Iterative Consolidation on Unorganized Point Clouds and Its Application in Design

CHAN, Kwan Chung

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Mechanical and Automation Engineering

The Chinese University of Hong Kong

August 2011

Thesis/ Assessment Committee

Professor Wang, C.L. Charlie (Thesis Supervisor)
Professor Hui, Kin-chuen (Chair)
Professor Chung, Chi-kit Ronald (Committee Member)
Professor Chen, Yong-Hua (External Examiner)

Abstract of thesis entitled

In geometric modeling, the fundamental units obtained from real world are usually points, which are often contaminated by noise, outliers and non-uniformities. In this article, we present a framework to consolidate models represented by those unorganized points through an iterative procedure of interlaced down-sampling and up-sampling steps. Based on this technique of consolidating un-orientated points, the design tool for shape modeling with points are also developed. The framework begins with a normal estimator which provides an option to compute the normal vectors. We down-sample the point and perform some steps of relaxation and repulsion followed by an up-sampling operator. After that, a selection operation is conducted to remove outliers while preserving geometric details; the uniformity of points is improved by up-sampling, and the geometric details are preserved by the selection of up-sampled points while the outliers are identified and removed by the selecting operator as well. Moreover, the selection also helps to speed up the down-sampling and up-sampling steps during the iteration. The design tool developed in this research enables the local deformation of a particular group of points, which is iteratively selected by users. Experimental results demonstrate the effectiveness of the proposed point processing framework and the functionality of the design tool developed under this framework.

Submitted by Kwan-Chung Chan
for the degree of Master of Philosophy
at The Chinese University of Hong Kong in March, 2011

Abstract of thesis entitled

(Chinese Abstract)

在幾何造型，從現實世界獲得的基本單位通常爲點數據，這些數據往往是受噪聲污染的，孤立的和非均勻的。在本文章中，我們提出了一個框架，通過迭代程序隔行降採樣和上採樣步驟，以鞏固這些散亂點數據模型的素質。我們在以鞏固散亂點數據的技術之上，同時開發了以點數據爲基礎的形狀模型設計工具。我們的框架由提供一個選項來計算法向量開始。然後向下採樣，並執行一些緩和，排斥及上採樣的操作。在這之後，選擇步驟會刪除孤立的點數據，同時保留幾何細節；點的均勻性因此提高了，並保存了模型的細節，而孤立點數據會在選擇步驟中被辨認及消除。此外，選擇步驟也有利於加速向下採樣和向上採樣的迭代過程。另外這項研究開發的形狀模型設計工具能讓用戶反覆選定一個特定群體的點數據並進行局部變形。我們會提供實驗結果證明框架和設計工具所提出的功能及其有效性。

Acknowledgements

I offer my heartfelt thanks to my supervisor Dr. Charlie C. L. Wang. His invaluable advice and guidance give me a clear view on my research direction. I admire him for both of his enthusiasm and scientific knowledge. I also thank my co-examiners for their suggestions of improvements. I had a great time at the Shape Modeling Group at the Department of Mechanical and Automation Engineering at CUHK and I hope that our cooperation will continue in the future. I am deeply appreciative of the help of my colleagues, Shengjun Liu, Yunbo Zhang, Mingdong Zhou, Mark Lam, Debbie Yuen-Shan Leung, ToM Tsz-Ho Kwok, Arthur Chau, Pu Huang, Yuwei Meng, Jun Wu, Ran Fan and Samuel Sai-Man Li. I have gained a lot of new knowledge from the discussions with these talents.

Last but not least, a very special thank to Coni Mok for proofreading my papers, reports and this thesis.

This research work is partially supported by CUHK Direct Grant CUHK/2050400, Hong Kong RGC CERG Grant CUHK/417508 and CUHK/417109, and ITF TS/026/07.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

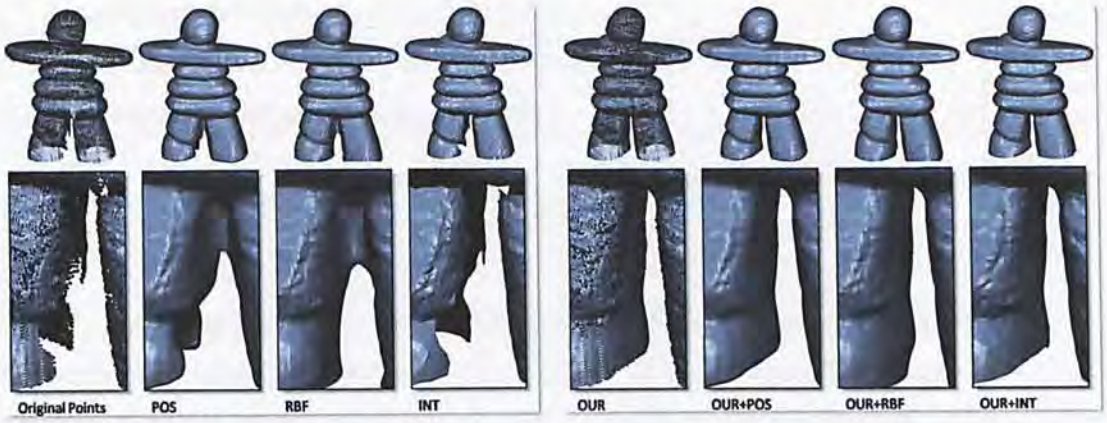

Figure 1.1: (Left group) The point cloud of an Inukshuk model obtained from a 3D scanner with incomplete sample points. The reconstructed surfaces generated by various algorithms in literature are poor at the regions with imperfect input samples. (Right group) The quality of surfaces reconstructed from the point cloud processed by our approach using various methods is all improved. The orientated normal vectors are generated by [LW10] for the approaches that need consistently orientated normal vectors.

The research of reverse engineering often involves the problem of acquiring and parameterizing structural information from a given object for later applications, such as computer graphics, virtual reality and CAD/CAM. Ascribed to the increase in accuracy and the decrease in cost, 3D scanning devices are becoming more and more popular in modeling and converting complex real world objects into digital 3D models. Most non-contact based devices like optical scanners acquire information in the form of unorientated point cloud data, and their preciseness usually bring the data to the amount of a few million points. Surface can be implicitly represented by points and therefore makes it relatively easier on shape modeling before defining the topological information [PKKG03]. However, reconstruction from preliminary data sometimes causes imperfection as these data are unavoidably contaminated with noise, outliers and incomplete

regions due to occlusions and physical limitation of the scanners. In the regions that are invisible to the cameras (e.g., deep cavities and bifurcations), the corresponding points of the resulting point cloud are absent. The under-sampled or completely missed regions on the scanned point cloud of a real-world geometry will lead to an imperfect shape on the surface reconstructed by most reconstruction algorithms. Normally, one may think of having some operations to the point source than to the mesh result. To obtain a desirable shape, we need some quality methods with a few input parameters and high automatically for the point treatment.

The abbreviations of the algorithms adopted in our tests are explained below.

(POS)	Poisson method of surface reconstruction [KBH06b]
(RBF)	Radial Basis Function (RBF) based surface reconstruction [OBS05a]
(INT)	Integrating meshing method [OBS05b]
(CON)	Consolidation of unorganized point clouds [HLZ*09]
(OUR)	Our iterative consolidation approach

Table 1.1: Abbreviations of algorithms

We propose a point cloud processing framework to improve the quality of point clouds and thus improve the quality of reconstructed meshes. Due to acquisition errors or misalignment of multiple scans, the input of our approach is an unorganized point cloud which may contain outliers, noises and non-uniformities in both thickness and spacing. Based on the point positions alone, we focus on how to make points evenly

Figure 1.2: The comparison of the Inukshuk model using **CON+POS** (Left) and **OUR+POS** (Right). The input after **CON** has severe drop in the number of points and many details are lost. In **OUR** method it selectively merges the **CON** result and the initial input. The structural connectivity is inherited from **CON** while the details are still preserved.

distributed by inserting samples into sparse regions using the **down-sampling**, **up-sampling** and **selection** mechanisms. During the iterations, noises and outliers are naturally detected and are discarded at the very beginning of the process by the two noise removal steps in different stages, which gives cleaner data to the down-stream process. For all approaches that need consistently orientated normal vectors in our framework, the orientated normal vectors are generated by [LW10] or [HLZ*09]. We also provide an option to calculate the orientation of the initial sample point by using either of these methods, which turns out to make the down-stream process more accurate with our modified consolidation algorithm (OAWLOP).

Another ingredient of our job is the point based shape modeling tool. Different from mesh based tools, the problems of self-intersection and topological changes have little influence on the data during deformation. In our framework points are inserted and/or removed using simple mechanisms in order to maintain a good speed of interaction. These mechanisms are mainly some linear interpolations with a weight formula that is controllable by the user, and their speed is usually very high compared with other surface defining tools. Some trade-off on quality includes the decrease in evenness of point distribution and normal accuracy. In our work the quality can be enhanced by running our consolidation framework once (see in Fig. 1.3).

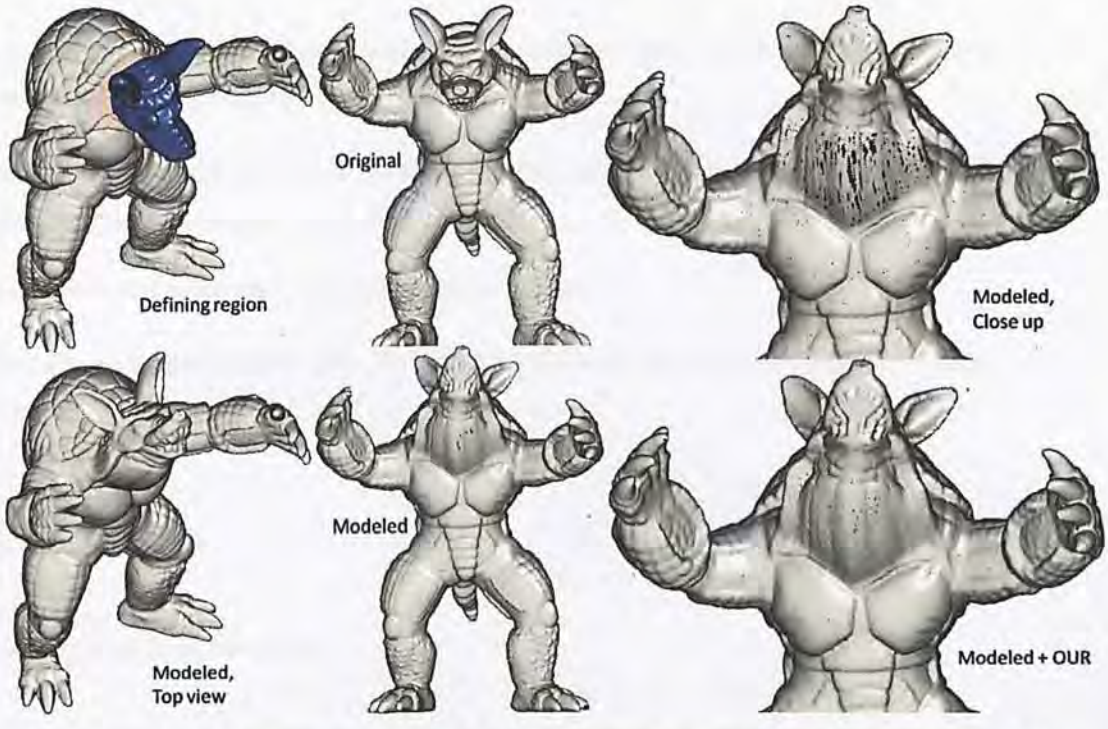

Figure 1.3: Deforming Armadillo model using our shape modeling tools: The blue points and yellow points are the handle region and the deform region respectively. The neck after deformation (Modeled, Close up) experiences some non-uniformity and is then improved by the **OUR** framework.

The resultant point cloud processed by **OUR** and the surfaces reconstructed by **POS**, **RBF** and **INT** are shown in the bottom row of Fig. 1.1. Compared with the results generated by non-iterative consolidation approach (**CON**) [HLZ*09] shown in Fig. 3.6, our iterative consolidation framework can preserve more geometric details. Under this framework, the quality of points is incrementally improved during the iteration of re-sampling and selection (see Fig. 3.1). After the process, we provide a simple, effective shape modeling tool that directly acts on point data. This tool performs real-time translation, rotation and twisting similar to those with mesh deformation tools expect that the surface is not yet defined during the deformation. Depending on which surface reconstruction method is used, the mesh topology of the point set is defined at the final stage of point processing.

1.1 Main contributions

Our aim is to process a raw point set typically contaminated with noise, outliers, uneven distributions and even holes. While the reconstructed result from the unprocessed points may experience structural defects, our processing fixes those drawbacks and results in an improved meshing by the following:

- an iterative framework for point cloud processing to improve the quality of point clouds while preserving the geometric details,
- several modifications on the down-sampling/up-sampling operators to make them better fit the framework of point processing,
- a new selection operation for this framework, and
- a fast shape modeling tool with its quality enhanced by the consolidation framework.

1.2 Overview

This report is organized as follows:

- In Chapter 2, we first review some previous work on point based processing. We examine the advantages and drawbacks of those methods and judge whether to modify and/or implement them into our algorithm. After that we search for various kinds of surface definition method. The aim is to investigate how the

consolidated point cloud affects the defined surface. Last, we investigate some papers about mesh based and point based deformations as a reference for our shape modeling tool. There are also some mesh reconstruction methods that polygonize the defined surface for our result comparison.

- Chapter 3 explains to you the detailed algorithm of our Iterative Consolidation. The sequence of the topics follows our program flow: We first give an **Overview** about our framework. This is followed by the **Down-sampling And Outliers Removal** section which states how to generate clean and uniform particles. Then in section **APSS Base Repulsion** we describe the further relaxation of the particles using an APSS based repulsion scheme in detail. The **Refinement** section mentions the procedure of inserting new points to the original point set using the $\sqrt{3}$-like refinement operator[KB00], as well as the sample point cleaning step and selectively point merging. These three sections are the key ingredients involved in our iterative framework.

- In Chapter 4, we describe the key feature of our point based shape-modeling tool and its implementation in detail. The chapter first presents the principle of our shape-modeling tool. It then explains the application guideline of our tools including point selection, stretching, compressing, bending twisting and extra point insertion after each deformation operation. The chapter ends with the illustrations of some deformation results.

- In Chapter 5, we present the results generated by our program. The results are classified into two categories, In the first part of the chapter, the results generated by iterative consolidation are shown. For comparison, results generated using other methods are also listed and their mesh reconstructions are studied. In the second part, we show the results of our de-noising step and how it improves the quality of reconstruction. The performance of the program and our machine specification are also listed.

- Chapter 6 summarizes the work conducted in this research. We point out some key factors affecting our algorithm, as well as the guidance on these factors to make the results better. Some possible enhancements are also suggested as potential future work.

- In Appendix, we attach some information about the technical details of our implementation. This includes the finding of k-th nearest neighboring points and the use of the PCA which is the classic way for normal estimation of point.

Chapter 2

Related Work

The problem of reconstructing a surface from points has been investigated extensively for more than three decades and the reconstruction methods have become a standard way of geometry creation. Many techniques have been developed. However, the points acquired by scanners are typically incomplete and highly non-uniform. Point pre-processing is sometimes necessary in order to get a better reconstruction. Some related approaches concerning the point processing and reconstruction of a surface from inhomogeneous sample density or missing data are reviewed below.

2.1 Point cloud processing

Well-known definitions, such as moving least squares (MLS) [Lev03] and extremal surfaces [ABCO*01, GM97, MLT00], can be applied to smooth or down-sample a raw point cloud. Amenta and Kil [AK04] used surfels (points equipped with normals) instead of points to define a surface with elliptical ball as distance function. Alexa et al.[AA09] introduced a new Hermite PSS Scheme which uses the projection of neighboring points instead of the original points for computation. Both [AK04] and [AA09] adopt the idea of planar MLS and can achieve the convexity of defined surface. Algebraic Point Set Surfaces [GG07] use the concept of MLS by directly fitting a higher order algebraic surface [Pra87] rather than a plane. This fitting can significantly improve the stability in situations where planar MLS fails. For instance, in the case where tight data approximation is accomplished or under-sampling, APSS performs much better and exhibits a high degree of stability. The nature of algebraic spheres can elegantly handle planar areas and regions around inflection points. We use APSS as the projection term in our repulsion operator.

To down-sample a point set, Carsten et al.[CN03] proposed the computationally and memory efficient FastFPS method which allows users to control the point density with coarse-to-fine uniform or feature-sensitive simplification. There are also algorithms for point cloud smoothing [LP05] and simplification [PGK02] guided by local geometry analysis, such as curvature estimation. Song and Feng [SF08] studied the problem of point cloud simplification by searching for a subset of the original input data set according to a user-specified number of points. Kalaiah and Varshney [KV03] represented surfaces by a sampled collection of differential points and offered a novel point-based simplification technique that factors in the complexity of local geometry. The decimation process in [ABCO*03] repeatedly removes the point that contributes the smallest amount of information to the shape. Liu et al. [LWL*08] applied quasi-Newton methods to compute Centroidal Voronoi Diagram (CVD) and demonstrated a faster convergence than Lloyd's method [Llo82]. However the time cost of computation in these approaches is very expensive. Valette et al.[VCP08] proposed a local update scheme, but not Kmeans [CSAD04] or Lloyd relaxation, to compute CVD on a given mesh surface, and then remesh the given surface according to CVD. Other methods include those introduced by Lipman et al.[LCOLTE07] who developed a highly effective, parameterization-free projection operator (LOP) which 'consolidates' a given point set to a uniform particles. Later Huang et al.[HLZ*09] proposed a weighting term to LOP (WLOP) to improve the LOP in the regions where points are highly non-uniform. However, due to the absence of normal, the consolidation of thin layer collapses into one. In the following chapter, we propose an improved version of WLOP - the orientation aware WLOP (OAWLOP) in our consolidation.

Defining a surface usually requires the point cloud to reach certain density and uniformity, which is one of our interested topics and some researches are studied. [Tur92] uses particle simulation procedure to control the sampling density. Zwicker et al.[ZPvBG01] proposes a novel Elliptical Weighted Average (EWA) filter for point-based rendering using "Surface Splatting" technique (a splat is a point equipped with normal and local supporting size). "Phong Splatting" like techniques [BK04a] improve the inner blur using second order informations, but at the same time also increase the memory consumption. [GBP05] uses a $\sqrt{3}$-like refinement procedure[KB00] combined with the points/normal interpolation of PN triangles[VPBM01] which have the smoothing property of inserting points to the region of low density. This method can naturally fill large holes in the geometry and plays a role in our up-sampling step. To better deal with outliers and delicate surface structures, the idea of mean-shifting method [FH75] is adopted to point cloud processing.

2.2 Model repairing

Ohtake et al. [OBS05a] and Carr et al. [CBC*01] exploited the extrapolation properties of radial basis functions to fill regions of sparse sampling. The work of Savchenko and Kojekine [SK02] warps a given surface model towards the missing region of the given surface using control points. This is followed by a fairing step along the boundary of the hole. This method is not automatic. It requires some manual interventions, and a prior model must be given in advance. Verdera et al. [VCBS03] also used an implicit function to represent the surface. They modeled a PDE for the smooth interpolation of a given hole based on the normal vector field around it. In [CDD*04] a surface is repaired by an optimization process. It minimizes the integral of the squared mean curvature to yield a smooth surface. Weyrich et al. [WPH*04] extended the volumetric diffusion method proposed by Davis et al. [DMGL02] to point-sampled models by replacing the distance estimation with a moving least square projection step. These methods are successful in repairing small deficiencies in the data, but have difficulties with complex holes or when large parts of the object are missing (e.g., the human model in Fig.11).

The method of Kolluri et al. [KSO04] requires filtering of the Voronoi diagram to obtain a correct pole graph. To compute a watertight surface, they used global normalized cuts that smoothly complete large missing parts. The surface synthesis methods [SACO04], [PMG*05] complete missing parts in the surface by integrating patches which are taken from a well annotated shape database or a given example set. If no appropriate examples exist, the result might be poor and the process might fail. The method of Hornung and Kobbelt [HK06] requires the definition of a watertight voxel crust in which the unknown surface is supposed to lie. To complete the crust, the authors used flood-fill and dilation operators. Sharf et al. [SLS*06] evolved an explicit mesh in a scalar field guided by the local feature size in a coarse to fine manner to avoid local minima and capture details. The method also requires a volumetric grid to evaluate the distance transformation, and the topological change has to be tracked. The computational implementation can be quite intricate (especially the topology variation on the two-manifold mesh surfaces). In [SLS*07], Sharf et al. interactively reconstructed the surface using only the positions of raw scanned data, where the user defines the general in/out orientation and assists the interpretation of data in automatically detected topologically unstable regions. Different from the mesh surface reconstruction algorithms, we focus on how to improve the quality of a given unorganized point cloud by adding sample points to change the uniformity of scattered points and removing outliers. After iteratively consolidating the given unorganized point cloud, a high-quality surface can be reconstructed from those points by various methods as shown in the right group of fig. 1.1 with **OUR** method.

2.3 Deformation and reconstruction

Many researches in free-form shape deformations have been studied extensively over the past decades: [WH94] provides a simple constraint that locks a set of particles onto a surface while the particles are able to flow on it. Later, [HBJF05] develops new techniques which allow constrained particles systems to sample and control more complex models. Other interactive modeling tools like [ST92] use orientated particles to model surface by short-range repulsion and long-range attraction to keep the particles from clumping or flying apart, therefore allowing users to move the particles on the surface. However, these interactive methods are not efficient enough to handle a massive number of points. Apart from point based deformation [Bar84, SP86, CR94, KE], some researches on mesh deformation[BK04b, KCVS98] are also studied. Our free-form modeling tool is quite similar to that proposed in [PKKG03], which allows arbitrary subsets of the distance function to define a smooth deformation field.

To visualize our effect of consolidation, we study on how to reconstruct a given point set into a mesh. The Marching Cube(MC) algorithm [LC87] and it's derivatives [JLSW02, KBSS01] generate one or more polygons for each cube in the grid that intersects the contours, where the in/out state and the required information is detected by various surface definitions like [OBS05a, KBH06b, AK04] etc.. The INT[OBS05b] method approximates a surface with triangle mesh by generating spherical cover and auxiliary points. Tight Cocone[DG03] reconstructs water-tight surfaces by computing an initial surface using Voronoi based approach followed by a subsequent marking and peeling step for filling all holes to complete the mesh reconstruction. Some re-meshing algorithms are also studied. Turk[Tur92] made the particles repel each other to sample polygonal meshes. Hoppe et al. [HDD*93] minimized an energy function that explicitly models the competing desires of conciseness of representation and fidelity to the data and can be effectively used in surface reconstruction from unorganized points.

Chapter 3

Iterative Consolidation on Un-orientated Point Clouds

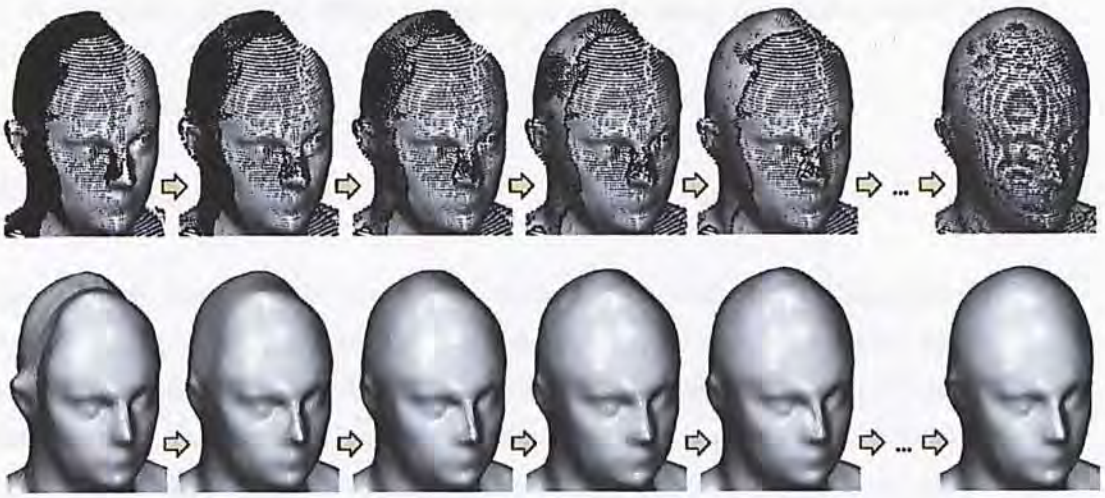

Figure 3.1: Progressive result of consolidating a head model: The intermediate results of both the point data (upper row) and their reconstruction (bottom row) are shown. The figure shows that our framework can progressively improve the quality of point set and thus its mesh.

Given an unorganized set $P = \mathbf{p}_j \subset \mathbb{R}^3$ with the presence of noise, outliers and non-uniformities, surface reconstruction from such data are likely to cause significant misinterpretation of the topology and leads to an erroneous surface. Our method presented in this paper aims to recover the structural information of P without losing the geometric details by iteratively inserting points in sparse regions to make the points evenly distributed and removing outliers that are far away from the up-sampled surface. Such a 'massage' procedure of point clouds is called consolidation [ABCO*03]. Our consolidation method consists of three steps: down-sampling, up-sampling and selection, which are iteratively applied to the input point set P.

Our point processing method is inspired by an image completion approach based on multi-resolution techniques [FL09]. Their method is based on the observation that lower resolution representation of an image contains stronger structural information while higher resolution representation contains more details. Therefore, the structural information is recovered at the lower resolution. The structural and nonstructural information on the 3D models represented by a set of sample points is analogous. And by combining both information after the recovery, the resultant models' holes are filled with their details preserved.

Another motivation is that conventional 3D scanners usually have difficulties in obtaining the data at regions where their normals are almost tangential to the scanner viewing direction. The resultant regions are highly sparse or even with holes because of the missing data. Therefore, we introduce a repulsion term which repels the point along that direction. The repulsion operator moves the points in the direction perpendicular to the 'surface' normal that is represented by the nearby points. Apart from tangential direction, we propose a spherical fitting to move the points along the circumference of the resultant sphere.

3.1 Algorithm overview

In our approach, we first estimate the normal vectors with consistent orientations on the given points in P. Then, the points in P are down-sampled into m particles, which perform relaxations and are redistributed on the surface defined by the samples in P. The redistribution of the particles is performed by iteratively applying an orientation-aware Weighted Locally Optimal Projection (OAWLOP) operator that is a variant of the WLOP in [HLZ*09]. After the iterations, a particle removal operator using the mean shifting principle is applied to detect and remove particle outliers. Then a new repulsion operator based on the Algebraic Point Set Surface (APSS) [GG07] is performed. While the normal is calculated by principle component analysis (PCA), the orientation of the particle can be obtained from the mean normal direction of its k closest point in P, followed by orientation-aware principle component analysis (OAPCA). After that, the redistributed particles are refined into a smooth point set surface by a $\sqrt{3}$-like interpolatory refinement scheme [GBP05] - this is an up-sampling step. The newly generated sample points are selectively merged into the given point set P, while the points in P are considered as outliers and removed if they are far away from the up-sampled points (i.e., a smooth surface interpolating the redistributed particles). Again, the orientation of the up-sampled points is inherited from the refinement scheme

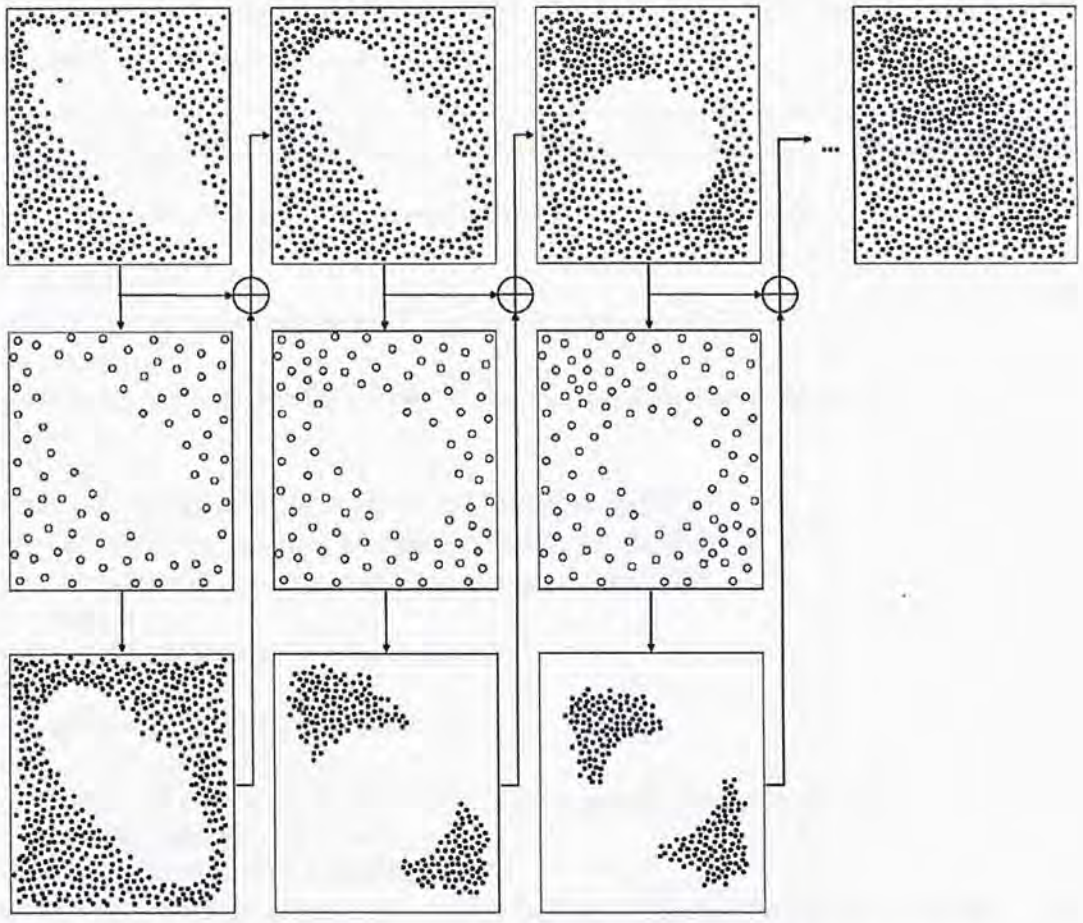

Figure 3.2: An illustration of our point processing framework, where the given unorganized points (in white small dots) are first down-sampled into particles (in yellow) and redistributed, and then up-sampled into a dense point set (in small green dots). Among the points generated by the up-sampling, the ones falling into the regions that are lack of samples in the given point set are selected (shown in red small dots) and retained. In the next iteration, the points retained from up-sampling are down-sampled into new particles (the yellow ones in the second and the third columns), redistributed, and up-sampled into new points (the green dots in the pictures of the last row). The iteration repeats until only a few up-sampled points are added.

in which the points will be selectively added to the original point set P. The down-sampling, up-sampling and selection steps are repeatedly applied to the point set. The iteration stops when very few new points are inserted into the point set P or some conditions are met. Nevertheless, the repeated application of down-sampling and up-sampling to the whole set of point samples in the framework proposed above wastes a lot of time in the regions that have been processed in the previous iteration steps. To reduce the redundant computations, an adaptive framework is investigated and used here. As illustrated in Fig. 3.2, we only down-sample the newly added points into 'alive' particles while retaining the particles used in the previous iterations as 'static' particles, and only the 'alive' particles are allowed to move in relaxation, repulsion and being up-sampled to new points (the inserted points during up-sampling are also regarded as

alive points). Specifically, the steps of our point processing algorithm applied to a given unorganized point set P are as follows:

Algorithm 1: Iterative-Consolidation

$P^0 \Leftarrow P$ and $i \Leftarrow 0$;
Initialize a particle set X by down-sampling all points of P into m particles;
repeat
  $X \Leftarrow X \cup X^i$;
  Repeatedly move the particles in X^i by the WLOP operator;
  if $i = 0$ **then**
    Remove the outliers particles from X^0 by a mean shift based selection operation;
  end if
  Estimate the orientation of particles by [LW10] or [HLZ*09];
  Apply the repulsion operator based on APSS to all particles in X^i;
  Refine the points in X^i into a set of up-sampled points Υ^i;
  if $i = 0$ **then**
    Remove the outliers in P according to Υ^0;
  end if
  Select the points of Υ^i into a subset P^{i+1};
  $P \Leftarrow P \cup P^{i+1}$ and $i \Leftarrow i + 1$;
  Down-sample all points of P^i into X^i with m^i particles ($m^i = 2m|P^i|/|P^0|$ with $|\ldots|$ being the number of points);
until the terminal condition is reached
Estimate the consistently orientated normals on the sample points in P by [LW10] or [HLZ*09];
return P;

In this adaptive framework, the points/particles that are processed in the previous iterations will not be further processed so that a lot of computational redundancies are removed. The speedup compared with the primary implementation introduced in the above paragraph is about 3-5 times. We employ a hybrid terminal condition for the iteration: 1) $\frac{P^{k-1}-P^k}{P^{k-1}} < 40\ \%$ or 2) more than a number of iterations that is specified by the user.

3.2 Down-sampling and outliers removal

3.2.1 Normal estimation

Our following operations require a point cloud to be equipped with consistently orientated normal vectors, which are usually missing in raw 3D scanning. In such a case we need a robust normal estimator which can handle noisy and unevenly-distributed data. We implement the method of **ORT** [LW10] for normal estimation. **ORT** is robust

in handling point data with a certain level of noise, non-uniformities and thin-sharp features where conventional orientating schemes using minimal spanning tree usually fail under such conditions. One minor drawback is that **ORT** is unable to detect whether the orientated normals are all 'inside' or 'outside', thus affecting the reconstruction during the iteration. To avoid it, we apply **ORT** on P at the beginning of the iteration and let users decide which orientation to flip. Our iteration consolidation involves the continuous changing of particle location X and the normal orientation of the particles needs to be recalculated each time. Repeatedly using the mentioned orientators are inefficient and time consuming. Besides, the inside/outside orientation is uncontrollable. In our framework, after applying **ORT** to P for the first time to get the orientated normal, we can simplify the normal orientation of X^k involved in the downstream operation. As the main idea of **ORT** is to generate a mesh M for a reference of orientation, which in our case the function of the mesh can be replaced by the orientated P. With the advantageof correct orientation of P as it is confirmed by users, we can first apply *Principle Component Analysis* (PCA) to each point set X^k, then we search for the k nearest neighbors (we choose 18) from P to obtain the average normal, which is also the orientation for those particular points. After that we perform an orientation-aware PCA (OAPCA) to further improve the normal direction. To be in detail, the algorithm is as follows:

Algorithm 2: Normal Estimation

```
Compute the orientated normal n_p of P using ORT;
Obtain the down-sampled and consolidated X;
for i = 1 to m do
    Compute the normal n_{x_i} of X by PCA;
    For each point x_i, search the k nearest neighbors in P;
    From the neighbors, compute the average normal n_{a_i};
    if (n_x · n_{a_i}) < 0 then
        Flip n_x;
    end if
end for
Compute the normal of X by orientation-aware PCA;
```

3.2.2 Down-sampling

For a given point set P^k, we randomly select m^k points to form a set X^k. The points in X^k are called **particles**, and m^k is selected as $m^k = \frac{2m|P^k|}{|P^0|}$ where m is a user input parameter - we usually choose $m = \frac{1}{20}|P^0|$ $\frac{1}{5}|P^0|$. The points in both P^k and X^k are all equipped with consistently orientated normal vectors. These normal vectors enable us to modify the Weighted Locally Optimal Projection(WLOP) operator in [HLZ*09] to a new orientation-aware relation operator of particles.

In WLOP, every particle $\mathbf{x}_i \in X^k$ is moved to a new position by the formula below. The update of position consists of two terms, where the first term attracts the particle to the given point set by the weighted local density

$$v_j = 1 + \sum_{\mathbf{p}_l \in (P \setminus \{\mathbf{p}_j\})} \theta(\|\mathbf{p}_j \mathbf{p}_l\|) \tag{3.1}$$

and the second term repulses the particles away from other particles by the density

$$w_p = 1 + \sum_{\mathbf{x}_q \in (X \setminus \{\mathbf{x}_p\})} \theta(\|\mathbf{x}_p \mathbf{x}_q\|) \tag{3.2}$$

The updated position of $\mathbf{x}_i$ is

$$\begin{aligned} \mathbf{x}_i = & \sum_{\mathbf{p}_j \in P} \mathbf{p}_j \frac{\theta(\|\mathbf{x}_i - \mathbf{p}_j\|)/v_j \|\mathbf{x}_i - \mathbf{p}_j\|}{\sum_{\mathbf{p}_j \in P} \theta(\|\mathbf{x}_i - \mathbf{p}_j\|)/v_j \|\mathbf{x}_i - \mathbf{p}_j\|} \\ & + \mu \sum_{\mathbf{x}_q \in (X \setminus \{\mathbf{x}_i\})} \mathbf{x}_q \mathbf{x}_i \frac{w_q \theta(\|\mathbf{x}_i - \mathbf{x}_q\|)/\|\mathbf{x}_i - \mathbf{x}_q\|}{\sum_{\mathbf{x}_q \in (X \setminus \{\mathbf{x}_i\})} w_q \theta(\|\mathbf{x}_i - \mathbf{x}_q\|)/\|\mathbf{x}_i - \mathbf{x}_q\|} \end{aligned} \tag{3.3}$$

where $\|\ldots\|$ is the L^2-norm and $\theta(r) = e^{-16r^2/h^2}$ is adopted as in [HLZ*09]. $\theta(r)$ is rapidly decreasing smooth weight function neighborhood. $\mu \in [0, 0.5)$ and h is served as two parameters selected by users to tune the performance of the operator. During our tests, the default values μ =0.45 and $h = 2L_{avg}$ work well on all models with L_{avg} being the average distance between particles and their k-nearest neighboring particles - we choose k=20.

However, the WLOP operator in Eq. (3.3) will lead to a degenerated shape according to the relocated particles on the sharp and thin features (as shown in the left of Fig. 3.3). To overcome this defect, we introduce an orientation-aware WLOP by modifying the function $\theta(r)$into

$$\theta(\|\mathbf{x}_i - \mathbf{p}_j\|) = \begin{cases} e^{-16\|\mathbf{x}_i - \mathbf{p}_j\|^2/h^2} & \mathbf{n}_{\mathbf{x}_i} \cdot \mathbf{n}_{\mathbf{p}_j} > 0 \\ 0. & \mathbf{n}_{\mathbf{x}_i} \cdot \mathbf{n}_{\mathbf{p}_j} \leq 0 \end{cases} \tag{3.4}$$

where $\mathbf{n}_{\mathbf{x}_i}$ and $\mathbf{n}_{\mathbf{p}_j}$ are the normal vectors at $\mathbf{x}_i$ and $\mathbf{p}_j$ respectively. Since the normal informations are missing in the original WLOP, we pre-calculate it to achieve our orientation-aware WLOP. By this modification, the particles redistributed by the operator can better preserve thin and sharp features.

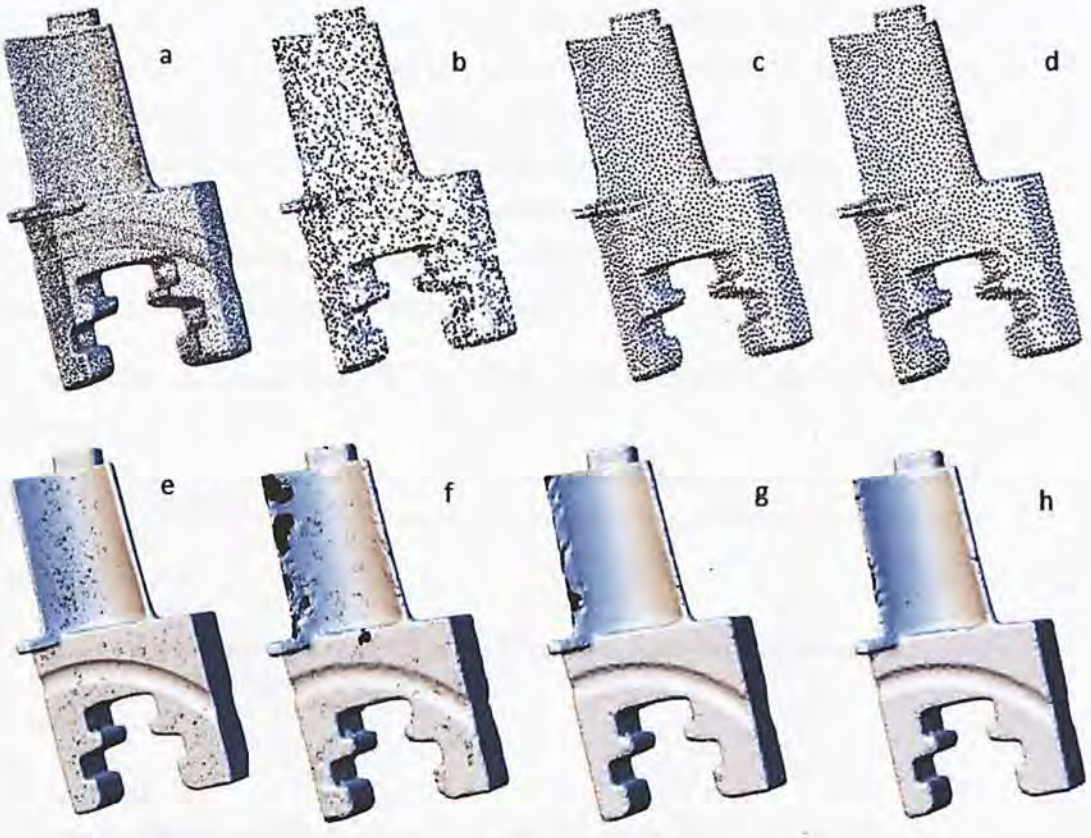

Figure 3.3: The sample points in the bottom row are the up-sampled result corresponding to the sample in the top row: **(a)** an original point set, **(b)** the down-sampled point set from **a**, **(c)** the down-sampled particles after WLOP, and **(d)** the down-sampled particles after OAWLOP.

If the input point cloud is very noisy, one may argue that the normal obtained by **ORT** is not accurate. In such a case we can still apply **ORT** and get the orientation of X from P. And instead of OAWLOP, we use WLOP during the consolidation process.

Starting from the second iteration, i.e., $k \neq 0$, only the particles in X^k are moved by having their positions updated based on Eqs. (3.3) and (3.4). All other particles in X are involved in the computation but with their positons fixed. When a particle is moved to a new position, its normal vector must also be evaluated again to preserve the correctness of the orientation-aware WLOP operator. This can be achieved by applying PCA with their orientation calculated by weighted normals from the neighboring P, followed by OAPCA, just like the previous method does.

3.2.3 Particle noise removal

WLOP has the ability of de-noising particles due to its MLS fitting in nature. This property can effectively deal with non-structural noise which is not far from the defined MLS surface. However, minor structural noise and noise far away from the structure,

remain after applying the WLOP operator. These unwanted particles will potentially affect the quality of the consolidation in the next APSS repulsion by affecting the algebraic sphere fitting and thus are necessary to be removed. It is observed that outliers are still present after WLOP; however, the number is much smaller than that of sample points and most of them are distant enough from the particle "surface" (see Fig. 5.7). We propose a mean shift method to detect and remove those unwanted outliers during the first iteration of the consolidation.

For the first iteration, we obtain the X^0 that is processed by WLOP and perform the following operation:

Algorithm 3: Particle Noise Removal

for all $\mathbf{x}_i \in X$ **do**
  Initialize $\mathbf{x}'_i \Leftarrow \mathbf{x}_i$;
  for $j = 1$ to m **do**
    Search the k nearest neighbors of $\mathbf{x}'_i$ in X and let them be the subset K;
    $\mathbf{x}'_i \Leftarrow \frac{1}{|K|}\Sigma_{\forall \mathbf{k}_i \in K}\mathbf{k}_i$; {Mean shift step};
  end for
end for
for all $\mathbf{x}_i \in X$ **do**
  Search the k nearest neighbors of $\mathbf{x}'_i$ in X and let them be the subset K;
  $\bar{d}_{\mathbf{x}'_i} \Leftarrow \Sigma_{\forall \mathbf{k}_i \in K}\frac{\|\mathbf{k}_i - \mathbf{x}'_i\|}{|K|}$; {The average distance}
  if $\|\mathbf{x}'_i - \mathbf{x}_i\| > s\bar{d}_{\mathbf{x}'_i}$, **then**
    $\mathbf{x}_i$ is considered as outliers;
  end if
end for
Remove all outliers from X;
return X;

The mean shifting method will iteratively move the query point to its nearby region having a higher point density, where the points within this region are more considerable to be the structural information of the model. We set up a threshold $s\bar{d}_{\mathbf{x}'_i}$ which is proportional to the local point density respective to this region. If a query point is likely to be an outlier, it is observed that the distance of movement is large enough to be distinguished. And any query point with its movement exceeding the threshold is regarded as an outlier and is discarded at the end of the operation. The weighting s controls the magnitude of noise removal: a smaller s results in stronger noise removal but less detail preservation, and vice versa. By performing some experimental testings, we believe that the number of iteration $m = 3$ and the distance factor $s = 3$ give a better balance between the noise removal and detail preservation.

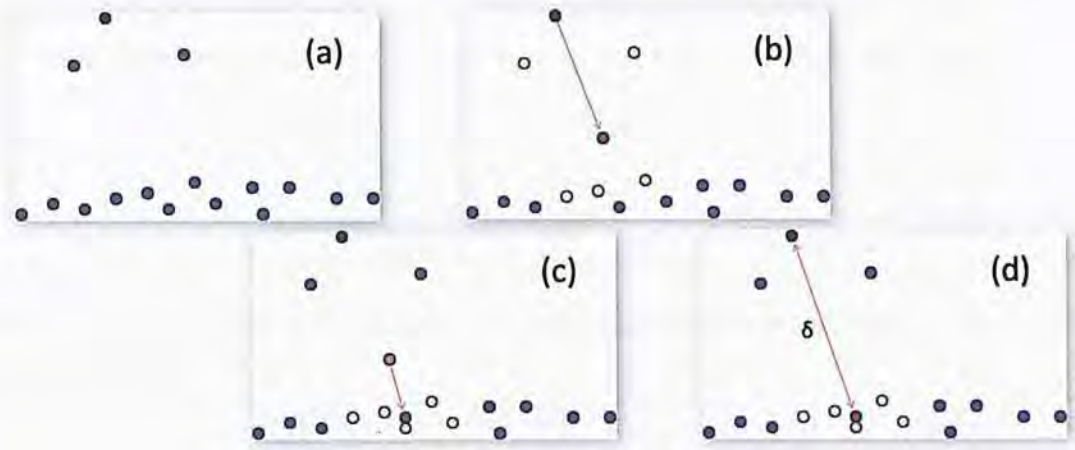

Figure 3.4: To determine whether a query point (green) is an outlier or not, we first search its kd neighbors (yellow) and calculate their average position(red). Then we iterate this process by updating the query point as the average position(pink). Finally the query point (green) is regarded as an outlier and is discarded if the distance δ between the query point and its final average position (red) exceeds certain threshold.

3.3 APSS based repulsion

The WLOP operator can evenly redistribute the particles along a surface defined by a given point set P. The mean shift method significantly improves the particles by further de-noising. However, if the point distribution of P is highly uneven, the ability of WLOP to move the particles into the highly sparse regions is still insignificant. The same observation has also been reported in [HLZ*09]. There are two reasons for this: First the down-sampled particles X are randomly chosen from P, which means that the density of P affects the initial density of X. WLOP aims to solve this effect but for some highly uneven density cases, WLOP needs a high number of iteration and thus becomes inefficient. Secondly, although WLOP operator is insensitive, it will still be affected by the density of the sample point P. This can be reflected from the first term of eq. (3.3). In this case large holes (sparse region) may take many iterations to be filled. To overcome these limitations, we introduce a repulsion operator below to move the particles into the highly sparse region efficiently.

As the particles in X have been locally evenly distributed after repeatedly applying the orientation-aware WLOP operators, an *Algebraic Point Set Surface* (APSS)[GG07] can be defined by the particles $\forall \mathbf{x}_i \in X$ and their normal vectors $\mathbf{n}_{\mathbf{x}_i}$. APSS is a kind of *Moving Least-square Surface* (MLS). Instead of plane fitting, APSS directly fits higher order algebraic spheres. The advantage is that APSS yields more stable results than planar MLS in regions with high curvature (i.e., thin and sharp features).

Using $\mathbf{u}=[u_0, ..., u_4]$ as a vector of scalar coefficient describing a general algebraic sphere in $\mathbb{R}^3$, the solution of algebraic sphere fitting at a given point $\mathbf{x} \in \mathbb{R}^3$ can be evaluated by

$$\mathbf{u} = \arg \min_{\mathbf{u}, \mathbf{u} \neq 0} \|\mathbf{W}^{\frac{1}{2}}(\mathbf{x})\mathbf{u}\|^2 \tag{3.5}$$

The solution of $\mathbf{u}$ can be found by solving the following linear equation system (details can be found in [GG07]).

$$\mathbf{W}^{\frac{1}{2}}(\mathbf{x})\mathbf{D}\mathbf{u} = \mathbf{W}^{\frac{1}{2}}(\mathbf{x})b \tag{3.6}$$

The coefficient matrices have $4n$ rows where n is the number of particles.

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} \ddots & & & & & & \\ & w_i(\mathbf{x}) & & & & & \\ & & \beta w_i(\mathbf{x}) & & & & \\ & & & \beta w_i(\mathbf{x}) & & & \\ & & & & \beta w_i(\mathbf{x}) & & \\ & & & & & \ddots & \end{bmatrix} \tag{3.7}$$

$$\mathbf{D} = \begin{bmatrix} \vdots & \vdots & \vdots \\ 1 & \mathbf{x}_i^T & \mathbf{x}_i^T\mathbf{x}_i \\ 0 & \mathbf{e}_0^T & 2\mathbf{e}_0^T\mathbf{x}_i \\ 0 & \mathbf{e}_1^T & 2\mathbf{e}_1^T\mathbf{x}_i \\ 0 & \mathbf{e}_2^T & 2\mathbf{e}_2^T\mathbf{x}_i \\ \vdots & \vdots & \vdots \end{bmatrix} \quad and b = \begin{bmatrix} \vdots \\ 0 \\ \mathbf{e}_0^T\mathbf{n}_{\mathbf{x}_i} \\ \mathbf{e}_1^T\mathbf{n}_{\mathbf{x}_i} \\ \mathbf{e}_2^T\mathbf{n}_{\mathbf{x}_i} \\ \vdots \end{bmatrix} \tag{3.8}$$

Here, the weight function $w_i(\mathbf{x}) = \phi(\frac{\|\mathbf{x}-\mathbf{x}_i\|}{h})$ describes the weight of the particle $\mathbf{x}_i$ for the local evaluation of the APSS at the position $\mathbf{x}$ with

$$\phi(r) = \begin{cases} (1-r^2)^4 & |r| < 1 \\ 0 & |r| \geq 1 \end{cases} \tag{3.9}$$

h is the supporting size of the repulsion where the same value is chosen as the orientation-aware WLOP above and $\{\mathbf{e}_k\}$ represents the unit basis vector of the coordinate system. $\beta = 10^6 h^2$ suggested in [GG07] is adopted to compensate the variance of scaling. Note that, in practical computations, only the particles whose distance to $\mathbf{x}$ is less than h are employed to determine the coefficients in $\mathbf{u}$ since the weight function $w_i(\mathbf{x})$ only

Figure 3.5: An illustration of a fish model performing one iteration of consolidation. The left model is consolidated without a repulsion step while in the right model, the repulsion step is applied two times. The comparison shows that the repulsion step can enhance the hole filling ability by moving points further to the sparse region and thus speedup our framework.

shows non-zero values on these particles. After computing the $\mathbf{u}$ vector, the center $\mathbf{c}$ and the radius of the algebraic sphere can be calculated as

$$\mathbf{c} = -\frac{1}{2u_4}[u_1, u_2, u_3]^T \text{and } r = \sqrt{\mathbf{c}^T\mathbf{c} - u_0/u_4} \tag{3.10}$$

By the APSS defined above, we can then move the particles along the APSS in repulsive manner. The movement consists of two components: the tangential component of a particle $\mathbf{x}_i$ is determined by rotating along an axis $\mathbf{r}_{\mathbf{x}_i}$ that passes through the center $\mathbf{c}_{\mathbf{x}_i}$.

- The tangential component of repulsion is derived from the second term of WLOP as

$$\mathbf{l} = \mu \sum_{\mathbf{x}_p \in (X\{\mathbf{x}_i\})} \mathbf{x}_p\mathbf{x}_i \frac{w_p\varrho}{\sum_{\mathbf{x}_p \in (X\{\mathbf{x}_i\})} w_p\varrho} \tag{3.11}$$

  with $\varrho = \theta(\|\mathbf{x}_i - \mathbf{x}_p\|)/\|\mathbf{x}_i - \mathbf{x}_p\|$. Generally speaking, it is not perpendicular to the normal vector $\mathbf{n}_\mathbf{x}$ at $\mathbf{x}_i$. We thus compute the corresponding orthogonal vector to $\mathbf{n}_{\mathbf{x}_i}$ by

$$\mathbf{l}' = \mathbf{l} - (\mathbf{l} \cdot \mathbf{n}_{\mathbf{x}_i})\mathbf{n}_{\mathbf{x}_i}. \tag{3.12}$$

- The rotation axis of the tangential component is

$$\mathbf{r}_{\mathbf{x}_i} = \frac{(\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}) \times \mathbf{l}'}{\|(\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}) \times \mathbf{l}'\|} \tag{3.13}$$

and the rotation angle is obtained by

$$\varpi = \frac{\|\mathbf{l}'\|}{2\pi\|\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}\|} \tag{3.14}$$

The tangential component of the movement is then defined by

$$\mathbf{R}_{\mathbf{r}_{\mathbf{x}_i}}(\varpi)(\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}) + \mathbf{c}_{\mathbf{x}_i} - \mathbf{x}_i \tag{3.15}$$

where $\mathbf{R}_{\mathbf{r}_{\mathbf{x}_i}}$ is the rotation matrix around the axis $\mathbf{r}_{\mathbf{x}_i}$. In order to improve the stability of the particle movement, we restrict the rotation angle by

$$\varpi = \min\{\frac{\|\mathbf{l}'\|}{2\pi\|\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}\|}, \frac{L_{avg}}{2\pi\|\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}\|}, \frac{\pi}{4}\} \tag{3.16}$$

where L_{avg} is the average distance between particles and their k-nearest neighboring particles (with k=20).

After applying the tangential component on a particle $\mathbf{x}_i$, we consecutively apply the following projective component on it for three times to retain the moved particles on the APSS defined by the particles equipped with normal vectors.

$$\mathbf{c}_{\mathbf{x}_i} + r(\mathbf{x}_i)\frac{\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}}{\|\mathbf{x}_i - \mathbf{c}_{\mathbf{x}_i}\|} - \mathbf{x}_i \tag{3.17}$$

The resultant particles obtained by our repulsion operator are distributed more evenly in the highly sparse regions.

3.4 Refinement

An up-sampling step is conducted to generate more sample points on the surface that interpolates the particles in X^i as well as the normal vectors on the particles. The up-sampled point set Υ^k is expected to regularize the scattered samples and converge on a smooth surface interpolating the particles and their normals. A good candidate up-sampling scheme that satisfies these requirements is the interpolatory refinement method presented in [GG07], which is therefore used in our framework for generating Υ^k from X^i. We integrate this method as a part of our program for application.

3.4.1 Adaptive up-sampling

To fit in with our adaptive down-sampling/up-sampling strategy, only the 'alive' particles (i.e., particles that can be moved in the repulsion) and the points up-sampled from

the 'alive' particles are used as the centers to generate refined points. Again, all the particles are involved in the calculation, i.e. the static particles can be the searching neighbor of the alive points. The refinement at a center is prevented if its distance to the closest neighbor is less than the mean minimum distance of $\overline{P_{min}}$. Here, $\overline{P_{min}}$ is calculated by:

$$\overline{P_{min}} = \frac{1}{n}\sum_{i=0}^{n} \min_{j\{j\neq i\}} (\|\mathbf{p}_i - \mathbf{p}_j\|) \tag{3.18}$$

The up-sampling is stopped when no refinement on any center is allowed. In fact, the repulsion operator and the up-sampling operator work together to generate the sample points on a smooth surface extrapolating the particles generated by orientation-aware WLOP. Such an extrapolatory helps to improve the uniformity of sample points in the highly sparse regions. Points are more likely to be added to the sparse regions than the dense regions.

3.4.2 Selection of up-sampled points

The points in Υ^k are selected to merge into the point set P^k to form a new point set P^{k+1}. This selection is very important for the preservation of geometric details on the shape represented by scanned points, as well as the surface reconstructed from the processed points. If we simply merge all the points of Υ^k into P^{k+1}, the points from Υ^k may then dominate the shape represented by P^{k+1} where the points in Υ^k represent a smooth surface interpolating the particles in X^k. In other words, the geometric details presented by the given cloudy points are blurred. Therefore, to preserve the geometric details on the given point cloud, we merge Υ^k and P^k by the criterion that:

- $\forall \mathbf{q}_j \in \Upsilon^k, \mathbf{q}_j$ should be excluded from P^{k+1} if $\exists \mathbf{p}_i \in P^k$ with $\|\mathbf{q}_j - \mathbf{p}_i\| < h/200$.

3.4.3 Sample noise removal

This is the second cleaning step performed when $k = 0$ to remove outliers from the given point cloud P. After down-sampling the points in P into uniformly distributed particles in X^0, we remove the outliers using mean shift, and then up-sample them back into points in Υ^0. The up-sampled point Υ^0 can be treated as a smooth surface that represents the structural information of P. Thus the points in P which are far away from the surface represented by Υ^0 are considered as outliers. This is based on the assumption that the 'noisy points' are sparse from each other so that after consolidation, no particles are left near those noisy regions. Therefore, we have:

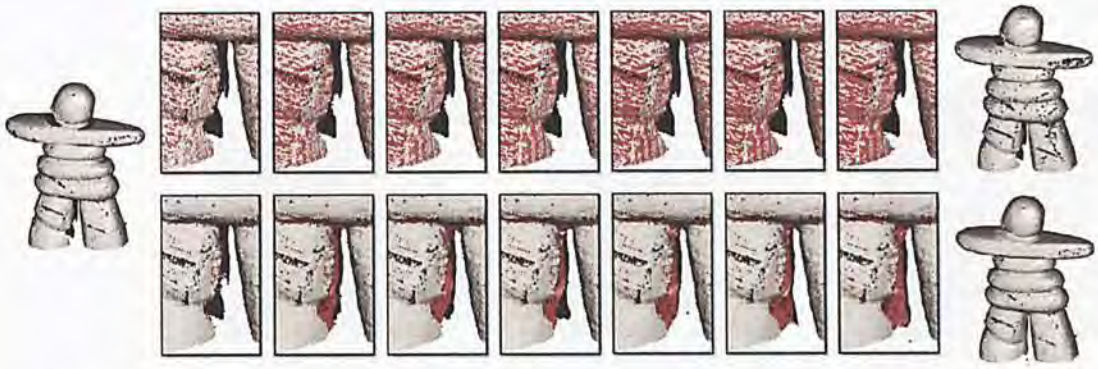

Figure 3.6: Progressive results using (top row) vs. without using (bottom row) the repulsion operator. Simply down-sampling and up-sampling the points with the newly proposed repulsion operator without a selection step cannot fill the missed regions on given point clouds (see the bottom row). On the contrary, our framework can progressively improve the quality of the points cloud (see the top row). The up-sampled and selected points are displayed in red.

- $\forall \mathbf{p}_i \in P$, it will be removed from P when $\|\mathbf{p}_i - \mathbf{q}_j\| > h(\forall \mathbf{q}_j \in \Upsilon^0)$.

This simple selection operation can effectively remove outliers embedded in the given point cloud. If the point set P is said to be noise-free, we can simply turn off this operation to preserve more details.

3.5 Set constraints to sample points

In general, it is easy to get models that have open end surfaces, which maybe inborn, caused by missing points or point preprocessing. Our algorithm further defines the surface along its tangential direction. As a result some burrs will be present in those regions, which is undesired for the future reconstruction. To solve this problem, we enable users to select a set of *anchor points* P_A at the beginning of iteration, so that the later processing simply ignores the selected region and the original point data are kept without point insertion or elimination in this region. To achieve this, the user first defines a set of P_A from the original P. After down-sampling and relaxation, we have the consolidated particles X and if for each $\mathbf{x} \in X$ its closest neighbor is an element of P_A, it is regarded as anchor point. These anchor points behave like *static points* and do not perform relaxation, repulsion and up-sampling for the rest of the iteration. The effect on setting anchor points can be seen in Fig. 3.7.

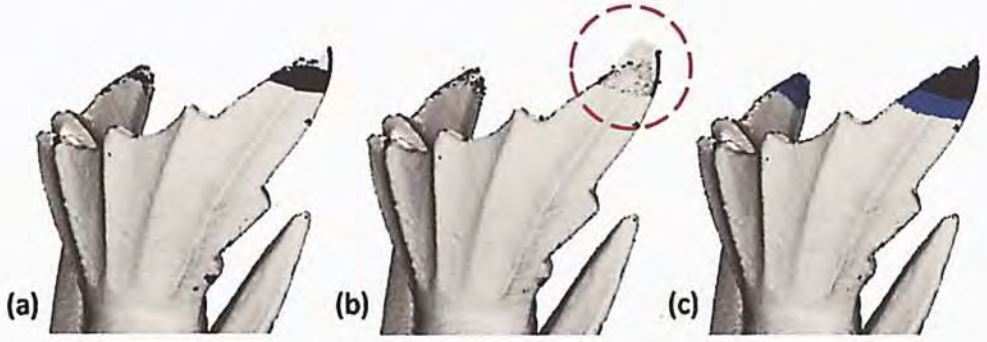

Figure 3.7: A consolidation of a fish fin with and without user constraints: **(a.)** the original model, **(b.)** the consolidate result without user constraints - note that there are some burrs caused by excess definition of the surface, **(c.)** the blue areas are the user constrained region where the original points are kept.

Chapter 4

Shape Modeling by Point Set

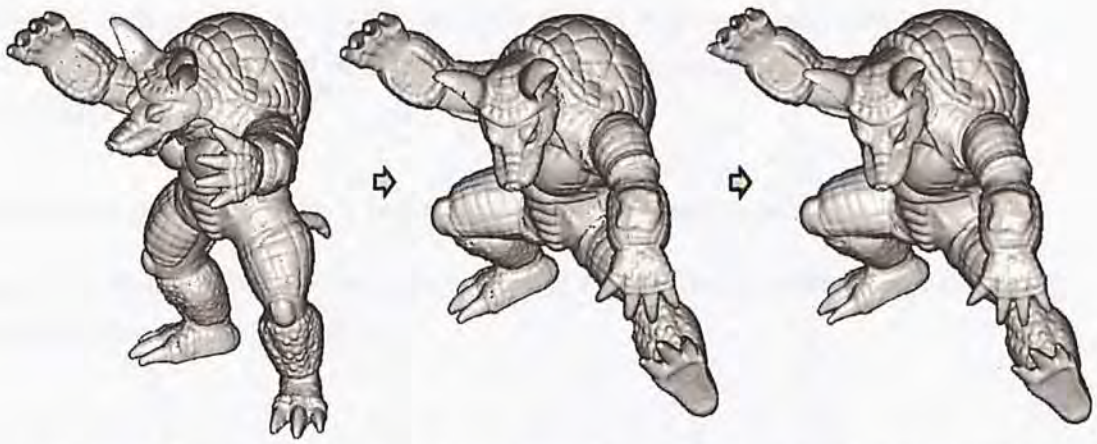

Figure 4.1: Shape modeling on Armadillo model: (Left) The original points, (Middle) points after deformation and (Right) its reconstruction. The meshes are generated by **POS** method.

After consolidation, we obtain an orientated point set which is hole filled, noise reduced and evenly distributed. In practice, it is more desirable to model the shape interactively before converting it to a mesh. Recently, there are various kinds of mesh deformation based on NURBS, on subdivision surfaces or on arbitrary triangular meshes [ZS00]. We provide a free-form shape modeling framework for point-sampled geometry, which enable users to locally deform the points by dragging, twisting and rotating. We implement the method which is similar to [PKKG03] as our deformation tool:

4.1 Principle of deformation

Given a point cloud $P = \{\mathbf{p}_i\}$, users first define the handle region $H = \{\mathbf{h}_i\} \subset P$, the deform region $D = \{\mathbf{d}_i\} \subset P \setminus \{H\}$ and the static region $S = \{\mathbf{s}_i\} \subset P \setminus \{H, D\}$. The

deformation occurs in region D, while region S remains static and region H undergoes rigid movement/rotation. The main calculations are in region D, in which we define a continuously varying scale parameter $t \in [0,1]$ that is the shortest relative distance from S to H. For $\mathbf{d}_i$ closer to H, t will be larger. More precisely, t is defined as

$$t_i(\mathbf{p}) = \begin{cases} 0 & \mathbf{p}_i \in S \\ 1 & \mathbf{p}_i \in H \\ \frac{\min_{\mathbf{s}\in S}(\|\mathbf{s}-\mathbf{p}_i\|)}{\min_{\mathbf{s}\in S}(\|\mathbf{s}-\mathbf{p}_i\|)+\min_{\mathbf{h}\in H}(\|\mathbf{h}-\mathbf{p}_i\|)} & \mathbf{p}_i \in D \end{cases} \tag{4.1}$$

We also call H the *one−region* and S the *zero−region*. Now we can simply apply t as a linear shape modeler, or together with another shape function β with the constraint $\beta(0) = 0$ and $\beta(1) = 1$. β can be a square function (i.e., $\beta(\mathbf{x}) = \mathbf{x}^2$), a root function (i.e., $\beta(\mathbf{x}) = \sqrt{\mathbf{x}}$), or any other smooth functions that satisfy the previous constraint. The position of each point $\mathbf{p} \in P$ after the deformation is $\mathbf{p}' = F(\mathbf{p}, t)$ where F is a deformation function composed of a translation and a rotation part, i.e. $F(\mathbf{p}, t) = F_T(\mathbf{p}, t) + F_R(\mathbf{p}, t)$, where

- $F_T(\mathbf{p}, t, \beta) = \mathbf{p} + t \cdot \mathbf{v} \cdot \beta$ with $\mathbf{v}$ being the translation vector, and
- $F_R(\mathbf{p}, t) = R(\mathbf{a}, t \cdot \alpha \cdot \beta) \cdot \mathbf{p}$ where $R(\mathbf{a}, \alpha)$ is the matrix that specifies a rotation around an axis $\mathbf{a}$ with an angle α.

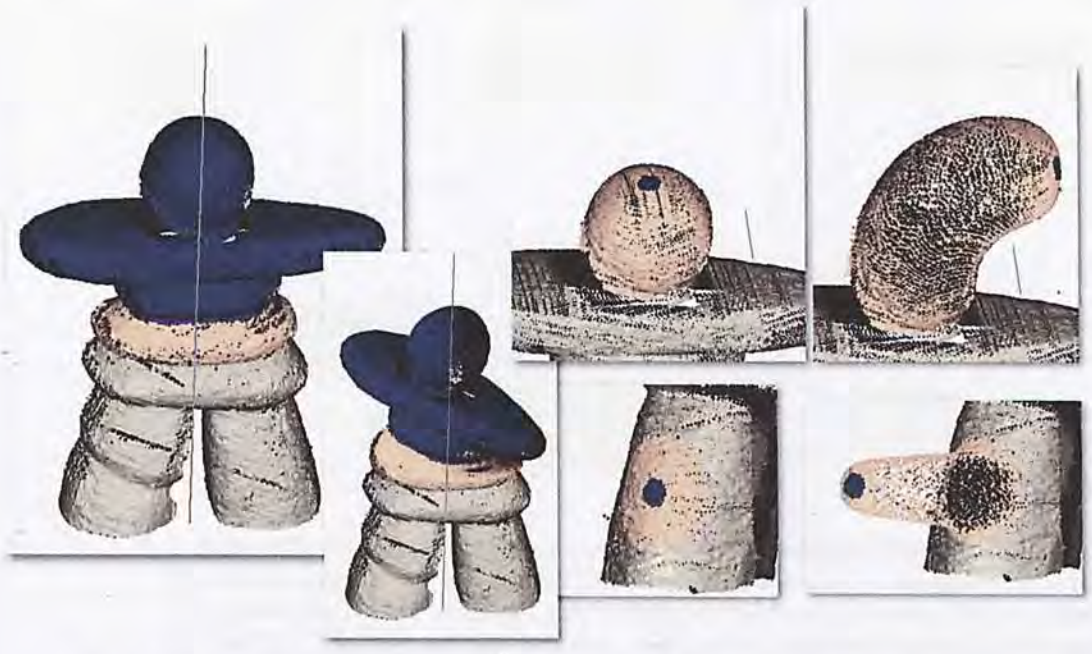

Figure 4.2: Point based shape modeling tools including (Left) twisting, (Top) rotating and (Bottom) stretching.

4.2 Selection

Our point selection tools are intuitive. Users can either select a group of points inside a boundary or the highlighted points and all points fallen within a preset radius. If users choose to select a group of points inside a boundary, they first define a polygon on the screen. All the points are then projected onto a plane which is parallel to the screen. Any points that falls into the polygon is marked as selected. To detect whether a point is in/out, we simply compare it with each line of the polygon using the following algorithm:

Algorithm 4: Normal Estimation

```
1: Initialize variable In = FALSE;
2: for each point do
3:    Project the point onto a plane that is parallel to the screen with coordinate x'y';
4:    for each line of the polygon do
5:       Get the starting point x'1y'1 and end point x'2y'2;
6:       if ((((y'1 <= y) AND (y < y'2)) OR ((y'2 <= y) AND (y < y'1))) then
7:          if (x < (x'2 - x'1) * (y - y'1)/(y'2 - y'1) + x'1)) then
8:             In = !In;
9:          end if
10:      end if
11:   end for
12: end for
13: return In;
```

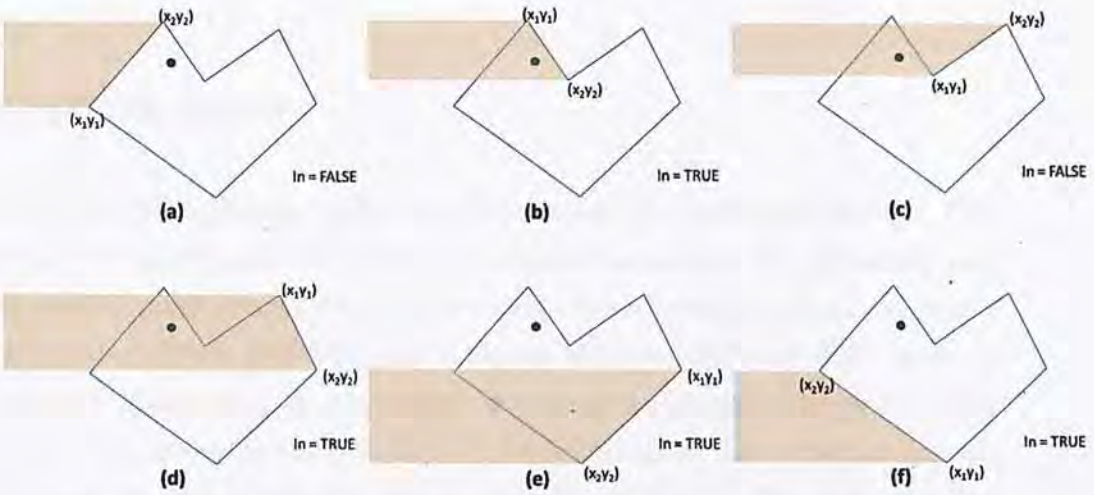

Figure 4.3: To check whether a point is inside a polygon: Starting from **(a.)**, each line of the polygon forms a region (yellow) using its **x** and **y** coordinates. If the point falls inside the region, we invert the boolean state of an indicator with its initial value of **FALSE**. This process will not stop until all the lines are checked. The final state of the indicator determines whether the point is inside(**TRUE**) or outside(**FALSE**) the polygon.

For the second selection, we first detect the upper most point in the selected region which is a few pixels around the mouse cursor. Then we highlight all the points with a Euclidean distance smaller than a user defined value.

4.3 Stretching and compressing

After defining the handle and region to be deformed, we can simply drag the handle to perform translational deformation. As we are using the mouse movement to control the magnitude of deformation and are mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ (screen coordinate to local object coordinate), we develop a better user interface for the 2-D to 3-D conversion: - We first find out the viewing vector $\mathbf{v}_v$ that points outward the screen direction. Then for each $\mathbf{h}_i$ and $\mathbf{d}_i$, their movement are constrained to be perpendicular to $\mathbf{v}_v$. .

4.4 Bending and twisting

To perform bending and twisting, we first arbitrarily define two points in 3-D space. These two points can be moved by the user. Same as the previous section, the way and the magnitude of moving the points are constrained to be perpendicular to $\mathbf{v}_v$. These two points will define the rotation axis $\mathbf{a}$ and all H and D are rotated along this axis. Again, the direction and the angle of rotation α are specified by the mouse movement.

4.5 Inserting points

During the deformation process, points are progressively changing their position. Our algorithm in sections 4.3 and 4.4 provide a real time interaction for calculating and displaying, yet the point density drops significantly when the magnitude of movement is large. The main problem is that the sparse regions will cause difficulty when applying surface definition tools. We propose a simple point mechanism which keeps the point density by inserting points as the final step of the deformation. Since the deformation is carried out in D, this mechanism will be effective in D only: We first build the connectivity of region D by searching the k-th nearest neighbors and storing them in a search table. The local density of D is then evaluated as:

$$kmd = \frac{\sum_{\|D\|} N_i/6}{\|D\|} \tag{4.2}$$

where kmd is the **k-d mean distance** of a point $\mathbf{d}_i \in D$ and N_i is the sum of L^2 norm of all the six nearest neighbors of $\mathbf{d}_i$. Once the deformation is confirmed, we search for the corresponding neighbors of each $\mathbf{d}_i$, and for each neighboring, if the distance is greater than $2kmd$, we insert points in between with the following algorithm:

Algorithm 5: Point inserting

```
1: for all d_i ∈ D do
2:    Retrieve its neighbors dn_j, j = {1,...,6} from the search table;
3:    for j =1 to 6 do
4:       dis = ||d_i − dn_j||;
5:       if dis > 2kmd then
6:          number of inserting points n = dis/(2kmd), round down;
7:          for k =1 to n do
8:             insert point at the position d_i + ((dn_j − d_i) * k)/(n+1);
9:          end for
10:      end if
11:   end for
12: end for
```

To avoid repeated point insertion of two mutual neighbors, we suggest a simple constraint when building the search table: During the searching, we label each searched neighbor with a flag and with a flag is not allowed to build its search list. The labeling is sequential, meaning that the search list of the visited $\mathbf{d}_i$ will not be affected by the flag. And after the searching we skip any $\mathbf{d}_i$ that does not have its corresponding search list at the beginning (first line) of the algorithm.

This algorithm is suitable for small scale deformation. When the deformation is large, some 'patterns' can be observed in the deformed region (see Fig. 4.6). In such a case we can apply our consolidation framework once to remove those artifacts. The following is some results obtained using our shape modeling tools:

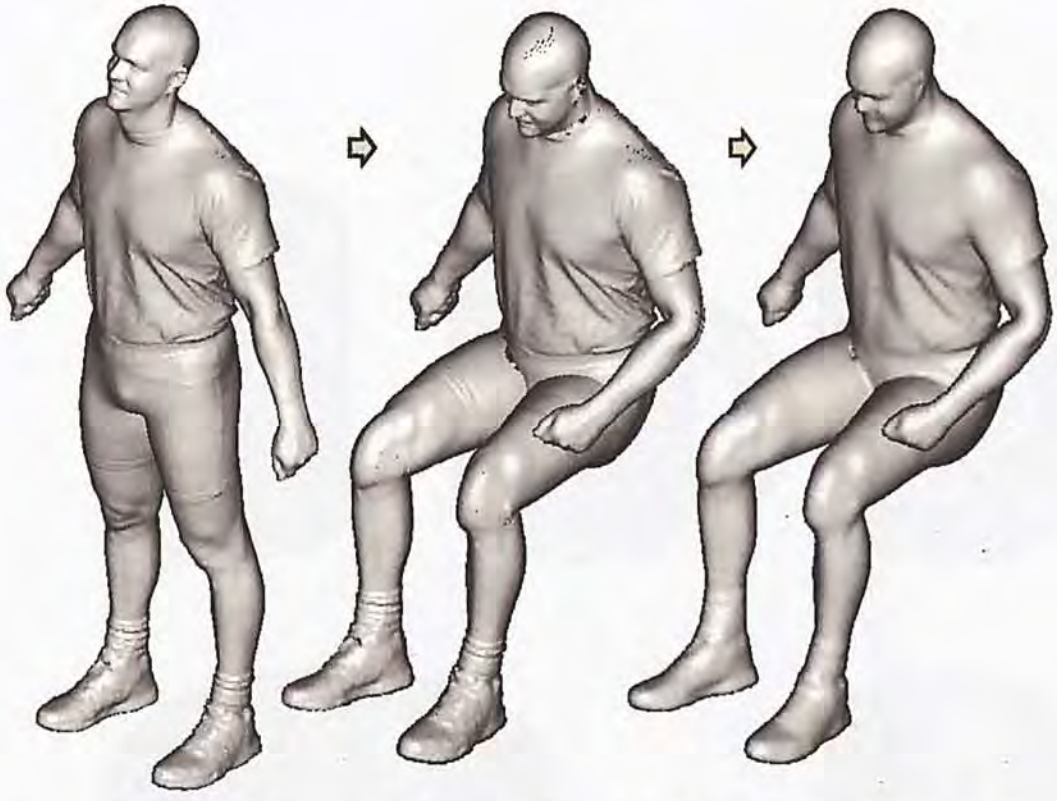

Figure 4.4: Shape modeling on a human model: (Left) The original points, (Middle) points after deformation and (Right) its reconstruction. The meshes are generated by **POS** method.

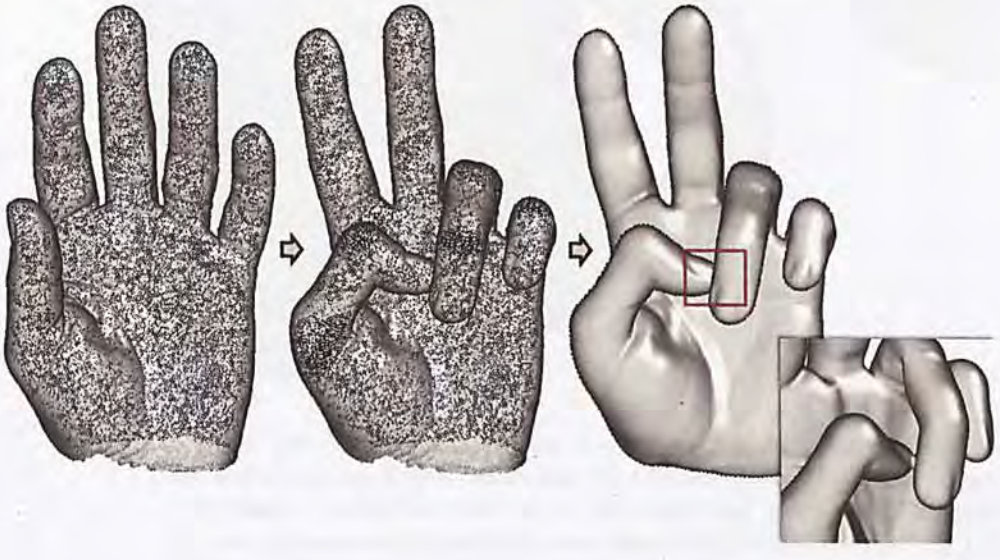

Figure 4.5: Shape modeling on a hand model: (Left) The original points, (Middle) points after deformation and (Right) its reconstruction. Note that there are topological changes but we do not necessarily have to handle it during the deformation period. The meshes are generated by **POS** method.

Figure 4.6: Shape modeling on a cylinder: The cylinder is being progressively deformed to a mug. When deforming the container area (First row, right fig.), the large deformation causes the point set to have undesired 'patterns'. The quality is improved by running our consolidation for one iteration (Second row, left fig.). The final mesh result is obtained using **POS** method.

Figure 4.7: Shape modeling on a rectangle: The rectangle is being progressively deformed to a guitar. The mesh is generated by **POS** method.

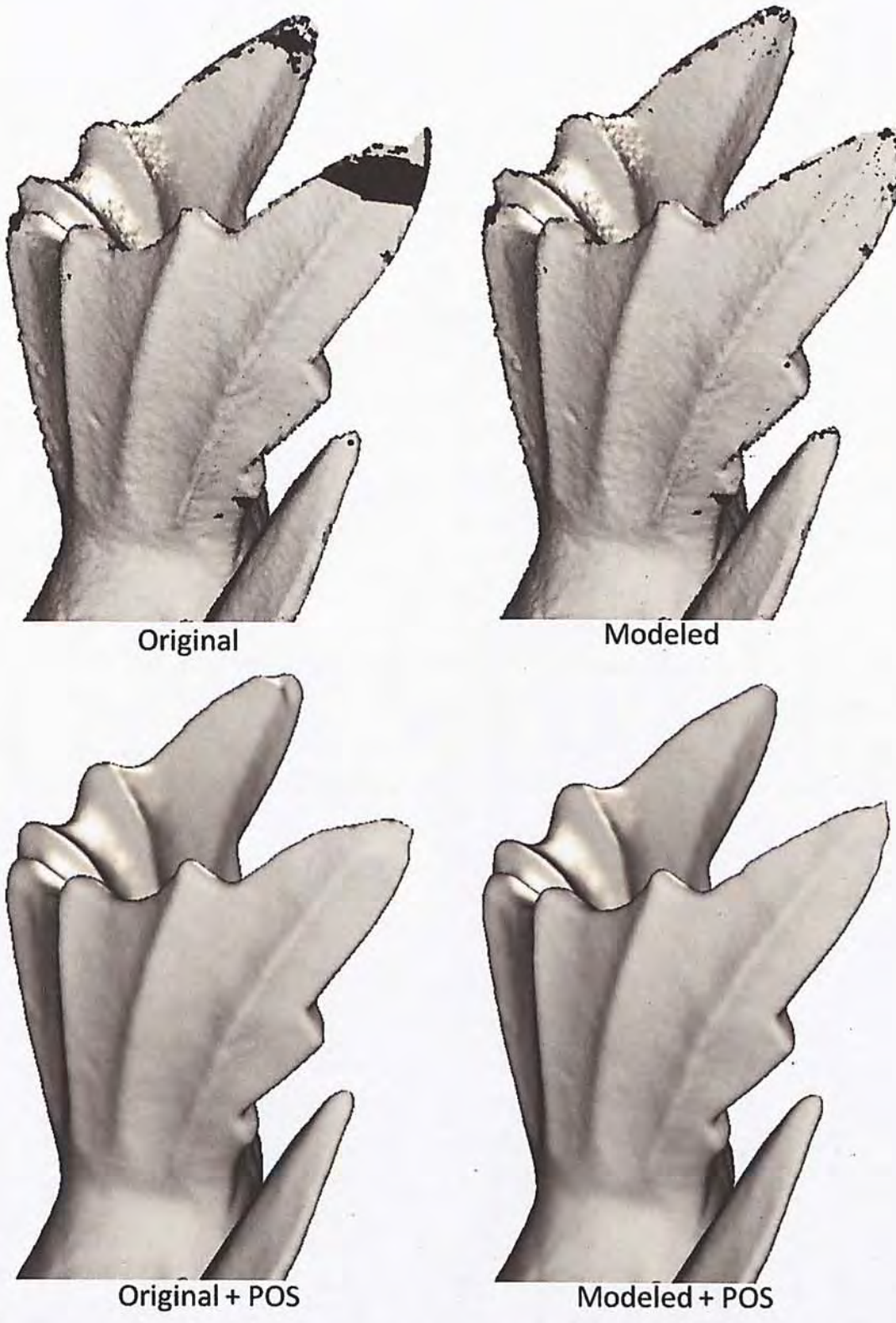

Figure 4.8: An alternative method for hole filling: In the previous chapter, we mention some situations where the iterative consolidation cannot give a desired output. Our shape modeling tool provides an alternative solution which handles the situations interactively. We improve both the point set (Top left) and its mesh (Bottom right).

Chapter 5

Results and Discussion

5.1 Program environment

We have implemented the pipeline of iterative consolidation on un-orientated noisy points into a prototype system by C++. The OpenMP library is employed to implement the WLOP/OAWLOP running part in multiple threads on a PC with multi-core CPU. All the tests and results are generated by a PC with core i5-750 CPU with 2.66 GHz frequency, 3.46GB main memory, with the hardware graphic card of model NVIDIA GeForce 9800 GT and runs in an environment of Windows XP SP3 Home Edition. The program is coded and compiled by Microsoft Visual Studio 2005.

5.2 Results of iterative consolidation on un-orientated points

To test the performance of our program, we use raw models found from the Internet that are un-orientated, contaminated with noise and have non-uniformities. These models are then passed to the program while some constraints and parameters (i.e. number of iterations, noise removal and selection of static points, etc.) are set before running the program automatically. The resultant points are reconstructed to mesh and the differences are studied below:

In Fig. 5.1, the input is a hand model with missing data between two fingers. The resultant mesh is reconstructed by **POS** method [KBH06b]. Note that due to the missing points, the fingers are connected together after the reconstruction. This is caused by incorrect surface definition of missing points. With our treatment which fills the missing area, a better surface is defined and the fingers are separated which are more desirable for future usage.

In Fig. 5.2, the input is a human model which is probably acquired with a scanner with two transceivers, one at the front and another at the back. The resultant boundaries (centre) are not connected. Direct reconstruction from the raw points has misinterpretation which causes defects on the mesh, especially in the underarm area. However, after our treatment, the point set is significantly improved and the underarm is reconstructed correctly. The same effect is also shown in Fig. 5.4. In addition, the treatment improves the area along the boundaries (e.g. head), where the concave appearance is avoided after our treatment. Moreover, after hole filling, the feet of the model have sufficient points to be reconstructed better.

Fig. 5.3 is another example showing the missing data resulted from using 3D scanners. The gap between the fins is difficult to be reached by the scanner rays and thus has a large area of missing data. Direct reconstruction is not suitable for those areas. With our treatment, which fills the missing area with points first, there is significant improvement on the resultant mesh compared with the original one. In addition, we compare our results with another hole filling tool which is the plug-in of PointShop3D[ZPKG02]. The results are very similar with slight differences onlu. However, the plug-in tool requires users to fill the holes manually by using a local MLS fitting, whereas our program fills the holes automatically, thus our method is more efficient.

Fig. 5.5 shows the results of seal models after our treatment on various kinds of reconstruction method. With the comparison between **POS**[KBH06b], **INT**[OBS05b], **RBF**[OBS05a], **CON**[HLZ*09], we can observe that prior to the situation of causing no negative effects, our method can produce results with a higher chance of successful reconstruction while **INT** fails to reconstruct the model using the original data. In the comparison between using the **CON** and **OUR** methods, it can be noticed that our selection operator can preserve more details for the models.

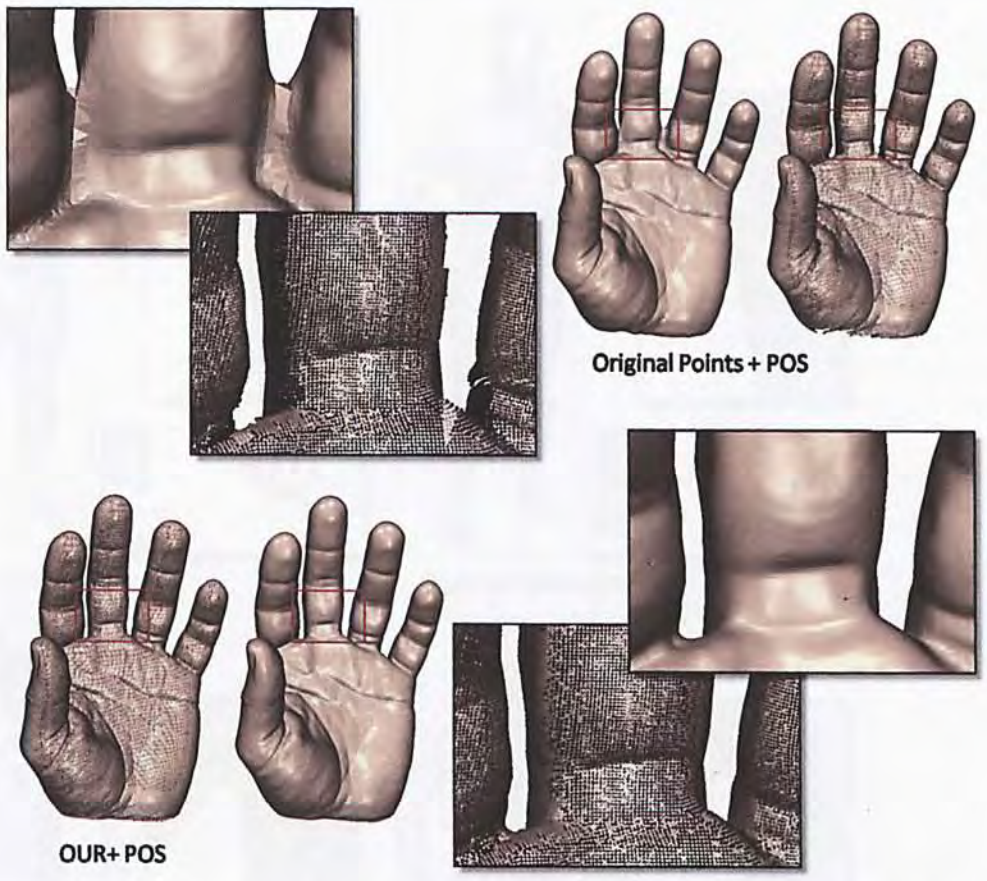

Figure 5.1: A point consolidation and reconstruction of a human hand model: (Top) Raw points and its mesh result. (Bottom) points after our point consolidation and its mesh result. The meshes are generated using the **POS** method.

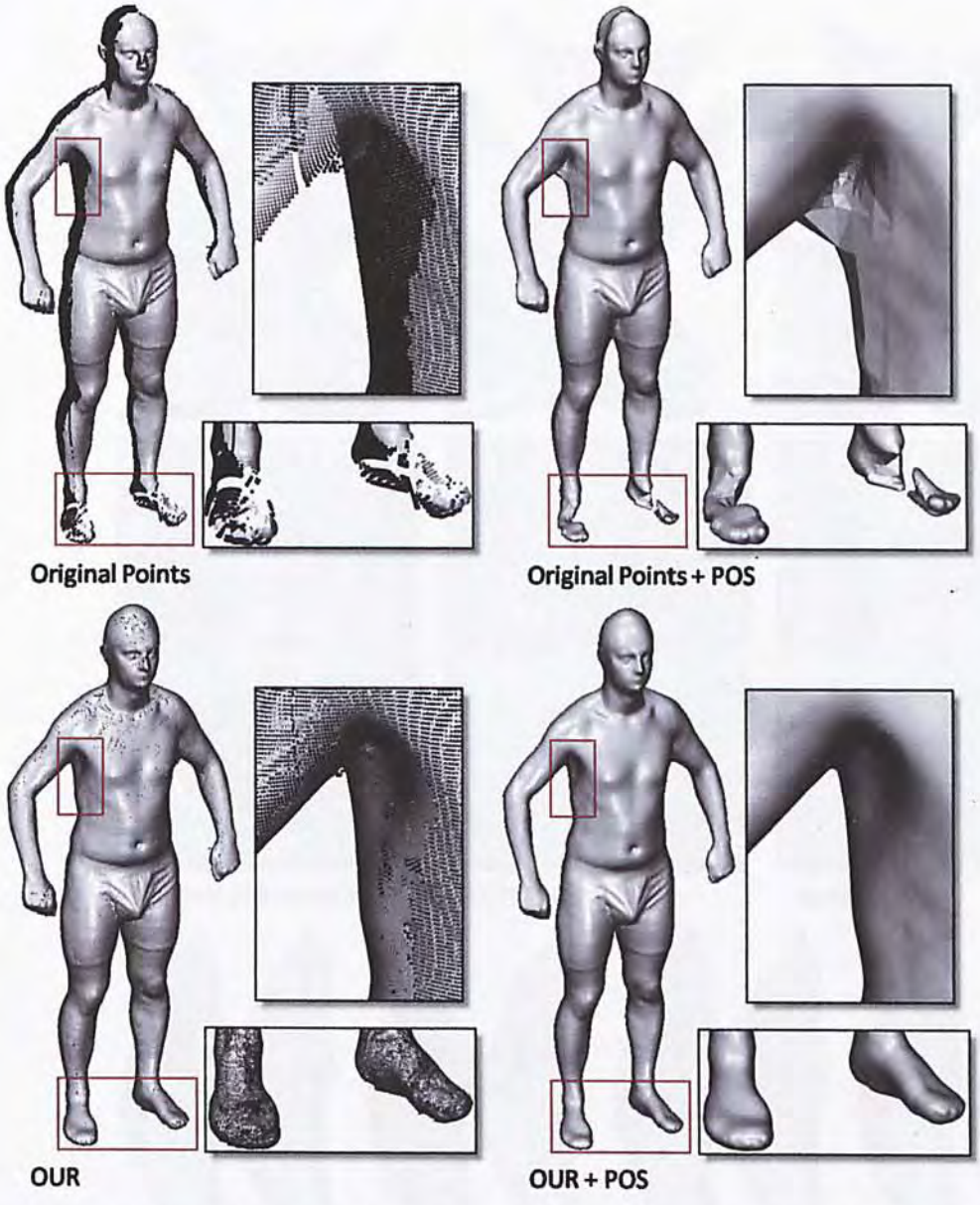

Figure 5.2: A point consolidation and reconstruction of a human model: (Top left) Raw points and (Top right) its mesh result. (Bottom left) points after our point consolidation and (Bottom right) its mesh result. The meshes are generated using the **POS** method.

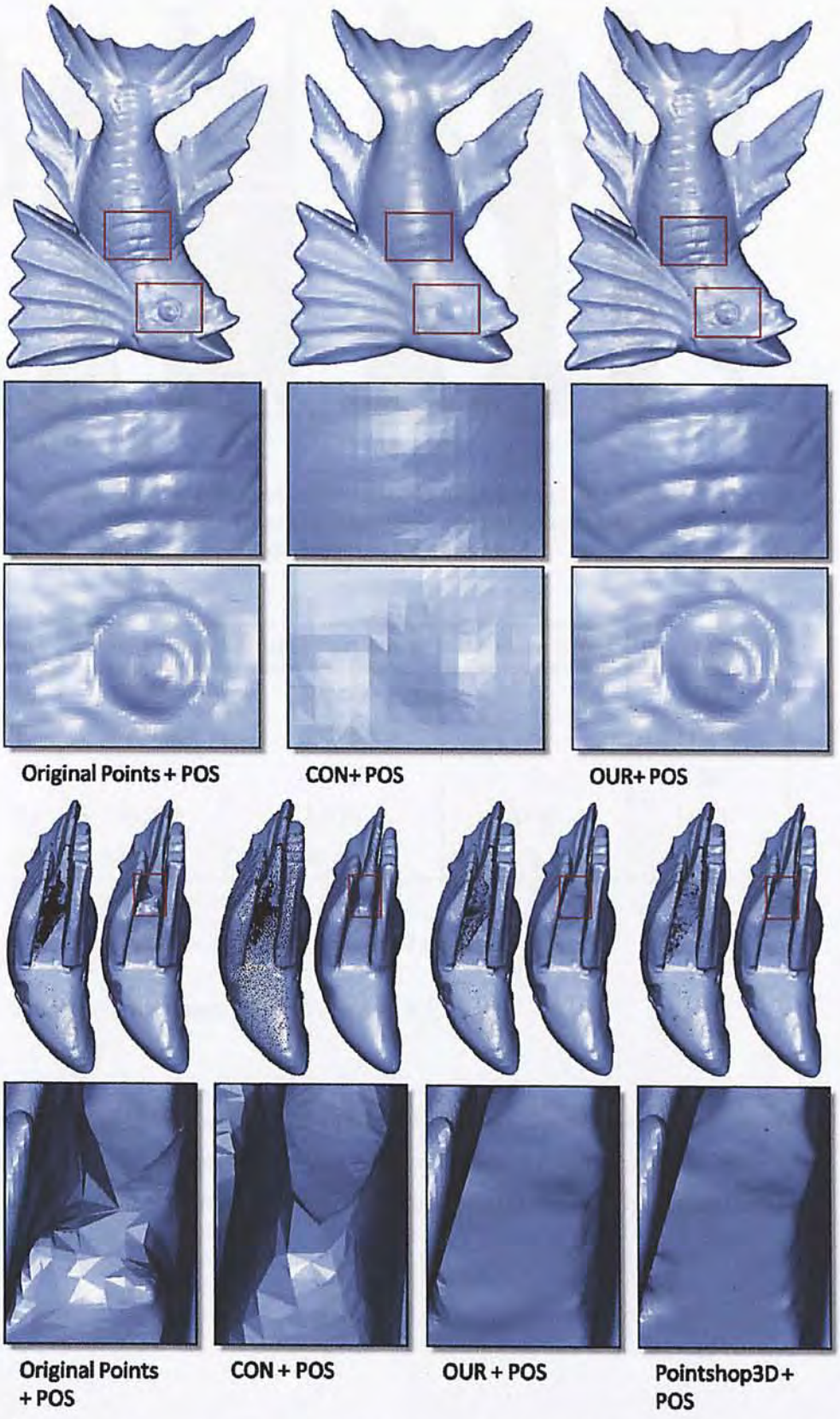

Figure 5.3: A comparison of consolidation using **CON**, **OUR**, and **PointShop3D**: (Top) Our method can successfully fill the holes between the fins and preserve more details compared with **CON**. (Bottom) The result obtained using **OUR** are similar to those using PointShop3D; however, **OUR** method can handle reconstruction automatically with a few parameters set up at the beginning.

Figure 5.4: A point consolidation and reconstruction of Japanese Lady model: (Left) Raw points and its mesh result. (Right) Points after our point consolidation and its mesh result. The meshes are generated using the **POS** method.

Model	Fig.	Input Points	Initial Particle #	Iterative Steps	Total time+
Inukshuk	1.1	205,858	10,293	6	3'13"
Hand	5.1	194,457	38,892	8	11'19"
Lady	5.4	175,514	35,103	2	4'26"
Body	5.2	170,346	8,518	6+8*	15'11"
Seal	5.5	889,076	44,454	2	6'59"

Table 5.1: Computational statistics

+ For system specification, please refer to section 5.1.

* To obtain a good result, we conduct the pre-processing on the human body example in two phases - in the first phase, the repulsion operation is not applied and a smaller value for h is used; and in the second phase, a normal procedure is fully applied.

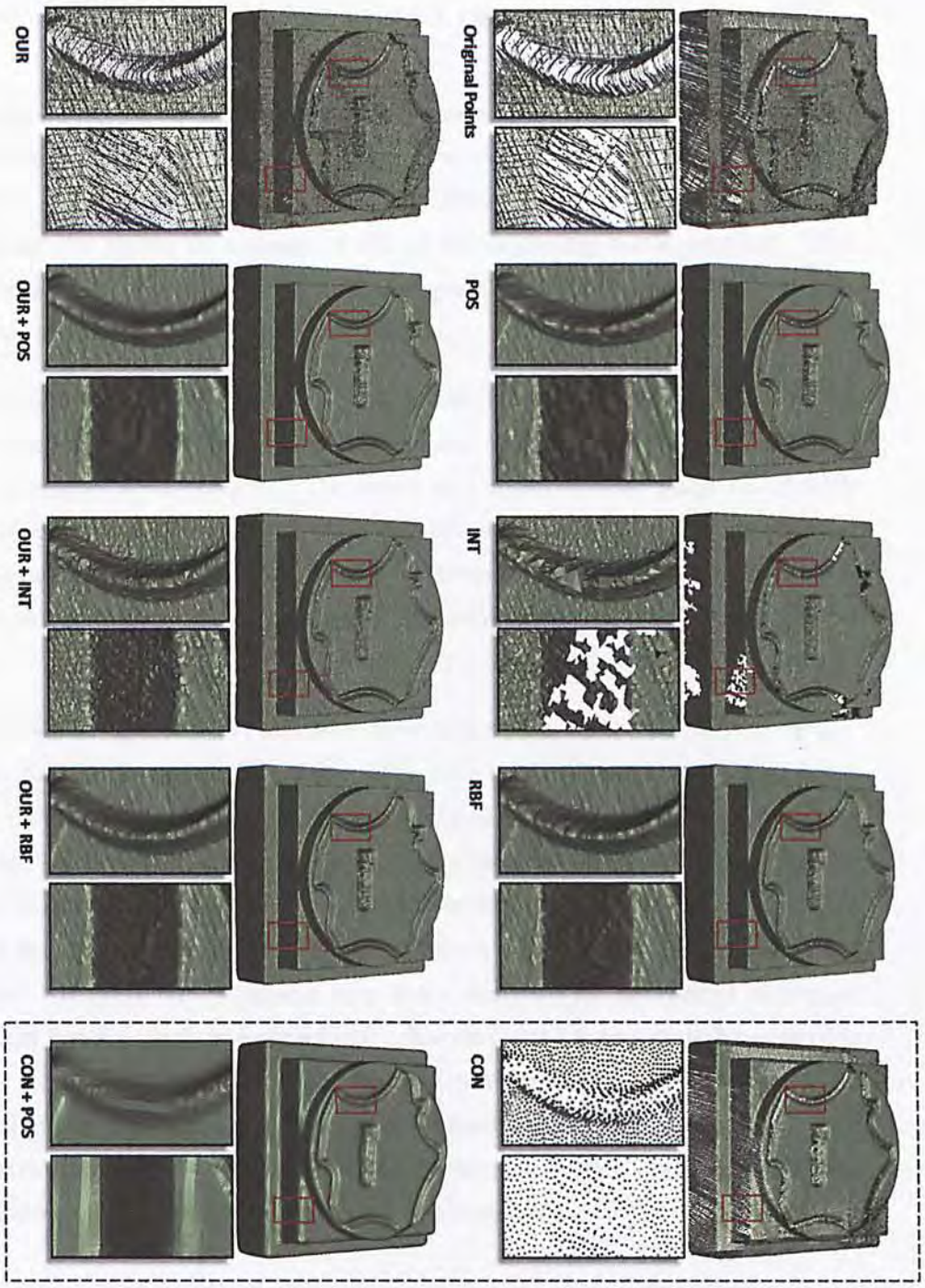

Figure 5.5: An example of the seal model. The point cloud processed under our framework results in higher quality surfaces reconstructed by various methods. The geometric details are preserved by our method, while the consolidation method of **CON** removes the geometric details.

5.3 Effect of our de-noising based on up-sampled points

In our program, we have a de-noising step that removes non-structural noise. We have tested the effect on both models with naturally present noise and models with artificially generated noise. To generate the noise, we apply Gaussian noise to the original point cloud by shifting the points in a range of 2% of the bounding box's diagonal. The Gaussian noise is randomly distributed with the specified percentage of amount. The resultant points are then reconstructed and studied.

Fig. 5.6 is the point set of a human face. For better rendering, we first find out the point orientation. Note that our algorithm does not necessarily require the original points to be equipped with normals. The result is a much cleaner point cloud with a significant amount of noise removed. And due to our consolidation framework, the holes become smaller. The same result can be observed in the mesh result, where the mesh surfaces are smoother than the original ones with the holes small enough to be reconstructed.

In Fig. 5.7, different amounts of Gaussian noise are added to the Armadillo model. Starting from the top, the amounts are 5%, 10%, 15% and 20% respectively which are shown in the middle column. The left column is the direct reconstructions from their respective noise levels while the right column is the results obtained by applying our noise removal before performing the same reconstruction method. It can be seen that our treatment can still result in a good meshing even on a model with a large amount of Gaussian noise. Although our de-noising step is not designed for removal of structural noise, noise that is very similar to structural noise can still be detected. The effect is shown on Fig. 5.8 where a 'piece' of noisy points is removed. We also test our program on models with structural noise. Fig. 5.9 is a skull model containing a large amount of random and structural noise. We can see that the result is a much cleaner model with significant amount of structural noise detected and removed.

Figure 5.6: Noise removal of a human face model: (Top) The original model and its reconstruction. (Bottom) The model after our framework and its reconstruction. Note that both the point data and the mesh are improved after the treatment. Meshes are generated using **INT** method.

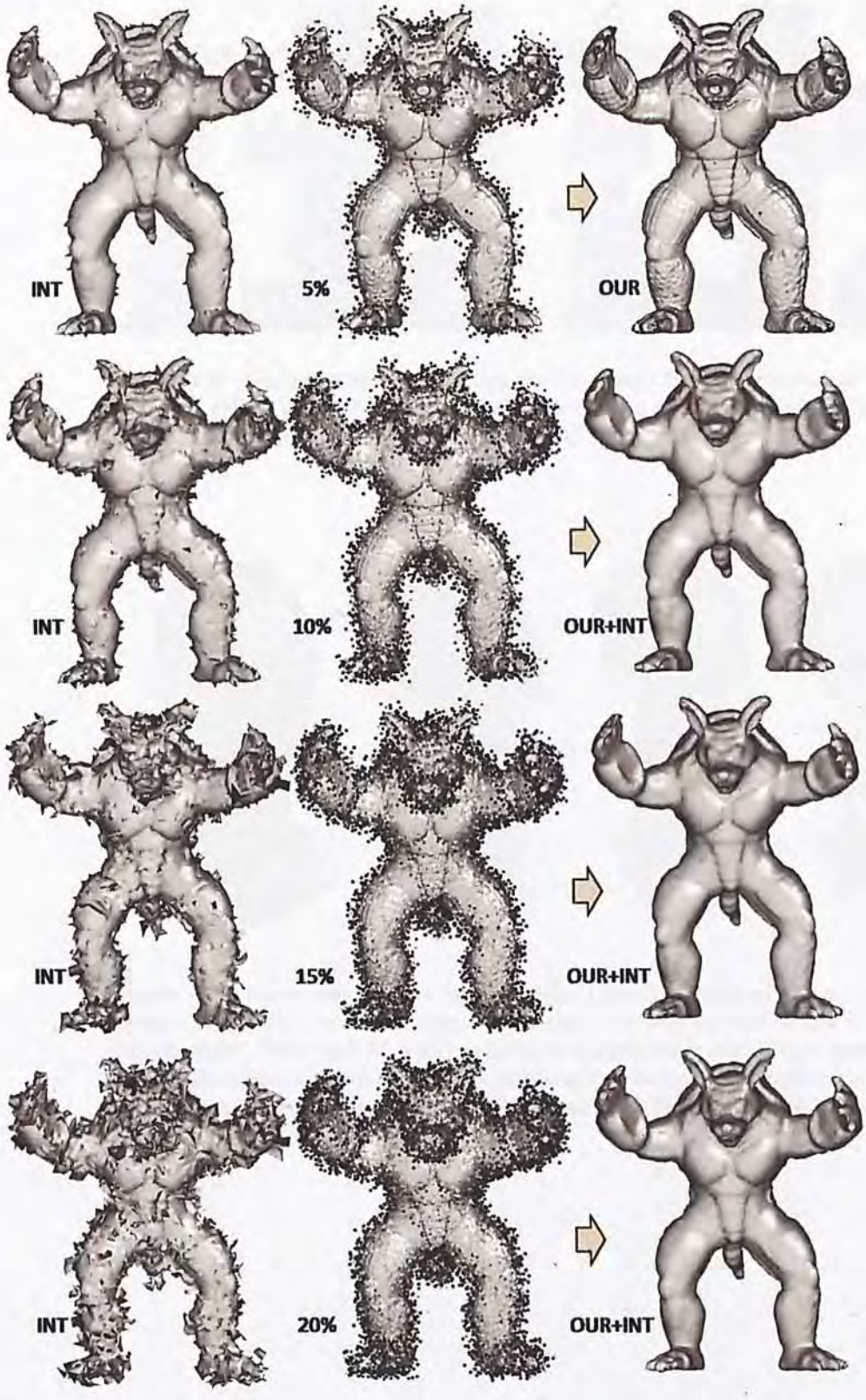

Figure 5.7: Noise removal of Armadillo model: The Gaussian noise is generated randomly with the maximum magnitude of 2% of the diagonal of the bounding box. The effect with/without our treatment is shown with (1st row) 5%, (2nd row) 10%, (3rd row) 15% and (4th row) 20% of noise. Meshes are generated using **INT** method.

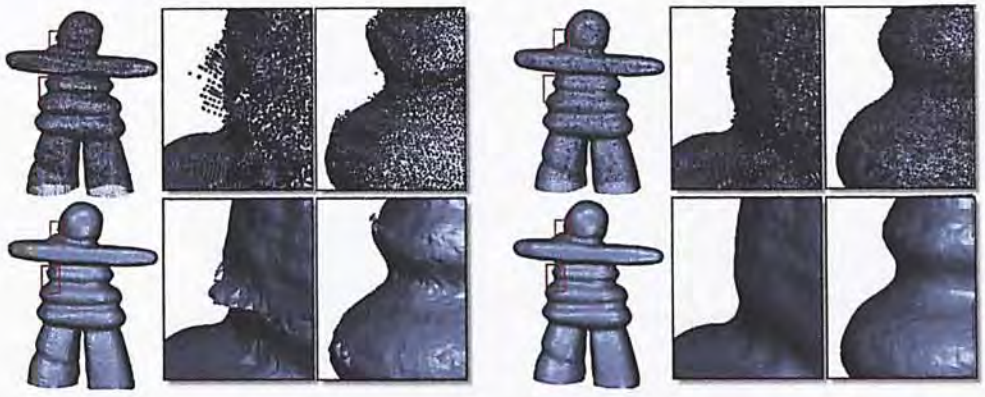

Figure 5.8: Noise removal of Inukshuk model: (Left) Original points and its reconstruction. (Right) Points with our treatment and its reconstruction. It can be seen that our method can remove a small amount of structural noise.

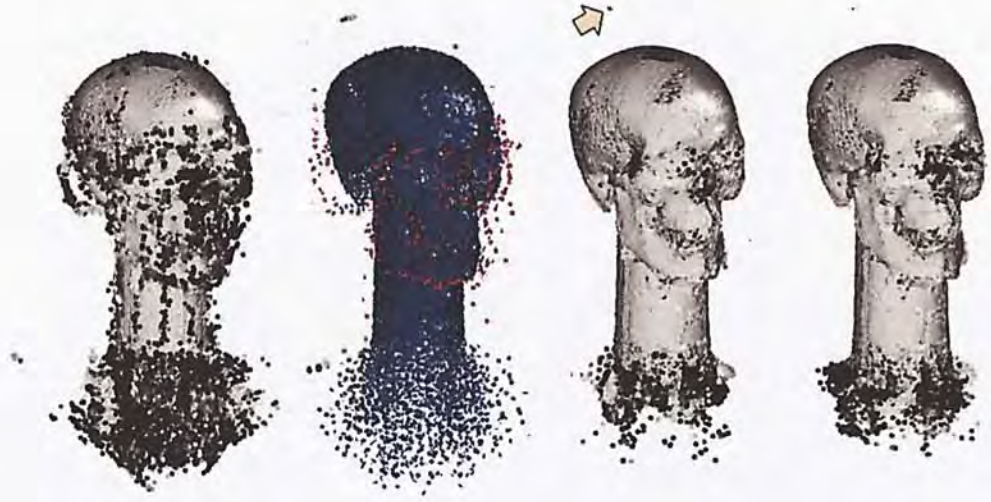

Figure 5.9: Noise removal of a skull model: (Left) The original points. Middle left During the "*particle removal*" step, red dots are identified as outliers and are removed. (Middle right) The result of skull model after applying one iteration of noise removal. (Right) The result of skull model after applying two iterations of noise removal. It can be seen that the results are significantly cleaner than that before the treatment.

Chapter 6

Conclusions

This chapter summarizes the key contributions and the limitations of the current implementation. The possible future research directions are pointed out at the end.

6.1 Advantages

In our work, we present and implement an iterative framework that progressively improves the quality of a given point set. Our idea is inspired from the image completion [FL09], which continually retrieves the missing data using a set of down-sampled particles. First we modify the WLOP operator by adding an orientation-aware term during the down-sampling and relaxation. The modification makes the consolidation more robust to regions with sharp and thin features. Next, a new repulsion operator further relaxes the particles by slightly repelling the particles to the sparse regions. This one results in a faster completion. The up-sampled points are then selectively merged back to the original point set to improve its uniformity. On the other hand, we arrange the mentioned steps as an iterative framework that continuously improve the quality of the point set. And to further speed up the framework, we introduce two terms of 'alive' and 'static' . Although the proportion of static to alive points is high in the later iterations, we prevent most of the work on static points and thus greatly shorten the computational time. Our consolidation framework is also advantageous to shape modeling, as we can use simpler algorithms to deform the points directly and reduce the interacting time of designing.

Compare to the recent works on point completion, we provide a very convenient tool that improves the quality of point set within a few user input. Our algorithm is also against some typical defect of models which are obtained from 3D scanners (Fig.1.1, 5.1,

5.2). Those models are oftenly missing points with their normal perpendicular to the scanner direction, where many point completion methods interpret the surface of the missing region unexpectedly and resulting in some indentation or imperfect shapes. On the other hand, we handle this kind of defect by naturally extent the 'surface' along its normal direction and insert new points onto it. The inserted points serve as supporting information which can improve the reconstruction using some conventional tools.

6.2 Factors affecting our algorithm

There are several factors affecting our algorithm. They are the density of points, the user input parameters, and the user selected points. In addition, some particular group of models are also affecting our result.

Inherited from [HLZ*09], our algorithm depends on the supporting factor h, which is initially some ratio controlled by the diagonal of the bounding box and the number of particles. This parameter controls the searching radius and also the supporting weight for the searched neighbors. We later modified it so that it depends on the particle density only(see section 3.2.2). This supporting factor plays a very important role in our framework as it affects the relaxation, repulsion and selection steps, in the form of weighting the kd nearest neighbors. A small h will cause undefined conditions (i.e. no neighbor searched within h) in some sparse regions, which results in insufficient data for computation. On the contrary, a large h may cause degenerations on regions of thin and sharp features, especially during WLOP for noisy data. Our new h depends on the particle density directly and is more natural and robust to various kinds of input models. In our program the value of h is initially estimated. However experience users can make changes refer to this value to get a better result.

At the beginning of computation, users have to define the value m which controls the number of particles. The down-sampled particles are then used to retrieve structural information for later repulsion and up-sampling. If m is small, there is an advantage of filling larger holes in a faster speed as the hole regions are more easily regarded as structural connected. However, if the model has thin layers, the results after WLOP are likely to have insufficient particles for separation. i.e. the thin layers are considered to be connected. This would lead to unwanted point insertion during the up-sampling step. Although most of the unwanted insertions can be detected and removed in the de-noising step, having a very small m is not preferred.

Another factor that affects our algorithm is the user selection. Recall that for models having open end surfaces (see Fig. 3.7), which maybe inborn or caused by missing points

or point preprocessing, our algorithm further defines the surface along its tangential direction. Users only needs to perform a few simple selections to define the regions to be constrained and will completely prevent further definitions.

We aim on processing various kinds of model with our algorithm. During our testing, we find out that most of the free form models are having a good result, both in consolidation and noise removal. In contrast, our algorithm has limitation on preserving sharp features. This can be happened when the inputs are some engineering models. The sharp features are often smooth out which we believe, is caused by the relaxation part and the APSS repulsion part. Recall that we are using MLS and APSS for surface definition. The smothing properties of these operators will hardly preserve sharp features. The up-sampled points from the particles are thus 'smoothed ' and after several iterations, some sharp features are lost.

6.3 Possible future works

Our framework can handle un-orientated points and improve their quality by an one-press iteration. In the future, there are still various improvements that can be done to our proposed algorithm. As our implementation is a block-like structure, different operators can be inserted to or removed from the pipeline stream easily, which makes our framework very adaptive to other algorithms.

6.3.1 Improve on the quality of results

During our implementation, we notice that some burrs are generated during the iteration. Those burrs are mainly appear on the unbounded open end surfaces defined by the points. Due to the local property of our framework, it is hard to detect such a situation by an effective algorithm. To solve this, we enable user selection to prevent such regions from contributing to the consolidation. However, the constrained region has the problem of noise and non-uniformity. Finding a way to detect such regions will surely have great improvements to our consolidation. In the future, we will consider using a global way to detect such regions. The authors of [HLZ*09] proposed a way of defining a boundary point, which we may consider taking advantage of to develop our new algorithm.

For models having large holes, the missing point areas after hole filling are flat and can be distinguished from the original sample points. In our current framework the surface of the down-sampled particles is defined using the APSS method, which is embedded

to have some smoothing effects. To retrieve the details of the missing area, we can take reference to the original sample points and extract the detailed information from them. Recently there are similar researches [KSW10] on image processing domain and we can study their effects on point based geometry.

6.3.2 Reduce user input

Our framework requires users to input some parameters before the one-press iteration. For some cases they also need to judge whether a region has to be constrained. Thus the consolidated result varies with different inputs. More importantly, users may have difficulty in deciding which area and how much of it is needed to be constrained. Although we suggest a range for the input parameters, it is better to reduce the number of parameters to as few as possible. While the constraint setting may be solved by detecting the boundary, finding the parameters m, the number of iterations and the number of repulsion step automatically will be the upcoming tasks for this project.

6.3.3 Multi-thread computation

Currently our framework is programed for a single core CPU environment, with limited parts that can be computed in parallel during the relaxations. In the future, the framework can be re-programed so that it is capable of multi-thread computation. There are various kinds of parallelization [BKBH07, KMXB08, KBH06a, ZHWG08] covering point processing, query point searching and mesh reconstruction on both multi-core CPU and GPU computation. Up to this stage, our adaptive computation plus the limited parallelization has speeded up the framework for around two to three times. We can speed up the processing by parallelizing the steps of kd tree construction, repulsion, up-sampling and mesh reconstruction.

Appendix A

Finding Neighbors

In point based processing, an effective way to build up the connectivity between points is by searching their neighborhoods. In our algorithm we often use the k-nearest neighbors (kNN) for computation. However, typical point set data consist of a few hundred thousands of points. Hence a fast and efficient searching method plays an important role in shortening the time of program execution. Recent researches state that a good data structure can result in faster searching. Thus many famous searching methods commonly consist of two parts: constructing and querying. For example, k-d tree and octree are the most common spatial data structure to decompose a space into different cells by constructing a special tree structure which stores the data. By searching through the structure, the speed is greatly enhanced.

A.1 k-d Tree

The k-d tree (short for k-dimensional tree) is a binary tree structure that stores k-dimensional points in each of its node. For each non-leaf node, one can imagine that there is a splitting hyperplane generated which subdivides the space into left and right parts called the subspace. While the points belonging to the left of this hyperplane fall into the left sub-tree of that node, the right sub-tree contains the points which are on the right of the hyperplane. Each node in the tree is assigned to one of the k-dimensions and the hyperplane direction is chosen to be perpendicular to the axis of that dimension. In our case, we have k = 3 which is represented by x,y and z axes. To construct a better k-d tree for searching, we use the library of [MA06] with the *standard kd-tree splitting rule* where the splitting point is the median of the coordinates along its assigned dimension.

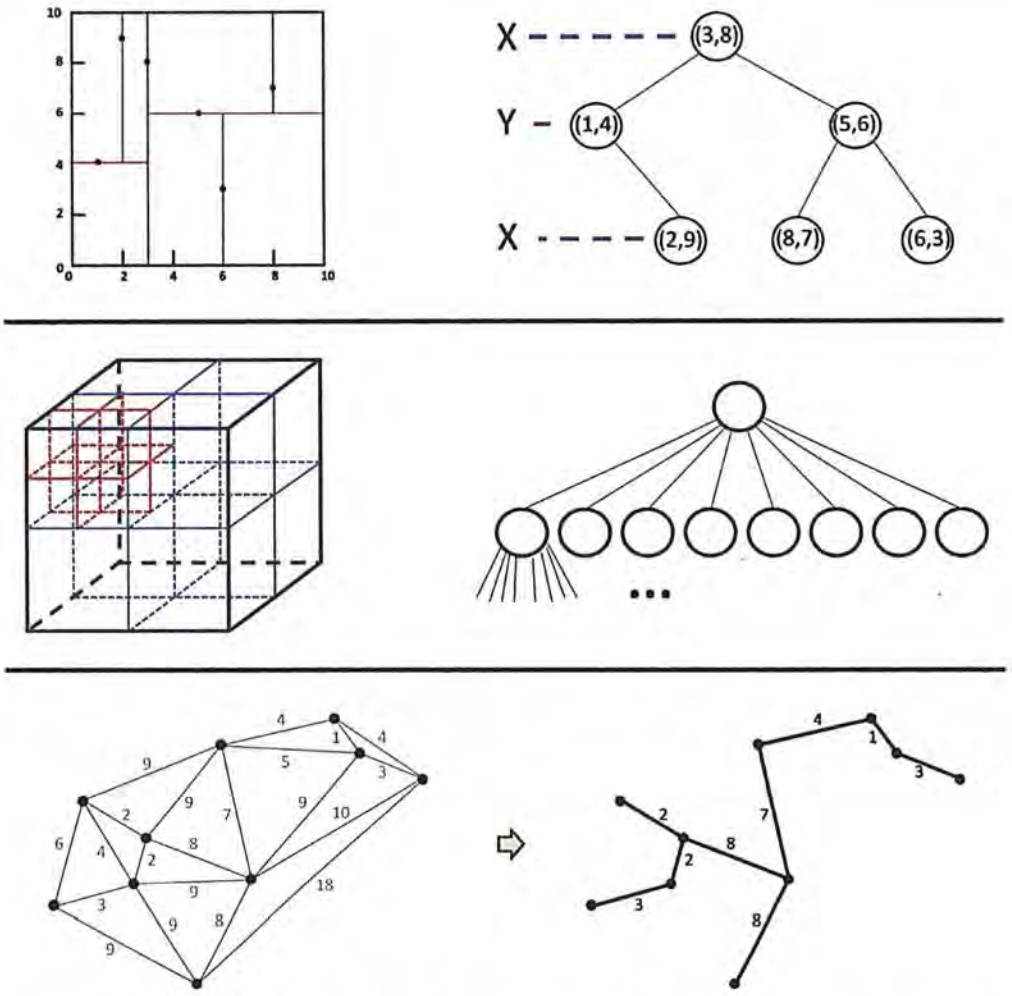

Figure A.1: Different tree like data structures including the (Top) k-d tree, (Centre) the octtree and (Bottom) the MST tree.

[MA06] is an open source kd tree constructing and searching method using single core CPU for computation. In our implementation we speed up the computation by storing the searched results on a table prior to CPU parallelization on investigation of the results. As most of the kNN searchings happen in the OAWLOP/WLOP step, the speedup for this step is about three to five times.

A.2 Octree

Different from k-d tree, octree[Mea82] has its non-leaf nodes subdivided into up to eight parts named the octants. During each subdivision, each octant will contain a datum that falls into its corresponding part. For an octant containing more than one datum, we call it non-leaf node and further subdivision is performed. The subdivision of the node can be ended by limiting the depth, or when every octant is a leaf node.

A.3 Minimum spanning tree

In our point geometry case, given a point cloud, we can treat every point pair to be connected (called edge) and form a graph with its respective edges weighting equal to the Euclidean distance. Then, a spanning tree of that graph is a subgraph that is a tree which links every point together with no cycles nor loops. A minimum spanning tree (MST)[HDD*92] is thus a spanning tree with a weight less than or equal to the weight of every other spanning tree. MST is a common way to find the normal orientation of a given point set: First a point is randomly picked and its normal direction is propagated to its connected neighboring point with the smallest weighting. We call these points the **visited** points. Then, among the visited points, we select another point with the smallest weighting connecting to them and propagate the normal direction. The process repeats until all the points are visited. And the result is a point set with orientated normal.

However, MST using Euclidean distance as weighting fails to orientate the normal in regions of sharp feature and thin layer, and regions with noise. Thus in [HLZ*09] a new method was proposed to specially deal with such cases. In our framework, we have both the method of [LW10] and the proposed method to estimate the normal orientation.

Appendix B

Principle Component Analysis

B.1 Principle component analysis

With the help of finding the k-nearest neighbors, one of the common ways to find the point orientation[PGK02] is by the principle component analysis (PCA). We first construct a 3×3 covariance matrix $\mathbf{C}$ and then find out the eigenvectors of the matrix. The orientation will be the eigenvector with the smallest eigenvalue. For instance, given a set $K(\mathbf{x})$ which stores the k-nearest neighbors of $\mathbf{x}$:

$$\mathbf{C} = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \cdots \\ \mathbf{x}_k - \bar{\mathbf{x}} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \cdots \\ \mathbf{x}_k - \bar{\mathbf{x}} \end{bmatrix}, \mathbf{x}_i \in K(\mathbf{x}) \tag{B.1}$$

where $\bar{\mathbf{x}}$ is the centroid of the neighbors $\mathbf{x}_i$ of $\mathbf{x}$ (see Fig. B.1). To find out the eigenvectors, we can solve the following eigen-problem:

$$\mathbf{C} \cdot \mathbf{v}_j = \lambda_j \cdot \mathbf{v}_j, j \in 0, 1, 2 \tag{B.2}$$

Since $\mathbf{C}$ is symmetric and positive semi-definite, we will have all the eigenvalues λ_j real-valued and the eigenvectors $\mathbf{v}_j$ form an orthogonal frame corresponding to the principal components of the point set defined by $K(\mathbf{x})$. λ_j is the eigenvalues which measure the variation of $\mathbf{x}_i \in K(\mathbf{x})$ along the direction of the corresponding eigenvectors. With the first principal component having as high variance as possible, each succeeding component in turn has the highest variance possible, but with the constraint that it is orthogonal to the preceding components. The total variation (sum of squared

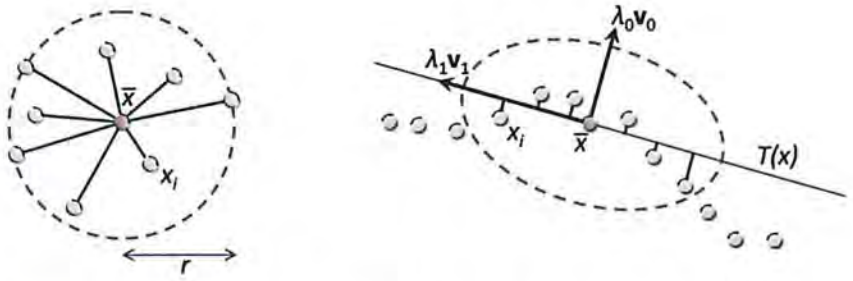

Figure B.1: Local neighborhood (Left) and covariance analysis (Right).

distances) of $\mathbf{x}_i$ from their center of gravity is thus

$$\Sigma_{i \in K(\mathbf{x})} |\mathbf{x}_i - \bar{\mathbf{x}}|^2 = \lambda_0 + \lambda_1 + \lambda_2 \tag{B.3}$$

Assume $\lambda_0 \leq \lambda_1 \leq \lambda_2$, it follows that the plane

$$T(\mathbf{x}) : (\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_0 = 0 \tag{B.4}$$

passing through $\bar{\mathbf{x}}$ minimizes the sum of squared distances to the neighbors of $\mathbf{x}$. Thus $\mathbf{v}_0$ approximates the surface normal at $\mathbf{x}$.

In our program, we use OAPCA which is the improved version of PCA in some cases. The OAPCA requires an initial orientation of the points first. Then for each query point $\mathbf{x}$, we search for its k-d neighbors $K(\mathbf{x})$. The difference of orientation between $\mathbf{x}$ and $\mathbf{x}_i$ affects its contribution by simply ignoring $\mathbf{x}_i$ with $\mathbf{n}_\mathbf{x} \cdot \mathbf{n}_{\mathbf{x}_i} < 0$ when composing the covariance matrix. This can improve the accuracy of orientation in regions of thin layers and sharp corners. Commonly, the initial orientation of the points can be found by PCA followed by [HLZ*09] using minimum spanning tree or [LW10] as chosen by users. In our program, we implemented both.

Appendix C

UI of the program

C.1 User Interface

During the program implementation, we need to continuously monitor the quality of our proposed algorithms. Therefore we implemented a precise and user friendly interface for control and display. The interface is also capable of various file format import/export. Here are some snapshots of our UI.

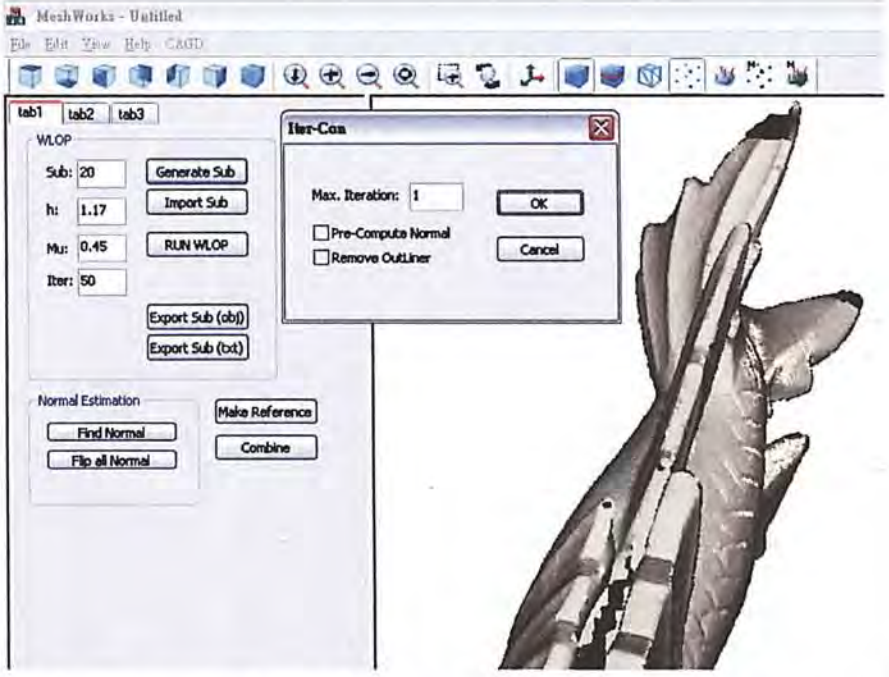

Figure C.1: User interface of the program: The right area renders the model which shows the qualities of our algorithm. In the left area there are tabs holding the input parameters and some control buttons. Users can drag the render area to change the viewing direction, as well as sliding and zooming. And depends on some operations, there are additional pop up dialogs requesting for parameter input.

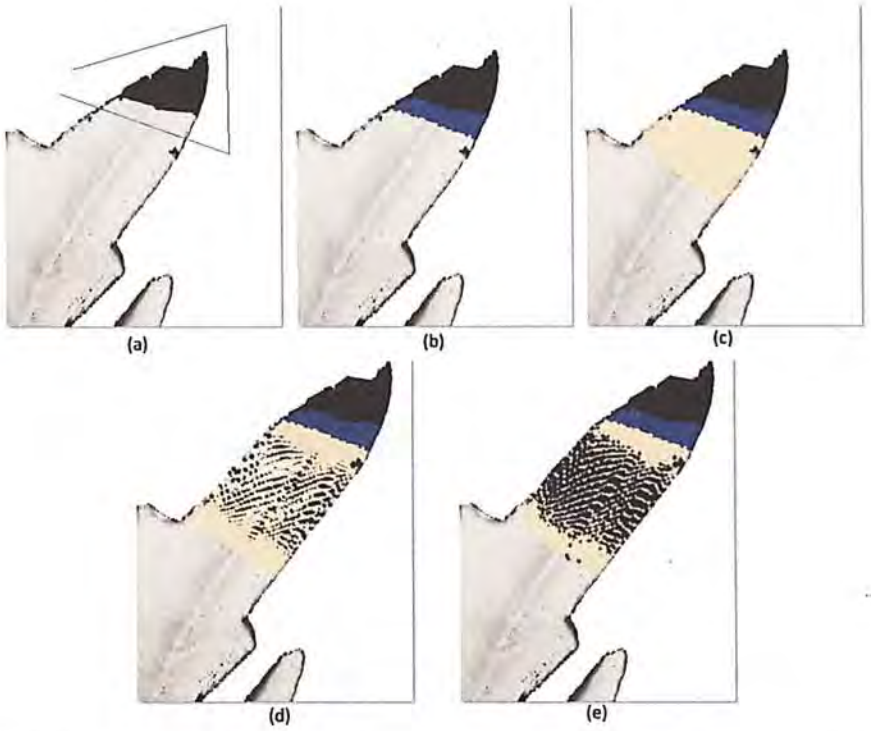

Figure C.2: To define different regions, we simply select an area by picking points (a). The regions are marked with different color (b and C). To deform, we drag the handle region (d) and some points will be inserted after moving (e). The same is true when setting the constraint for our iterative consolidation.

Appendix D

Publications

- Shengjun Liu, Kwan-Chung Chan, and Charlie C. L. Wang, "Iterative Consolidation of Unorganized Point Clouds", IEEE COMPUTER GRAPHICS AND APPLICATIONS, accepted.

Bibliography

[AA09] ALEXA M., ADAMSON A.: Interpolatory point set surfaces - convexity and hermite data. *ACM Trans. Graph. 28* (May 2009), 20:1–20:10.

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 21–28.

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph. 9*, 1 (2003), 3–15.

[AK04] AMENTA N., KIL Y. J.: Defining point-set surfaces. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 264–270.

[Bar84] BARR A. H.: Global and local deformations of solid primitives. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 21–30.

[BK04a] BOTSCH M., KOBBELT L.: Phong splatting. In *In Proceedings of Point-Based Graphics 2004* (2004).

[BK04b] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), SGP '04, ACM, pp. 185–192.

[BKBH07] BOLITHO M., KAZHDAN M., BURNS R., HOPPE H.: Multilevel streaming for out-of-core surface reconstruction. In *SGP '07* (2007), pp. 69–78.

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and

representation of 3d objects with radial basis functions. In *SIGGRAPH '01* (2001), pp. 67–76.

[CDD*04] CLARENZ U., DIEWALD U., DZIUK G., RUMPF M., RUSU R.: A finite element method for surface restoration with smooth boundary conditions. *Comput. Aided Geom. Des. 21* (May 2004), 427–445.

[CN03] CARSTEN M., NEIL A. D.: A new point cloud simplification algorithm. In *Presented at 3rd LASTED International Conference on Visualization, Imaging, and Image Processing* (8-10 Sep 2003).

[CR94] CHANG Y.-K., ROCKWOOD A. P.: A generalized de casteljau approach to 3d free-form deformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 257–260.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. In *SIGGRAPH '04* (2004), pp. 905–914.

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: a water-tight surface reconstructor. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications* (2003), pp. 127–134.

[DMGL02] DAVIS J., MARSCHNER S., GARR M., LEVOY M.: Filling holes in complex surfaces using volumetric diffusion. In *First International Symposium on 3D Data Processing, Visualization, and Transmission* (2002), pp. 19–21.

[FH75] FUKUNAGA K., HOSTETLER L.: The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on 21*, 1 (Jan. 1975), 32 – 40.

[FL09] FANG C.-W., LIEN J.-J. J.: Rapid image completion system using multiresolution patch-based directional and nondirectional approaches. *IEEE Transactions on Image Processing 18*, 12 (2009), 2769–2779.

[GBP05] GUENNEBAUD G., BARTHE L., PAULIN M.: Interpolatory refinement for real-time processing of point-based geometry. 657–666.

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. *ACM Trans. Graph. 26* (July 2007).

[GM97] GUY G., MEDIONI G.: Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data. *IEEE Trans. Pattern Anal. Mach. Intell. 19* (November 1997), 1265–1277.

[HBJF05] HART J. C., BACHTA E., JAROSZ W., FLEURY T.: Using particles to sample and control more complex implicit surfaces. In *ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), SIGGRAPH '05, ACM.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92* (1992), pp. 71–78.

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 19–26.

[HK06] HORNUNG A., KOBBELT L.: Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 41–50.

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph. 28* (December 2009), 176:1–176:7.

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21* (July 2002), 339–346.

[KB00] KOBBELT L., BOTSCH M.: $\sqrt{3}$ subdivision. *In Proceedings of ACM SIGGRAPH 2000* (2000).

[KBH06a] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70.

[KBH06b] KAZHDAN M. M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Symposium on Geometry Processing* (2006), pp. 61–70.

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 57–66.

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th*

annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 105–114.

[KE] K. S., E. F.: Wires: A geometric deformation technique. In *SIGGRAPH 1998.*

[KMXB08] KUN Z., MINMIN G., XIN H., BAINING G.: Highly parallel surface reconstruction. *Microsoft Technical Report, MSR-TR-2008-53* (2008). http://www.kunzhou.net/2008/MSR-TR-2008-53.pdf.

[KSO04] KOLLURI R., SHEWCHUK J. R., O'BRIEN J. F.: Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), SGP '04, ACM, pp. 11–21.

[KSW10] KWOK T.-H., SHEUNG H., WANG C.: Fast query for exemplar-based image completion. *Image Processing, IEEE Transactions on 19*, 12 (2010), 3106 –3115.

[KV03] KALAIAH A., VARSHNEY A.: Modeling and rendering of points with local geometry. *IEEE Trans. Vis. Comput. Graph. 9*, 1 (2003), 30–42.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169.

[LCOLTE07] LIPMAN Y., COHEN-OR D., LEVIN D., TAL-EZER H.: Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. 26* (July 2007).

[Lev03] LEVIN D.: Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization* (2003), Springer-Verlag, pp. 37–49.

[Llo82] LLOYD S. P.: Least squares quantization in pcm. *IEEE Transactions on Information Theory 28* (1982), 129–136.

[LP05] LANGE C., POLTHIER K.: Anisotropic smoothing of point sets. *Comput. Aided Geom. Des. 22* (October 2005), 680–692.

[LW10] LIU S., WANG C.: Orienting unorganized points for surface reconstruction. In *Computers and Graphics, Special Issue of IEEE International Conference on Shape Modeling and Application (SMI 2010)* (June 21-23, 2010), vol. 34, pp. 209–218.

[LWL*08] LIU Y., WANG W., LVY B., SUN F., YAN D. M., LU L., YANG C.: *On Centroidal Voronoi Tessellation - Energy Smoothness and Fast Compu tation.* Tech. rep., Hong-Kong University and INRIA - ALICE Project Team, 2008. Accepted pending revisions.

[MA06] MOUNT D. M., ARYA S.: Ann: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/ mount/ANN/.

[Mea82] MEAGHER D. J.: Geometric modeling using octree encoding. In *Computer Graphics and Image Processing 19,2 (June 1982)* (1982), pp. 129–147.

[MLT00] MEDIONI G., LEE M.-S., TANG C.-K.: A computational framework for segmentation and grouping. *Elsevier.* (2000).

[OBS05a] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graph. Models 67*, 3 (2005), 150–165.

[OBS05b] OHTAKE Y., BELYAEV A. G., SEIDEL H.-P.: An integrating approach to meshing scattered point data. In *Symposium on Solid and Physical Modeling* (2005), pp. 61–69.

[PGK02] PAULY M., GROSS M. H., KOBBELT L.: Efficient simplification of point-sampled surfaces. In *IEEE Visualization* (2002).

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 641–650.

[PMG*05] PAULY M., MITRA N. J., GIESEN J., GROSS M., GUIBAS L. J.: Example-based 3d scan completion. In *Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association.

[Pra87] PRATT V.: Direct least-squares fitting of algebraic surfaces. *SIGGRAPH Comput. Graph. 21* (August 1987), 145–152.

[SACO04] SHARF A., ALEXA M., COHEN-OR D.: Context-based surface completion. *ACM Trans. Graph. 23*, 3 (2004), 878–887.

[SF08] SONG H., FENG H.-Y.: A global clustering approach to point cloud simplification with a specified data reduction ratio. *Computer Aided Design 40*, 3 (2008), 281–292.

[SK02] SAVCHENKO V., KOJEKINE N.: An approach to blend surfaces. In *Computer Graphics International* (2002), pp. 139–150.

[SLS*06] SHARF A., LEWINER T., SHAMIR A., KOBBELT L., COHEN-OR D.: Competing fronts for coarse-to-fine surface reconstruction. *Computer Graphics Forum 25*, 3 (2006), 389–398.

[SLS*07] SHARF A., LEWINER T., SHKLARSKI G., TOLEDO S., COHEN-OR D.: Interactive topology-aware surface reconstruction. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 43.

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20* (August 1986), 151–160.

[ST92] SZELISKI R., TONNESEN D.: Surface modeling with oriented particle systems. *SIGGRAPH Comput. Graph. 26* (July 1992), 185–194.

[Tur92] TURK G.: Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), SIGGRAPH '92, ACM, pp. 55–64.

[VCBS03] VERDERA J., CASELLES V., BERTALMIO M., SAPIRO G.: Inpainting surface holes. In *International Conference on Image Processing* (2003), pp. 903–906.

[VCP08] VALETTE S., CHASSERY J. M., PROST R.: Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 369–381.

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 159–166.

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 269–277.

[WPH*04] WEYRICH T., PAULY M., HEINZLE S., KEISER R., SCANDELLA S., GROSS M.: Post-processing of scanned 3d surface data. In *Symposium on Point-Based Graphics* (2004), pp. 85–94.

[ZHWG08] ZHOU K., HOU Q., WANG R., GUO B.: Real-time kd-tree construction on graphics hardware. *ACM Trans. Graph. 27* (December 2008), 126:1–126:11.

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 322–329. http://www.cs.umd.edu/ mount/ANN/.

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 371–378.

[ZS00] ZORIN D., SCHRODER P.: Subdivision for modeling and animation. *SIGGRAPH 2000 Course Notes* (2000).