Indoor scene reconstruction from a sparse set of 3D shots

Cedric Bobenrieth, Hyewon Seo, Arash Habibi, Frédéric Cordier

Indoor Scene reconstruction from a sparse set of 3D shots

Cédric Bobenrieth
IGG, Icube
300 Boulevard Sébastien Brant
Illkirch-Graffenstaden, France 67400
cbobenrieth@unistra.fr

Hyewon Seo
IGG, Icube
300 Boulevard Sébastien Brant
Illkirch-Graffenstaden, France 67400
seo@unistra.fr

Arash Habibi
IGG, Icube
300 Boulevard Sébastien Brant
Illkirch-Graffenstaden, France 67400
ahabibi@unistra.fr

Frédéric Cordier
LMIA
4 rue des Frères Lumière
Mulhouse, France 68093
frederic.cordier@uha.fr

ABSTRACT

The use of cost-efficient and portative devices, as the Kinect, enables the easy reconstruction of interior scenes. Thus 3D reconstruction has become a field of interest to many researchers. A large number of applications have thus been created, seeking to provide the most accurate reconstruction of a scene through a complete 3D scan. Although these applications are efficient, the scanning of the scene by the user remains a time-consuming process. Moreover this process tends to be difficult for users since the scan required by those methods must not contain any missing data. Our work, on the other hand, requires only a few shots without any overlapping requirement, which makes the scanning process very fast. Furthermore our method works also on shots obtained from virtual scenes which makes it able to create new architectures by assembling shots from different real or virtual scenes. Since we are working with sparse data, our reconstruction will be less detailed than a reconstruction based on a full scan.

CCS CONCEPTS

• **Computing methodologies → Computer graphics**;

KEYWORDS

3D Reconstruction, Plane Alignement

ACM Reference format:
Cédric Bobenrieth, Hyewon Seo, Arash Habibi, and Frédéric Cordier. 2017. Indoor Scene reconstruction from a sparse set of 3D shots. In *Proceedings of ACM Yokohama conference, Yokohama, Japan, June 2017 (CGI'17),* 5 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recognition, analysis and modeling of the world around us are long-sought goals in Computer Graphics. One of the most important problems is to obtain a digital representation of an object or a scene of the real world. The basis of the problem is to succeed in recovering a 3D point cloud of our scene and to use a surface reconstruction method.

Numerous researchers have been working on this subject, and the Kinect has increased the amount of work in this area ([3],[10],[1]). Indeed, the Kinect device has a number of features that made it quite popular among researchers : its low price, its ease of use and its ability to provide fast and real-time acquisition of large areas.

Many applications already exist with the common goal of providing the most accurate reconstruction of the scene. In order to virtualize a given scene, these methods need a complete 3D scan which is carried out by pointing the Kinect towards the contents of a scene, possibly turning around different objects. Although portable devices, like the Kinect, have made this step easier than before, it still requires the user to spend some time on it. Moreover those methods fail if there is any missing data in the scan, or if the user makes a mistake.

Furthermore these methods merely reconstruct the reality. Some designers may want to create original virtual scenes inspired by more than one real scene, or by mixing real scenes with other virtual scenes. Existing methods does not enable such combinations.

We develop a method which can work with only a few shots, either taken with 3D scanning devices, or directly from other virtual scenes. Since our shots can arise from different scenes, we can have missing part in our data and shots may not overlap each other. Aligning each point cloud together is, in this case, a difficult task. In order to successfully overcome this difficulty, we decided to use the only information present in each shot : the planes, and especially their alignment. Therefore our work will focus on indoor scenes. Indeed planes are a main component of most human buildings and manmade items.

2 RELATED WORK

A lot of work has been done in the field of 3D reconstruction of real scenes. Some methods ([8]) use the symmetry of objects to restore the missing parts, others use the repetition of the objects present in a room in order to complete occlusions with the help of a 3D database ([7] ,[6], [5]).

When it comes to Kinect, the best-known method for detailed reconstruction is KinectFusion ([3]). KinectFusion is implemented to use the GPU, allowing real-time reconstruction. The user moves with the camera around the object or scene he wants to scan. The

CGI'17, Yokohama, Japan

DOI: 10.1145/nnnnnnn.nnnnnnn

reconstruction is carried out by simultaneously tracking the camera's 6 degrees of freedom and aligning the point clouds of each frame according to the process known as *SLAM (Simultaneous Localization and Mapping*. Subsequently, many methods have been developed based on KinectFusion in order to overcome some of the problems raised by this method.[1] presents a more effective use of the KinectFusion memory and [10] proposes the Kintinuous extension which removes the spatial limit inherent to KinectFusion.

In order to align the frames, KinectFusion, and all methods derived from it, use the ICP (Iterative Closest Point) method, an algorithm that seeks to align the views by minimizing the distance between point clouds. However, this does not work when the scenes do not have distinct geometric information to rely on, for example when they are mainly composed of planar surfaces such as walls. Therefore those methods would fail on certain indoor scenes.

Other methods have been proposed to present a functional solution where ICP fails. [2] presents a technique based on the alignment of the planar surfaces with the help of RANSAC (RANdom SAmple Consensus). This method uses the wall's normal vectors and the angles between them in order to correctly align the frames. This idea of using planes and their alignment is also the idea underlying our work. However, our method requires only a few shots and does not require overlaps between them. Moreover, our method does not necessarily reconstruct reality, but can also create a scene by using shots from multiple origins.

3 THE SOLUTION

3.1 Problem formulation

Our objective is to provide the most plausible 3D scene from an input data consisting of X distinct shots of one or more indoor scenes. Each shot is a point cloud which can contain occluded parts. In order to simplify our problem, we consider each cloud as already postprocessed, i.e. without any noise.

In order to achieve this goal our method must find all the transformations (translation and rotation) to apply to each point cloud in order to correctly align them. Since we do not necessarily have overlapping areas between point clounds, methods like ICP cannot be used. Our strategy is to adopt planar alignment. Indeed, man-made building contain mostly planar surfaces (floor, walls, ceilings, table, chair, etc.). This makes our method well suited for most scenes.

It is important to remind that our method proposes **one solution among all possibilities**. Indeed, since our shots may not present overlapping areas, there may be an infinite number of ways to align them, and consequently, an infinite number of solutions. The final solution should meet two criteria : it should maximize the number of plane alignments with the smallest number of rotations.

From a certain point of view, our problem resembles that of a **Jigsaw 3D Puzzle** commonly presented in archaeology ([9], [4]). Indeed our shots can be viewed as the pieces of a puzzle whose position and orientation must be determined. Our method must therefore be able to find this information on the basis of the planes present in each room, with the possibility of retrieving missing information. Since our problem is like a puzzle, for the remainder of this paper we will consider each point cloud (each shot) as a piece of puzzle. Therefore the term *piece* will refer to a shot and its corresponding point cloud hereafter.

3.2 The Algorithm

In order to simplify our problem, we made an assumption that the upright orientation of each shot is correct, i.e. the "bottom" of the shot is at the "bottom" in the data. This assumption is not very restrictive but it implies that the user must not hold the camera upside down at the time of scan.

The input shots to our algorithm meet the following criteria :

(1) in a given pair of shots, each shot has at least one face that corresponds to none of the faces of the other shot

(2) they do not necessarily contain common parts

(3) they may contain areas hidden by occlusions

(4) they are not vertically inverted

Moreover each shot is characterized by a set of border edges (shared by only one face). Two shots can be assembled only along these border edges.

Our method computes a possible reconstruction by aligning the supplied pieces. It uses planes as principal elements for alignment. The pieces will be placed so as to maximise the number of aligned planes, and the occlusion zones will be completed by extending those planes. Planes will be detected in our point cloud thanks to the **RANSAC** method,which is an iterative method to estimate parameters of mathematical models.

In order to find the optimal solution, our method tests all possible solutions. Our algorithm breaks down into 2 main parts :

- Search all possible rotations and translations to align pieces two by two, and thus create a non-oriented alignment graph
- Use the above constructed graph to search a global solution among all the possible transformations.

3.2.1 Determine transformations. The first part of the method determines, for each pair of pieces, the transformations (translations and rotations) that would align the greatest number of planes of one piece with those of the other piece.

For each pair of pieces, and for each face f1 and f2 in each of the pieces, we determine the transformations that should be applied to one of the pieces such that f1 and f2 be aligned (i.e. lie on the same plane). A transformation that aligns two faces is valid only if the edges facing each other on each piece are border edges. **Figure 1** shows an example where the transformation is not valid since the edges facing each other, represented in green, are not border edges.

If this transformation is valid, we evaluate its quality by counting the number of other planes that it aligns. The method only keeps transformations with the best quality.

The validity of the transformations is ensured by adding *magnet points* on border edges in our shots, as shown in **figure 2**. These magnet points represent the edges where a connection between pieces is possible. Therefore, they are not placed on the side of a face where intersects other faces. The magnet points attract each other, add supplementary translation and lead to a correct positioning of each piece, as shown in the bottom row of **figure 2**.

Prerequisite (1) ensures that each piece adds information for the reconstruction. This means that transformations leading to completely overlapping pieces (**figure 3 b**) must be discarded. Thus our algorithm leads to **figure 3 c**.

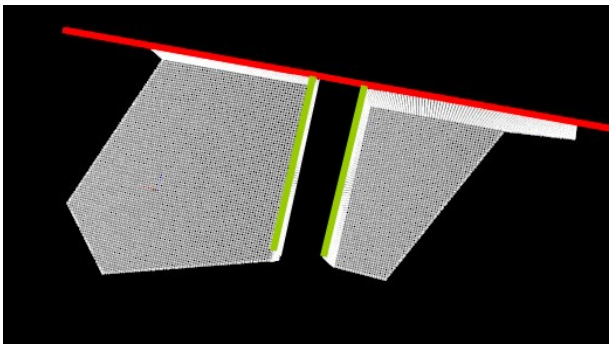

Figure 1: The two pieces are aligned with the plane represented by the red line. However, this alignment is not valid for the scene reconstruction.

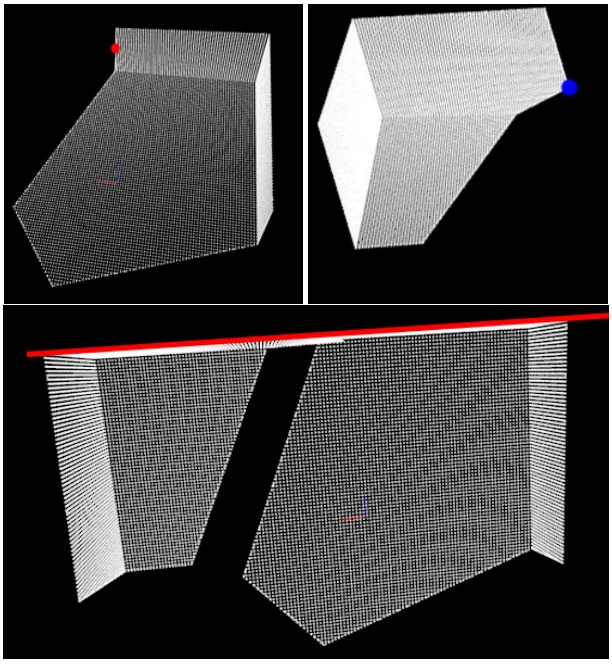

Figure 2: First row : Magnets points at planes extremities. Second row : Solution found by our algorithm with the help of magnets points.

3.2.2 Construction of the Alignment Graph. Once one or more valid transformations maximizing plane alignment are found between each pair of pieces, our method add this information in an *alignment graph* which will enable us to search a global solution as detailed in 3.2.3. The alignment graph is a non-oriented where each node represents a piece and the rotation applied to it, and each edge between two nodes represents the translation that would align the corresponding pieces.

If a transformation contains a rotation of one piece which is not yet represented in the graph, the algorithm creates a new node associated with the same piece but with a different rotation.

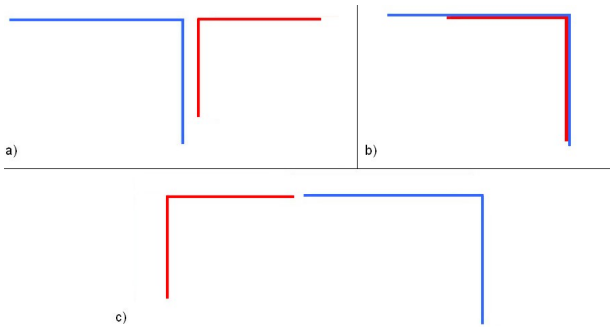

Figure 3: a) Two pieces we want to align. b) What would provide our algorithm without taking care that the alignment add information. c)The wanted solution.

In our experiments only rotations above 2° are considered in order to avoid rotations which could be derived from the inaccuracies of the calculations.

Finally, an edge is added between two nodes of the graph, representing the translation aligning the two corresponding pieces. This translation can also be zero if the two pieces are already aligned.

Therefore our algorithm can be described as in Algorithm 1.

```
Input  : Set S of pieces
Output : The Set of pieces completed by the possible rotations
         of each one, the alignment graph G=(N,E), with N =
         Nodes and E = Edges
foreach Piece P1 in S do
    foreach Piece P2 in S such as P1 != P2 do
        [Translation,Rotation] =
        SearchValidTransformation(P1,P2); if Rotation >
        Threshold then
            Piece PRotation = applyRotation(P1,Rotation);
            Add PRotation to S; Add PRotation to N;
            AddEdgeBeetween(PRotation,P2) to E;
        end
        else
            AddEdgeBeetween(P1,P2) to E;
        end
    end
end
```

Algorithm 1: Alignment Graph construction

An example of a graph resulting from our algorithm can be found **figure 4**. At first the set of pieces contained only 3 pieces P1,P2,P3. In order to keep this example as simple as possible, we decided that only one rotation was created by the method for each piece (symbolised in the graph by P1r,P2r and P3r); but, depending on the data there's no limit to the number of different rotations which can be found.

3.2.3 Find the solution with the Graph. We find the alignment solution by finding a cycle in the alignment graph.

Indeed a link between two nodes of the graph represents a translation allowing to align the corresponding pieces. Therefore the

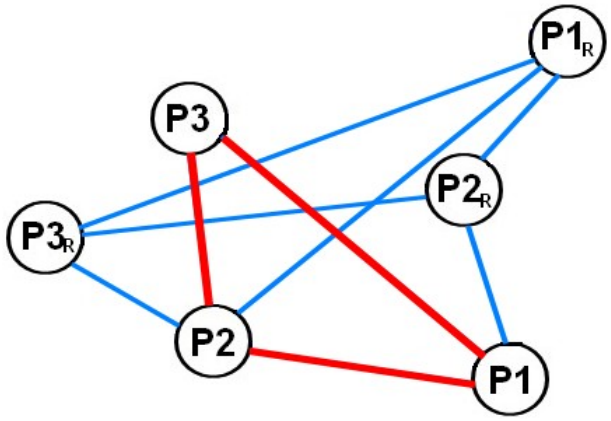

Figure 4: The alignment graph that could be obtained from our algorithm for 3 pieces. Nodes P1_R, P2_R and P3_R are obtained respectively from R1, R2 and R3 by rotating the corresponding pieces. An example of cycle in this graph is shown in red.

presence of a cycle between a set of nodes means that there is a set of translations that can align all the corresponding pieces. For example, for 3 pieces it is necessary that there exist an edge, i.e. a way to align, between pieces 1 and 2, between pieces 2 and 3 and between pieces 1 and 3. Therefore **Figure 4** shows this kind of solution in red. Moreover, it is a cycle because it is necessary that the *path* travelled on the graph passes once and only once by each piece, or one of its rotations. In this way we are certain that a piece is not used twice in our solution.

Let N be the total number of shots. For each node n of the alignment graph, we seek all the paths leaving n and returning to n and containing N nodes that represent N distinct pieces.

```
Data: The alignment graph from 1 and the set S of Pieces
Result: All the valid cycles.
Cycles allCycle[] ; foreach Piece P in S do
    Cycles c[] = SearchCycleStartingFromP() ;
    AddToallCycle(c) ;
end
foreach Cycle C in allCycle do
    if isValid(C) == false then
        removeFromAllCycle(C) ;
    end
end
```

Algorithm 2: Search for all cycles.

4 RESULTS

Our algorithm was implemented in C++ with **PCL (Point Cloud Library)**. This library provides a lot of functionalities for point clouds processing such as Filtering,Segmentation, Surface reconstruction, etc as well as the RANSAC method already implemented. Our algorithm was firstly tested on the three pieces represented on **figure 5**.

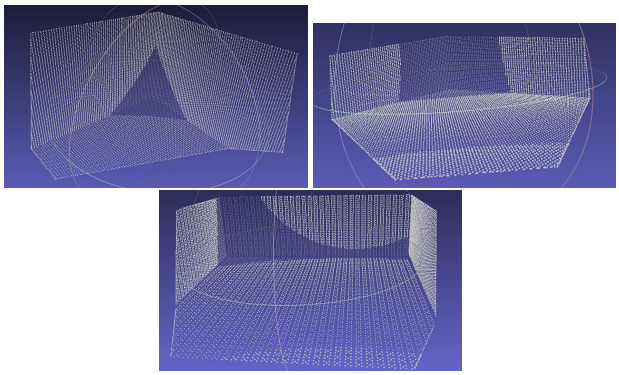

Figure 5: The three pieces of our simple test set.

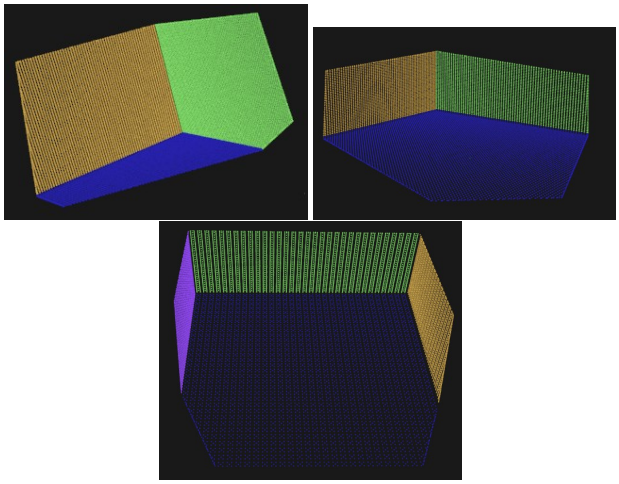

Figure 6: The planes detected by RANSAC on our test set.

These three shots make it possible to reconstitute the simplest scene possible : an empty quadrangular piece. The planes corresponding to the walls and the floor of this scene can easily be identified on these images. Each of the shots consists of a point cloud with a density between 30 and 45 thousand points.

Our program needs 24 seconds to load the data and execute the RANSAC method to find all the planes in each point cloud (shown in **figure 6**). Our algorithm can then start its work.

After nine seconds our algorithm is able to constitute the alignment graph. For this set of data our method finds six plausible solutions. One of them is shown on **figure 7**. As we can see in this figure, the result is what was expected. Only the placement of the pieces is shown here. The surface reconstruction is not yet implemented.

Some of our other results are represented on **figure 8 and 9**. The first one shows an indoor scene viewed from two different angles (bottom row), reconstructed from four shots with overlapping areas (top row). Conversely, the second one shows two different views (bottom row) reconstructed from three shots without overlapping areas (top row).

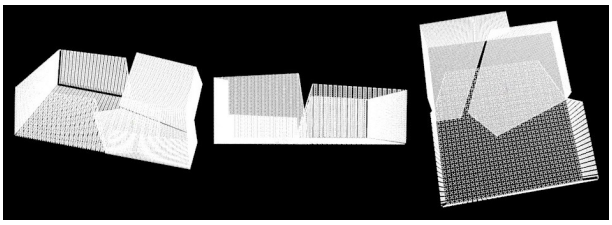

Figure 7: One of the possible solutions returned by our algorithm for the 3 shots of the figure 5, from 3 different points of view.

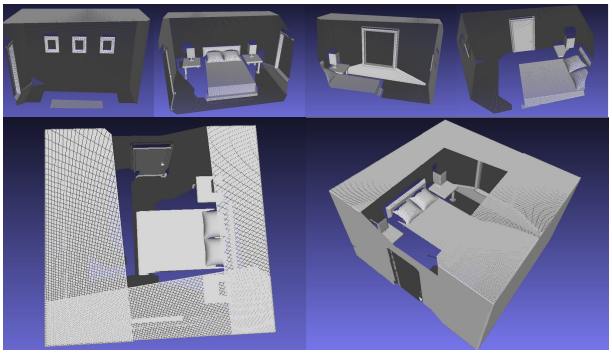

Figure 8: A result from our method. The first row presents the 4 shots given to our algorithm, the second row presents the result seen from 2 different angles.

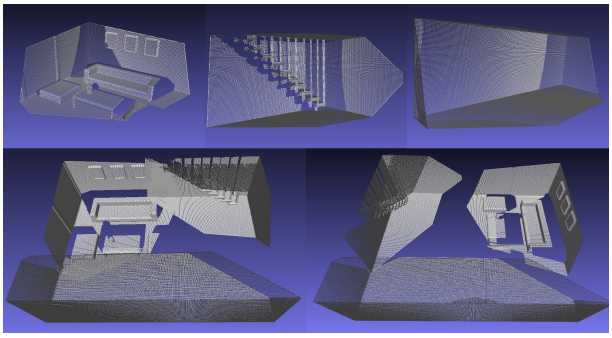

Figure 9: A result from our method. Three shots containing no overlapping area are given as input (top row). The second row shows two different results provided by our method.

5 CONCLUSION

We present a method for the reconstruction of 3D indoor scenes from only few shots of the scene. The provided shots are aligned by our method, so that a seamless scene can be generated ; even if some of those shots come from 3D virtual pieces.

However, they also show the limitations of our algorithm. Indeed,even if the execution time of our method in a simple case is only nine seconds, it increases rapidly and exponentially to the number of shots provided as data, as well as to the number of planes they contain. This is due to the fact that our method actually examines all possible solutions. Nonetheless we are convinced that the principle of our method is good, as it manages to place the pieces as they should be. Thus, we plan to optimize the execution time in the future, by using, a solution tested as invalid for not having to test other similar solutions.

RÉFÉRENCES

[1] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. 2013. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 113.

[2] Mingsong Dou, Li Guan, Jan-Michael Frahm, and Henry Fuchs. 2012. Exploring high-level plane primitives for indoor 3D reconstruction with a hand-held RGB-D camera. In *Asian Conference on Computer Vision*. Springer, 94–108.

[3] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and others. 2011. KinectFusion : real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 559–568.

[4] Martin Kampel and Robert Sablatnig. 2004. On 3d mosaicing of rotationally symmetric ceramic fragments. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, Vol. 2. IEEE, 265–268.

[5] Young Min Kim, Niloy Mitra, Dongming Yan, and Leonidas Guibas. 2012. Acquisition of 3D Indoor Environments with Variability and Repetition. *ACM Trans. Gr* 31, 6 (2012).

[6] Liangliang Nan, Ke Xie, and Andrei Sharf. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 137.

[7] Mark Pauly, Niloy J Mitra, Joachim Giesen, Markus H Gross, and Leonidas J Guibas. 2005. Example-based 3D scan completion. In *Symposium on Geometry Processing*. 23–32.

[8] Sebastian Thrun and Ben Wegbreit. 2005. Shape from symmetry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2. IEEE, 1824–1831.

[9] Gokturk Uccoluk and I Hakki Toroslu. 1999. Automatic reconstruction of broken 3-D surface objects. *Computers & Graphics* 23, 4 (1999), 573–582.

[10] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. 2012. Kintinuous : Spatially extended kinectfusion. (2012).