Shrink Wrapping Small Objects

Sricharana Rajagopal

Master of Science

School Of Computer Science

McGill University

Montreal,Quebec

2015-06-22

Requirements Statement

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Prof. Kaleem Siddiqi for supporting me throughout my research with his patience and knowledge and for funding me during my research. Without his encouragement and experience, this thesis could not have been compiled or written.

I would also like to thank my lab mates Emmanuel Piuze, Chu Wang, Morteza Rezanejad and Babak Samari for their helpful comments related to research as well other technical issues and for creating a friendly environment to work in. Special thanks to Pierre Ablin for helping me with the french translation of my abstract.

Finally, I should also mention my sister, Swetha and my parents without whose unwavering support, I would have never come this far.

ABSTRACT

Reconstruction of 3D surface models has been a fundamental problem in computer vision and there has been a lot of work done in this area. In this thesis, we introduce a novel approach to 3D surface reconstruction by "shrink wrapping" a surface around 3D point clouds using weighted geometric flows.

Specifically, we consider the problem of scanning and modelling small objects using easily available RGBD sensors such as the Kinect. These sensors allow for the acquisition of view-based registered depth and color data. We register the point clouds by recovering the rigid transformation between successive pairs in a sequence of views, and then demonstrate the utility of surface evolution for shrink wrapping the result using geometric flow based approaches for surface modelling. In our experiments we use a PrimeSense Carmine 1.09, which is a high resolution short range RGBD sensor suited to capture small objects.

We present results of our experiments on four different objects. We obtain high quality surface and appearance models of small objects that are competitive with commercial software packages while better capturing surface detail.

ABRÉGÉ

La reconstruction de modèles de surfaces en trois dimensions est un problème fondamental en vision par ordinateur, et de nombreux travaux ont été effectués sur ce sujet. Dans cette thèse, nous introduisons une nouvelle approche de la reconstruction de surface en trois dimensions qui consiste à “emballer” la surface comme un film plastique autour d’un nuage de points en trois dimensions à l’aide d’équations d’évolution géométrique pondérées.

Plus spécifiquement, nous considérons le problème de scanner et modéliser de petits objets en utilisant des capteurs RGBD facilement accessibles tels que le Kinect. Ces capteurs permettent d’acquérir à la fois des données de profondeur de champ et de couleurs. Nous acquérons le nuage de points en retrouvant la transformation rigide entre deux prises de vue successives, puis utilisons une surface évolutive pour “emballer” le résultat en utilisant des équations d’évolution géométrique afin d’obtenir un modèle de surface.

Nous présentons les résultats de nos expériences sur quatre différents objets. Nous obtenons des surfaces de haute qualité ainsi que des modèles d’apparence pour de petits objets qui sont au niveau de ceux qui sont obtenus avec des logiciels commerciaux, tout en capturant mieux les détails des surfaces.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Introduction

Computer vision is a sub-discipline of artificial intelligence that studies how to understand the real world by acquiring and analysing images projected by the real world. The data, besides 2D images, can also be video sequences, images from multiple cameras, three dimensional data from RGBD sensors, laser scanners or multidimensional data from medical scanners, etc. The goal of computer vision is to replicate human vision by extracting and interpreting useful information from the image data so as to be able to make inferences about the visual world.

Current research in computer vision has allowed machines to reliably perform tasks such as face recognition, tracking objects in a video sequence and 3D scene reconstruction. However, computer vision as a discipline has a long way to go. This is because of the inherent challenges of modelling the human visual system without fully understanding how it works. Another reason why computer vision is non trivial is because of its inverse nature, i.e., we try to reconstruct the world we see given images or other visual data, but without being able to make direct physical measurements. Zucker summarizes the vision problem in his Early Vision paper [33] as follows,

"Three dimensional physical structure in the scene projects into two-dimensional structure in the image. This process must be inverted i.e., somehow, physical structures must be inferred from image structures."

Generally, low-level computer vision tasks have to do with extracting low level features such as edges and corners from images. Mid and higher level vision problems deal with tasks like inference of surfaces from low level data. They also include high level tasks such as object recognition, scene reconstruction etc. These stages of computer vision tasks were based on the views of two vision scientists, Helmholtz (1821-1894) and Mach (1838-1916) [33]. Helmholtz separated the vision problem into low level and high level processing. On the other hand Mach suggests that a separation should be made between the analysis of a task and the mechanism proposed to accomplish it [33]. The first attempt at a computer vision system by L.Roberts was also based on both these themes. In Roberts' system, low level processing involved the extraction of a "cartoon-like" drawing out of an image and high level processing involved recognition of objects. The mechanism applied at the low-level was called "edge detection" and the high level mechanism being called object matching into a database.

This thesis focuses on the specific mid level vision problem of 3D surface reconstruction. 3D surface reconstruction involves trying to understand and model the shape of physical objects from visual data which is described in the following section.

1.2 Surface Reconstruction

The goal of surface reconstruction is to find a surface, that approximates a physical surface, using points sampled from the physical surface. These points can

be obtained either from RGB images, in which case the depth has to be computed or as 3D point clouds obtained from 3D sensors such as the Microsoft Kinect [31].

Surface reconstruction from unorganized point clouds, like most vision problems, is ill-posed, i.e., it has no unique solution. The connectivity of the data set and the topology of the real surface can be complicated. A good surface reconstruction algorithm should have a surface representation that is good for both static rendering and also for deformation and other dynamic operations on surfaces. It should also be able handle a variety of topologies as well as noisy and non-uniform data sets [20, 32].

1.2.1 Surface Representations

There are two types of surface representations: explicit and implicit. Explicit representations describe precisely the location of the surface, i.e., they explicitly define the set of points belonging to a surface. Surfaces can also be represented implicitly as a particular isocontour of some scalar function.

Consider for example a curve $\phi(\vec{x}) = x^2 + y^2 - 1$ seen in figure (1–1). Here the border between the inside, Ω^-, and the outside, Ω^+, is called the interface. The interface can be implicitly defined by the $\phi(\vec{x}) = 0$ isocontour. The explicit representation of this interface is the unit circle defined by $\partial\Omega = \{\vec{x}||\vec{x}| > 1\}$ [20].

Generally, in two spatial dimensions, the explicit representation of an interface needs to specify all the points on the curve. A general approach to do so is to parametrize the 2D curve with a vector function $\vec{x}(s)$, where s is parameter, that moves along the length of the curve, in the interval $[s_o, s_f]$ [20]. For closed curves, $\vec{x}(s_o) = \vec{x}(s_f)$. A straightforward method of discretizing this explicit representation

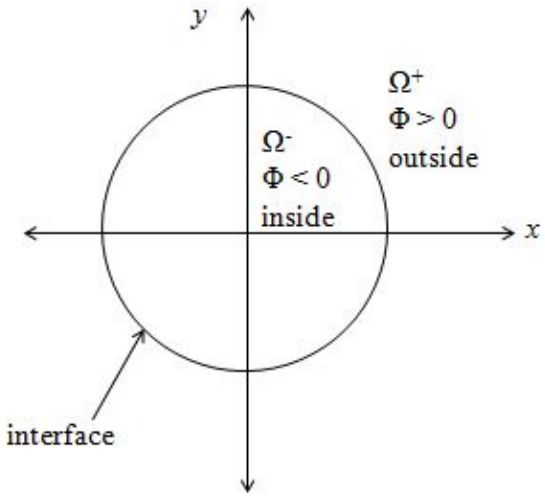


Figure 1–1: Implicit representation of the curve $x^2 + y^2 - 1$ (adapted from [20])

is to discretize the parameter s into a finite set of points such that $s_o < ... < s_{i-1} < s_i < s_{i+1} < ... < s_f$. Thus a point s_i in the parameter space has a corresponding point on the interface represented by the vector function as $\vec{x}(s_i)$. In an explicit discretization we only know the location of the 2D curve in these finite set of points. To approximate the location of the curve in the remaining infinite set of points, interpolation is usually used.

An implicit representation of a 2D curve, is expressed as an isocontour of an implicit function ϕ. For such a representation, we need to discretize the implicit function ϕ. Just as in the case of the explicit representation, we know the location of the implicit function ϕ only in finite number of points. To locate the interface, we use the values of ϕ at the known data points and interpolate the isocontour, $\phi(\vec{x}) = 0$.

In three spatial dimensions, explicit surface representations are difficult to discretize when connectivity is not known. Connectivity is straightforward, however,

in 2D; it is based on the ordering of the points. For example, $\vec{x}(s_i)$ is connected to $\vec{x}(s_{i-1})$ and $\vec{x}(s_{i+1})$ and so on. However, it is not as straightforward in 3D. If we know the connectivity and the exact surface, then surface reconstruction by tiling using triangles is very simple. But if the connectivity is not known, surface representations can be very inaccurate.

Implicit surfaces have several advantages over explicit ones in that they are flexible to topological changes, they use simple data structures and they are more efficient to store in memory. A major drawback of implicit surfaces is that they can be computationally more expensive. To be precise, in $\Re^n$, an explicit discretization needs to resolve only a $(n-1)$-dimensional set [20]. For example, in the two dimensional example from above, only a one dimensional set in the interval $[s_o, s_f]$ has to be resolved. However for the implicit approximation has to resolve an $n-$dimensional set. One easy way to deal with this issue is to only consider points close to the isocontour $\phi(x) = 0$ since we are only concerned with the zero isocontour.

One nice thing about implicit surfaces is that connectivity need not be determined for discretization. But the most powerful property of implicit surfaces is that it is straightforward to go from two spatial dimensions to three or higher spatial dimensions.

There has been a lot of work done in surface reconstruction using each of these representations. This is discussed in detail in Chapter 2.

1.3 Objective Of this Thesis

In this thesis, we consider the problem of scanning and modelling small objects using RGBD data acquired from 3D sensors like the Kinect [31]. The goal of this

thesis is to use geometric flow based approaches to "shrink wrap" a surface around registered dense 3D point clouds of small objects that can be scanned on a desktop with a turn table.

1.4 Contributions of this Thesis

The main contributions of this thesis are listed below.

- We are able to generate accurate dense 3D point clouds of objects using registered depth and color point cloud data from easily available RGBD cameras.
- We obtain high quality surface and appearance models of small objects by wrapping a surface around dense 3D point clouds using active surface flows.
- Our method is able to capture better surface detail and is robust to holes because of the doublet term in the geometric flow. We are able to create watertight models even in the presence of some missing data.
- We are also able to obtain accurate estimates of surface normals and mean curvature.

1.5 Publications arising from this thesis

Shrink wrapping small objects. Sricharana Rajagopal and Kaleem Siddiqi. In Proceedings of the 2015 Conference on Computer and Robot Vision, Halifax, Canada, June 2015.

The majority of the algorithm development, implementation, experimental work and writing was carried out by the author of this thesis. K. Siddiqi assisted in methodological development and writing.

1.6 Thesis Overview

The chapters in thesis are organized as follows. Chapter 2 discusses some of the related work in surface reconstruction. Chapter 3 describes how an incremental Iterative Closest Point algorithm is used to register pairs of views of 3D point cloud data to create a dense point cloud representing the surface of the object.

Chapter 4 introduces the "shrink wrapping" method of surface reconstruction used in this thesis based on weighted geometric flows. We discuss in detail the theory of front evolution and level set methods or curve and surface evolution. We then, describe how these are modified to attach themselves to a dense 3D point cloud. We also present experimental results of our surface reconstruction algorithm in Chapter 5. Finally, Chapter 6 discusses advantages and caveats of our approach and suggests future directions for this work.

CHAPTER 2
Related Work

There has been a lot of work in surface recovery from unorganized data sets over the years. Based on the two kinds of surface representations, there are two popular surface reconstruction approaches: explicit and implicit.

2.1 Explicit Surface Representations

Well-known explicit surface representation methods use either parametric surfaces or triangulated surfaces. Parametric surface reconstruction approaches use energy minimizing methods to constrain the class of possible solutions by fitting models whose topology is known, such that an error is minimized.

This section discusses popular surface reconstruction approaches that use explicit surface representations that globally parametrize the surface such as NURBS [22], superquadrics [27] as well as those that fit local surfaces such as the works of Sander and Zucker [24], Fua and Sander [11] and Mathur and Ferrie [17]. We also include discussions of works based on Delaunay triangulation and Voronoi diagrams like the Power Crust Algorithm of [2].

Surfaces reconstructed using parametric surfaces, such as NURBS, [15, 22] are quite smooth and these approaches work well for non-uniform data sets. However, the parametrization of the data set can be difficult for arbitrary data sets. It is also difficult to deal with noise in the data sets.

In [27], superquadrics with parametric deformations are used to recover compact volumetric models for single-part objects. Solina and Bajcsy in [27], use a least squares minimization of a cost function (which for example, could depend on the distance to data set) to recover the model. By enforcing constraints on the search in the parameter space, they arrive at a specific solution such that most of the points on the data set lie close to the model surface.

The idea of using algebraic curves and surfaces as geometric models or shape priors have also been explored by Taubin *et al.* in [28]. Algebraic curves and surfaces suffer from instability issues, despite having nice properties that make them suitable for object recognitions and positioning algorithms. This is because unlike the data sets which are always bounded, the resulting algebraic curves or surfaces are always unbounded. In [28] a method to constrain polynomials to a family with bounded zero sets is proposed and only this family is used for the fitting process. This approach works well as long as data is available i.e. they cannot "invent" data where it is not available. Another nice thing is that unlike superquadrics [27], they do not force structure onto objects which do not possess any. However, it is still not sufficient enough to capture surface detail.

The parametric approaches discussed above impose a global parametrization on the surface. There are other parametric surface recovery methods which fit local parametric patches to the data set like the works of Sander and Zucker [24] and Fua and Sander [11]. In [11] local quadric patches are fit to a small neighbourhood of each 3D point. These patches are then used to impose a graph structure by defining the points as connected components of a graph. The surface is then interpolated

using Delaunay triangulation. Similar work is done in [17] by Mathur and Ferrie, where a curvature consistency algorithm is used for surface recovery by first fitting local surface descriptors to each 3D point and then iteratively refining them so that each surface descriptor is consistent with its neighbours.

Delaunay triangulation and Voronoi diagrams are commonly used approaches in Computer Graphics to reconstruct triangulated surfaces from point sets. There has been a lot of work done in this line and there are several efficient algorithms to compute Delaunay triangulations and Voronoi diagrams. One such work based on Voronoi diagrams is the Power Crust algorithm of [2]. Given a set of sampled points from the surface of a 3D object, the power crust algorithm constructs a piecewise-linear approximation to both the object surface and the Medial Axis Transform (MAT). The medial axis transform or MAT is a skeletal shape representation that represents a solid by the set of maximal balls completely contained in the interior rather than the set of points in the boundary [2]. They use the sampled set of points to first approximate the MAT and then use inverse transformation on the MAT to produce a piecewise-linear surface approximation.

These approaches are capable of dealing with more general data sets. However, the reconstructed surfaces are only piecewise linear and it does not work well in the presence of noise and non-uniformity in the data.

2.2 Implicit Surface Representations

This section examines some work done in implicit surfaces or volumetric representations. The two main classes of approaches discussed here as the following: approaches that represent the reconstructed surface as an isocontour of an implicit

function and those that define a signed distance function based on the data set and then express the reconstructed surface as the zero isocontour of the distance function.

The approaches used in [18] and [29] fall into the first class where the reconstructed surface is expressed as an isocontour of an implicit function. In [18], Muraki uses primitives such as blobs and minimizes an energy function which measures the shape difference between the range data and the "blobby model". The approach in [29], however is more concerned with shape transformation between two N dimensional objects and thus they place constraints on the starting and final objects. Turk and O'brien [29] formulate the transformation between two N dimensional objects by casting it as a scattered data interpolation problem in N+1 dimensions. For a 2D shape, for example, they create an implicit function in 3D and intermediate shapes are zero isocontours of the implicit 3D function.

However these approaches suffer from high computation costs for large data sets since the construction is global and a single data point change can greatly affect the coefficients thus making deformation, incremental updates and human interaction difficult. In [5], polynomial radial basis functions (RBF) are used to model large data sets with single RBF which greatly reduces storage and computational costs.

Another class of approaches based on implicit surface representation uses the data set to define a signed distance function on a rectangular grid with the zero isocontour of the signed distance function representing the reconstructed implicit surface [9, 12]. In [3] alpha shapes are used to construct the signed distance function to which implicit polynomials are fitted. The representational power of such approaches is only limited by the size and resolution of the 3D grid in which they

are embedded and they must transformed back to parametric meshes for efficient storage [9].

Level-set approaches to surface reconstruction have been used in both [30] and [32]. Whitaker [30], uses a computational technique for level set modelling, called the sparse-field algorithm, that combines the advantages of the level-set approach with the computational efficiency and accuracy of parametric representation. The surface reconstruction problem is posed as the process of finding the surfaces that are most likely to have given rise to the data in a Maximum A Posteriori (MAP) strategy.

Zhao *et al.* use in [32] a new weighted minimal surface model based on variational and partial differential equations (PDE) methods. The flow used in their method is a special case of the conformal flow introduced in [6] and [14] where the distances to the data are used to weight various terms in the surface evolution equation. Their algorithm however is very dependent on a good initial approximation to the real surface.

2.3 KinectFusion

One important work in real-time 3D reconstruction is KinectFusion [13]. Izadi *et al* in [13] use a Kinect camera and a novel GPU pipeline to rapidly create detailed 3D reconstructions of indoor scenes. There is no explicit feature detection and the whole depth map is used. KinectFusion allows users to segment objects by having them physically move the object in a large, almost, unchanging scene. By detecting large changes in the 3D scenes when an object is physically moved, it allows users to cleanly segment objects from the scene. Then, they use a volumetric surface representation

based on [9] which raycasts the volume to extract views of the implicit surface for rendering to the viewer.

The surface reconstruction approach used in this thesis is based on active surfaces. The idea of using active contour based methods for implicit surface representation has been used earlier in [6] and [16]. The advantage of such active surface approaches is that the flows can be data driven, while being derived from first principles such as weighted length (2D) or surface area (3D) geometric flows [6, 14, 32]. Such weighted geometric flows have been used in [26], [6] and [14] for shape segmentation by adjusting them so that they cling to features of interest.

In [26], they introduce a gradient flow based on a weighted area functional with image dependent weighting factor. They combine this with the weighted length gradient flow of [6, 14] to obtain a PDE for the purpose of shape segmentation. The gradient flow used in this thesis is different from the one in [26] and uses only the weighted length component (or weighted surface area in 3D), for surface reconstruction. This is because although the weighted area component is robust in the presence of small gaps, in the presence of larger gaps the weighted area flow will leak through [26]. Because of the way we acquire point clouds (details in Chapter 3) as well as other reasons like self occlusion for example, we would expect the point cloud data to have large gaps. So the weighted length flow is more suited for the surface reconstruction method used in this thesis.

Motivated by these geometric flows and their demonstrated success in fitting point clouds as in [32], in this thesis we develop a method that uses a weighted surface area gradient flow to shrink wrap a dense point cloud obtained from a small

object, one that can be scanned by placing it on a desktop with a turntable. For such objects it is possible to obtain a dense point cloud such that a co-dimension 1 geometric flow is applicable.

In the next chapter, we introduce the incremental ICP algorithm used in this thesis to build 3D point cloud models of small objects.

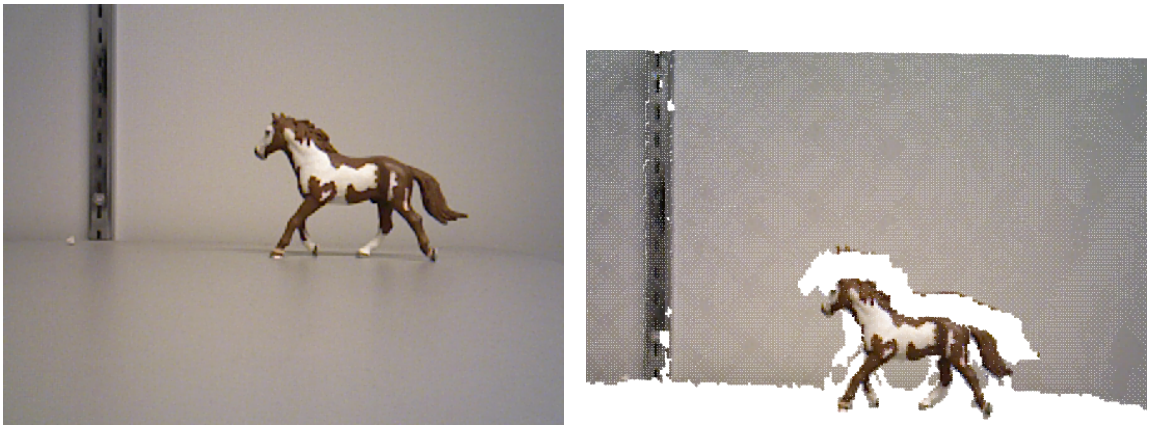
CHAPTER 3
Scanning dense 3D Points from a small object

We now consider the problem of building a dense 3D point cloud. First, we sense point clouds of surfaces of small objects using a RGBD camera. We then have to stitch these point clouds, taken from different viewpoints, to build a dense 3D point cloud of the sensed object.

3.1 Acquiring the data

The input is in the form of 3D Point clouds with xyz point values and RGB color values. The input point clouds are grabbed using the Primesense Carmine 1.09 short range sensor, using OpenNI [8], a multi-language, cross-platform framework for writing APIs to grab RGB and depth data from RGBD sensors. Figure (3–1) shows a single frame captured by the PrimeSense sensor.

The PrimeSense sensor is kept fixed at one side of the table and the object is placed at a distance of about 0.3m - 0.35m from the sensor and rotated manually by small amounts to capture a 360° view of the object. Since the background is unchanging, it provides no information to the registration and therefore is removed. Since the objects are sensed from a fixed distance, the object being sensed is bounded by certain dimensions. Thus the background can be eliminated by simply thresholding the depth values using a fixed parameter.



(a) RGB image of a single viewpoint (b) 3D Point Cloud of the same scene

Figure 3–1: RGB Image and 3D Point Cloud of a single viewpoint of the small model horse. The background is removed before registration

3.2 Registration of the Point Cloud: ICP

ICP or the Iterative Closest Point algorithm introduced in [4] is one of the most widely used methods to align 3D point clouds. This algorithm is used to minimize the distance between two point clouds [4]. To be precise, given two point clouds, the source and the target, ICP estimates the transformation matrix that would transform the source point cloud onto the target point cloud. The original ICP algorithm proposed in [4] can be used in many representations of geometric data such as point sets, line segment sets, implicit curves, parametric curves, triangle sets, implicit surfaces and parametric surfaces. However, regardless of what data sets are used for the data (source) and the model (target), the ICP algorithm proposed in [4] matches a point in the source data set to one of the points in the target data set, i.e., it minimizes a point-to-point error metric. This is done by pairing each point in the source data set with the closest point in the target data set to form correspondences. Then the point-to-point error metric is used to minimize the sum

of the squared distance between points in each pair of correspondences. This process is then iterated until the error falls below a threshold or it stops changing.

Since the introduction of the original ICP in [4], many variants have been introduced such as the one in [7] which minimizes a point-to-plane error metric as opposed to the point-to-point of [4]. These variants have been surveyed by Rusinkiewicz, Szymon and Levoy in [23].

In this thesis, we use the original ICP introduced in [4], one which minimizes point-to-point error metric and estimates the rigid transformation matrix using Singular Value Decomposition (SVD). The point-to-point metric used here takes into account both the distance between the points as well as the difference in colors. This section outlines the main steps of the ICP algorithm.

The first step of ICP is to estimate correspondences. For each point in the source point cloud, the closest point in the target point cloud is its corresponding point. Here "closest" point would mean the point in the target data set that has the smallest Euclidean distance from the point in the source cloud. The next step is to estimate the rigid transformation between the two point clouds using the correspondences. Let $\vec{q} = [\vec{q}_R|\vec{q}_T]$ be the complete registration state vector where $\vec{q}_R = [q_0 q_1 q_2 q_3]^t$ is the unit rotation quaternion and $\vec{q}_T = [q_4 q_5 q_6]^t$ is the translation vector [4]. Then if $P = \{\vec{p}_i\}$ is a source data set to be aligned with $X = \{\vec{x}_i\}$ a target data set, the mean objective function to be minimized as given in [4] is

$$f(\vec{q}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\vec{x}_i - \mathbf{R}(\vec{q}_R)\vec{p}_i - \vec{q}_T\|$$

where N_p is the total number of points in P and $N_p = N_x$ the total number of points in X. $\mathbf{R}(\vec{q}_R)$ is a 3×3 rotation matrix generated by the unit quartenion as given in

[4]:

$$\mathbf{R}(\vec{q}_R) = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2+q_2^2-q_1^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2+q_3^2-q_1^2-q_1^2 \end{bmatrix}$$

An effect of the least square solution, $f(\vec{q})$, is that the two point sets P and X should have the same centroid [10]. Therefore, we can define,

$$\bar{P} = \frac{1}{N_p}\sum_{i=1}^{N_p}\vec{p_i} \qquad \vec{p_{c_i}} = \vec{p_i} - \bar{P} \tag{3.1}$$

and

$$\bar{X} = \frac{1}{N_x}\sum_{i=1}^{N_x}\vec{x_i} \qquad \vec{x_{c_i}} = \vec{x_i} - \bar{X}. \tag{3.2}$$

$f(\vec{q})$ can be rewritten as,

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|\vec{x}_{c_i} - \mathbf{R}(\vec{q}_R)\vec{p}_{c_i}\| \tag{3.3}$$

$$= (\vec{x_{c_i}}^T\vec{x_{c_i}} + \vec{p_{c_i}}^T\vec{p_{c_i}} - 2\mathbf{R}\vec{p_{c_i}}) \tag{3.4}$$

The above equation is minimized when the last term is maximized, i.e., when $Trace(\mathbf{RH})$ is maximized [10]. Here, $\mathbf{H}$ is a correlation matrix given as

$$\mathbf{H} = \textstyle\sum_{i=1}^{N_p}\vec{p_{c_i}}\vec{x_{c_i}}^T$$

If the SVD of $\mathbf{H}$ is given by $\mathbf{H} = U\Lambda V$ then the optimal rotation matrix $\mathbf{R}$ can be calculated as

$$\mathbf{R} = VU^T \tag{3.5}$$

and the translation vector $\vec{q_T}$ can be computed as

$$\vec{q_T} = \bar{X} - \mathbf{R}\bar{P} \tag{3.6}$$

At each iteration ICP estimates the 4×4 transformation matrix T (see equation (3.7)) based on the rotation matrix and translation vectors as computed above and transforms the source point cloud using it. The ICP algorithm [4] converges when the mean square error falls below a preset threshold $\tau > 0$.

$$T = \begin{bmatrix} R_1 & R_2 & R_3 & q_4 \\ R_4 & R_5 & R_6 & q_5 \\ R_7 & R_8 & R_9 & q_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

In equation (3.7), the "R"s are elements of the rotation matrix and the last column is the translation vector previously defined.

It is also necessary to take into account the special case when the SVD computes a reflection instead of a rotation, i.e., the determinant of $\mathbf{R}$ is -1 instead of 1. This can happen when the two point sets are planar, or if there are large amounts of noise.

In this case, the rotation matrix $\mathbf{R}$ can be computed using the equation as $\mathbf{R} = V'U^T$ where $V' = [v_1, v_2, -v_3]$ is formed from the columns of V and v_3 is the column that corresponds to the singular value of $\mathbf{H}$ that is zero [10].

3.3 Construction of the Final PointCloud

The pairwise ICP is then implemented incrementally to stitch several pairs of point clouds taken from slightly different viewpoints. For each object around 25-30 views were used to generate a 3D point cloud. The basic steps are outlined below:

- For each point cloud in the dataset do
  - Pick one point cloud as source and the next one in the list as target.
  - Estimate correspondences between source and target
  - Compute the rigid transformation between source and target
  - Align source to target using the transformation matrix
  - Replace source with a new point cloud that fuses the transformed source and target.

Figure (3–2) shows the incremental ICP results for the humanoid object.

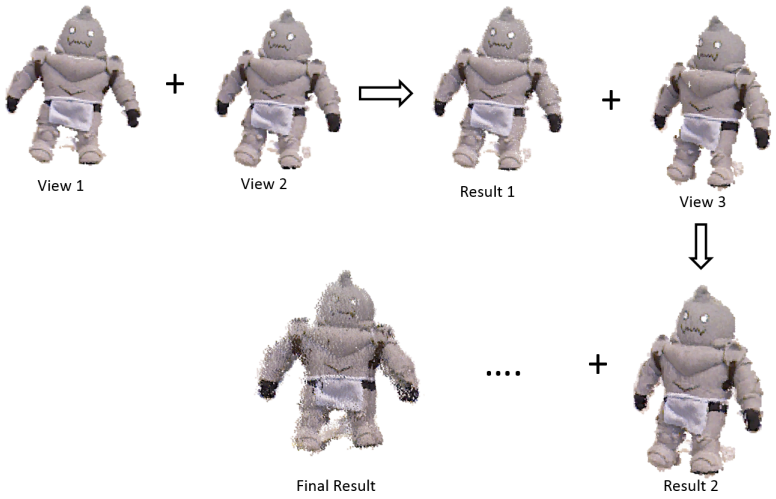
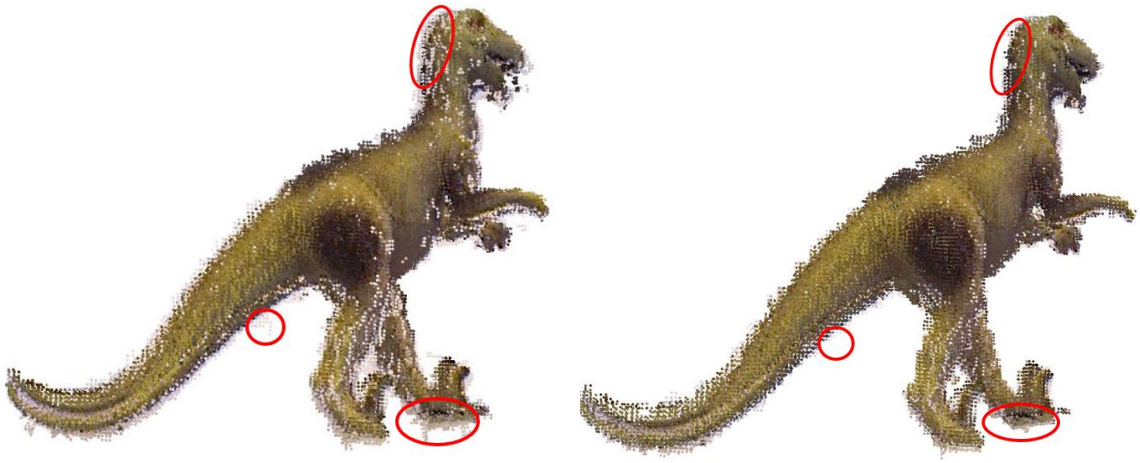


Figure 3–2: Incremental ICP



(a) Before Pruning (b) After Pruning

Figure 3–3: Pruning Strategy to remove outliers: The figure highlights some areas where our pruning method has successfully removed outliers.

One major issue with the sensing of point cloud data is the presence of outliers (can be seen in figure (3–3a)) in the regions where there is a sharp change in depth (the occluding contour) due to an artefact of the sensing device. In this thesis, we tried two methods to deal with these outlier points. In the first method, we use a "voting" strategy where we remove the outlier points as they are sensed by looking at several frames and picking only those points which are persistently present over all or most of the frames. For example, suppose we look at ten frames of the same scene. We pick the first frame and add all points in this scene. We then look at the next frame and add only those points which appear in both frames. We do this for all ten frames and finally pick only those points which appear in the majority of the frames considered.

In the second method, we "prune" the point cloud after the final reconstruction as a post processing step. We do this by first placing the point cloud in an octree. Each voxel in the octree is three-dimensional grid containing a small portion of the point cloud. We look at the voxel density (the number of points in a voxel) at each voxel of the octree and eliminate those with low density. The size of the voxel i.e. the resolution of the octree depends on the density of the final point cloud.

Both these methods, however, have their drawbacks. The former slows down the acquisition process because we are dynamically checking how many times each point appears over several frames and the latter is heuristic and depends on the density of the final point cloud. Also, since these outliers are caused due to the sensing device, they can sometimes persist over several frames and thus can be missed by both voting and pruning strategies. Despite these issues, using one or both of these methods, we have managed to get reasonably good dense point cloud data as can be seen in the figure.

CHAPTER 4
Shrink Wrapping a Surface

Once we have a dense 3D point cloud using the incremental ICP method of the previous chapter, we can consider the problem of shrink wrapping a surface around it using a geometric flow.

Before we discuss surface evolution we first introduce its two dimensional counterpart, the concept of front evolution or curve evolution in 2D.

4.1 Front Evolution

Suppose we have a closed curve in 2D or closed surface in 3D that separates two regions of $\Re$ from each other. If the curve or surface is evolving in the direction of its normal (the normal direction can be oriented towards the inside or outside) with a speed function F then the goal of front evolution is to track the motion of the interface (i.e boundary) as it evolves, given the initial position and the speed function F.

The speed function F may depend on local geometric properties of the evolving front such as the curvature or the normal, global properties that depend on the shape and position of the front and other independent properties like an external velocity field.

In general, if we have a closed curve C then a simple curve evolution equation is of the form,

$$\frac{\partial \vec{C}}{\partial t} = F\vec{N}. \tag{4.1}$$

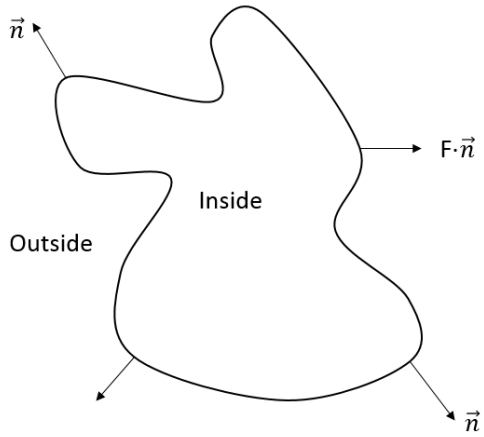


Figure 4–1: Curve propagating with speed F in the direction of the local normal (adapted from [25])

A straightforward approach to the front evolution problem would be to parameterize the interface by some variable and then discretize this parameterization into a set of "marker points". By approximating the equations of motions in time as the interface evolves, the positions of the marker points are updated. This approach is very accurate for small-scale motions of interfaces. However, for more complex motions of the interface,it suffers from numerical instabilities as the marker particles come together in regions of high curvature. The computed curvatures change drastically from one marker to the next causing an unstable growth of small errors in the positions of the markers [25].

The problems with marker methods discussed above deal with stability and local singularity. Another issue with these methods has to do with topological changes of the moving front. As regions on a plane merge together or pinch apart, their boundaries become a single curve or split into two or more curves. The task of

keeping track of the marker points, removing, redistributing or connecting them is complex, particularly for higher dimensions.

Another approach to front evolution is the "volume-of-fluid" technique which tracks the "interior" of an interface by discretizing the interior with a grid. Each cell on the grid is assigned a "cell fraction" corresponding to the amount of the cell within the interior of the grid. The original algorithm [19] was called "SLIC", i.e., "Simple Line Interface Algorithm". The idea is to evolve the front by updating the cell fractions on this fixed grid corresponding to the progress of the evolving front. This method has an advantage over the parameterized method in handling topological changes. However, there are some drawbacks. These techniques are inaccurate, requiring a large number of cells to obtain reasonable results. Also, it is difficult to calculate intrinsic properties of the front such as its curvature and the direction of the normal.

The surface evolution technique used in this thesis is based on the notion of level sets for front evolution introduced by Osher and Sethian in [21]. The main idea of the level set methodology is to embed the evolving front as the zero level set of a higher dimensional function Ψ. The next section introduces the concept of level sets for front evolution.

4.1.1 The Level Set Formulation for Front Evolution

Let $\Gamma(t)$ be a family of closed $(N-1)$ dimensional hyperspaces parameterized by t, moving along the direction of the inward normal with speed F. Here F can be a function of curvature, normal direction and other quantities. As mentioned above,

the idea of level set front evolution is to embed Γ as the zero level set of a higher dimensional function Ψ [25].

We define Ψ such that $\Psi(x, t = 0) = \pm d$ where x is a point in $\Re^N$ and d is the Euclidean distance from x to $\Gamma(t = 0)$. d is positive if the point x is outside the initial hypersurface $\Gamma(t = 0)$ and negative if it is inside. The initial function $\Psi(x, t = 0) : \Re^N \rightarrow \Re$ is defined as:

$$\Gamma(t = 0) = [x|\Psi(x, t = 0) = 0]. \tag{4.2}$$

We now have to produce an equation for the evolving function $\Psi(x, t)$ such that the level set $\Psi = 0$ contains the embedded function $\Gamma(t)$. Suppose we consider the path, $x(t)$, of a point, x, on the evolving front. As mentioned earlier, we have to ensure that the zero level set of Ψ matches the propagating hypersurface. This means that,

$$\Psi(x(t), t) = 0 \tag{4.3}$$

By the chain rule,

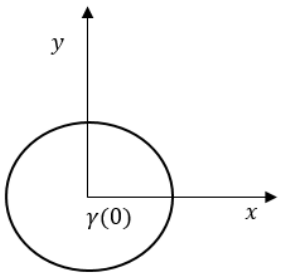
$$\Psi_t + \nabla\Psi(x(t), t) \cdot x'(t) = 0 \tag{4.4}$$

If $F = x'(t) \cdot \vec{n}$ is the speed in the direction of the outward normal and $\vec{N} = \nabla\Psi/\|\nabla\Psi\|$ is the outward normal then we can write,

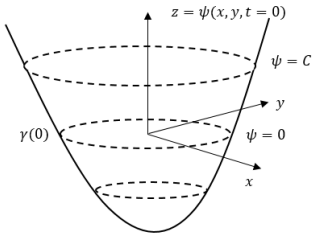
$$\Psi_t + F\|\nabla\Psi\| = 0 \tag{4.5}$$

given $\Psi(x, t = 0)$.

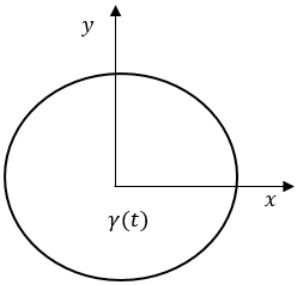
Figure (4–2) illustrates the outward propagation of an initial curve and the motion of the level set function Ψ that accompanies it. (Figure adapted from [25]).



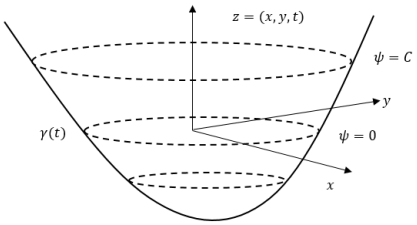
(a) Initial Circle



(b) Initial Position of level set function Ψ



(c) Circle at later time t



(d) Position of level set function Ψ at later time t

Figure 4–2: Propagating Circle: This figure illustrates the outward propagation of an initial curve and the accompanying motion of the level set function Ψ (adapted from [25])

There are several advantages to this level set formulation. First, as long as F is smooth, the evolving function $\Psi(x,t)$ will remain a function. This means that the level set surface $\Psi = 0$ and in turn the propagating hypersurface $\Gamma(t)$ may change topology, i.e., break or merge and form sharp corners as the function evolves.

Second, it is easy to determine intrinsic properties of the front such as the normal and curvature from the level set function Ψ, as follows. The unit outward normal to the front is given by,

$$\vec{N} = \frac{\nabla \Psi}{\|\nabla \Psi\|} \tag{4.6}$$

and the curvature of each level set can be given as the divergence of the unit outward normal as follows,

$$\kappa = \nabla \cdot \frac{\nabla \Psi}{\|\nabla \Psi\|}. \tag{4.7}$$

Since the level set function Ψ remains a function as it evolves, it is also easy to numerically discretize the function using finite difference schemes to approximate both the spatial and the temporal derivatives. For example, if we use a forward first order in time difference scheme then we can discretize equation (4.5) as follows:

$$\frac{\Psi_{ij}^{n+1} - \Psi_{ij}^{n}}{\Delta t} + (F)\|\nabla_{ij} \Psi_{ij}^{n}\| = 0 \tag{4.8}$$

where $\|\nabla_{ij} \Psi_{ij}^{n}\|$ is the finite difference approximation of the spatial derivatives of Ψ.

Finally, with the level set formulation, it is very simple to extend the approach to the case of surfaces evolving in 3D as will be covered in the next section.

4.2 Surface Evolution

In this section, we discuss the level set front evolution in three dimensions and introduce the specific weighted gradient flow used in this thesis. This is the weighted

surface area minimizing flow, which is used in [14], [6] and [26] for shape segmentation. Such flows when applied to 3D point cloud data could have weights that depend upon local distance to 3D points [6, 14, 32].

If $\vec{S}$ is the surface to be evolved, then the general form of the evolution equation is the same as the curve evolution equation of (4.1)

$$\frac{\partial \vec{S}}{\partial t} = F\vec{N} \tag{4.9}$$

where the surface $\vec{S}$ moves in the direction of its normal $\vec{N}$ and F is the speed function.

Given that $\vec{S}$ evolves according to (4.9), the level set surface Ψ satisfies the level set equation:

$$\frac{\partial \Psi}{\partial t} = F \left\| \nabla \Psi \right\|. \tag{4.10}$$

This equation can be solved by discretization and numerical techniques based on hyperbolic conservation laws.

4.2.1 Weighted Gradient Flows

The weighted gradient flow used in this paper is based on the "Weighted Surface Area Gradient Flow" in [26].

Let $S : [0,1] \times [0,1] \rightarrow \Re^3$ denote a compact embedded surface with mean curvature κ and unit inward normal $\vec{N}$.

Let $\phi : \Omega \rightarrow \Re$ be a positive differentiable function defined on an open subset of $\Re^3$. This function can be considered as "stopping" function. Here we define ϕ as a scalar function of location in 3D space, which plays the role of a weight. In practice,

ϕ, is an unsigned outward distance function with each voxel containing the distance to the closest point in the sensed 3D point cloud.

The weighted surface area flow derived in[6] and [14] is given as:

$$\frac{\partial \vec{S}}{\partial t} = \phi \kappa \vec{N} - \nabla \phi. \tag{4.11}$$

Equation (4.11) can be written in level set form as:

$$\Psi_t = \left[\phi \kappa + \left\langle \frac{\nabla \Psi}{\|\nabla \Psi\|}, \frac{\nabla \phi}{\|\nabla \phi\|} \right\rangle \right] \|\nabla \Psi\| . \tag{4.12}$$

The geometric flow in equation (4.12) has two main terms. The first is the "$\phi\kappa$" term and the second is a "doublet" term that has the advantage of trapping an evolving surface at a local minimum of ϕ because it switches its sign where that occurs.

4.2.2 Numerical Implementation

We now review some of the key numerical steps that are needed to implement the surface evolution. We first need an arbitrary initial surface that contains the data. Since we do not know the topology of the surface to be reconstructed in advance, we use a bounding box that is known to contain the data set. In practice, the level set function Ψ is set up as a signed distance function to the data, where distances within the box are negative and distances outside are positive. As mentioned in the previous section, ϕ is the outward distance function containing distances to the point cloud data.

The two terms of equation (4.12) also have to be carefully computed. The mean curvature κ can be calculated using the equation (4.7) in the previous section as

follows.

$$\kappa = \nabla \left(\frac{\nabla \Psi}{\|\nabla \Psi\|} \right) \tag{4.13}$$

This expands to,

$$\kappa = \frac{1}{\|\nabla \Psi\|^3}(\Psi_x^2 \Psi_{yy} - 2\Psi_x \Psi_y \Psi_{xy} + \Psi_y^2 \Psi_{xx} + \Psi_x^2 \Psi_{zz} - 2\Psi_x \Psi_z \Psi_{xz} + \Psi_z^2 \Psi_{xx} + \Psi_y^2 \Psi_{zz} - 2\Psi_y \Psi_z \Psi_{yz} + \Psi_z^2 \Psi_{yy}) \tag{4.14}$$

Here, the gradient of Ψ is calculated using central differences everywhere except at the boundaries where one-sided differencing is used. The "doublet" term is slightly more complicated. Unlike in the case of the "$\phi\kappa$" term, the computation of spatial derivatives for Ψ, is not as straightforward and we have to use an upwind differencing scheme to calculate it. This is because this term can lead to singularities in the evolving surface. The upwind differencing scheme given here is based on [20] and [25]. A first order time discretization of the equation (4.10) using the forward Euler method is given as:

$$\frac{\Psi_{t+1} - \Psi_t}{\Delta t} = F_t \|\nabla \Psi_t\| \tag{4.15}$$

where F_t is the speed at time t and $\nabla \Psi_t$ is the spatial gradient of Ψ at time t. The gradient term in equation (4.15) can be expanded to

$$\frac{\Psi_{t+1} - \Psi_t}{\Delta t} = F_x^t \Psi_x^t + F_y^t \Psi_y^t + F_z^t \Psi_z^t. \tag{4.16}$$

This expression can be evaluated in each dimension (x, y and z) separately. Suppose we consider one dimension $F_x^t \Psi_x^t$ where the sign of F_x^t along x determines the direction of flow.

At each grid point i, we determine the spatial derivative $(\Psi_x)_i$, at i based on the sign of $(F_x)_i$ at i. As given in [20], if $(F_x)_i > 0$, the values of Ψ are propagating from left to right which implies that, according to the method of characteristics, we have to look to the left of the point x_i to determine the value of Ψ at x_i at the end of the time step, Δt, (i.e., we use the first order accurate backward difference, D^{-x}). Likewise if $(F_x)_i < 0$ then the information is flowing from right to left and we have to use the first order forward difference denoted as D^{+x} The same approach is taken for spatial derivatives Ψ_y and Ψ_z. This method of choosing spatial derivatives based on the sign of the speed function is called *upwinding* or *upwind differencing*. In summary, equation (4.15) can be approximated with simple upwinding using a first-order space convex [25] as:

$$\Psi_{t+1} = \Psi_t + \Delta t[\max(F(i,j,k),0)\nabla^+ + \min(F(i,j,k),0)\nabla^-] \tag{4.17}$$

where,

$$\begin{aligned} \nabla^+ = [&\max(D^{-x},0)^2 + \min(D^{+x},0)^2 + \max(D^{-y},0)^2 \\ &+ \min(D^{+y},0)^2 + \max(D^{-z},0)^2 + \min(D^{+z},0)^2]^{1/2} \end{aligned} \tag{4.18}$$

and

$$\begin{aligned} \nabla^- = [&\max(D^{+x},0)^2 + \min(D^{-x},0)^2 + \max(D^{+y},0)^2 \\ &+ \min(D^{-y},0)^2 + \max(D^{+z},0)^2 + \min(D^{-z},0)^2]^{1/2} \end{aligned} \tag{4.19}$$

where, D^+ and D^- are forward and backward finite differences as mentioned previously.

The complete algorithm to implement the weighted geometric flow of equation (4.11) is given in algorithm (1). In order to speed up the flow, the "$\phi\kappa$" term is

Algorithm 1: Level Set Surface Evolution.

for *each iteration* t **do**
  Estimate the "$\phi\kappa$" term
  Estimate the doublet term, $\left\langle \frac{\nabla\Psi}{\|\nabla\Psi\|}, \frac{\nabla\phi}{\|\nabla\phi\|} \right\rangle$.
  for *each voxel* (i, j, k) **do**
    if $\phi(i, j, k) < threshold$ **then**
      speedTerm $= \phi\kappa + \left\langle \frac{\nabla\Psi}{\|\nabla\Psi\|}, \frac{\nabla\phi}{\|\nabla\phi\|} \right\rangle$
    else
      speedTerm $= \left\langle \frac{\nabla\Psi}{\|\nabla\Psi\|}, \frac{\nabla\phi}{\|\nabla\phi\|} \right\rangle$
    end
  end
  Update equation: $\Psi_{t+1} = \Psi_t + \Delta t(speedTerm)$
end

added only when the level set surface gets within *threshold* voxels of the data set.

CHAPTER 5
Experiments and Results

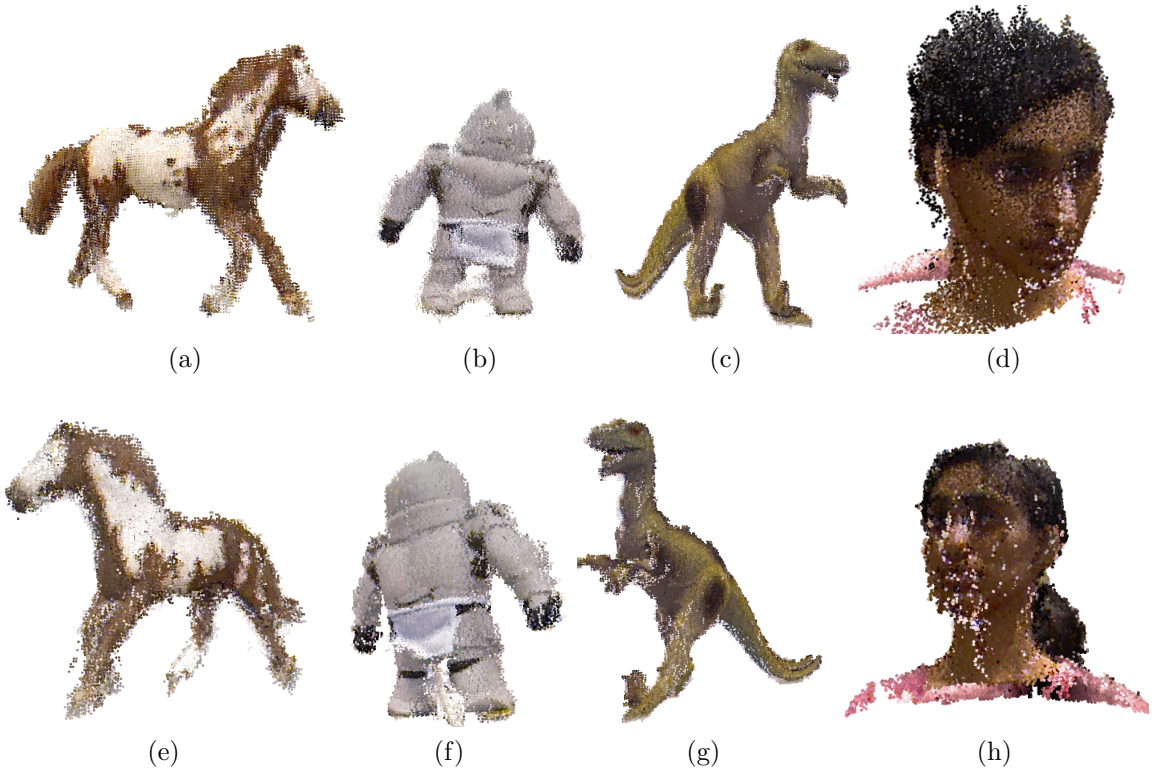
In this chapter, we present experiments demonstrating the application of the ICP method to generate dense point clouds of small objects, followed by surface evolution to obtain surface and appearance models of them. The results of our "shrink wrapping" approach are presented on both a prototype point cloud, obtained from sampling points from a mesh object, and real data obtained from ICP method.

We also present color plots to show how accurately our method represents the point cloud data. A qualitative comparison of our results against that of a commercial tool, Skanect, is also presented.

5.1 Results of the ICP

In this section, we present the results of our incremental ICP algorithm to build dense 3D point clouds from RGBD data obtained from several views of scanning the small object.

Figure (5–1) shows two views of the final 3D point cloud obtained for 4 objects, a small model horse, a humanoid object, a model dinosaur and a human head. As can be seen in the figure, due to way in which we capture point clouds, there are some holes in our 3D point cloud model (such as the back of the horse or the top of the head). Although this can be rectified by obtaining new views by moving the camera, it is not needed for our surface evolution method because of the way the



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5–1: The results of our ICP method. The top row shows registered dense point clouds for 4 objects: (5–1a) a horse, (5–1b) a humanoid object, (5–1c) a dinosaur and (5–1d) a human head. The bottom row shows different views for the same object.

surface evolution equation (4.12) is set up. This is demonstrated in the following sections.

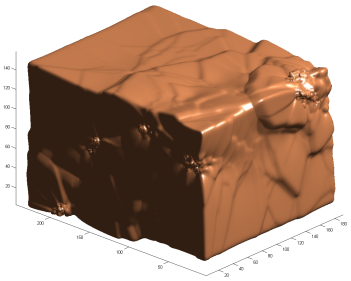
5.2 Results of the "shrink wrapping" flow

This section reviews the results of our shrink wrapping flow. We first illustrate the stages of our surface evolution method and then present some results on the 4 objects.

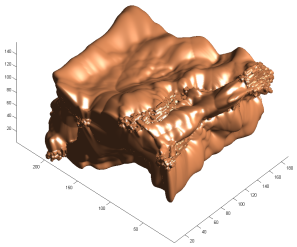
Figure (5–2) illustrates the stages of surface evolution applied to a point cloud obtained from a model horse. As explained in Algorithm (1), in order to speed up the flow, until the level set surface Ψ gets "close" (i.e within a distance of *threshold* voxels) to the data, the speed function only contains the "doublet" term (Figures (5–2a), (5–2b) and (5–2c)). Figure (5–2d) shows the result of including the "$\phi\kappa$" term as well. In all experiments the value of the *threshold* is 2 voxels and the time step Δt is 0.5 when the speed term is only the doublet term and is 0.1 when the "$\phi\kappa$" term is also included. These constants are in units of voxels. As it can be seen in figure (5–2), our surface reconstruction method effectively fills holes like the top of head or the back of the horse for example because of the "doublet" term trapping the flow at these locations. The last row shows results of applying a mean curvature smoothing flow. This is done as a post processing step to smooth the bumps in the zero isocontour of Ψ.

5.2.1 Results on a prototype point cloud

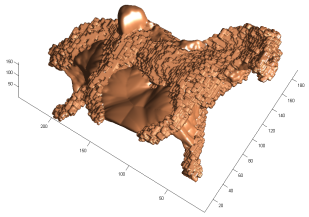
We first demonstrate the results of our surface evolution method on a prototype bull point cloud. This point cloud has been obtained by densely sampling points on



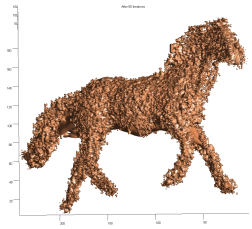
(a) After 25 iterations



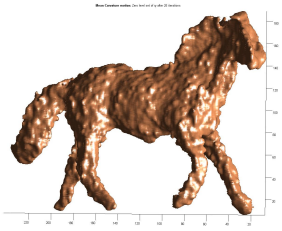
(b) After 100 iterations



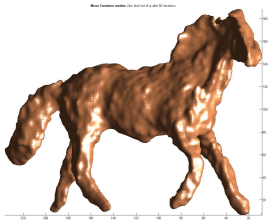
(c) After 250 iterations



(d) After 40 iterations of the combined flow (i.e., including the "$\phi\kappa$" term)

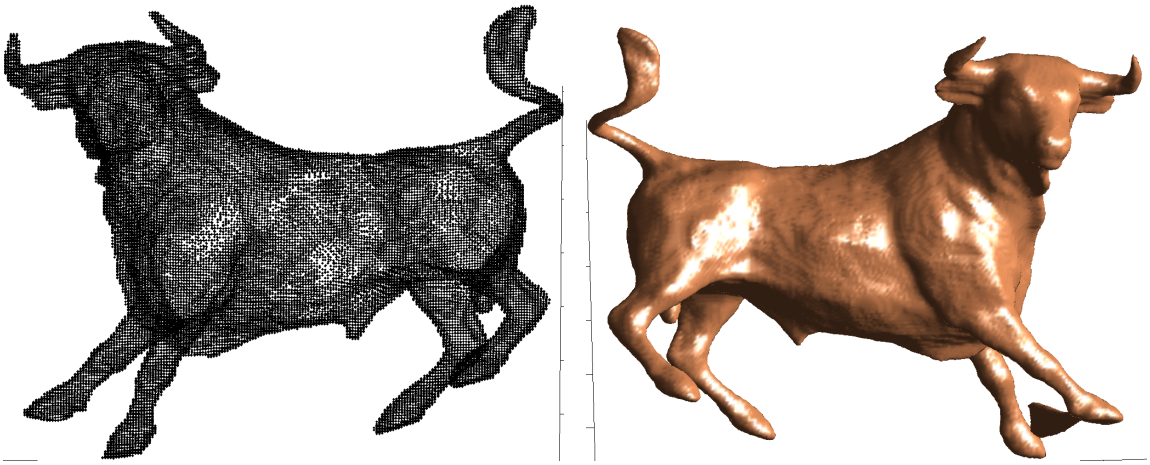


(e) After 25 iterations of mean curvature flow



(f) After 50 iterations of mean curvature flow

Figure 5–2: An illustration of various stages of surface evolution. Panels (5–2a), (5–2b) and (5–2c) show the level set surface Ψ, after 25, 100 and 250 iterations of running the weighted gradient flow of equation (4.11) flow on the model horse point cloud but with only the "doublet" term. This is done to speed up the flow. Panel (5–2d) shows the result after a further 40 iterations of running the flow with the "$\phi\kappa$" term included. The last row shows results of a mean curvature smoothing flow applied as a post processing step.



(a) Bull Point Cloud. (b) Reconstructed Surface

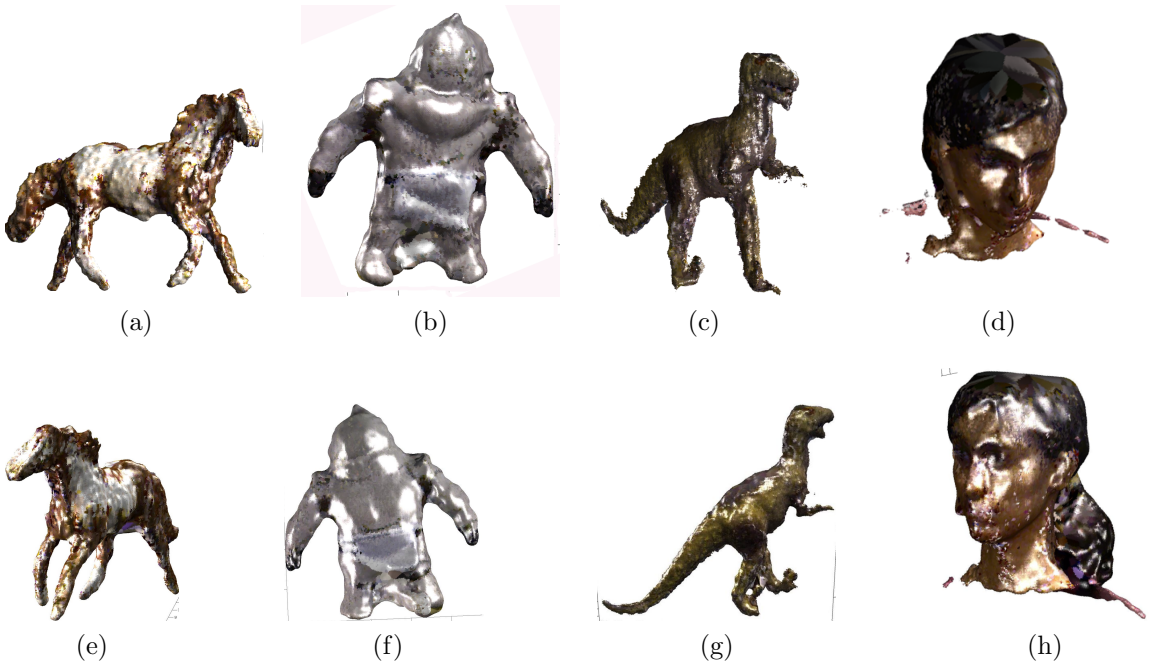
Figure 5–3: Results on a prototype point cloud data. This point cloud was obtained by sampling points on a mesh. The reconstructed surface was smoothed using the mean curvature flow.

a mesh as can be seen in figure (5–3a). The result shown in figure (5–3b) is obtained after smoothing using few iterations of the mean curvature smoothing flow.

As it can be seen in the figure, our method works extremely well if the point cloud is dense. The result picks up a lot of shape and curvature detail in the body of the bull and the shape of its legs and tail, for example.

5.2.2 Results on registered 3D point clouds

Figure (5–4) the results of our shrink wrapping approach on the registered point clouds of (5–1). The shrink wrapping results reveal the level of surface detail that can be recovered. The results are shown after post process smoothing and patching the colors. The smoothing was done by running a few iterations of the mean curvature flow as seen in the bottom row of figure (5–2). The advantage in using this flow for smoothing over a simple Gaussian is that, the mean curvature flow smooths along



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5–4: The results of our surface reconstruction method. These results are shown after few iterations of mean curvature smoothing flow. The bottom row shows different views of the same result.

the curvature, thereby preserving the surface shape details. However, too many iterations of this flow would cause regions of high curvature to break off. Thus there is a trade off between the smoothness and the level of detail acquired.

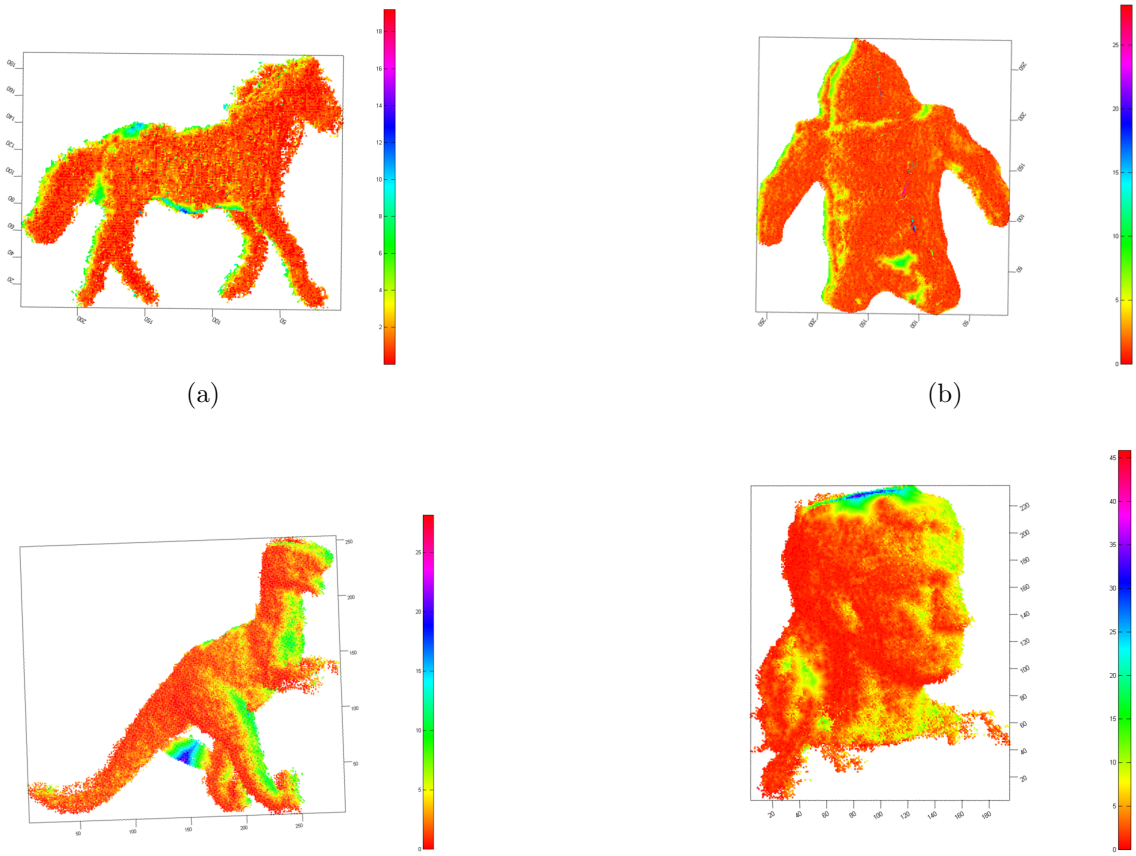
5.3 Quantitative and Qualitative summary of results

Table 5–1: Quantitative Results

No Of Views	**Object**	**Dimensions** (voxels)	**Avg. Distance** (voxels)	σ
25	Horse	$244 \times 193 \times 160$	4.4171	4.0038
25	Humanoid	$264 \times 286 \times 97$	6.4073	5.8649
30	Dinosaur	$289 \times 257 \times 144$	5.9817	5.7692
39	Human Head	$199 \times 236 \times 288$	10.1955	9.7089

The results in figure (5–5) show the distance of each point on the shrink wrapped surface to the closest point on the point cloud in voxel units, as explained by the color bars. These plots demonstrate the accuracy of the method, particularly at locations where 3D point cloud data has been sensed, with the regions of larger error typically corresponding to locations where the point cloud data is sparse or is missing, such as the top of the human head. A quantitative summary of these results for the various models is shown in Table 5–1.

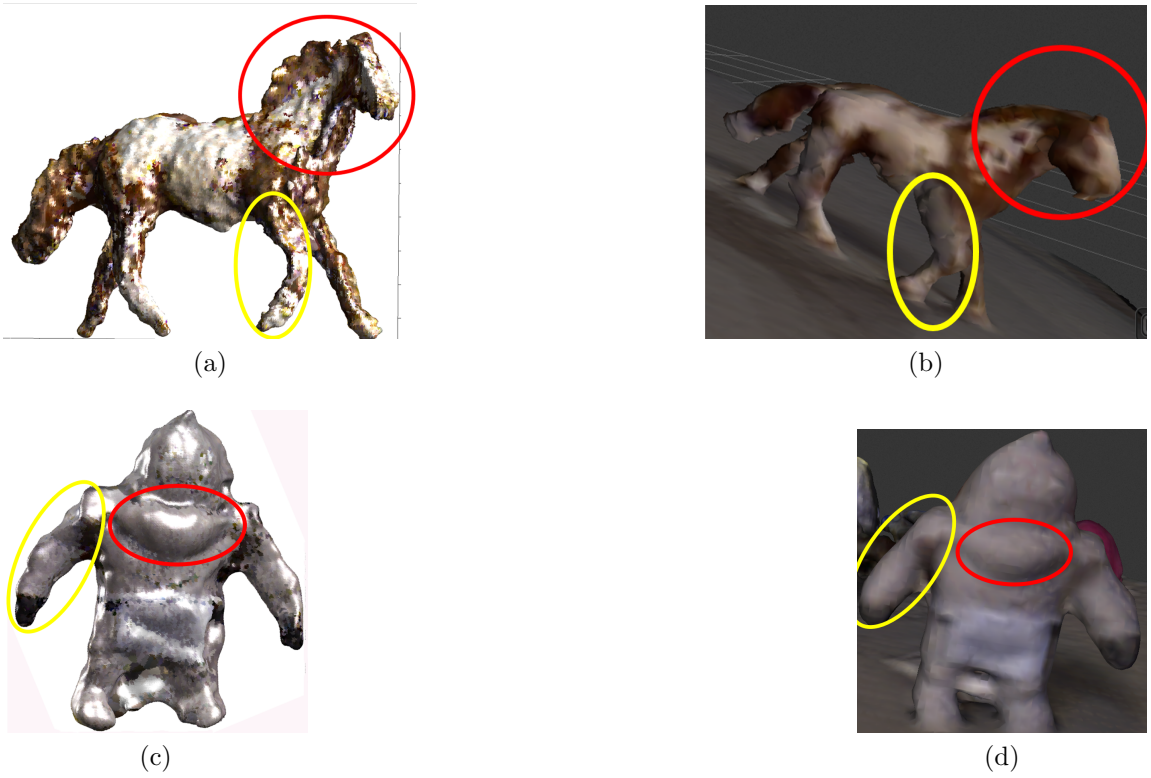
Figure (5–6) provides a qualitative comparison of the results obtained by our approach and those obtained using the commercial software package Skanect. [1].



(a) (b)

(c) (d)

Figure 5–5: This figure shows the distance errors summarized in Table 5–1 as color plots.



(a) (b) (c) (d)

Figure 5–6: Qualitative comparison of our results against those of Skanect [1]. The first column shows the results of our surface evolution method. The second column depicts screen shots of results obtained using Skanect. Our method picks up more detail in the mane of the horse and the shape of the face and leg. It also better recovers the curvature of parts of the body of the humanoid object.

CHAPTER 6
Conclusion and Future Work

In this thesis, we demonstrated the use of weighted geometric flows to reconstruct surfaces using registered point clouds of 3D data obtained from RGBD sensors. We first used a simple ICP algorithm to stitch together several views of an object obtained from an RGBD Sensor. We then used a specific geometric flow, the "Weighted Surface Area Minimizing Flow" and adjusted it to cling on to the points on the point cloud surface.

Our experimental results show the level of surface detail that can be captured using this approach. We also presented distance plots that show how accurately our geometric flow based method is able to reconstruct the 3D point cloud. As figure (5–6) shows, the result of our reconstruction is qualitatively comparable to that of Skanect.

The main advantages of our method is that it captures better surface detail as seen in figure (5–6). It also shows some robustness to holes because of the doublet term trapping the flow. Thus, we are able to create watertight models even if some data is missing. Since we use a surface evolution method, we have good estimates of surface normals and mean curvature which as demonstrated in figure (5–4) has been used for post processing to smooth using the mean curvature flow.

The main caveat of the method is that we don't explicitly deal with smoothing noisy 3D point clouds; we have assumed that outliers have been eliminated and that

the point cloud is dense, which is a fair assumption for RGBD sensors particularly given that new views can easily be acquired to fill in regions with missing data. Also, the ICP algorithm used in this thesis was the standard algorithm introduced in [4]. Some pre-processing and/or post-processing (voting and pruning strategy, ref. Chapter 3) were done to reduce noise in the final constructed point cloud. One way to improve registration could be incorporating some of the more efficient variants of ICP [23], to weight correspondences and/or reject bad ones. This might eliminate the need for the pre and post processing strategies to obtain higher quality point cloud data.

In this thesis, we evaluate the accuracy of our surface evolution method in terms of the distance errors with respect to the registered point cloud. These results are summarized in table 5–1 along with the error color plots shown in figure (5–5). A drawback in the present thesis is a lack of comparison against other surface reconstruction algorithms. We could run our surface evolution method on standard data sets and compare in detail the recovered models against those obtained from other surface reconstruction algorithms. This could be a fruitful direction for future work.

Some applications of surface reconstruction are 3D printing and object recognition. It would be interesting to see how well the model generated using our shrink wrapping approach would perform in these applications. We could perhaps, generate a database of many objects for such applications. Our approach also provides accurate 3D curvature estimates that can be used in shape analysis and object recognition tasks.

It would also be interesting to see how well our approach would work if we used regular RGB cameras and computed depth values from the 2D images to generate 3D point clouds. This would circumvent some issues of the RGBD sensor such as dealing with transparent and/or reflective objects.

References

[1] Skanect — http://skanect.occipital.com/.

[2] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM, 2001.

[3] Chandrajit L Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 109–118. ACM, 1995.

[4] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[5] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001.

[6] Vicent Caselles, Ron Kimmel, Guillermo Sapiro, and Catalina Sbert. Minimal surfaces: A geometric three dimensional segmentation approach. *Numerische Mathematik*, 77(4):423–451, 1997.

[7] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729. IEEE, 1991.

[8] OpenNI Consortium et al. Openni, the standard framework for 3d sensing.

[9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[10] David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290, 1997.

[11] Pascal Fua and Peter Sander. Reconstructing surfaces from unstructured 3d points. In *Second European Conference on Computer Vision (ECCV90)*. Citeseer, 1992.

[12] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

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

[14] Satyanad Kichenassamy, Arun Kumar, Peter Olver, Allen Tannenbaum, and Anthony Yezzi. Gradient flows and geometric active contour models. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 810–815. IEEE, 1995.

[15] Piegl Les and Tiller Wayne. The nurbs book. *Springer Verlag, Berlin*, 1997.

[16] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(2):158–175, 1995.

[17] Shailendra Mathur and Frank P Ferrie. Edge localization in surface reconstruction using optimal estimation theory. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 833–838. IEEE, 1997.

[18] Shigeru Muraki. Volumetric shape description of range data using blobby model. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 227–235. ACM, 1991.

[19] William F Noh and Paul Woodward. Slic (simple line interface calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in*

Fluid Dynamics June 28–July 2, 1976 Twente University, Enschede, pages 330–340. Springer, 1976.

[20] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2003.

[21] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. volume 79, pages 12–49. Elsevier, 1988.

[22] David F Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2000.

[23] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[24] Peter T Sander and Steven W Zucker. Inferring surface trace and differential structure from 3-d images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(9):833–854, 1990.

[25] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[26] Kaleem Siddiqi, Yves Bérubé Lauziere, Allen Tannenbaum, and Steven W Zucker. Area and length minimizing flows for shape segmentation. *Image Processing, IEEE Transactions on*, 7(3):433–443, 1998.

[27] Franc Solina and Ruzena Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(2):131–147, 1990.

[28] Gabriel Taubin, Fernando Cukierman, Steven Sullivan, Jean Ponce, and David J Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(3):287–303, 1994.

[29] Greg Turk and James F O'brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses*, page 13. ACM, 2005.

[30] Ross T Whitaker. A level-set approach to 3d reconstruction from range data. *International journal of computer vision*, 29(3):203–231, 1998.

[31] Jeffrey L Wilson. Microsoft kinect for xbox 360. *PC Mag. Com*, 2010.

[32] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *Variational and Level Set Methods in Computer Vision, 2001. Proceedings. IEEE Workshop on*, pages 194–201. IEEE, 2001.

[33] Steven W Zucker. *Early vision.* McGill University, Computer Vision and Robotics Laboratory, 1986.