{S} - \mathbf{T}_i$. These identifiers are visualized in Figure 2.

The state of the art indicates two kinds of user support: the automatic closing of holes and the user guidance. However, only because a hole can be closed automatically, it is not always in the user's interest to do so. Therefore, we distinguish between three kinds of holes a) that can be closed automatically and the user agrees, b) that can be closed automatically and the user disagrees, and c) that can not be closed automatically. In our previous work [3] and [4] a user-defined parameter was introduced, the so called *structure size* δ_s . The structure size indicates, which geometric features (considering their size) can be ignored. Therefore, all holes smaller than the structure size can be closed automatically, as they are too small to be relevant. Otherwise, user guidance should be provided. The definition of the size of a hole has to take different shapes of holes into account. As the size of a hole the maximum diameter ($k = 1$), the maximum distance to the missing edge ($k = 2$) or the distance to the missing corner ($k = 3$) can be used. For holes with a cardinality $k > 3$ or the orientation outer, user guidance is the only possibility, as the hole can not be closed automatically.

3.2 Detection of holes

The following algorithm detects all holes of a B-Rep, starting with a half-edge without a twin-edge. This *start* edge is added to the set *hole*. Beginning with *start*, all following half-edges (considering the direction in a B-Rep) are visited until we reach the start again. If we encounter an edge with an opposite, we do not add this to the set (as this half-edge is a transition-half-edge) and return to the hole in the next step. In the end, the set *hole* is added to the overall set *holes*. This whole process is repeated until no half-edge without twin-edge exists, which has not been visited yet. The cardinality is determined by counting the participating faces. The procedure is summarized in Algorithm 1. The function $\text{twin}(c)$ denotes the half-edge with the twin relationship to half-edge c . Analogously, the function $\text{successor}(c)$ denotes the half-edge after c for this hole.

Based on the now known holes and their cardinality, we calculate the orientation of holes, starting with $k = 2$. The idea is, to check whether the transition-

Algorithm 1 Determining all holes of a B-Rep

```

1:  $holes = \{\}$ 
2: while  $\exists$  An unvisited half-edge without twin do
3:    $start = \text{random, unvisited, half-edge without twin}$ 
4:   mark  $start$  as visited
5:    $hole = \{start\}$ 
6:    $c = \text{successor}(start)$ 
7:   while  $c \neq start$  do
8:     if  $c$  has a twin then
9:        $c = \text{twin}(c)$   $\triangleright$  return to the hole
10:    else
11:       $hole = hole \cup c$   $\triangleright$  add  $c$  to the hole
12:      mark  $c$  as visited
13:       $c = \text{successor}(c)$ 
14:    $holes = holes \cup hole$   $\triangleright$  add current hole to the set

```

half-edge points in the same direction as the vector from one transition-vertex to the next one. However, the transition-half-edges are not necessarily twins. Therefore, we take the normals of the corresponding faces into account:

$$\text{outer hole} \iff \begin{cases} (\mathbf{n}_1 \times \mathbf{n}_2) \circ \mathbf{v}_{12} < 0 & \text{if } f_1, f_2 \text{ convex} \\ (\mathbf{n}_1 \times \mathbf{n}_2) \circ \mathbf{v}_{12} > 0 & \text{if } f_1, f_2 \text{ reflexive} \end{cases} \quad (1)$$

The definition can be easily extended for the case of $k = 3$:

$$\text{outer hole} \iff \begin{cases} (\mathbf{n}_i \times \mathbf{n}_j) \circ \mathbf{s}_i < 0 & \text{if } f_i, f_j \text{ convex} \\ (\mathbf{n}_i \times \mathbf{n}_j) \circ \mathbf{s}_i > 0 & \text{if } f_i, f_j \text{ reflexive} \end{cases} \quad (2)$$

$$\forall i \in \{1, 2, 3\}, \quad j = (i - 1 + k) \bmod k.$$

This calculation fails, if participating faces are linearly dependent, because no intersection point exists. Due to the definition of an inner hole, that faces meet in one point, it is clear that a hole with linear depending faces is an outer hole. To calculate the orientation we use whether two faces are convex or reflexive. This relationship is only defined for edges (as faces can be both on different edges) and is calculated by $\text{edge convex} \iff (\mathbf{v}_{ij} \times \mathbf{n}_i) \circ \mathbf{n}_j > 0$. In our case, we want to know the convexity of two faces in one transition-vertex. We define a plane $\pi = (\mathbf{n}_i \times \mathbf{n}_j - (\mathbf{n}_i \times \mathbf{n}_j) \circ \mathbf{T}_i)^T$ which is orthogonal to both faces and contains the transition-vertex t_i . We project both faces onto this plane and get two lines g_i, g_j . For these lines there are four possible cases, how the transition-half-edges can lie in the plane π . For two of these the decision of the convexity is obvious. The other two are undefined, as both convex or reflexive is possible. However, an estimation is possible by projecting the hole-half-edges onto π and comparing the overlap a_i resp. a_j onto the other side (see Figure 3). Depending on which value is greater, the two faces are convex or reflexive in this transition-vertex.

Fig. 3: The four possible cases how transition half-edges h_i and h_j can lie in plane π . The convexity in transition-vertex t_i (red-dotted) is estimated.

Fig. 4: Closing of inner holes (top) by replacing hole-half-edges (middle) and merging of separated edges (bottom).

3.3 Automatic closing of holes and Generation of User Guidance

Based on the attributes, we close holes automatically which are smaller than the structure size δ_s . For $k = 1$ the inner boundary is removed, in which all half-edges of the hole lie. For $k = 2$, all half-edges of the hole are replaced by one twin-half-edge. If there are already half-edges on this physical edge, they are merged afterwards. For $k = 3$ a new vertex v_s is generated, which lies in the intersection of all involved faces. All hole-half-edges are deleted and replaced by pairs of half-edges, with v_s as one vertex, and each transition-vertex as the other vertex. If the edges are separated, they are merged into one (see Figure 4).

Based on the automatic closing, three combinations of cardinality and orientation exist, where an user guidance during the recording is necessary: Inner holes with $k \leq 3$, outer holes with $k \leq 3$, and holes with $k > 3$. This separation is based on the idea, that holes with $k \leq 3$ can be closed by taking a look from one proper pose. The user guidance aims on the calculation of an arrow consisting of a position $\mathbf{P} \in \mathbb{R}^3$ and a direction $\mathbf{r} \in \mathbb{R}^3$ with $\|\mathbf{r}\| = 1$.

The idea for the method for inner holes with $k \leq 3$ is based on the functionality of depth sensors. The easiest way to capture planar surfaces occurs when the angle between the face's normal and the sight ray is as small as possible. Therefore, the direction of the arrow can be calculated as the sum of the inverted normals of all involved faces. The arrow directs towards a point which is calculated either as the centroid $\mathbf{P}_s$ of the hole or as the weighted sum of the transition-vertices:

$$\mathbf{r} = \sum_{i=1..k} -\mathbf{n}_i / \left\| \sum_{i=1..k} \mathbf{n}_i \right\|_2, \mathbf{P} = \begin{cases} \mathbf{P}_s - d \cdot \mathbf{r} & k = 1 \\ \frac{1}{2}(\mathbf{T}_1 + \mathbf{T}_2) - d \cdot \mathbf{r} & k = 2 \\ \frac{1}{3}(\mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_3) - d \cdot \mathbf{r} & k = 3 \end{cases} \quad (3)$$

The parameter $d \in \mathbb{R}$ corresponds to a distance from the hole in the direction of $-\mathbf{r}$ which can be adjusted by the scaling of the B-Rep and the user's preferences. For the case of outer holes with $k > 3$, the missing patches lie on the other side compared to inner holes, therefore it is reasonable to use the inverted direction. However, when a hole is the inner boundary of a surface, the direction may still not fit, because the arrow is on the wrong side of the face. This can be fixed by checking, whether the hole is an inner or an outer boundary of this face:

$$\mathbf{r} = \sum_{i=1..k} a_i \cdot \mathbf{n}_i / \left\| \sum_{i=1..k} \mathbf{n}_i \right\|_2 \quad \text{with} \quad a_i = \begin{cases} +1 & \text{inner boundary} \\ -1 & \text{outer boundary} \end{cases} \quad (4)$$

The closing of holes with $k > 3$ is ambiguous. The calculation of the orientation fails, because no intersection point is available. The idea is to determine triangles, which would close the hole and calculate a direction, from which most of them are visible. The first step is to calculate the mean $\mathbf{M}$ of the transition-vertices as $\mathbf{M} = \frac{1}{k} \sum_{i=1..k} \mathbf{T}_i$, which can be used as the position. Together with the transition-vertices T_i , this point is used to triangulate the hole and calculate the normal $\Delta_i = (\mathbf{M} - \mathbf{T}_i) \times \mathbf{v}_{ij}$ with $j = (i + 1) \bmod k$ for each triangle. These normals are not normalized, so they can be weighted by the area of each triangle:

$$\mathbf{r} = - \sum_i \Delta_i / \left\| \sum_{i=1..k} \Delta_i \right\|_2 \quad (5)$$

4 Validation

The generation and merging of B-Reps [3, 4] was implemented and extended by the developed methods. It can be used as an application with common depth sensors (Microsoft Kinect) to reconstruct objects by moving the hand-held sensor through the scene. Also, we implemented an Android application which can be used on devices with a depth sensor (Lenovo Phab 2 Pro) as a proof of concept. One possible validation technique is the complete exploration of the problem space. For this, we limit the cardinality to $k \leq 3$ and determine all possible cases considering convexity of faces, orientation, and for outer holes whether the hole is an outer or inner boundary. For each possible case, a synthetic test case is generated and the calculated user guidance evaluated. The decision, whether the user guidance is reasonable, is an individual decision. The cardinality is fixed with $k \leq 3$ due to the fact that the number of possible cases increase exponentially in k and no automatic closing is possible with higher values for k . Another method is a validation based on real data. A first step is to evaluate the automatic closing of holes. In this case, the reconstruction is done by hand without any support. This result can be compared to a reconstruction which uses the automatic closing of holes and the same input. A second method is to analyse the amount of user guidance on real world data. This experiment is based on our previous work [12], in which a robot cell is reconstructed in an offline step.

Fig. 5: Inner holes with cardinality $k = 3$ for all four possible cases considering the convexity of the three edges. Top: User guidance. Bottom: The model after automatic closing. The direction of the face normal is indicated by a black line.

This reconstruction was used for a real-time path planning. The developed user guidance and hole closing should help in generating a complete model.

Based on this setup for the validation we obtain the following results: As a proof of concept we reconstruct complete objects. The user starts in an arbitrary place and surrounds the object. After each merged segment the user guidance is calculated, and the operator may take another image. After one round the model is completed to a high degree, with some indications left (see Figure 6). Considering the complete evaluation we generated synthetic models for every possible case: If the half edges of the hole lie in more than one inner boundary, the resulting user guidance is not very helpful due to a poor direction. The same problem occurs with models in which one face is twisted around another. However, these two cases are not highly relevant as exactly this situation with no other faces changing the cardinality must arise. Additionally, holes which lie in more than one inner boundary are rare in reality. The results for inner holes with $k = 3$ and different convexity can be seen in Figure 5. The result of the comparison between real world data with and without the automatic closing is presented in Figure 7. The difference between these two reconstructions is small, but several holes are closed in a reasonable way. Due to the fact, that a edge can only be merged when it exists in both input B-Reps (due to the method from [3]), most of the holes exist on edges and corners. If the corresponding faces are well reconstructed, the hole is barely visible. The scene in Figure 6 visualizes the user guidance. The amount of arrows increases in the beginning and reaches the maximum number when the object is circled for the first time. When the user actively closes these holes, the number of arrows decreases until the object is completely reconstructed and no arrows remain.

Fig. 6: Test-scene and the reconstructed model with a Lenovo Phab 2 Pro. During the recording, the user guidance was calculated and displayed.

Fig. 7: A testscene (c) was reconstructed with a Microsoft Kinect for XBOX. Model (a) is the result without automatic closing. In model (b) sufficient small holes were closed automatically. In detail (d+e) several holes were closed.

5 Conclusion

We presented a new approach to close holes automatically and to generate user guidance during recording. We found that different types of holes can exist,

which need different methods depending on the attributes of the hole. A method to determine all holes of a B-Rep and their attributes was presented. Based on this, the holes are closed automatically or user guidance was generated. We evaluated our approach with a complete exploration of a problem space and by comparing the results of a real world scene with and without automatic closing of holes. Future work may include a collision test for the arrows to prevent them from being stuck inside the model. Additionally, a user study could verify the usefulness of the user guidance and the decrease in recording time.

Acknowledgements This work has partly been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant agreement He2696/16 HandCAD.

References

1. T. Breitbach: Effizienzsteigerung der Zellenmodellierung für Industrieroboter durch hybride Simulationsmodelle. RWTH Aachen, 2015.
2. H.-J. Schweizer: Experimentelle Untersuchungen über den Auge-Hand-Regelkreis bei der Durchführung einer Balancieraufgabe unter stroboskopischer Beleuchtung. Kybernetik, 1971.
3. M. Sand and D. Henrich: Incremental reconstruction of planar B-Rep models from multiple point clouds. The Visual Computer, 2016.
4. M. Sand and D. Henrich: Matching and Pose Estimation of Noisy, Partial and Planar B-rep Models. Proceedings of the Computer Graphics International, 2017.
5. S. Izadi et al.: KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. User Interface Software and Technology, 2011.
6. E. Dunn and J. M. Frahm: Next best view planning for active model improvement. British Machine Vision Conference, 2009.
7. S. Chen et al.: Active vision in robotic systems: A survey of recent developments. The International Journal of Robotics Research, 2011.
8. T. Ju: Fixing Geometric Errors on Polygonal Models: A Survey. Journal of Computer Science and Technology, 2009
9. X. Guo et. al.: A survey on algorithms of hole filling in 3D surface reconstruction. The Visual Computer, 2018.
10. G. Barequet and S. Kumar: Repairing CAD models. Visualization, 1997.
11. S. Bischoff and L. Kobbelt: Structure Preserving CAD Model Repair. Computer Graphics Forum, 2005.
12. T. Werner et al.: Sparse and Precise Reconstruction of Static Obstacles for Real-Time Path Planning in Human-Robot Workspaces. International Symposium on Robotics, 2018.
13. L. S. Tekumalla and E. Cohen: Reverse Engineering Point Clouds to Fit Tensor Product B-Spline Surfaces by Blending Local Fits. CoRR, 2014.
14. J. Branch et al.: Automatic Hole-Filling of Triangular Meshes Using Local Radial Basis Function. 3D Data Processing, Visualization, and Transmission, 2006.
15. G. Harary et al.: Context-based Coherent Surface Completion. Transactions on Graphics, 2014.
16. Y. Jun: A piecewise hole filling algorithm in reverse engineering. Computer-Aided Design, 2005.
17. M. Wei et al.: An Integrated Approach to Filling Holes in Meshes. Artificial Intelligence and Computational Intelligence, 2010.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.