A ROBUST MESH-BASED SURFACE INTEGRATION ALGORITHM

SAURABH GARG

List your previous degrees here

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

2013
Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Saurabh Garg
June 2013
Acknowledgments

Write your acknowledgments here.
Abstract

Surface integration is an important step for automatic 3D reconstruction of real objects. The goal of a surface integration algorithm is to reconstruct a surface from a set of range images registered in a common coordinate system. Based on the surface representation used, existing algorithms can be divided into two categories: volume-based and mesh-based. Volume-based methods have been shown to be robust to scanner noise and small features (regions of high curvature) and can build watertight models of high quality. It is, however, difficult to choose the appropriate voxel size when the input range images have both small features and large registration errors compared to the sampling density of range images. Mesh-based methods are more efficient and need less memory compared to volume-based methods but these methods fail in the presence of small features and are not robust to scanning noise.

This paper presents a robust algorithm for mesh-based surface integration of a set of range images. The algorithm is incremental and operates on a range image and the model reconstructed so far. Our algorithm first, transform the model in the coordinate system of the range image. Then, it finds the regions of model overlapping with the range image. This is done by shooting rays from the scanner, through the vertices in the range image and intersecting them with the model. Finally, the algorithm integrates the overlapping regions by using weighted average of points in the model and the range image. The weights are computed using the scanner uncertainty and helps in reducing the effects of scanning noise. To handle small features robustly the integration of overlapping regions is done by computing the position of vertices in the range image along the scanner’s line of sight. Since for every point in a range image there is exactly one depth value, the reconstructed surface in the regions of high curvature will not have self-intersections.
Contents

List of Figures vi
List of Tables vii
List of Algorithms viii

1 Introduction 1
  1.1 First section 1

References 4
List of Figures

1.1 Boundary edges 2
## List of Tables

1.1 Statistics for the time taken to merge various models 2
List of Algorithms

1.1 Laminar shape generation algorithm for multilobed leaves 3
Chapter 1

Introduction

Automatic 3D reconstruction of real objects plays an important role in many applications such as animation, virtual reality, and gaming. A widely used approach to reconstruct an object is by using a 3D laser range scanner. Since range scanners have limited field of view, acquisition of 3D geometry of an object requires taking several scans from different viewpoints. The main difficulty with this method is to combine these scans into a unique surface representing the object. This typically involves two steps. First, all the scans must be transformed into the same coordinate system by a process known as surface registration. Second, the registered scans must be integrated into a single model by a process known as surface integration. In this paper we focus on surface integration, assuming the scans have been acquired and registered in a common coordinate system.

1.1 First section

Existing surface integration methods for range images can be classified into two categories: volume-based and mesh-based. Volume-based methods [CVGS05, CL96, Mas02, PDH+97, SWI97, SPKA03] convert range images into an intermediate volumetric representation using a signed distance function and extract the final surface using a polygonizing algorithm. These methods can handle objects of arbitrary topology and are considered to be robust with respect to scanning noise, outliers and registration errors. The choice of appropriate voxel size is important for these methods [CVGS05, CL96]. If the voxel size is too large, then features smaller than the voxel size are missed and opposite surfaces of a narrow region will be merged. If the voxel size is too small, then in the presence of scanning noise or registration errors, corresponding surfaces will be reconstructed as separate surfaces. It still remains unanswered as to what extent the signed distance function computed on the discretized space is sensitive to the presence of noisy data. It is also not clear how to choose an appropriate voxel size when the input range images have both small features and registration errors.

Mesh-based methods [Pit96, RST94, SG00, SL92, SL95, SDA00, TL94, ZLL06] directly integrate range images into a single mesh. These methods do not need an intermediate
Chapter 1. Introduction

representation. Hence, they are faster and require less memory compared to volumetric methods. In addition, they are more amenable to data reduction during processing which makes them more feasible for use with very large meshes. Existing methods are, however, sensitive to scanning errors and do not work well on objects with large surface curvature [CL96].
Algorithm 1.1: Laminar shape generation algorithm for multilobed leaves.

**Input**: Parameters of the leaf model.

**Output**: Laminar shape \( M \) as a triangle mesh.

1. \( \{\alpha_i\} \leftarrow \text{GenerateAlphaVeins}(s_0^l, s_0^r, \Delta s) \)
2. \( L \leftarrow \text{GenerateUnilobedLeaf}(\theta(B^l), \theta(B^r), \theta(A^l), \theta(A^r), W^l, W^r) \)
3. \text{foreach} \( \alpha \text{-vein} \alpha_i \text{ do} \)
4. \( L_i \leftarrow L \)
5. \( L_i \leftarrow T \cdot L_i \)
6. \text{end}
7. \text{for} \( i = 1 \text{ to } n - 1 \text{ do} \)
8. \( p^l(v_i) \leftarrow \text{IntersectLobes}(L_i, L_{i+1}) \)
9. \( d(v_i) \leftarrow \frac{d(\alpha_i) + d(\alpha_{i+1})}{2} \)
10. \text{if} \( p(v) \text{ or } \theta(v) \text{ is specified} \) \text{ then} \)
11. \( p(v_i) \leftarrow p^l(v_i) + p(v)l(\alpha_i)d(v_i) \)
12. \text{end}
13. \text{end}
14. \text{for} \( i = 1 \text{ to } n - 1 \text{ do} \)
15. \text{if} \( p(v) \text{ or } \theta(v) \text{ is not specified} \) \text{ then} \)
16. \( M \leftarrow M \cup \{p_j \mid j \in L_i \land \text{Index}(q(\alpha_i), L_i) \leq j \leq \text{Index}(p^l(v_i), L_i)\} \)
17. \( M \leftarrow M \cup \{p_j \mid j \in L_{i+1} \land \text{Index}(p^l(v_i), L_{i+1}) < j \leq \text{Index}(q(\alpha_{i+1}), L_{i+1})\} \)
18. \text{else} \)
19. \( b_1 \leftarrow \text{FitBSpline}(q(\alpha_i), \theta(q(\alpha_i)), W^l(\alpha_i), r(v_i), \theta(v_i)) \)
20. \( b_2 \leftarrow \text{FitBSpline}(r(v_i), \theta(v_i), W^r(\alpha_{i+1}), q(\alpha_{i+1}), \theta(q(\alpha_{i+1}))) \)
21. \text{Discretize} \( b_1 \) and append the points to \( M \)
22. \text{Discretize} \( b_2 \) and append the points to \( M \)
23. \text{end}
24. \text{end}
References


