3D Surface Analysis for the Automated Detection of Deformations on Automotive Panels

By:

Arjun Yogeswaran

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements for the degree of

Master of Applied Science in

Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School for Information Technology and Engineering

University of Ottawa

April 2011

Abstract

This thesis examines an automated method to detect surface deformations on automotive panels for the purpose of quality control along a manufacturing assembly line.

Automation in the automotive manufacturing industry is becoming more prominent, but quality control is still largely performed by human workers. Quality control is important in the context of automotive body panels as deformations can occur along the assembly line such as inadequate handling of parts or tools around a vehicle during assembly, rack storage, and shipping from subcontractors. These defects are currently identified and marked, before panels are either rectified or discarded. This work attempts to develop an automated system to detect deformations to alleviate the dependence on human workers in quality control and improve performance by increasing speed and accuracy.

Some techniques make use of an ideal CAD model behaving as a master work, and panels scanned on the assembly line are compared to this model to determine the location of deformations. This thesis presents a solution for detecting deformations of various scales without a master work. It also focuses on automated analysis requiring minimal intuitive operator-set parameters and provides the ability to classify the deformations as dings, which are deformations that protrude from the surface, or dents, which are depressions into the surface.

A complete automated deformation detection system is proposed, comprised of a feature extraction module, segmentation module, and classification module, which outputs the locations of deformations when provided with the 3D mesh of an automotive panel. Two feature extraction techniques are proposed. The first is a general feature extraction technique for 3D meshes using octrees for multi-resolution analysis and evaluates the amount of surface variation to locate deformations. The second is specifically designed for the purpose of deformation detection, and analyzes multi-resolution cross-sections of a 3D mesh to locate deformations based on their estimated size. The performance of the proposed automated deformation detection system, and all of its sub-modules, is tested on a set of meshes which represent differing characteristics of deformations in surface panels, including deformations of different scales. Noisy, low resolution meshes are captured from a 3D acquisition, while artificial meshes are generated to simulate ideal acquisition conditions. The proposed system shows accurate results in both ideal situations as well as non-ideal situations under the condition of noise and complex surface curvature by extracting only the deformations of interest and accurately classifying them as dings or dents.

Acknowledgements

First and foremost, I am genuinely thankful to my academic supervisor, Dr. Pierre Payeur, whose help, patience, and understanding throughout my time as a Masters student had no threshold. I appreciate him accepting me as a student and allowing me to work in my own way while conducting research. This thesis truly would not have been possible without him.

I would like to thank my colleague and friend, Valentin Borsu, who worked alongside me in research, classes, conferences, and publications. Though I did not spend much time in the lab, Alain Boyer, Phillip Curtis, and Fouad Khalil all provided me with valuable knowledge which I appreciate.

Also, I am grateful for the financial support from Precarn Inc., along with the collaboration between Neptec Design Group Ltd. and Honda Canada for this research.

For the people outside of school, I would like to thank all of my friends for being supportive of me pursuing my education, especially Koli Basu and Ben Miller who I inadvertently dragged into my world of research. Finally, I would like to thank my mother and father for their continued support through all my years of hard work. I cannot express how much I appreciate everything they have done for me.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

1.1 Context and Motivation

Automation in the manufacturing industry has become quite commonplace in the past few decades, as evidenced by the major role of computers and robotics on assembly lines. This automation has provided a notable increase in the speed and efficiency of production in various fields ranging from semiconductors to pharmaceuticals. With this increase in speed and efficiency of production, quality control of these automatically manufactured products must also increase in both speed and accuracy. The role of a human worker in quality control is to identify potential inconsistencies and defects on the manufactured products, and to ensure that those defective parts are dealt with, either by being discarded or rectified. As the speed of production accelerates, the human worker often becomes the limiting factor in speed, accuracy, and consistency.

The automotive industry is working to automate more and more of their production lines, including quality control stations that currently employ human workers to visually identify and mark defects over unpainted automotive body parts. Certain techniques aid the worker with their visual inspection, such as shining a light at different orientations along the surface such that the reflection in areas of deformations will stand out compared to the reflection along the smooth areas of the panel. This time-consuming process can be difficult for a human, especially when dealing with small deformations that require close inspection, and may result in a decreased accuracy when the repetitive task is performed over the course of an entire shift. Automation of quality control could significantly improve the accuracy and speed of the assembly line, thus increasing the number of panels inspected within an allotted time, maximizing the number of accurately detected defects, and minimizing the number of falsely detected defects.

To sufficiently automate this process, a system would have to analyze the surface of the body part to be inspected, determine the position of deformations, and mark those deformations on the body part. An automated approach could use an acquisition system to capture the surface of the automotive panel in 3-dimensions. Using a computer to analyze the 3D data, the locations of the deformations can be found through the use of surface shape analysis techniques. Finally, a robotic marking system would mark those locations on the panel itself.

This thesis focuses on the analysis of 3D surfaces and automatic detection of deformations. When provided with a 3D mesh of the automotive panel, the system should output the locations of the deformations with minimal operator interaction. One of the imposed

requirements of this system is that it must be able to detect deformations without knowledge of the ideal shape of the part, meaning it cannot use a master work or CAD model for comparison. Some automated deformation detection techniques focus on the difference between the scanned model and an existing ideal model or master work [1, 2]. By performing a subtraction between the two models, all similar areas can be removed while leaving only the deformations behind. Certain challenges lie within this approach. The first constraint is that a very precise ideal model must be used, because small faults in the master work can result in incorrectly identified defects during execution. The second challenge, and arguably the most important one, is related to the registration of the scanned part with the ideal model. Due to vibrations on the assembly line and slight inconsistencies in the acquired model, inaccurate registration may occur resulting in a subtraction that incorrectly identifies defects. Also, if panels of different models are processed on the same assembly line, or a new piece is introduced to the system, significant calibration and set up is required to synchronize the master work with the acquired model. Given these difficulties, such a system would not be very robust and it would require significant synchronization to deal with assembly lines containing various automotive surface parts. This thesis aims to develop a more generic technique, which does not require an ideal model.

1.2 Objectives

This thesis deals primarily with the design of a deformation detection subsystem. Its requirements are to identify deformations of interest over the surface of automotive parts, with minimal human interaction and independently from the type of acquisition system used. The deformations of interest are dings and dents, where dings are surface deformations which protrude from the surface and dents are depressions into the surface. Dings and dents, which can vary in size between several micrometres to several millimetres, are caused by a variety of situations including inadequate manipulation of parts or tools around a vehicle during assembly, rack storage, and shipping from subcontractors. Limited literature has been published on the topic of automated surface deformation detection for quality control, with most techniques requiring a master work serving as the ideal model to compare the measured model against [1, 2]. This thesis focuses on deformation detection when no ideal model of the automotive part is provided, similar to an approach which is alluded to by Döring *et al.* [3] and explored by Chen [4]. Since there is no CAD model of a master work to compare the measured model to, the deformation detection must be done without knowledge of the expected surface and requires certain assumptions to be made based on common characteristics of surface deformations

compared to the characteristics of an undeformed surface. However, not all characteristics can be assumed, known, or easily defined. Therefore some basic parameters need to be set by the operator to provide the system with a minimal knowledge of the approximate size or scale of the deformations that the manufacturer wants to detect and eliminate from its products. This is not unrealistic, as the operator generally has a clear idea of the approximate range of sizes for the deformations to be detected.

Given these requirements, a system is proposed which analyzes the digital 3D model of an automotive part collected along the assembly line, determines the locations of only the deformations of interest, and classifies them as dings or dents. Areas of significant surface variation could be deformations. But other features of an automotive panel such as aesthetic curves and door handles, or inaccurate surface measurements such as acquisition artifacts and noise, also represent surface curves that must not be falsely detected as dings or dents. For this reason, more is required than a simple feature extraction method. The deformation detection system is comprised of a surface shape analysis phase to extract areas of interest, a segmentation phase to group areas containing pieces of deformations together into segments, and a classification phase to determine which segments contain dings, which segments contain dents, and which segments contain neither. Design features of the automotive panel are generally much larger than the deformations to be detected, therefore they can likely be classified by size and scale to make sure they are not outputted as a deformation of interest.

Two original surface shape analysis techniques that extract 3-dimensional features from a 3-dimensional model are proposed. The first technique is an octree-based feature extraction, which is a more general approach; and the second technique is a contour-based analysis, which is specifically suited to the problem of extracting deformations from relatively smooth surfaces. Both algorithms support multi-resolution analysis of the 3D model of a panel, providing the capability of extracting deformations regardless of the resolution or scale of the model. The techniques also rely on intuitively adjustable parameters for the operator to target the feature extraction towards desired characteristics of deformations. Both techniques have strengths and limitations which are analyzed both theoretically and empirically.

A group of segmentation techniques are presented to create consistent segments out of the results of surface shape analysis, with each one representing a complete and unique deformation of interest. The final proposed stage after segmentation classifies the extracted segments, selects features which correspond to the set characteristics of deformations, and ignores other features. The classification stage also determines whether the deformation is a ding or a dent.

This research takes place in the context of developing an experimental automated deformation detection and marking system for the automotive industry. A miniature representation of this system is created in a lab setting [5], which is primarily broken up into two subsystems: the deformation detection subsystem, which analyzes an automotive panel and locates the deformations and is detailed in this thesis, and the robotic marking subsystem, which tracks the automotive panel as it moves along the assembly line and marks the deformations, which is beyond the scope of this work [6].

1.3 Thesis Structure

A literature review, which covers existing works which may be beneficial to this application, is proposed in chapter 2. An overview of the proposed deformation detection system, an overview of the testing, as well as a description of the complete deformation detection and marking system and the corresponding experimental laboratory setup are all presented in chapter 3. The two proposed surface shape analysis approaches are described in chapter 4 and 5, respectively, and an experimental analysis is conducted for each technique. Chapter 6 provides a description of the segmentation and classification phases to refine the results and presents an empirical comparison between the various techniques. Also, recommendations are made on which combination of techniques yield the best results. Finally, in the conclusion, chapter 7, a summary of the techniques and recommendations is offered, contributions of this thesis are presented, and future work is outlined.

Chapter 2 Literature Review

The process to automatically determine the location of a defect on an automotive body part requires several steps, some of which are still complex research topics. This chapter will provide a review of important existing research that is relevant to the topic of automatic deformation detection proposed in this thesis. In order to analyze the surface for automotive body parts, the parts must first be digitally represented as a 3-dimensional object. Various techniques in 3D acquisition and surface reconstruction will be explored in section 2.1 and 2.2 respectively. To determine the location of deformations in the digitized 3-dimensional surface, the surface must be analyzed for certain characteristics. Techniques such as 2D and 3D segmentation and feature extraction, as well as mathematic characterization of surfaces can be useful for surface analysis, and will be discussed in section 2.3. Finally, the extracted areas of strong surface variation must be classified to determine if it is a deformation of interest, and if so, what type of deformation it is. A short evaluation of classification techniques will be presented in section 2.4. This chapter concludes with section 2.5, which provides a summary of the reviewed techniques, as well as briefly explains how they can be applied and how they inspire new techniques which are developed for the objective of this thesis.

2.1 3D Acquisition

In order to analyze the surface of a real-world object in 3-dimensions, it must be scanned and converted into digital 3-dimensional data. Various 3-dimensional scanners exist that provide this function.

Laser scanners are very commonly used highly accurate 3D acquisition tools. Some laser scanners use triangulation as the main principle behind their method of distance calculation [7-9]. By emitting a laser that falls on the surface of an object or scene, and acquiring the position of that laser relative to a sensor, such as a camera, the distance to that surface can be calculated. Other laser scanners use time-of-flight calculations as the main principle behind their method of distance [10, 11]. The time-of-flight laser rangefinder emits a pulse of light, and this pulse of light should reflect off the object being acquired and return to the sensor. The time taken for the laser to return to the sensor is proportional to twice the distance from the sensor to the part of the object or scene that the laser hits. Based on moving these lasers around an object or scene, a 3D representation can be created after acquiring sufficient distance readings to create a 3D point cloud. These 3D acquisition tools might be able to produce high resolution, high accuracy scans. However they are usually expensive systems

that take a long time to complete a full scan, and often require some mechanical system to move the laser and acquire readings before accumulation into a point cloud.

For lower cost, lower scan times, and minimal mechanical complexity, stereoscopic vision systems are a very popular way of digitizing a 3-dimensional object. By viewing an object with two cameras, separated by some distance, it is possible to recreate that scene in 3-dimensions [12]. As each camera sees the scene differently, depth information can be calculated based on the disparity of the scene between the two cameras. Two problems must be solved in order to successfully determine the 3-dimensional position of a part of the scene: the correspondence problem and the reconstruction problem. A piece of the scene may appear in both cameras, and the location of that piece with respect to each of the cameras must be determined. This challenge is known as the correspondence problem. If a feature can be located with respect to each of the cameras, the distance to that feature, with respect to the cameras, can be calculated via triangulation by knowing the characteristics of the cameras and their positional relationship to one another. This is known as the reconstruction problem. Fortunately, the reconstruction problem can easily be solved through triangulation by knowing the configuration of the cameras, assuming the correspondence problem has been solved. The correspondence problem is much more difficult to solve, and has been the topic of extensive computer vision research for the past three decades. Many research projects attempt to solve both problems for each part of the scene seen by both cameras, creating a 3-dimensional reconstruction of the surroundings for robotic navigation [13-15]. The correspondence problem is usually computationally intensive as part of the scene in one camera must be compared to each part of the scene in the other camera to find a match, however the epipolar constraint can reduce the complexity of the search and reduce the likelihood of false matches [12]. This constraint allows the search for the matching feature in one camera, only along a certain line in the other camera, which can be determined by the location of the feature in the first camera, and is known as the epipolar line. There can be greater accuracy in solving the correspondence problem by selecting unique features carefully. However, this inherently prevents the entire scene from being reconstructed in 3-dimensions, since only certain parts are used for the depth calculation. Using stereoscopic sensors alone to create dense 3-dimensional models is difficult due to the possibilities of featureless surfaces. If sufficient feature points cannot be located, the sparse model will be lacking too much detail. Also, the correspondence problem may be difficult to solve under certain circumstances, considering that many parts of the scene will contain features with similar characteristics resulting in false matches, again causing a lack of detail.

One popular technique to overcome the limitations of using traditional stereoscopic imaging is to acquire 3-dimensional models using structured light scanners. This type of system projects a set of artificial features onto a model or scene that is going to be scanned, and then uses a vision system to acquire the model in 3D. Most structured lighting systems use a single camera along with a projector to do the 3D acquisition [16, 17]. Similar to stereoscopic vision systems, the reconstruction problem is present here, but instead of requiring the relationship between two cameras, the relationship between the projector and a singular camera must be known. The correspondence problem in this situation is to match each of the features that are projected to the singular camera's view of them. By projecting a pattern of artificial features on the object or scene, both problems of determining sufficient feature points to create a dense enough model, and allowing an easier task for the correspondence of the stereo images are solved with greatest ease.

Payeur and Desjardins [18] also implement such a system, using a bi-dimensional pseudo-random pattern of coloured squares for projection to create the artificial features. Each artificial feature point is a uniquely coded pattern to decrease the likelihood of incorrect correspondence. Dissimilar to other structured light systems, a stereo pair of cameras is used, instead of the singular camera proposed by other structured light solutions. This way, the calibration is between the two cameras, independent of the projector. Also, it allows focus, zoom, and brightness adjustments to the projector, without requiring re-calibration of the system. The output of this system is a point cloud, where each point represents a feature point that was projected onto the object or scene.

2.2 Surface Reconstruction

From the acquisition of data from most mechanisms, whether it be a point cloud or a voxel representation, most feature extraction algorithms require that the surface of the object is reconstructed to determine the orientation of each part of the surface. 3D scanners usually take samples of the surface of an object, recording only positional information. Photometric 3D scanners are capable of determining orientation information along with positional information [19, 20]. They use multiple lighting angles while viewing the scene with a camera and keeping the viewing angle constant, creating radiance maps by which the orientation of each pixel in the scene can be determined through analysis of the lighting characteristics. This technique can be attached to stereo imaging or structured light systems to provide surface orientation information. However, such a process is not usually flexible enough to be acceptable in a manufacturing setting, considering that the lighting conditions must be very precise and the complexity of the

system which contains both the 3D acquisition tool and a lighting system. Also, the results provided may not warrant the time and effort required to implement such a system, especially if surface reconstruction techniques exist for the more flexible 3D acquisition tools required for the application at hand, even if they only provide positional information.

There are a number of techniques which allow the creation of a surface from an unordered point cloud, containing only 3D positions, with very high accuracy and efficiency. Unordered point clouds are a set of points where the order of the points is undefined, such that the adjacency of points cannot be determined simply by analyzing the order of the series of points. Triangulation is the process of dividing a set of points into regions of non-overlapping triangles, and such a mesh of triangles is what is required to represent the surface.

Delaunay triangulation is a triangulation of points, such that no point falls inside the circumcircle of any triangle other than the points that form that triangle [12]. The Delaunay triangulation of a set of points is not always unique, and it is possible that Delaunay triangulation cannot be completed for all sets of points because the criteria cannot be met. However, if such a triangulation is possible, it will construct a triangular mesh surface from a set of points, while avoiding skinny triangles and maximizing the angles of the triangles created. The Bowyer-Watson algorithm is an incremental point insertion algorithm [21, 22]. Beginning with an empty mesh, points are added one at a time to create the triangulation, ensuring that the triangles created with the point meet the Delaunay criteria, and that existing triangles still meet the Delaunay criteria in spite of the point being added. This algorithm is computationally expensive, since all existing triangles must be re-evaluated when a new point is inserted. To improve on the extra computation required to re-evaluate each triangle, Lee and Schachter [23] propose a divide-and-conquer algorithm, where the points are recursively divided into halves, until a minimum set of points can be triangulated. Then, each of these subsets is merged to form a triangulated surface. Though the efficiency of the algorithm is an improvement over the incremental technique, the sorting of the points for division can be computationally expensive on large datasets.

One of the most commonly used techniques for surface reconstruction from an unordered point cloud is the ball pivoting algorithm, proposed by Bernadini *et al.* [24]. The algorithm uses the idea of a ball “walking” along the surface of the object by rolling this ball around the points in the point cloud, and analyzing the intersection of its surface with three points of the point cloud at a time. Provided that no other point lies within the sphere, a triangle is formed between those three points, and that triangle becomes part of the mesh surface. The algorithm starts by selecting an initial triangle, as the seed triangle, in the point cloud, to place

the ball on, such that the three points fall on the surface of the ball and no other point is contained inside the ball. The ball then pivots about an edge of the triangle, meaning it stays in contact with two of the points. It pivots until another point is reached, and no other point is contained inside the ball, at which point it creates a new triangle to add to the mesh. This process is repeated until all reachable edges from the initial triangle have been used as a pivot. Then a new triangle, which has not yet been reached, is used as the new seed triangle and the process is repeated. This entire process occurs until all points in the point cloud have been evaluated. A voxel representation of the point cloud, with a voxel size equal to the ball radius, is used to find points within a neighbourhood such that the search for points is more efficient than a full search. A 2-dimensional example is shown in Figure 2.1, where the ball pivots about one point in 2-dimensions. An important parameter is the radius of the ball. If the radius is too small, relative to the sampling density of the point cloud, the ball may not be large enough to consistently get three points to fall onto its surface, and many holes in the mesh will be present. If the radius is too large, parts of the mesh that should not connect to each other may be connected, generating an incorrect surface and missing potential features. This algorithm is more efficient at generating meshes than the previously mentioned techniques, since it aggregates pieces of the mesh incrementally, and does not require sorting since neighbouring points can be searched for through the voxel representation. Also, the ball radius parameter can be set very easily by knowing the characteristics of the 3D acquisition tool. Provided that the 3D acquisition tool gives a point cloud representation of the model as an output, and an appropriate ball radius is selected, the ball pivoting technique is an ideal surface reconstruction technique as it does not alter the original data in any way and can reconstruct the surface without holes or losing features.

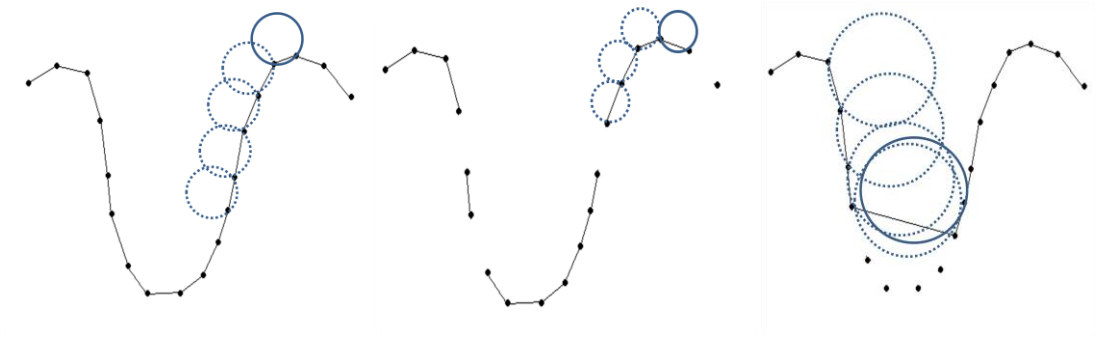

Figure 2.1. a) Ball pivoting with proper radius. b) Ball pivoting with radius too small, holes in the mesh are left because of lower sampling density. c) Ball pivoting with radius too large, features are lost. (Adapted from [24])

In the domain of graphics, on a 3-dimensional grid, each cell can be called a volumetric pixel, also known as a voxel. Voxel representations are a different form of representing 3D data, where an object can be described by a set of voxels that it occupies as opposed to a set of polygons [25]. The Marching Cubes algorithm, proposed by Lorensen and Cline [26], is a technique that generates a mesh from a voxel representation, though the technique can be extended to generating a mesh from any 3-dimensional representation. The principle behind marching cubes is to subdivide the space containing the object into a grid of cubes. The vertex of each cube is evaluated to see if it lies within the object or outside of the object. If all of the vertices of a cube lie outside or inside an object, none of the surface intersects that cube. But if some of the vertices of the cube lie on or inside the object while other vertices lie outside of the object, some of the object's surface passes through that cube, and a composition of triangular pieces can be created to represent that surface. The algorithm is based on replacing each of those cubes with the appropriate surface approximation, and connecting them to form a mesh. Since there are 8 vertices in a cube, and 2 possibilities for each vertex (inside the object or outside the object), there are 2^8 possibilities for combinations. Each of those possibilities corresponds to a certain surface approximation composed of triangles. Some examples of those possibilities are shown in Figure 2.2.

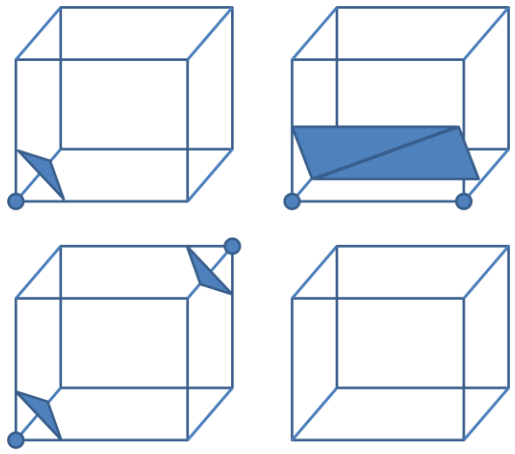

Figure 2.2. Examples of marching cubes possible surfaces based on which points of the cube lie inside the object. (Adapted from [26])

The selection of the surface approximation could result in a mesh that has significant error from the initial scan. Post-processing can be used to smooth the mesh and reduce the error, however this involves further assumptions. Shu and KanKanhalli [27] proposed an adaptive marching cubes method, which adjusts the size of the triangles to the shape of the

surface. The complexity of these algorithms lies in determining whether a vertex lies inside the object or outside the object. In the case of a voxel representation, it is easy to determine this, since the representation itself identifies which voxels are inside the object. That relationship is more difficult to determine if the scanner provides another representation, such as the more likely point cloud representation, and would require further steps to prepare the data for surface reconstruction using this technique. This makes marching cubes an unnecessarily complex surface reconstruction algorithm for the purposes of this application.

Mederos and Figueiredo proposed a method that uses a moving least-squares surface to generate a triangular mesh [28], as well as reducing the size of the point cloud to create a simpler mesh. The algorithm splits the point cloud into clusters, such that the points in each cluster likely belong to a surface with very little curvature, which is evaluated by calculating the variation in a least-squares surface fitted to those points. A representative point is found for each clusters, which is near the centroid of the cluster and falls on a moving least-squares surface fitted to the cluster [29]. This reduces the size of the point cloud, without losing features, so triangulation can be performed. Triangulation is then performed, using an algorithm to create the Delaunay triangles to form the mesh [30]. Though a mesh has now been created, a more accurate mesh is possible given the data. A further refinement step is used, which uses the orientation of the calculated least-squares surface for each representative point and the original point cloud, to get a more accurate mesh that fits the original point cloud better. This multi-step algorithm provides excellent results for more complex meshes, but also provides steps that are not required for good results with simpler meshes. The extra step that reduces the number of points in the point cloud may be counterproductive for this application, as any change to the original point cloud may alter features that need to be detected as deformations. In the interest of minimizing the error between the 3D acquisition and surface analysis, this technique is less suited for this particular application.

2.3 Surface Shape Analysis

The most important component of a surface deformation detection system is surface analysis to locate the defects in question. No ideal model of the automotive part is provided, so the algorithm has no *a priori* knowledge of what the surface should look like without deformations. If such knowledge was provided, the focus on the shape analysis would be registration to match the ideal model with the scanned model, and a simple subtraction would give the surface deformations as they deviate from the provided model. Since no knowledge is

available, registration techniques are not required, but advanced surface shape analysis must be performed instead to determine the locations of probable deformations.

Some techniques such as noise removal and surface reconstruction will be evaluated for components that can aid in this task, but the majority of this section will be evaluating feature extraction and segmentation techniques as those pertain directly to analyzing the shape of the surface. Feature extraction techniques are often designed without regard for what part of the object is extracted, as long as it is distinct and consistent, for purposes such as registration or tracking. Feature extraction is important for this application since the deformations will resemble features as they are sharp and distinct from the rest of the surface. Typical shape fitting or segmentation techniques may not detect deformations sufficiently, since they are very small and shallow, so feature extraction provides methods to extract these areas of change in the smooth surface of the automotive part as areas of interest, such that they can be investigated further. Segmentation techniques are often used for classification purposes to separate areas of differing characteristics and to extract segments as a whole. Segmentation techniques are important for this application due to the need of classifying these extracted areas of surface variation to determine if it is a deformation of interest or not, so the entire area of strong surface variation must be segmented from the rest of the surface. This thesis requires surface shape analysis to identify the location of changes in the surface, as well as the surface characteristics of these areas of change such that they can be classified as a deformation of interest or some other surface characteristic that is of no importance. Therefore literature from both types of techniques will be evaluated. Section 2.3.1 will introduce 2-dimensional techniques that can be used for surface shape analysis and segmentation. Section 2.3.2 will introduce 3-dimensional techniques for the same purposes. It is assumed that the model will be represented as a 3D point clouds. For this reason, techniques specific to that format will be evaluated. However, some segmentation techniques used on volumetric representations, such as volumetric grids or octrees, will also be evaluated.

2.3.1 2D Analysis

2D image processing techniques can be worth evaluating for surface analysis purposes. Provided that the 3-dimensional data is converted from a range image to a 2-dimensional image where each pixel intensity represents the depth of that point on the object from the viewpoint, features can be extracted and images can be segmented using traditional 2D image processing techniques. Also, these 2D image processing techniques can be extended to three dimensions,

especially if the 3D data is in the form of a volumetric grid, such as voxels or octree representations.

Edge detection techniques are commonly used in 2D applications, to determine areas of significant change in pixel intensity. Well-known edge detectors, such as the Sobel and Canny operators, can highlight the areas that belong to features [12]. The result of applying these on a range image, after an appropriate threshold is selected, is a binary edge image where the locations of sharp changes in the surface of a model will be highlighted. These techniques can outline a deformation feature, however they may not always highlight all of it due to a lack of connectivity on the edges. These disconnects between edges can be due to gradual changes in intensity between pixels not being large enough to cross the edge detection threshold, and thus not being labelled as an edge. Slight lack of connectivity can be overcome using dilation and erosion [31], but these are very dependent on the situation and may not always provide good results.

The extraction of a complete deformation feature bears characteristics of problems that segmentation techniques are effective at solving. Instead of using edge detection and connecting the outline of the deformation for classification, it might be more effective to segment the range image directly, into regions that share similar characteristics. Ideally, such segmentation would group areas of high surface variation and all areas of low surface variation, creating a distinction between the regular surface and the deformations.

The mentioned edge detection techniques usually determine whether each pixel in the image belongs to an edge or not, and produce a binary image where lit pixels are the ones that belong to an edge. For classification purposes, knowledge of each pixel's correspondence to an edge is usually not sufficient. Edge pixels often outline important areas in an image, or highlight certain shapes and patterns. In order to acquire more information about the features in the image, these edge pixels can be grouped to form a shape or outline of a complete feature to be classified. Blob extraction, also known as connected component labelling, is a commonly used segmentation technique to group adjacent lit pixels in a binary image [32], whether it be an edge detected image or some other form of segmentation. Such groups can be analyzed for shape, size and other characteristics for classification.

The k-means algorithm is a very well-known clustering algorithm, which is also used for segmentation purposes, to partition a dataset into a specified number of clusters, denoted as *k* [32]. *k* cluster centers are picked in an image, either randomly or using some sort of selection algorithm. A distance is calculated for a certain pixel to each cluster, and the pixel is assigned to the cluster which is the least distant from it. This distance value can be any combination of

characteristics which can determine which cluster a pixel should belong to, such as actual distance from the center, colour, intensity, or texture. Cluster centers are recalculated by an average of all pixels belonging to the cluster, and then the algorithm is repeated for the next pixel. This process should divide the image into *k* clusters, with differing results based on the initial selection of the cluster centers and the distance metric. However, selecting the value of *k* is most important, and in the case of an unknown amount of deformations, this value cannot be known for sure.

If the intensity of the pixels in an image is the primary indicator of what segment each pixel belongs to, thresholding can be used for segmentation. A histogram-based method can also be used, in this manner, for the selection of thresholds. In the context of threshold selection for edge detection, appropriate thresholds must be selected such that the important features are extracted while minimizing the extraction of undesired features. This is a type of segmentation based on intensity, and automatic threshold selection algorithms exist that are capable of determining that value. Some basic algorithms attempt to clarify the location of the peaks and valleys in a histogram, such that thresholds can be determined to separate them [33]. Otsu [34] attempts to analyze the gray-level histogram of an image, and selects an optimal threshold based on certain criteria such as maximizing the variance of the classes the histogram is separated into. Sometimes the performance of these algorithms are affected significantly, since determining the peaks and valleys in histograms with lots of noise or varying characteristics can be difficult. Also, intensity is not always a reliable metric for segmentation due to noise or lighting changes.

Seeded region growing methods start with a set of pixels, each belonging to a different object to be segmented [32]. Then neighbouring pixels of each of those objects is evaluated to determine if they also belong to the object. This evaluation is done based on comparing a neighbouring pixel's intensity to the object's pixels' mean. The pixel with the smallest difference is added to the object's pixel list. This process is repeated until all pixels are allocated to an object. This technique, however, requires that at least one pixel inside each object is known, and that information is not available in the application considered. Unseeded region growing uses a similar process, but requires no initial set of pixels. It does require a threshold. The difference with the seeded region growing method is that a random initial pixel is selected in the image as the seed. If the evaluation of neighbouring pixels falls beneath a difference threshold, the pixel is added to the object's region. Otherwise, a new region is created, and the process repeats. Unseeded region growing can overcome the problems discussed with seeded region growing where the location of pixels in each deformation is not known. It also overcomes the

problems with k-means algorithms by not requiring an initial knowledge of the number of clusters for division. The limitation is that a threshold must be selected carefully such that it correctly separates region boundaries in all parts of the image. Similar to the limitations of edge detection, gradually changing pixel intensities between actual regions of the image may not be sufficient to overcome the threshold, and a new region may not be created. This can cause deformations to be connected to areas of smooth surfaces which are omnipresent over automotive panels, and being unsuccessful in segmenting the regions required.

These techniques can all be extended to 3 dimensions by using points or voxels instead of pixels and adjacency can be determined by distance or connectivity in a grid or tree, as is done by Palagyi and Kuba [35]. Also, the data being used as the intensity value in an image can be adjusted to using distance in a range image or 3-dimensional surface deformation metrics such as standard deviation of normals or the curvedness value which will be discussed further in section 2.3.2.

2.3.2 3D Analysis

Various techniques from the field of 3D data analysis can be used for the purpose of deformation detection. Section 2.3.2.1 reviews surface reconstruction and noise removal methods, based only on point data, which can be applied to determine significant surface variation. Section 2.3.2.2 reviews feature extraction techniques that are based on information obtained on a triangulated mesh generated by surface reconstruction. Finally, section 2.3.2.3 discusses parametric representations of surface variation that can aid in determining surface deformations.

2.3.2.1 Analysis based on Point-based Surface Estimation

A basic method of determining deformations is by applying noise removal techniques. Simple deformations in a mesh can resemble outliers on a smooth surface, and using noise removal techniques to identify areas of noise-like characteristics can be beneficial to determining the location of the defect.

Schall and Seidel propose a noise removal method that also provides applications in outlier removal [36]. This algorithm takes noisy point cloud data and fits the data points into their most probable position based on approximated surfaces to filter out noise. The statistical method estimates the density of each area of the point cloud, and uses the neighbouring points to adapt a probable surface to each point. After fitting a surface to a predefined number of neighbouring points, the current point and neighbouring points are adjusted accordingly. The

further their distance from the current point that is being assessed, the less their adjustment is. In the end, since points are moved to their most probable location along a surface, the spatial density of the resulting point cloud is relatively consistent throughout the surface of the object. To remove outliers, which were not removed by the filtering process, a simple threshold can be applied. If the spatial density surrounding a point is too low, beneath the threshold, it is likely that the point is an outlier.

By determining which points are outliers from a smooth surface, the points that belong to deformations may be detected as outliers. Though a threshold needs to be determined, this is a feasible method of extracting deformations. However, the algorithm might be too dependent on the sampling density to determine whether the points are part of a deformation or not, and the parameter selection is not very intuitive and may cause unpredictable performance. Sometimes, deformations may not have the characteristics of being outlier points in the point cloud, and this algorithm may fail to be robust enough to deal with such a situation.

The moving least-squares surface reconstruction technique proposed by Mederos and Figueiredo [28], which was discussed earlier, uses a hierarchical segmentation technique that finds redundant points such that the point cloud density can be reduced before surface reconstruction. This process is based on a technique discussed by Shaffer and Garland [37]. This segmentation technique results in clusters of points, where the surface variation within each cluster is minimal. The boundaries between those clusters could define a significant deformation in the surface. Analyzing the eigenvalues and eigenvectors of the covariance matrix of a cluster of points can estimate local surface properties [37, 38]. The 3x3 covariance matrix for a neighbourhood is given by:

$$\vec{c} = \frac{1}{N}\sum_{i=1}^{N} \overrightarrow{x_i} \tag{2.1}$$

$$M = \begin{bmatrix} \overrightarrow{x_1} - \vec{c} \\ \overrightarrow{x_2} - \vec{c} \\ \overrightarrow{x_3} - \vec{c} \\ \dots \\ \overrightarrow{x_N} - \vec{c} \end{bmatrix}^T \begin{bmatrix} \overrightarrow{x_1} - \vec{c} \\ \overrightarrow{x_2} - \vec{c} \\ \overrightarrow{x_3} - \vec{c} \\ \dots \\ \overrightarrow{x_N} - \vec{c} \end{bmatrix} \tag{2.2}$$

where M is the covariance matrix for N points, $\overrightarrow{x_1} \dots \overrightarrow{x_N}$ are the points in the neighbourhood, and $\vec{c}$ is the centroid of those points.

The eigenvectors of this matrix represent the vectors that form an orthogonal frame, and each of the eigenvalues represents the variation of the points along their respective eigenvectors. The eigenvector corresponding to the smallest eigenvalue is the normal to the plane of best fit. A binary space partitioning tree is used to partition the model into clusters of

points that lie on surfaces of low variation. The root node of the tree contains the entire point cloud, and the subdivision is based on the flatness criterion, which represents variation within a group of points, as described by Pauly *et al*. [39]. This flatness criterion is calculated as follows:

$$\sigma = \frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3} \tag{2.3}$$

where σ is the flatness criterion, and $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues in order from smallest to largest.

If the flatness criterion is too high, that cluster of points is separated further. That separation is done through the centroid of the cluster, in the direction of greatest variation, which is the eigenvector corresponding to λ_3. This process allows the point cloud to be split into clusters that belong to the same relatively flat surface. The boundaries between these clusters can be classified as features, since they are areas of variation between seemingly flat clusters. Such an algorithm is effective at determining the characteristics of a model for surface reconstruction or resampling, but requires an extension to be used for feature extraction efficiently, since the boundaries must be determined instead of just the clusters. The use of a binary space partitioning tree is very effective to separate the mesh, but might become too deep of a tree to traverse efficiently, since each node can only be subdivided into 2 nodes at a time. Finally, the calculation of the covariance matrix, and the eigenvectors and eigenvalues each time a subdivision happens, is computationally intensive, especially at nodes closer to the root where many points are being evaluated at a time.

2.3.2.2 Analysis based on Triangulated Mesh Properties

When simple noise removal and surface reconstruction techniques do not provide sufficient tools for finding deformations, more specialized 3D feature extraction techniques exist. Similar to the binary space partitioning tree subdivision, Woo *et al.* [40] introduce a technique based on octree structures, and use recursive subdivision of the volume of a 3D model to identify features. It removes segments of the mesh as the octree is generated, and leaves parts of the mesh that belong to features in the final octree data structure. It requires a model with a reconstructed surface and partitions the model into subsections which represent varying levels, or scales, of features. Given a 3D mesh, surface normal vectors can be calculated for all triangles composing the surface, and ultimately for each point by averaging the normals of the triangles that the point belongs to. Variations in the orientation of the surface within a given region are estimated from the standard deviation of normal vectors within that region. This

method facilitates the partitioning process. All of the points that make up the surface of the object are initially added to the root of the tree structure. The standard deviation of their normals is calculated, and compared to a threshold. First the mean normal is computed:

$$\overline{N} = \sum_{i=0}^{n} N_i / n \tag{2.4}$$

where n is the number of points at the node, $\overline{N}$ is the mean normal, and N_i is the unit normal of each of the points. Then the standard deviation, σ, of the normals can be estimated as:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n}(N_i - \overline{N})^2}{n}}$$

$$= \sqrt{\frac{\sum_{i=0}^{n}(x_i - x)^2 + \sum_{i=0}^{n}(y_i - y)^2 + \sum_{i=0}^{n}(z_i - z)^2}{n}} \tag{2.5}$$

A threshold must be defined for the subdivision as the maximum standard deviation allowed in a volume before it should be further divided. This threshold is defined by the user. If the standard deviation is larger than the threshold, the volume represented by the root is divided into 8 octants. 8 children are added to the root and each of those children represents each of the 8 octants that the volume is divided into. The points from the root are redistributed based on their spatial location into each of the 8 octants, and thus into each of the 8 children of the root. This process is repeated recursively for each of the children, and for their children, and so on until either there are no children remaining, or a sufficient level of feature details is discovered. The depth of the tree determines how detailed the feature level is. Figure 2.3 details the recursive process of the feature segmentation at various scales.

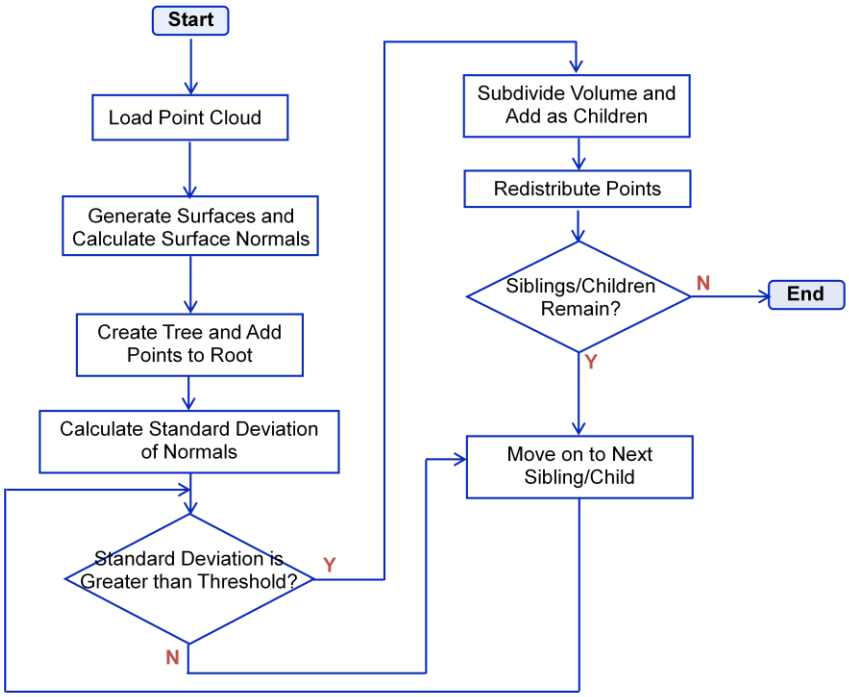

Figure 2.3. Octree segmentation subdivisioning flowchart.

The final structure provides a tree where the points are distributed amongst the tree nodes. Leaves at greater depths represent finer detailed features of the mesh contained in smaller volumes. Leaves at lesser depths represent larger scale features contained in larger volumes. An example of this can be seen in Figure 2.4.

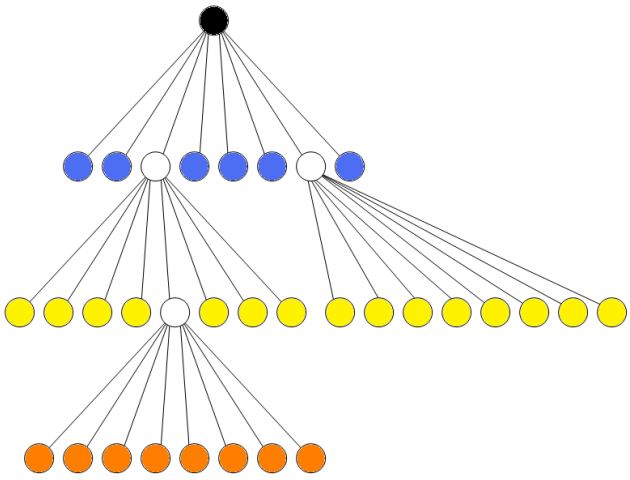

Figure 2.4. An example octree where the black node represents the entire mesh. Blue nodes represent local features at scale 1 (Largest scale). Yellow nodes represent local features at scale 2. Orange nodes represent local features at scale 3 (Finest scale).

The technique is very useful in generating a multiresolution representation of the features in a mesh. However, because it uses a single threshold value throughout the entire tree, its ability to detect features can be unpredictable. A feature has to sufficiently affect the standard deviation of the surface normals across the selected volume for the method to investigate the mesh at a higher resolution. Sometimes this is not the case and the feature is not identified. The criteria for setting the threshold is that it must be high enough such that smooth curvatures and noise are not detected as features, but low enough such that the deformation features are detected. This remains a subjective criterion that varies with the point cloud. Since standard deviation is used for this distinction, it can be hard to find values which meet the defined criteria.

The technique is very conservative with its memory usage due to its ability to remove unnecessary nodes as the tree is being generated. It generates broad shallow trees, as opposed to deep narrow trees generated by the binary space partitioning method in [28, 37], which makes it easier to traverse. This allows analysis at much higher resolutions, with reduced computational load. An added benefit is that the computational complexity for the octree subdivision criteria is less than the binary space partitioning method, since a simple standard deviation calculation is simpler than calculating a covariance matrix and the appropriate eigenvalues and eigenvectors. Also, similar to quadtrees for 2-dimensional data, the octree data structure is better-suited to 3-dimensional analysis than a binary space partitioning tree. The octree representation allows features to be represented in the point cloud dataset, as well as representing features in a volumetric grid, giving the flexibility of using a variety of techniques for added segmentation. A limitation, for the purposes of this research, is that the octree method determines which pieces of the mesh show a change in surface shape. However a collection of these pieces is required to form a complete deformation feature for classification, and this algorithm is not capable of that. An extra step would be required to connect these mesh pieces to form a feature, such as 3D extensions of several 2D segmentation techniques previously discussed.

Feature extraction can also be done without converting the mesh into a binary space partitioning tree or octree representation. Pauly and Gross present a technique that allows feature extraction from a 3D object composed of surfaces, at various detail levels [41]. This allows features to be classified as general larger areas, or as fine variations in surfaces. Weights are assigned to each point in the point cloud, representing the amount of local variation in the surface normals. At different scales, different local neighbourhood sizes are used, which correspond to how many points are involved in estimating the feature. By increasing the

neighbourhood size, more data is involved in estimating the surface feature, behaving like a smoothing filter. Changing the scale of the feature extraction is like changing the amount of filtering before the feature extraction is performed. Features in the point cloud are determined by identifying the local maxima in the weights. Introducing the idea of feature persistence, a threshold can be selected, such that variations over that threshold can be considered feature nodes. As a feature persistently exceeds that threshold, across multiple scales, it can be classified as a strong feature, not just a small local feature. For example, some features may only be visible at a fine detailed level, while others may only be visible when looking at a very coarse level. If one persists across many scales, it can be beneficially identified as a useful feature.

Their results show that this method performs well under noise and is effective at identifying prominent features, more so than single scale methods, where unimportant features can be identified just as frequently as important ones. Also, the idea of feature persistence is introduced, where prominent features appear over multiple scales, and can be very important in using multiresolution information to identify important features. However, as in the octree method, each area is defined as exhibiting a change in surface shape or not, but this is not satisfactory for classification purposes, since a collection of these areas must be joined to shape a deformation feature.

The techniques described so far analyze each point or node individually to determine if it belongs to a feature or not. This inherently requires an extra step before classification, since the individual points or nodes must be connected together to constitute a feature. Lee *et al.* concentrate on a method to detect edges in 3D point clouds, which aggregates pieces of the mesh, connecting pieces that belong to a feature [42]. Edges are classified as sharp edges and smooth edges, where sharp edges are drastic changes in adjacent surface normals and smooth edges are gradual changes in adjacent surface normals, as more of a curve would do. All adjacent triangular surfaces, with a variation in their surface normals beyond a certain threshold are regarded as boundary meshes. Ideally, several of these boundary meshes would be connected to form a continuous feature, sometimes until they form a circular continuous loop. This is not always the case in real-world free form scans, and features may not appear consistently connected throughout. Due to variation in the density of the point scans, or the inherent shape of the object, the edges may change from two adjacent surfaces with large surface normal variations, to several adjacent surfaces that gradually change orientation to form a smooth curve as shown in Figure 2.5.

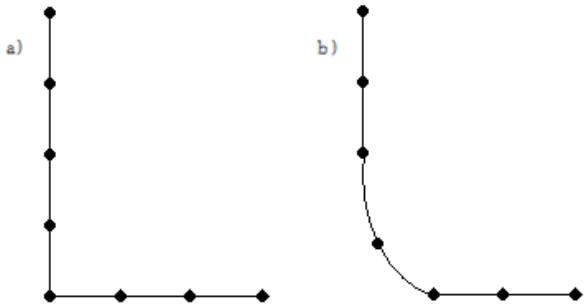

Figure 2.5. a) Sharp edge can be detected by comparing adjacent edges b) Smooth edge cannot be detected by comparing adjacent edges (Adapted from [23])

In smooth rounded edges, the change in orientation between adjacent normals may not be large enough to overcome the threshold to define a boundary mesh. To overcome this problem, both the areas of the polygon meshes and the angles between adjacent surface normals are accumulated separately. One threshold, the area threshold, defines the maximum area of a group of polygons that will be attached together to form a mesh, and a second threshold, the angle threshold, defines the maximum accumulated angle before the mesh is found to be an edge. The technique attaches adjacent polygons together to form a larger mesh, until that mesh's aggregated angle exceeds the angle threshold, and it is decided that this mesh is an edge. If the surface area of the mesh exceeds the area threshold before the accumulated angle exceeds the angle threshold, the mesh is not an edge. This is how longer gradual curves are detected as edges, just as well as abrupt edges are detected, since both types should cross the angle threshold before crossing the area threshold. This technique succeeds at detecting smooth, more gradual curvatures as features. Also, this process automatically connects mesh pieces to form a feature for classification, and does not require an extra step for grouping or labelling connected pieces. Even though it does not provide multiresolution capabilities without explicitly changing the parameters and doing multiple passes, it is capable of determine features of different scales due to its ability to extract sharp features, which may appear at low resolutions, as well as smooth curved features, which may appear at higher resolutions. This approach also stops aggregating edges when the thresholds have been crossed, meaning that several pieces of the same feature may be left unconnected, thus requiring an additional connecting step for adjacent edge pieces.

Hubeli and Gross focused on using a variety of operators on the edges between points in a point cloud to extract features [43]. Similar to the technique proposed by Pauly *et al.* [41], weights are assigned to the points representing the likelihood that it belongs to a feature. Requiring a reconstructed surface, four different operators are defined and applied to assign

weights to the appropriate edges of the mesh, which are made up of connected points. Some operators quantify the difference between a best-fit surface and the points while others quantify the difference between points and their neighbours. The edges are then assessed to determine if they belong to prominent features of the 3D object, based on their weight and the weight of the surrounding edges exceeding a defined threshold. Adjacent edges exceeding the threshold are important, while the remaining edges are unimportant. By using a single threshold and removing unimportant edges, the possibility exists that there may be discontinuities in a set of connected edges. To ensure that certain prominent features do not become discontinuous, two thresholds are defined to help bridge those gaps. The edges with weights higher than the high threshold are attached to be part of a feature mesh whereas edges with weights lower than the low threshold are discarded. Edges with weights that are between the thresholds are added to the feature mesh if they are connected to other edges that are, or will be, part of the feature mesh, preserving some edges that may be weaker but are important to bridge gaps in features. Multiresolution feature extraction is achieved by separating the original mesh into progressively lower resolution meshes and reapplying the same techniques. As the vertices are added to the feature mesh, different edges are assessed by looking in similar areas in the multiresolution scan. Like the algorithm described by Lee *et al.* [42], this technique results in a feature mesh of connected pieces instead of list of pieces that belong to features, effectively reducing the step of connecting those pieces after the feature extraction. The limitation of this process is that several operators must be applied to each section of the mesh, slowing down the feature extraction. The setting of multiple thresholds may decrease robustness. Also, its multiresolution capabilities are limited to reapplying the feature extraction on different resolution versions of the mesh and combining the results.

Assuming the availability of ordered point cloud data, Vosselman *et al.* [44] exploit the knowledge of the ordered point cloud in the form of scan lines, and combine various techniques to segment point clouds by recognizing geometric shapes and flat smooth surfaces specifically for the analysis of industrial and city scans from LIDAR data. Working with an ordered point cloud, each scan line can be broken into line segments based on orientation and proximity, and a plane of best fit equation is calculated. The same operation can be performed on adjacent scan lines, and line segments from these scan lines are all compared based on some similarity criterion to be connected as a planar surface [45]. By just using a different similarity criterion, without evaluating a plane of best fit, adjacent scan lines representing smooth surfaces can also be connected [46]. Fitting other shapes such as spheres and cylinders is also discussed. This technique is an effective segmentation technique, which also classifies the extracted segments

as various primitive shapes in the process. The dependency on ordered point cloud data is a limitation of the techniques described, since data can come from various sources and may not always be in the form of the ordered point data found in scan lines. Also, the very distinct shapes that are being extracted are more associated to certain scenes in the city or in an industrial setting, and the techniques are less suited to more curved and variable surfaces of automotive body parts, since such shapes do not fall into the category of basic geometric primitives.

2.3.2.3 Analysis based on Parametric Representations

A more in-depth mathematical analysis of the surface of an object can provide a different perspective on feature extraction. Gaussian curvature and mean curvature are commonly used techniques to describe the curvature of a surface [47]. Curvedness is a useful metric which combines Gaussian curvature and mean curvature into one value, to describe the curvature of a surface as shown in Eq. 2.8 [47, 48].

$$k_1 = H + \sqrt{H^2 - K} \tag{2.6}$$

$$k_2 = H - \sqrt{H^2 - K} \tag{2.7}$$

$$c_p = \sqrt{\frac{k_1^2 + k_2^2}{2}} \tag{2.8}$$

where k_1 and k_2 are the principal curvatures at the point p, H and K are the mean curvature and Gaussian curvature at point p, respectively, and c_p is the curvedness at point p.

Ho and Gibbins [49] use curvedness, which is both scale and rotation invariant, to evaluate the amount of curvature at each point in the point cloud. To make use of multiresolution information, the curvedness metric is calculated for each point at multiple scales by using different local neighbourhood sizes to fit a surface. The points that have extreme values compared to the points in their neighbourhood at that scale and extreme values in both the immediately higher and lower scales are labelled as keypoints. A confidence measure is also calculated for each point to help distinguish between important features, and transient features between scales or the effect of noise, similar to the use of feature persistence in the multiresolution feature extraction method proposed by Pauly and Gross [41]. Confidence measures are calculated by comparing each keypoint to the characteristics of the surrounding points in its neighbourhood, at the current scale as well as the immediately adjacent scales.

The feature extraction proposed here concentrates more on determining important points along distinct curvature for registration purposes. Therefore distinct key points are selected without much support to aggregate pieces of a mesh for segmentation. This algorithm

could very well determine the centre of deformations in a surface, since they would appear as significant curvatures, however a major extension would be required to apply this technique to segment the complete deformations that are the focus of this research.

Jagannathan and Miller [50] use curvedness for segmentation, to extract regions of the mesh with high curvature. The curvedness is calculated for each point in the mesh. Using iterative graph dilation and filtering of outlier curvedness values, the mesh is broken up into sub-meshes with similar curvedness values. This type of algorithm would do very well in segmenting deformations in the mesh of an automotive body part, as the curvedness values of the points belonging to deformations would be significantly higher than the points belonging to other parts of the mesh. Though it is not explicitly a multi-resolution technique, since it can detect curvatures of various sizes it is capable of extracting deformations of varying scale. Based on the results shown, the algorithm has great success in segmenting many 3D models. But the tests models have very large distinct form changes, and do not resemble the types of meshes that are the focus of this thesis. It might be difficult to predict the success of this algorithm when faced with finding subtle shallow deformations on the surface of a flat or curved mesh, especially when dealing with significant noise and acquisition artifacts. Outlier removal might also be challenging when there are not many distinct changes in the surface of the mesh to describe a deformation. However, it seems like this algorithm, especially when using curvedness as a descriptor for deformation, could be very useful for the application if those potential problems can be overcome.

Döring *et al.* tackle a similar problem to this work, by detecting deformations in car body panels [3]. The deformation extraction is only briefly explained as finding the differences between the point cloud and an inertial surface approximation of a low polynomial degree. The experiments in this paper work under similar assumptions to this thesis, in that there is no ideal model or *a priori* knowledge to compare to the model being analyzed. Surface deformations must be extracted by analysis of the model surface against what is assumed to be a smooth ideal surface instead of being compared to an existing model of what the surface should look like. This thesis is more concerned with the extraction of surface deformations than the classification, but the paper emphasizes the classification of the feature as one of many types of known deformations, and will be discussed further in section 2.4.

2.4 Classification

The surface shape analysis can extract strong deviations from a smooth surface as a deformation. Since no prior knowledge about the panel is available, and there is no ideal model

to compare the deformation information to, it is possible that regular characteristics of the panel can be identified as deformations. Regular characteristics include aesthetic features, such as curves and shapes, and functional features, such as keyholes and door handles. In order to separate these regular characteristics from deformations, such as dents and dings, a classification system must be used. The data being classified is assumed to be in the form of a point cloud or as a group of voxels (or similar volumetric grid cells).

One of the simplest methods of classification is based on thresholding. A surface deformation will have differing physical attributes than regular characteristics found on automotive body parts. The most obvious characteristic is size. The deformations in question are small and shallow, so they are of less size in all axes than any regular characteristics found in the automotive body part. By setting thresholds for the maximum size of a deformation, deformations can be separated from regular characteristics just be evaluating the size of the segmented regions.

This may be sufficient for the limited scenario of having small shallow deformations. However, to improve robustness, classification must be done involving further characteristics. As introduced in section 2.3.2, parameters such as curvedness and error in a least-squares surface can also be useful, since they are methods for characterizing surface shape, as discussed in [47] and [48].

As described in the surface shape analysis section, Döring *et al.* [3] tackle the problem of extracting and classifying deformations from automotive body parts. The segmented deformation is classified as one of many different types of possible deformations, under the assumption that the segmented region is indeed a deformation and not another feature. For the purposes of this application, it is more important to determine the difference between a deformation and a regular feature of the automotive body part, than to classify a deformation into multiple categories. This paper focuses more on the separation of training examples for a clustering or neural network technique, which is beyond the scope of this thesis.

There are several techniques which attempt to fit geometric primitives to segmented volumes [44]. Unfortunately, the majority of the regular characteristics and deformations occurring on automotive panels will not have any resemblance to basic geometric primitives. Fitting a sphere to a deformation may generate less error than fitting a sphere to regular body part characteristics like door handles and keyholes. However that is based on the assumption that deformations are symmetrical and sphere-like. This is only the case with some deformations, and such a solution would not be sufficient.

Spin image features are very useful to determine information about a point and its surrounding points, and can be beneficial when describing 3D information for classification purposes [51]. Assfalg *et al.* [52], in the context of object retrieval, use spin image features to compute a spin image signature for each point in a point cloud. This signature is a compressed set of information, describing the point with information regarding its surroundings. Instead of having such a signature for each point in the point cloud or segment, they are compressed down to a smaller set of spin signatures which should provide sufficient shape information about the point cloud or segment. This group of spin signatures is a rotation invariant description of the segment being evaluated. Such a technique can be very useful to describe shapes for classification purposes, and should be helpful in differentiating deformations and regular characteristics.

2.5 Conclusion

Several techniques were surveyed that have potential applications to the problem of determining surface deformations in automotive panels. The discussed 3D acquisition systems are important to digitize automotive panel surfaces in 3D. 3D acquisition through structured light provides a very low cost, low complexity solution, and will play a role in the experimental setup to test the deformation detection system, providing unordered point cloud data. 3D laser scanning provides a high accuracy solution, which is commonly used in industrial applications, that provides ordered point data useful to some of the surface shape analysis techniques. The deformation extraction algorithms should be robust to data from both sources such that the techniques developed in this thesis can easily be ported to different systems.

The surface reconstruction techniques are all important to gather extra information from a 3D point cloud representation of the automotive part, particularly the Delaunay triangulation and ball-pivoting technique.

The surface shape analysis section evaluated several types of techniques that could aid in extracting deformations from a 3D surface. 2D analysis was reviewed and, though it has inherent limitations when dealing with 3D data, can be very effective when using traditional edge detection and segmentation techniques provided the 3D data can be translated into a 2D image. 3D techniques are much more suited to dealing with the type of data produced by the acquisition system. Areas such as surface reconstruction, noise removal, and both point-based and mesh-based feature extraction provide an abundance of strategies and metrics to characterize surface variation in 3D data. Since there is no CAD model, and no prior knowledge of what the inspected surface is supposed to look like, there is significant challenge in

determining areas that are potential deformations. A simple comparison of the acquired surface and an ideal surface would not suffice.

The LIDAR data analysis techniques dealt with ordered point clouds, which is a useful way of analyzing cross-sections of the scanned object to look for important geometric characteristics. Though it is not robust to working with different types of data, such as unordered point clouds, and does not use much of the information generated from mesh triangulation, the idea of analyzing strips of data in 2 dimensions simplifies the problem of geometric shape extraction. This method of feature extraction inspires the contour analysis technique developed by this application, discussed in chapter 5.

The octree segmentation method can be very useful for multi-resolution analysis, and provides a robust data structure that provides statistical summaries of the surface contained in each node. These statistical summaries can determine areas of high surface variation, corresponding to surface deformations. This type of surface analysis will be enhanced and applied to the objective of this thesis in chapter 4. The octree data structure is also very important to this thesis, in the context of allowing unordered point data and a 3D mesh to be analyzed in an ordered manner, bridging the gap between techniques that can only operate on ordered point clouds and more robust techniques. This will be further detailed in chapter 5. The idea of feature persistence is also useful when dealing with multi-resolution analysis, and can be beneficial for extracting deformations of various scales while isolating them from noise and aesthetic surface curvature. This provides inspiration for an important enhancement to the octree feature extraction technique, which will be further discussed in chapter 4.4, and to the idea of segmenting and classifying features over multiple resolutions, which will be further discussed in chapter 6.

Finally, the classification section reviewed some techniques to ensure that the extracted surface curvatures are actually deformations of interest, while removing areas that this thesis is not interested in. Not all areas extracted by the techniques in surface shape analysis will be deformations of interest, so the classification must use known information about deformations of interest and the extracted information to make decisions on which areas are deformations and which are not. Though this component may not always be perfect, at least some basic techniques such as thresholding characteristics or fitting basic geometric primitives will reduce the number of falsely extracted deformations, increasing the overall accuracy of the algorithms. These techniques are adapted for classification, as the final step in the deformation detection pipeline, further detailed in chapter 6.

Chapter 3 Automated Surface Deformation Detection

The complete automated deformation detection and marking system and its experimental setup, which contains the deformation detection subsystem proposed in this thesis, is composed of several subsystems and will be discussed in section 3.1. The deformation detection subsystem is what this thesis is concerned with, and the proposed system and its components are overviewed in section 3.2. Finally, the details of the testing and the test data used are detailed in section 3.3.

3.1 Automated Deformation Detection and Marking System

The entire research project that this thesis belongs to involves the development of an automated deformation detection and marking system [5]. The primary objective is to identify deformations in an automotive panel and physically mark those deformations while the automotive part moves along an assembly line. The system is broken up into two subsystems: the deformation detection subsystem and the robotic marking subsystem.

The deformation detection subsystem, which is the focus of this thesis, analyzes the surface of the automotive panel and determines the locations of all deformations of interest. The robotic marking subsystem tracks the moving automotive panel along the assembly line, and marks the deformations with a robotic arm, based on the locations given by the deformation detection subsystem [6]. Also, the deformation detection subsystem relies upon a 3D acquisition system to provide it with 3D point cloud or mesh information. The relationships between the subsystems are shown in Figure 3.1.

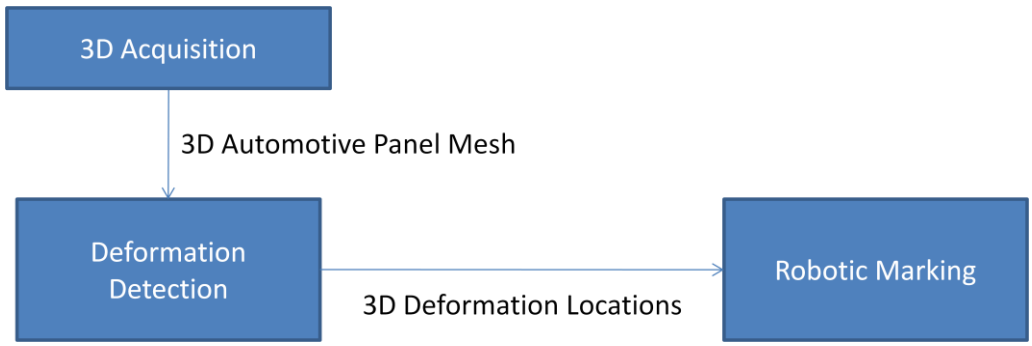

Figure 3.1. Relationship between subsystems of automated deformation detection and marking system.

The experimental setup, which is a miniature assembly line used in the lab for testing, is discussed in section 3.1.1. The subsystem dealing with 3D acquisition, which provides the 3D

data to the deformation detection subsystem, is detailed in section 3.1.2. The robotic marking subsystem is explained in section 3.1.3.

3.1.1 Experimental Setup

The automated deformation detection and marking system is created on a smaller scale in a lab setting. This serves as a test bed for the developed techniques, and demonstrates that they can work in a real-world setting. The entire system, if proven effective on a small scale, can be adapted to large-scale industrial settings. An image of the setup is shown in Figure 3.2.

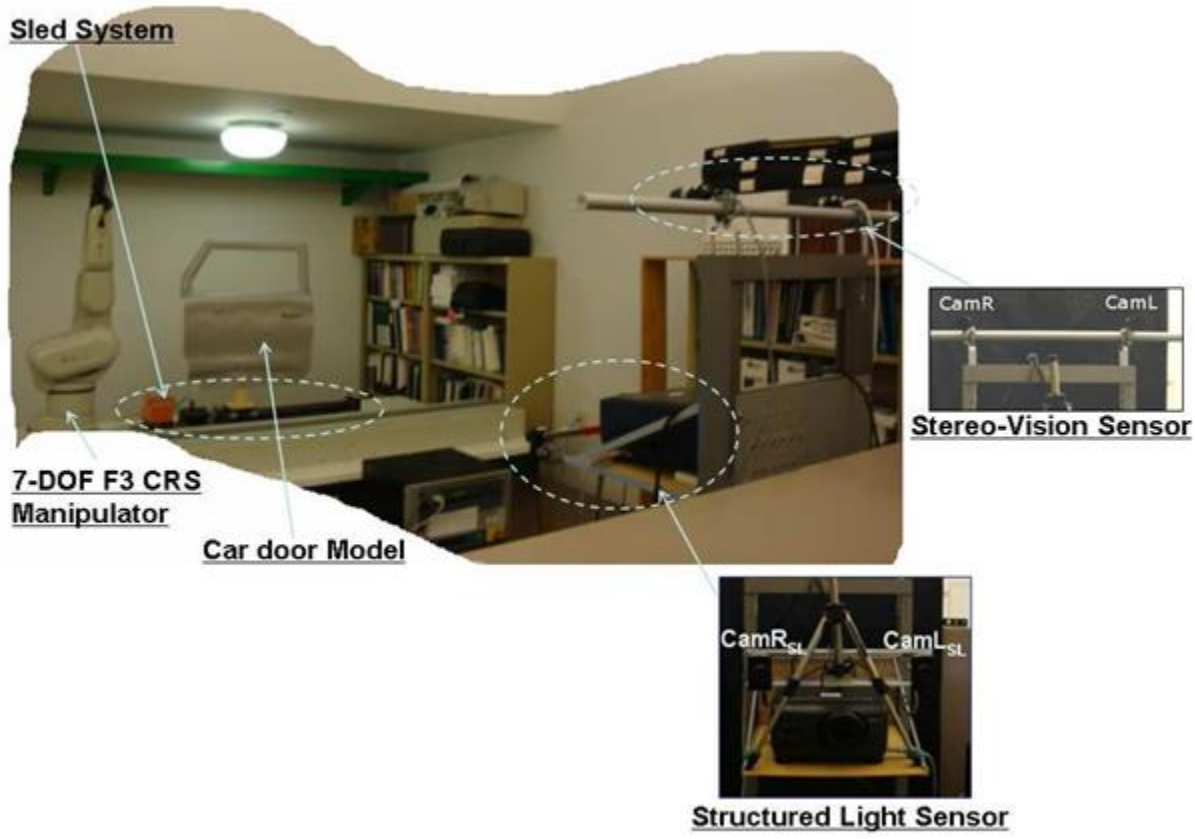

Figure 3.2. Experimental lab setup.

To represent the idea of a moving assembly line, a PC-operated sled system is used and simulates a shortened conveyor in a lab setting. One of several real or imitation automotive panels is mounted on the sled system to imitate a real automotive panel. The imitation models contain the general characteristics of an automotive panel that would be found on an assembly line, and are used to test the complete system.

At the beginning of the assembly line, when the automotive panel is static, a structured light sensor is used to generate a dense 3D reconstruction of the surface of the automotive panel [53, 54]. The automotive panel must remain static for the structured light sensor due its requirement of the model being in fixed pose and its time-consuming nature in generating a high-resolution scan. The final 3D mesh is outputted by the 3D acquisition system, and serves as the input for the deformation detection subsystem. Further details on the acquisition system will be explained in section 3.1.2.

The deformation detection subsystem processes this 3D data, and uses multi-resolution feature extraction and segmentation techniques to acquire the location of the deformations. The location and orientation of these deformations are passed on to the robotic marking subsystem.

The panel moves along the sled system to test the robotic marking system [6]. The robotic marking subsystem uses a stereo pair of cameras to estimate the pose and motion of the automotive panel as it moves along the assembly line [55]. Then, based on the location provided by the deformation detection subsystem, the CRS robotic manipulator is moved and angled to mark the deformation on the surface of the automotive panel. Further details about the robotic marking system are given in section 3.1.3.

3.1.2 3D Acquisition

The 3D acquisition system is a fundamental component of the automated deformation detection and marking system, as it provides the only input used by the deformation detection system to identify the location of deformations of interest. In order to generate a dense 3D reconstruction of the evaluated panel surfaces, a structured light sensor (SLS) is used in the experimental setup as the 3D acquisition system [53, 54]. The SLS is shown in Figure 3.3. Note that the SLS is not part of the work done in this thesis, and is used only as a tool to provide the deformation detection system with a mesh generated from scanning a real-world object.

Figure 3.3. Structured light sensor.

A pair of cameras is mounted on top of a video projector. The video projector projects a 2-dimensional pseudorandom colour pattern of squares onto the scene to be reconstructed in 3D [18]. In the case of this thesis, the pattern is projected onto the surface of the automotive panel to be analyzed by the deformation detection system, as shown in Figure 3.4.

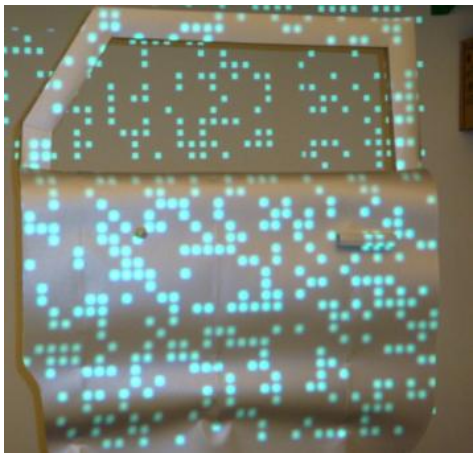

Figure 3.4. SLS pattern projected onto automotive panel.

As a dense reconstruction of the scene requires a sufficient amount of feature points, which may be lacking in a typical automotive panel containing only subtle surface variations, the projected pattern serves as a set of artificial feature points to be used by the stereoscopic reconstruction system. The number of squares in the pseudorandom pattern is fixed for optimal performance, however to increase resolution, they are shifted across the surface and reacquired by the stereoscopic system after each shift, resulting in an increased point density in the

reconstruction. The stereoscopic reconstruction uses traditional 2D feature matching techniques and triangulation to determine the distance of each point from the stereoscopic cameras. By determining the distance of each point, a 3D reconstruction can be made in the form of a 3D point cloud. Surface reconstruction, through the use of the ball-pivoting algorithm that was discussed in chapter 2, is used to generate a mesh triangulation out of the acquired 3D points. The output of this module is given as input to the deformation detection subsystem, which is the starting point for the original work presented in this thesis.

The structured light acquisition system is very susceptible to noise, especially when operating at high enough resolutions to acquire small deformations on a large surface. It is also very susceptible to artifacts along the boundaries of the panel. Finally, the maximum resolution is often not high enough to accurately acquire deformations smaller than 1cm in diameter and 1cm in depth. These limitations are shown in the models acquired for testing, shown in section 3.3.2. There are significant constraints of the overall deformation and detection marking system because of the structured light sensor, though it is sufficient to demonstrate that such a system could work with advances in the acquisition module. For these reasons, it is important that the deformation detection subsystem is made to be independent of the acquisition system, so that it can be applied with ease on more accurate data acquired by a different sensor. At the same time, to successfully demonstrate the deformation detection subsystem's robustness, it must be effective despite the described limitations of the acquisition system.

3.1.3 Robotic Marking Subsystem

The robotic marking subsystem, developed by Borsu [5, 6], is a major component of the overall integrated solution, and uses the information provided by the deformation detection subsystem to mark the deformations of interest on the automotive panel. The first goal of the subsystem is to estimate the motion and pose of the moving panel along an assembly line [55], which is represented by the PC-operated sled system in the experimental setup.

A stereoscopic pair of cameras comprises the vision system that is used to determine the pose and motion of the automotive panel in 3D space. Note that this stereo pair is different than the one used for the SLS. A set of feature points, which are chosen on the automotive part by an operator and refined by the Shi and Tomasi corner detector [56], are extracted at positions relative to both cameras such that their 3D positions are calculated relative to the stereo system through triangulation, effectively estimating the shape of the rigid automotive panel in 3D space. This estimation is done once, with the first frame grabbed with each of the cameras, and will be used as the basis of the pose and motion estimation in subsequent

frames. The subsequently acquired frames are used to estimate the motion of the panel and to recalculate the pose of the rigid body. Using the pyramidal Lucas-Kanade tracker [57, 58], along with the geometric constraints of the rigid body, the transformation that the panel has undergone between successive stereo pair frames is calculated.

In order to use the pose and motion estimation results in conjunction with the robotic manipulator and the deformation locations provided by the deformation detection system, two inter-calibrations must be done. The deformation detection system results in the 3D locations of deformations relative to the SLS, in the coordinate system used by the SLS. The first inter-calibration is done between the pose and motion estimation system and the SLS. Through this inter-calibration, the deformation positions can correspond to the representation of the automotive panel relative to the pose and motion estimation system. For the robot to mark locations on the automotive panel, it must operate relative to the pose and motion estimation system. So the second inter-calibration is done between the pose and motion estimation system and the base of the robotic manipulator.

Given an input of the deformations locations relative to the SLS and the stereoscopic view of the automotive panel moving along the assembly line, the robotic marking subsystem can accurate track the panel and mark the deformations.

3.2 Deformation Detection Subsystem

The proposed system takes a 3D mesh as an input, and outputs the pieces of the mesh which are deformations of interest along with whether they are a ding or a dent. Since no CAD model is provided of the ideal surface, the proposed system must locate and classify the deformations of interest using assumptions based on common characteristics of dings and dents. Since some assumptions regarding size and scale of deformations cannot be made without more information, a minimal and intuitive set of parameters must be set by the operator to ensure accurate detection with minimal human interaction. This also ensures that design features of the automotive panel are not accidentally extracted as deformations, since they are generally much larger than the deformations of interest and can easily be separated by size and scale. The 3D mesh is provided by the 3D acquisition system, described in section 3.1.2, however the deformation detection subsystem must be robust to the type of data acquired, meaning it should work whether the input mesh is reconstructed from an unorganized point cloud or a range image. The outputs of the proposed system are passed onto the robotic deformation marking system, described in section 3.1.3.

The proposed system contains 3 major components, as shown in Figure 3.5.

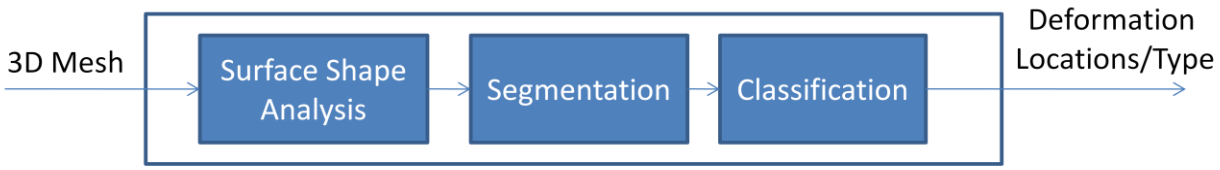

Figure 3.5. System diagram of proposed deformation detection system.

The surface shape analysis component is tasked with dividing the 3D mesh into pieces and analyzing each piece for the amount of deformation contained in that piece. This results in a multi-resolution description of estimated deformations in the mesh. The surface shape analysis can be one of two techniques, each of which has its own strengths and weaknesses. The first method, the octree-based technique, will be discussed in chapter 4, while the second method, the contour analysis technique, will be discussed in chapter 5.

The segmentation component combines pieces from the surface shape analysis which seemingly belong to the same deformation. It separates deformation areas of importance from the rest of the mesh, and results in a set of segments, each of which is a piece of the mesh that resembles a deformation. There will be a different type of segmentation, corresponding to whether the octree-based surface shape analysis was used or the contour analysis was used. This segmentation can be executed at either a single resolution or it can make use of multi-resolution information for a more informative analysis.

The classification component classifies each segment from the segmentation as either a ding or dent. Also, it removes segments which do not meet the criteria of being a deformation of interest, outputting only segments which are deemed to be dings or dents. These non-deformation segments can be areas of noise, acquisition artifacts or design features on the automotive panel.

Since there is some overlap between the segmentation and the classification components, both components will be discussed in chapter 6.

Two different pipelines will be evaluated to achieve the location of deformations. The octree-based system, shown in Figure 3.6, is the first pipeline and uses more traditional 3D feature extraction techniques to locate deformations in the 3D mesh. It contains techniques specific to octree-based feature extraction for its surface shape analysis, segmentation, and classification components. The second pipeline is the contour analysis system, shown in Figure 3.7, and contains techniques specific to the proposed contour analysis methodology for surface shape analysis and segmentation. The second pipeline does not require classification, as it is done inherently in the surface shape analysis and segmentation modules. The final

recommendation on which system is best suited for deformation detection will be provided in the conclusion to Chapter 6, in section 6.6.

Figure 3.6. System diagram of octree-based deformation detection system.

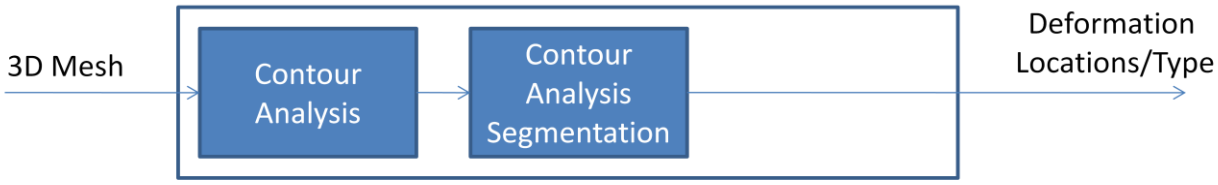

Figure 3.7. System diagram of contour analysis deformation detection system.

3.3 Testing Overview

This section will overview the details of how the deformation detection system is tested. The criteria to determine the effectiveness of each subsystem in the deformation detection system will be discussed in section 3.3.1. The data that will be used for the testing is presented in section 3.3.2.

3.3.1 Effectiveness

The complete system can be said to be effective if it can determine the locations of the deformations and classify them accurately as dings and dents. It is also desirable that the system extracts only the deformations, as the extraction of non-deformation areas is not ideal. Different applications have different definitions of what is required to be extracted. Some applications require only a single point in the deformation, while others require all of the points in the deformation. Though no quantitative criteria will be given, this application's goal is to extract only deformations, in part or in full, while being able to correctly classify them as dings or dents.

Each subsystem must also be given criteria for effectiveness. There are three subsystems that must be tested: surface shape analysis, segmentation, and classification.

The surface shape analysis uses one of two techniques: the octree-based method and the contour analysis method. For the octree-based method, the criterion is that all areas

containing any piece of the deformation are extracted, while minimizing the area that is extracted surrounding the deformation. This allows the segmentation to accurately create a segment containing just the deformation, and classification will determine the deformation type based on the information contained in the segment. If the entire deformation cannot be extracted, the segmentation will not create a segment for the complete deformation, and the classification will behave with less effectiveness as it will not have the correct information to determine whether the contained data is a dent or a ding. For the contour analysis method, since the classification of deformation type is built into the technique, any amount of contours containing the deformation can be detected as long as the classification is correct, while minimizing the amount of non-deformation contours that are detected.

The criterion to determine the effectiveness of the segmentation is that all areas extracted from the surface shape analysis, which contain the same deformation, must be grouped together. Any additional processing done by the segmentation which can aid in determining the deformation type or removes non-deformation segments is helpful but not required.

Finally, the criterion to determine the effectiveness of the classification system is primarily how accurately it can classify deformation segments as dings and dents. Its secondary ability to remove additional non-deformation segments is not a requirement.

3.3.2 Test Data

Test data is important to determine the effectiveness of the system. The unit of measurement is specific to the scanner, so this thesis will designate *mesh units* (*mu*) as the general unit of measurement. Different scanners may have different conversions from their coordinate system to a real-world coordinate system. The designed deformation detection system operates independently of the real-world coordinate conversion and does all calculations relative to the provided unit of measurement, and the robotic marking system is calibrated relative to that unit of measurement also. In the case of the SLS used to provide data for this thesis, 1 mesh unit is equal to 1mm. This is important when dealing with the size and magnitude of deformations.

The 3D acquisition system used does not provide ideal meshes for this application. The reflective characteristics of the surfaces, the subtle variations in the surface, and the large distance the panel is kept from the acquisition system, cause the acquisition system to introduce an abundance of noise and acquisition artifacts. Regardless of the sensor's limitations, the designed algorithms will need to be demonstrated to work even when these undesired

characteristics are present. These real-world meshes will serve as test cases for non-optimal acquisition and surface characteristics because of the abundance of acquisition errors.

The first real world mesh is a desktop CPU panel modified by hammering 3 dents into it. Though this is not an automotive part, it simulates real-world deformations on a relatively flat surface. The panel is 20cmx15cm and each dent is circular, with dimensions of approximately 2cm in diameter and 0.25cm in depth. The panel was scanned at three different resolutions: the highest resolution contains 14626 points, the medium mesh contains 3647 points, and the low resolution mesh 398 points. Also, the resulting meshes were filtered using a basic Laplacian averaging filter. The unfiltered panel meshes are shown in Figure 3.8 and the filtered meshes are shown in Figure 3.9.

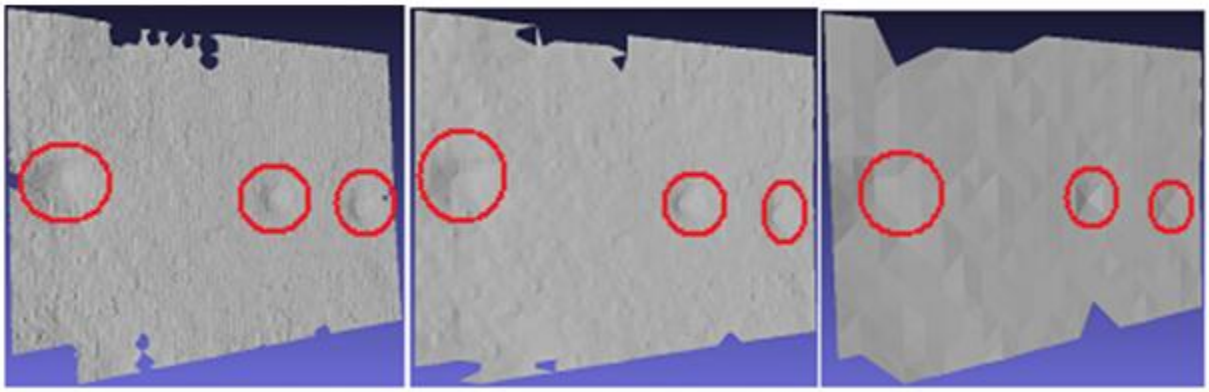

Figure 3.8. CPU panel high resolution (Left), medium resolution (Center), low resolution (Right).

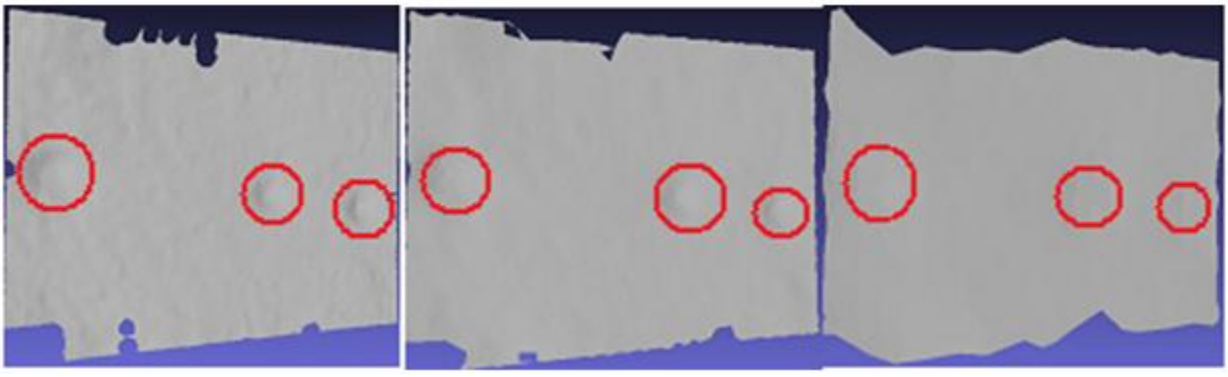

Figure 3.9. Laplacian filtered CPU panel high resolution (Left), medium resolution (Center), low resolution (Right).

The amount of surface variation, holes along the boundaries, and noise is visible in the images of the unfiltered CPU panel mesh. Also, at the lowest resolution, the deformations are barely visible, and are defined by very few points with very minimal variation. Since it might be

unrealistic to expect an accurate extraction of deformations from these meshes, filtered versions are created that use a basic Laplacian smoothing filter to remove the noise while maintaining the deformations.

A mock car door was crafted out of cardboard, consisting of a curved body, a door handle, and a window frame. The door is approximately 70cmx78cm. Three dings, made up of paper were stuck to the door at various positions, where each ding is circular and approximately 1cm in diameter and 1cm in depth. This car door was scanned at two different resolutions: with the high resolution mesh being 32202 points and the low resolution one being 13944 points. Also, the resulting meshes were filtered using a basic Laplacian averaging filter. The unfiltered meshes are shown in Figure 3.10 and the filtered meshes are shown in Figure 3.11.

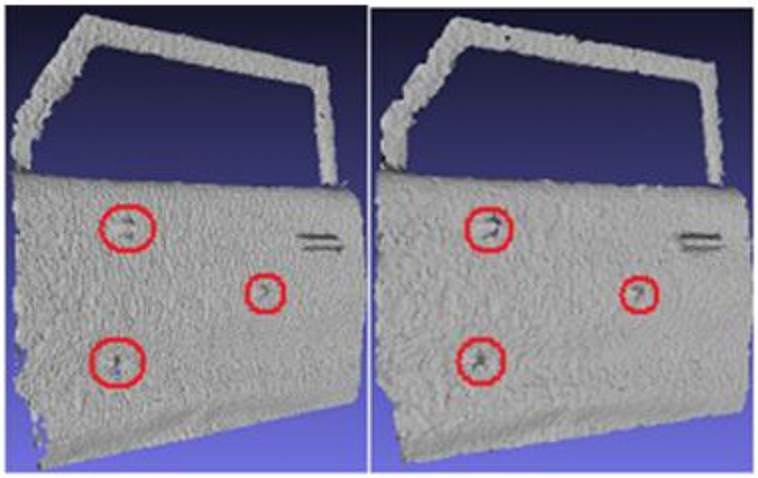

Figure 3.10. Car door high resolution (Left), low resolution (Right).

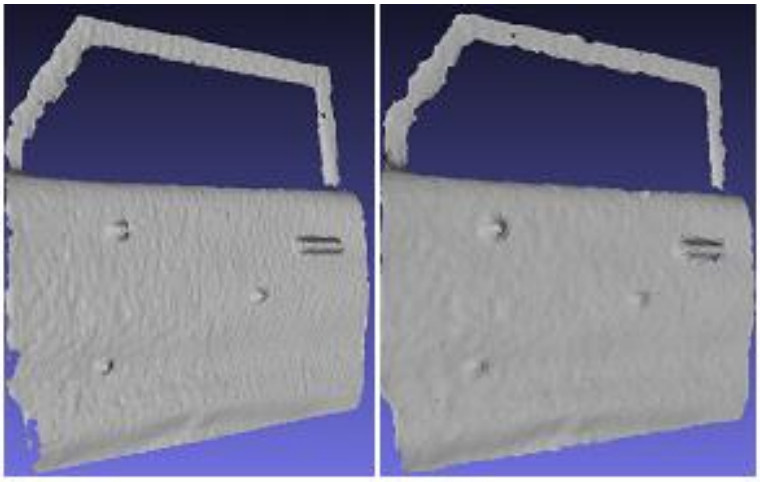

Figure 3.11. Laplacian filtered car door high resolution (Left), low resolution (Right).

The car door is acquired relatively well, with the deformations, door handle, surface variation and window frame all appearing. There are very few holes, but lots of surface variation along the borders due to the acquisition errors. Also, there is a significant amount of noise, which may interfere with isolating the deformations from parts of the noise as the peaks of the noise are almost as high as the peaks of the deformations. The filtered versions reduce the noise levels, minimizing the peaks and further separating them from the deformation peaks. Also, though the variation along all of the borders is reduced, it is still significant, especially along the window frame.

These real-world meshes might be sufficient to test the proposed system's behaviour when applied upon meshes with noise, acquisition artifacts, and real-world characteristics. However, it is not a complete enough set of test data to test the various situations that the system might be exposed to. For this reason, a set of artificial test meshes were generated, with characteristics that were not found in the acquired real-world meshes, that may occur in other real-world meshes. These meshes resemble deformations of interest, of various sizes and scale, under different surface conditions. These meshes also attempt to test the functionality of the system in working under ideal acquisition scenarios where there is no noise and no acquisition artifacts. Note that these meshes were not acquired by a real scanner, so there is no conversion between their unit of measurement and a real-world unit of measurement. As a result, when referring to actual measurements, again, the term *mesh unit* will be used.

A flat mesh with a small dent was created. A flat mesh with a large ding was also created. These meshes are shown in Figure 3.12.

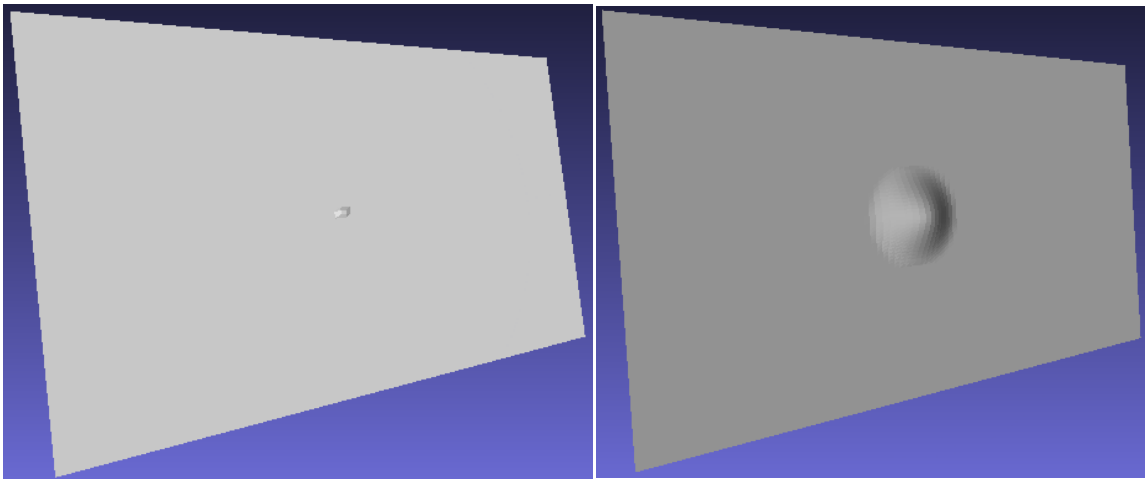

Figure 3.12. Flat mesh with small dent (Left). Flat mesh with large ding (Right).

A curved mesh with a small dent was created, and a curved mesh with a large ding was created. These meshes are shown in Figure 3.13 and Figure 3.14, respectively.

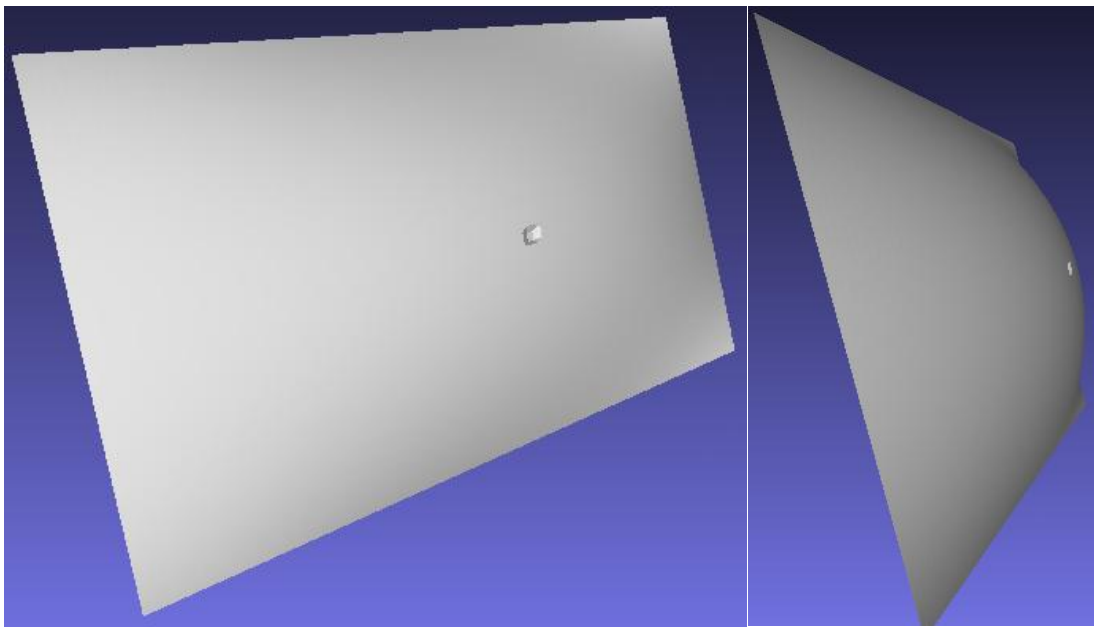

Figure 3.13. Curved surface with small dent.

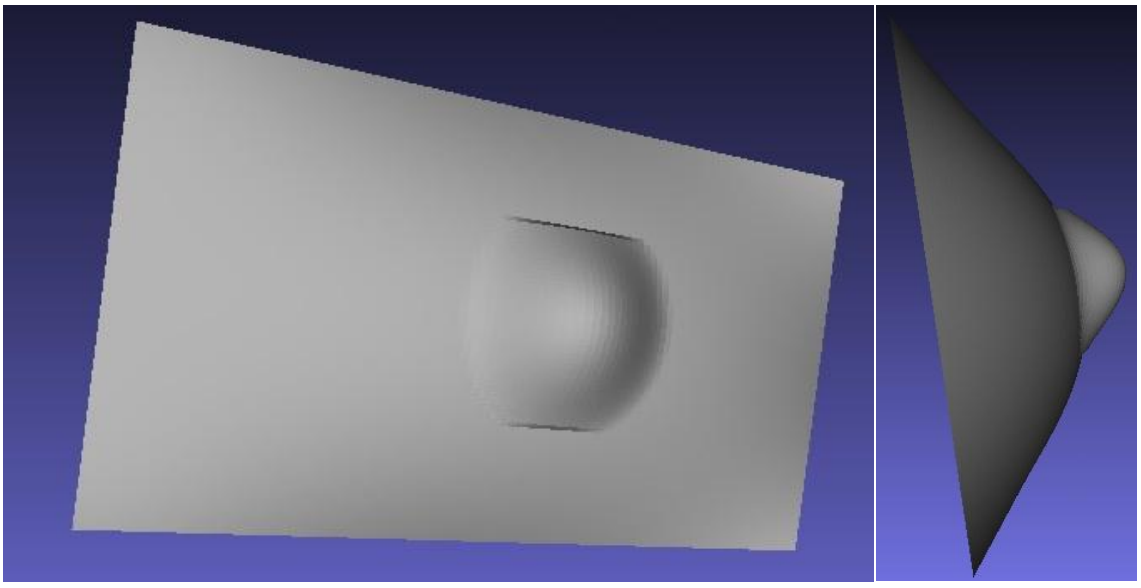

Figure 3.14. Curved surface with large ding.

The flat meshes are used to determine if the designed algorithms can detect small scale deformations as well as large scale deformations. The curved meshes are used to determine if

the designed algorithms can detect deformations in spite of a curved or uneven surface around it.

An additional artificial mesh was created specifically for the contour analysis technique, known as the recursion test mesh. A flat mesh is created with large ding-shaped curvature, and a small ding deformation on top of the ding-shaped curvature. This situation can be handled by appropriate thresholding in the octree-based surface shape analysis technique. However, due to the nature of the contour analysis algorithm, the small deformation would be difficult to handle. This mesh is used to test the contour analysis' ability to isolate deformations which may reside on deformation-like surface curvature. It is shown in Figure 3.15.

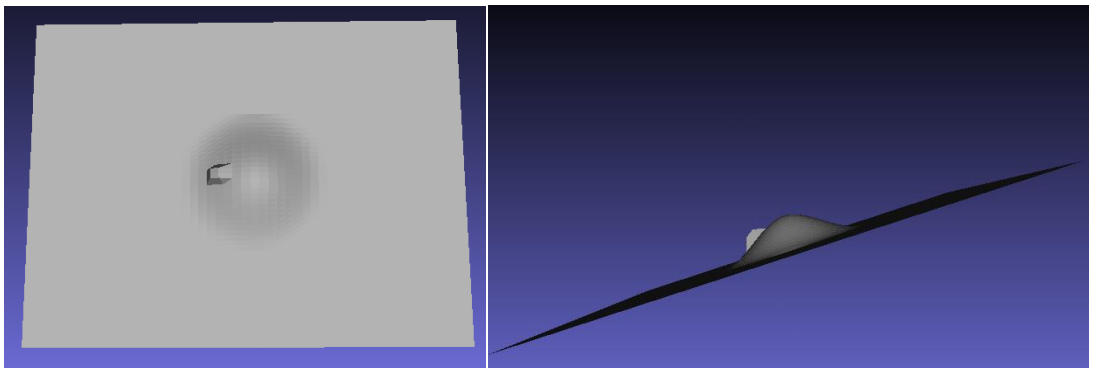

Figure 3.15. Recursion test mesh.

Chapter 4 Surface Shape Analysis Method Based on Octrees

The most important component of the defect detection system is surface shape analysis. The goal of this module is to break the mesh up into pieces and determine which of those pieces likely belong to the defects in question, as shown in Figure 4.1.

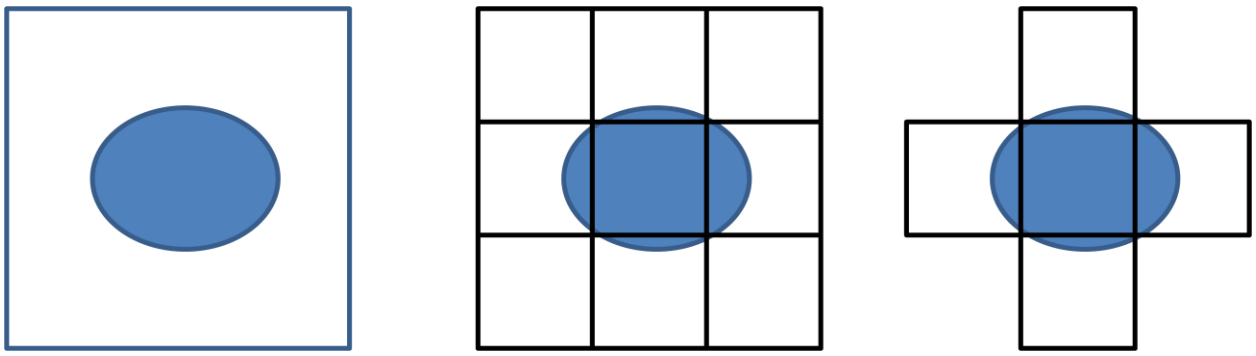

Figure 4.1. Mesh with oval feature (Left). Mesh broken into pieces (Center). Feature extraction results (Right).

The result of the methods presented here are the labelling of all parts of the mesh as either belonging to a feature or not. There are also sub-properties that are unique to the different techniques, which can be used to aid in the feature grouping phase. This module does not determine what collection of mesh pieces define a deformation, however it will determine which mesh pieces likely contain part of a deformation. The output of this module is passed to the segmentation phase, to determine which collection of mesh pieces define a deformation.

Two methods are considered and examined as candidates for surface shape analysis, each with their own strengths and weaknesses. This chapter introduces the first method, which is based on octrees. The octree-based technique divides the mesh into cubic volumes and analyzes the mesh contained in those volumes to determine if they belong to a feature. It is also the selected surface shape analysis technique for the octree-based deformation detection pipeline.

This thesis introduces a set of improvements to the octree-based segmentation method proposed by Woo *et al. [40]*, explained in section 2.3.2. The original technique represents the entire volume surrounding the point cloud in the root node of a tree. Then, by evaluating the standard deviation of the point normals, σ, against a threshold, it determines if there should be a subdivision of that volume. If σ is higher than a threshold, that volume is subdivided into eight octants. Each of those octants is a volume, and is represented by a child node. The points in the original volume are redistributed into the new octants, based on their position, and stored in

the child node which represents the new volume it belongs to. This process is repeated for each node, until a tree of sufficient depth is generated or no more volumes require subdivision.

The original technique is designed to find significant changes in a 3D point cloud, such as sharp edges. Deformation detection requires detection to slight variations in a smooth surface, which may not be consistent over multiple resolutions. For this reason, the technique must be enhanced. Three major aspects are introduced to enhance the algorithm's flexibility and performance for the purposes of the application considered in the present work: triangle-based analysis rather than point-based analysis of surface shape, non-uniform weighting of surface normals, and incremental thresholding. These enhancements will be discussed in section 4.1, 4.2, and 4.3 respectively. A fourth improvement is also made, that uses the octree to aggregate multi-resolution information into performing the feature extraction after the tree generation is complete. This will be discussed in section 4.4.

4.1 Triangle-based Analysis

The original method [40] uses the point cloud, with knowledge of the reconstructed surface, to calculate the appropriate values for subdivision of the octree. The calculation of the point normal uses all the triangles surrounding the point, and the subdivision of the octree uses the standard deviation of the point normals contained inside each node. The triangles surrounding a point will provide several pieces of information about the surface, yet reducing this information to a point normal using a simple averaging calculation acts as a smoothing filter, inherently inducing a loss of valuable surface information. For some applications, such a smoothing filter can be useful, especially in the case of noise. However, for the purposes of this thesis where deformations to be detected are of a small size compared to the remainder of the panel surface curves, such a filter should not be included directly in the feature extraction technique.

Each triangle contains a surface area which is described by a surface normal. However by accumulating several triangles into a surface normal being represented at a point, a larger surface area is being described by the same value, which means it may be less specific in describing the surface. An example of this is shown in Figure 4.2. For these reasons, a first improvement that is proposed consists of using more information to describe the surface of a model by using the triangles for surface calculations rather than points, since they describe smaller areas of surface. This provides higher resolution information at the same level of an octree, compared to that of the original technique.

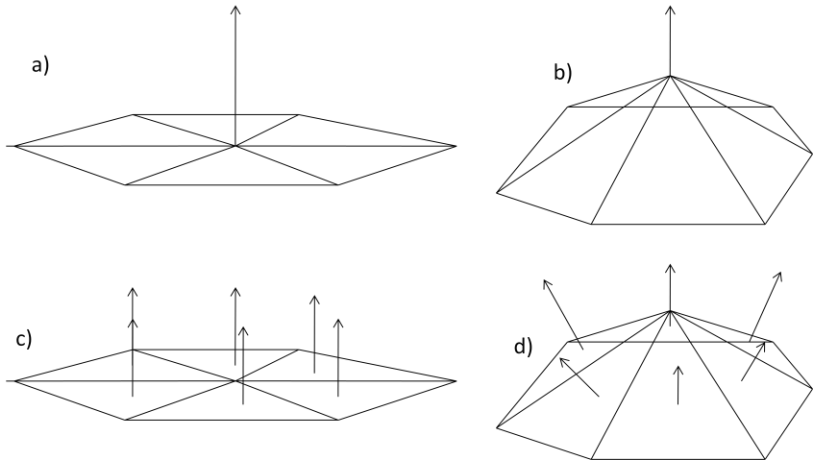

Figure 4.2. a) Point normal describing flat surface. b) Same point normal describing non-flat surface. c) Triangle surface normals describing flat surface d) Different triangle surface normals describing non-flat surface.

A limitation of this improvement is that it can be difficult to perform the subdivision of volumes, since triangles may belong to multiple subdivided volumes. In the original approach of Woo *et al*., points can easily be redistributed amongst subdivided volumes, and since none of the volumes overlap no points will be shared between two volumes. The difficulties that appear when considering the redistribution of triangles instead of points can be separated into two problems: determining if a volume contains a triangle, and triangle re-allocation. Those problems are discussed in section 4.1.1 and 4.1.2, respectively.

4.1.1 Determining if a Volume Contains a Triangle

The first major problem is that calculating whether a triangle is contained in a volume can be complicated. Two scenarios can be considered. The first scenario is if one or more of the triangle's vertices are contained in a volume, at least part of the triangle is definitely contained in the volume, as shown in Figure 4.3. Determining if a point is contained in a volume is a trivial task, so the first scenario is not difficult to resolve. The second scenario is if none of the vertices of a triangle are contained in a volume, it is still possible that the triangle passes through the volume, as shown in Figure 4.4. This scenario is more complicated, and requires extensive calculation to determine if a triangle intersects with a volume. Such a calculation, occurring for every triangle in the surface, can slow this technique down considerably. For the purposes of the current application, the second scenario is ignored, and the check to determine if a triangle with no vertices in a volume still intersects that volume is not done. Currently, this enhancement determines if a triangle is in a volume by only checking if any of the points that belong to that triangle are contained in the volume. If at least one point is contained, that

triangle is said to be contained in the volume. Though only one of the two scenarios is evaluated, this is still an improvement over the original technique, since only points were evaluated, this information would be ignored anyway if none of the points are found in the volume.

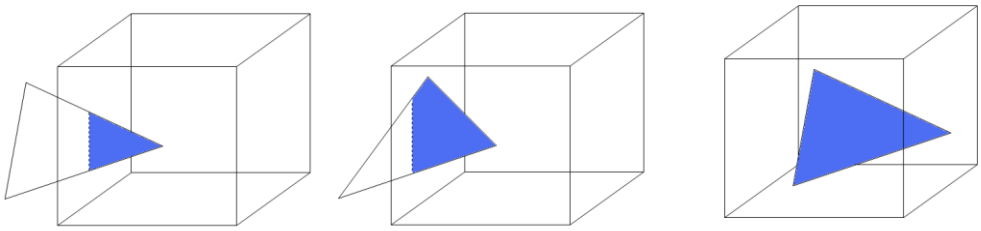

Figure 4.3. One vertex inside the volume (Left). Two vertices inside the volume (Middle). All three vertices inside the volume (Right).

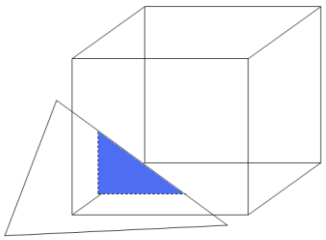

Figure 4.4. No vertices inside the volume, but triangle still intersects the volume.

4.1.2 Triangle Re-allocation

The second major problem to overcome is that sometimes only a part of a triangle belongs in a volume. If part of the triangle is contained in a volume, there are several options for subdivision. The first option is to only evaluate triangles that are wholly contained in a volume. The second option is to allocate triangles that are shared between multiple volumes to one of those volumes, either randomly, or based on some criteria. The third option is to copy the triangle such that it belongs to all volumes that it is shared between. The fourth option is to partition the triangle such that only the part of the triangle contained in a volume is included.

The first option would contradict the reason why the triangles were being used in the first place, in that the algorithm requires three points of a triangle to be contained in a volume to be evaluated at all. This limitation, in theory, will reduce the amount of information used for the surface shape analysis in each volume. The second option could provide inconsistent results, since triangles are allocated in one volume or another based on some criteria. The volume containing the greater part of the triangle getting the triangle exclusively would be the most reasonable criterion for allocation of the triangle, however even this would cause a loss of information for one of the volumes, again, contradicting the initial reason for the inclusion of

triangles. Also, the calculations may skew slightly as some information from outside the volume, which is the piece of the triangle belonging to another volume, is being used to calculate information about the surface contained inside the volume. The third option skews the calculations just as the second option did, however a maximum amount of information is being used to determine the surface shape in each of the volumes. The fourth option, mathematically, would be the best option as only the surface contained in the volume affects the calculation for that volume, solving all of the limitations of the previous three options. However, such a calculation, for each triangle, would be extremely time consuming and would impede the performance of the algorithm. For these reasons, the third option appears as the best option for the purposes of the application considered in this work.

Though no immediate results can be shown for this procedure, the proposed enhancement is crucial for the extra enhancements described later in section 4.2 and 4.4.

4.2 Non-Uniform Weighting of Surface Normals

The second strategy to enhance the performance of the algorithm proposed in [40] is to use information about the size of the triangles represented by the surface normals to improve the σ calculation from Eq 2.5, as reported in [5].

In the case of non-uniform sampling density, the original technique will give equal weight to each point. Sometimes there will be more points to describe a certain area within a volume, and the variation in that area will be more strongly accounted for in the σ calculation than another area, even if there is no disparity in the surface area represented by these two areas as shown in Figure 4.5. A solution to this problem is determined based on the proposed enhancement described in section 4.1.

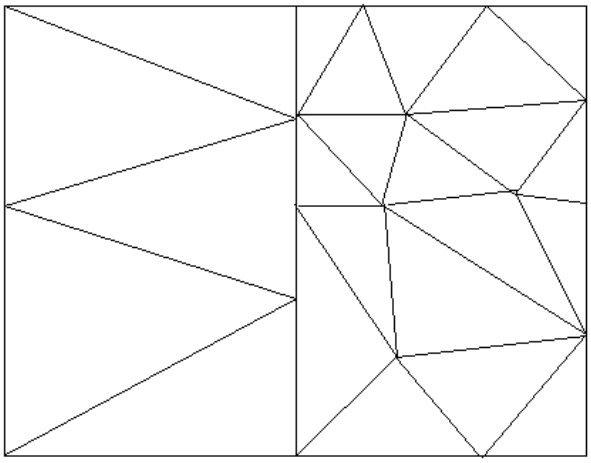

Figure 4.5. Non-uniformly distributed points. Right side of mesh contributes more to σ value than left side due to more points/triangles.

Some triangles are large, and should contribute more to the σ value, since they contribute more to the surface. The improvement is accomplished by using the area of each triangle as a weight to calculate the mean normal and σ values. This solution helps to minimize the effect of small noisy areas, to overcome the effect of non-uniformly distributed points, and to provide a more accurate representation of the variation over the region being analyzed. The evaluation of σ facilitates the partitioning process, where high values indicate a potential feature, and low values indicate a relatively smooth, and ultimately flat surface.

First, the area of each triangle is calculated:

$$a_i = \frac{1}{2}|v_{i1} \times v_{i2}| \tag{4.1}$$

where v_{i1} and v_{i2} are any two of the edge vectors that define triangle T_i. Then the weighted average normal is calculated over all the triangles that are contained within a given node of the octree, with the areas of each triangle as weights:

$$A = \sum_{i=0}^{n} a_i \tag{4.2}$$

$$\bar{N} = \sum_{i=0}^{n} \frac{a_i}{A} N_i \tag{4.3}$$

where n is the number of triangles contained in the volume represented by the node of interest, $\bar{N}$ is the weighted average normal, and N_i is the unit normal of each of the triangles. Finally, the weighted standard deviation, σ, of the normals can be estimated as:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n} a_i (N_i - \bar{N})^2}{A}}$$

$$= \sqrt{\frac{\sum_{i=0}^{n} a_i (x_i - x)^2 + \sum_{i=0}^{n} a_i (y_i - y)^2 + \sum_{i=0}^{n} a_i (z_i - z)^2}{A}} \quad (4.4)$$

As in the original method, if σ is greater than the defined threshold for that resolution, the volume is subdivided for further investigation at a higher resolution, until the entire tree has been generated.

The selection of the threshold is important. Most importantly, it must be sufficiently low to extract as much of the surface area of the deformations as possible. But it must also be sufficiently high such that the extraction of non-deformation features is minimized and the areas between adjacent deformations are not extracted such that there is separation. If all of the extracted features are to be classified as deformations, then a very high threshold can be used to only mark the sharpest part of the deformation, which is usually near the center. But in real world data, any area of high variation may be extracted, including noise, acquisition artifacts and other non-deformation features, so it is important to get as much of the surface area of the deformations of interest as possible, such that they can be classified while other extracted areas of high surface variation can be removed by the techniques discussed in chapter 6.

This improvement makes the standard deviation values more accurately represent the amount of deformation within a volume. Surface variations over a small area will be weighted less, since the areas of those triangles will be small relative to the entirety of the surface contained in the volume. With this improvement, a higher accuracy in determining nodes that belong to features, and a much better separation between feature nodes and non-feature nodes are expected.

4.2.1 Results

Before results are presented, it is important to visually demonstrate how the octree algorithm extracts deformations. The octree-based algorithm is applied to the artificial flat mesh with a large deformation to demonstrate how it strips away non-feature areas as the octree level increases, and the results are shown in Figure 4.6.

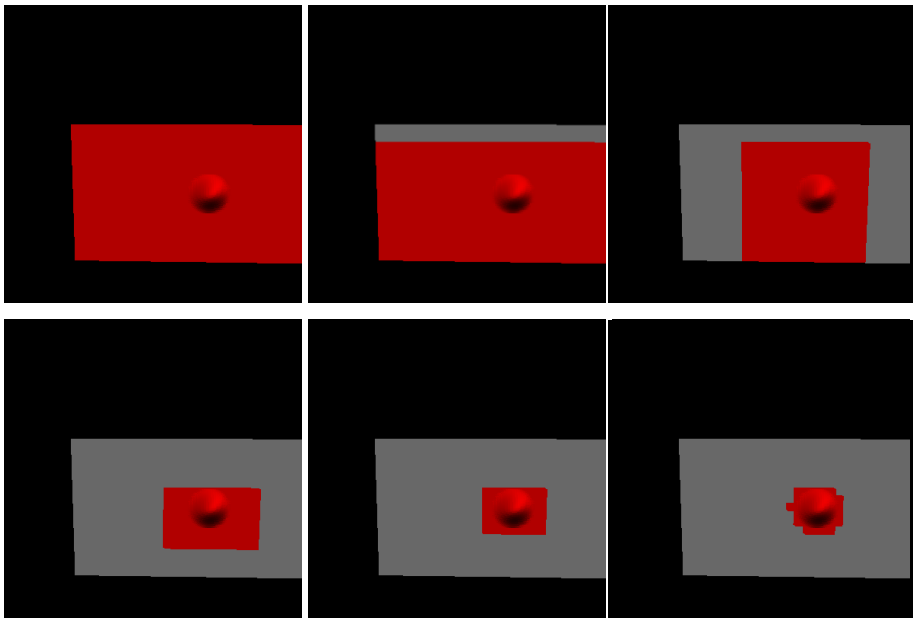

Figure 4.6. Feature extraction on artificial flat mesh with a large deformation: Octree level 1(Top Left). Octree level 2 (Top Center). Octree level 3 (Top Right). Octree level 4 (Bottom Left). Octree level 5 (Bottom Center). Octree level 6 (Bottom Right).

As the octree level increases, more and more of the mesh is stripped away. Nodes that do not meet the threshold are removed, and as the volume of those nodes get smaller through each increasing octree level, the extracted surface corresponds more to the deformation's shape.

The discussed enhancement of the octree algorithm was applied to the experimental data, and the results are presented below. Optimal thresholds for the various 3D meshes were experimentally determined. For the artificial meshes, many thresholds give similar results, while for the real world meshes, a threshold was selected as the maximum threshold that all deformations could be extracted. For the real world meshes, if the threshold was selected to be any higher, at least one of the deformations would not be extracted.

This enhanced algorithm is applied to the artificial flat mesh with a small deformation and also to the artificial flat mesh with a large deformation. A threshold, a, is set at 0.005, for both meshes. The algorithm is also applied artificial curved mesh with a small deformation and artificial curved mesh with a large deformation. A threshold for those meshes, a, is set at 0.12. The results are shown in Figure 4.7.

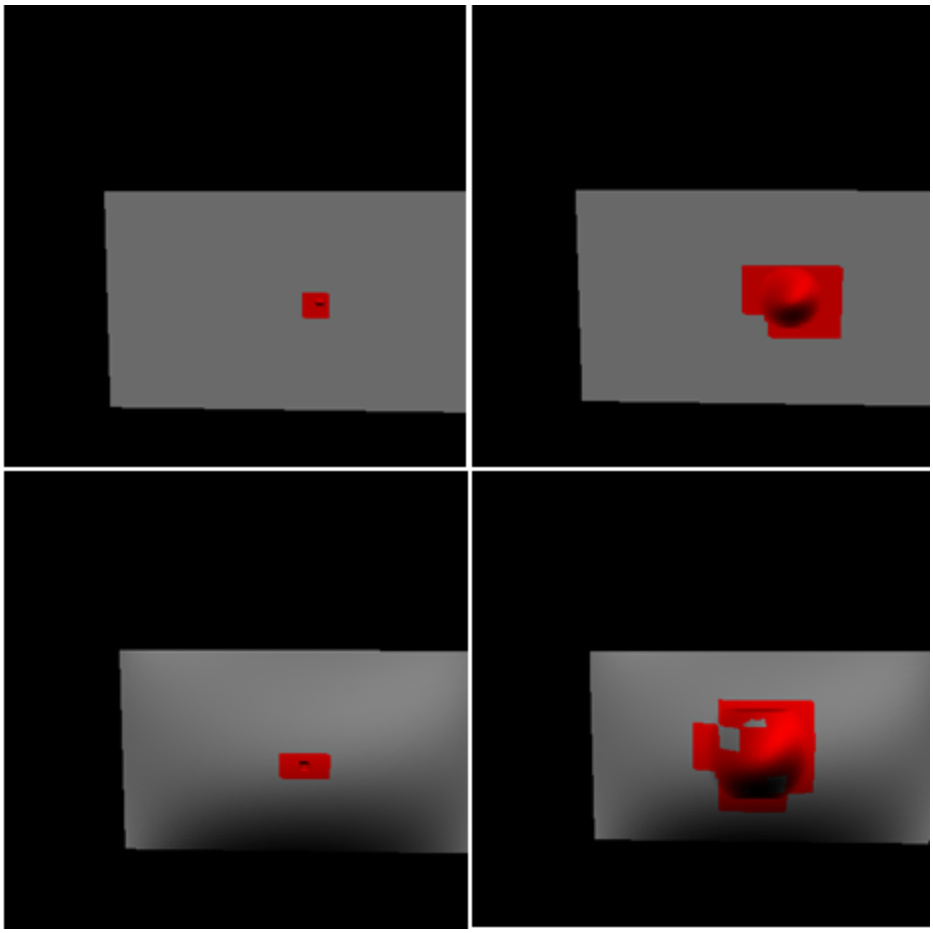

Figure 4.7. Enhanced feature extraction on artificial flat mesh with a small deformation (Top Left). Enhanced feature extraction on artificial flat mesh with a large deformation (Top Right). Enhanced feature extraction on artificial curved mesh with a small deformation (Bottom Left). Enhanced feature extraction on artificial curved mesh with a large deformation (Bottom Right). All results are from level 6 of the octree.

In the artificial meshes, appropriate results are found by level 6 of the octree, regardless of the size of the deformation or the curvature of the body. The same threshold is set for both flat meshes. Since there is no noise and no curvature aside from the deformation, the σ value of the nodes in the flat meshes are 0 in areas that do not contain the deformation. As the octree level increases, the nodes get smaller, resulting in more nodes that do not contain any piece of the deformation, thus isolating the deformation sufficiently by the 6^{th} level. It would be unnecessary to generate further levels of the octree, as the objective has already been accomplished by the 6^{th} level.

The same threshold is used for both curved meshes, and since there is no noise and the body curvature is tolerable the threshold can be set very low. The reason that there is isolation of the deformation by the 6^{th} level of the octree is because there are no other significant

deformations nearby. Also, more and more of the non-deformation areas are removed as the levels increase, since they are of lower surface variation than the threshold, until 6th level where only the deformations of interest remain. The entire ding in the curved mesh with large deformation is not extracted, and has some parts missing. This is because those areas do not meet the threshold, however the majority of the deformation is detected and this is sufficient since most of the deformation is extracted, and no other areas are extracted.

To compare the proposed enhancement to the original technique proposed by Woo *et al.* [40], the original technique is applied on the same meshes using the same thresholds indicated above. The results are shown in Figure 4.8.

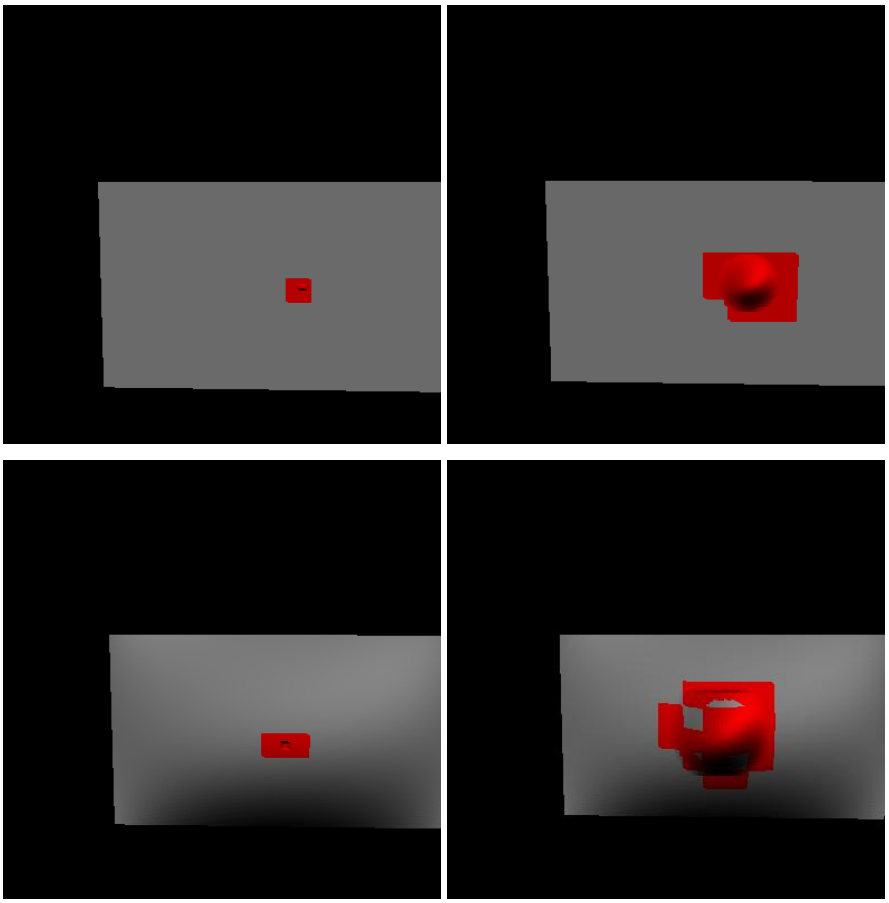

Figure 4.8. Original feature extraction on artificial flat mesh with a small deformation (Top Left). Original feature extraction on artificial flat mesh with a large deformation (Top Right). Original feature extraction on artificial curved mesh with a small deformation (Bottom Left). Original feature extraction on artificial curved mesh with a large deformation (Bottom Right). All results are from level 6 of the octree.

The results show that the enhanced feature extraction provides similar results to the original method by Woo *et al.*

To test the enhanced algorithm's ability to deal with real world data, which contains noise, surface curvature characteristics, and acquisition artifacts, it is applied on data acquired through the acquisition system details in chapter 3. This algorithm is applied to the unfiltered medium resolution CPU panel mesh, with the threshold, *a*, set at 0.223, and the results are shown in Figure 4.9.

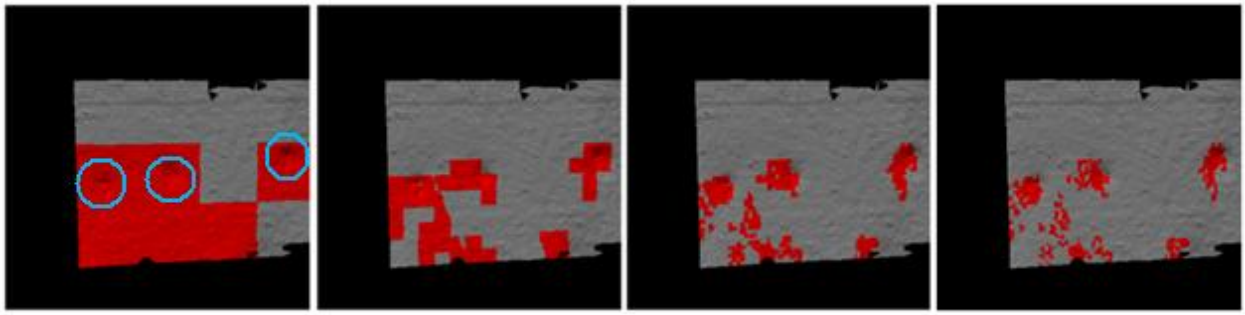

Figure 4.9. Enhanced feature extraction on unfiltered CPU panel mesh at octree level 4 (deformations circled), 6, 8, and 10.

This enhanced algorithm is applied to the filtered high resolution CPU panel mesh, with the threshold, *a*, set at 0.197, and the results are shown in Figure 4.10.

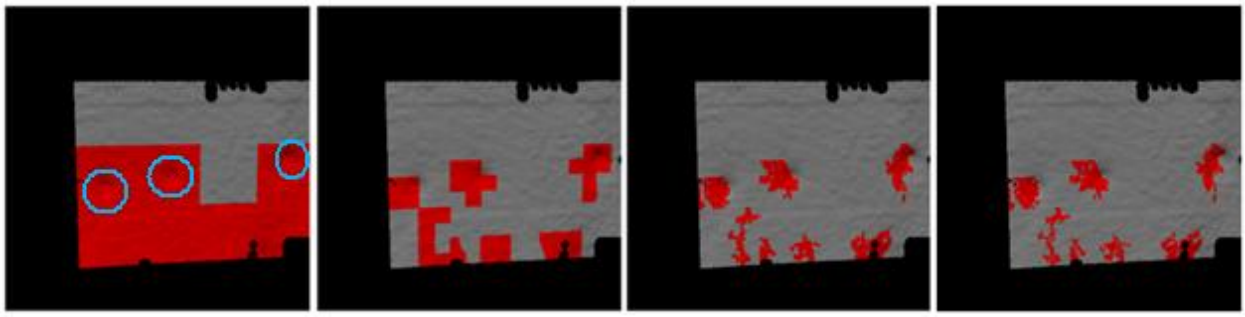

Figure 4.10. Enhanced feature extraction on filtered CPU panel mesh at octree level 4 (deformations circled), 6, 8, and 10.

In both the unfiltered and filtered meshes, areas of low surface variation are removed by the 4^{th} level of the octree but the nodes are not small enough in volume to extract areas in the shape of deformations. By the 6^{th} level of the octree, more areas of lower surface variation are removed, and it is seen that the deformations begin to become isolated, however other surface variations remain near the bottom. Finally, by the 8^{th} level of the octree, the algorithm highlights the areas of the deformations in the CPU panel mesh clearly, and isolates them from the surrounding surface variation which is also detected. Unfortunately, it also highlights areas

around it, including small deformations near the edge of the panel, as well as noise or transient features. However, there is a very good separation between the different features that are extracted. This separation between the extracted deformations of interest and the other extracted features is beneficial for segmentation and classification, which will be discussed in chapter 6, where erroneously extracted areas can be removed to leave only the deformations. There is not much of a change between level 8 and level 10, so only 8 levels of the tree need to be generated to extract the deformations of interest.

The difference between the filtered and unfiltered mesh is not very significant. Fewer surface variations near the bottom of the mesh are extracted in the filtered mesh compared to the unfiltered mesh. Also, the overall surface variation is diminished because of the filtering, so naturally, the threshold must be less to achieve similar results.

To compare the proposed enhancements to the original technique proposed by Woo *et al.* [40], the original technique is applied on the filtered high resolution CPU panel mesh using a threshold of 0.08. The results are shown in Figure 4.11.

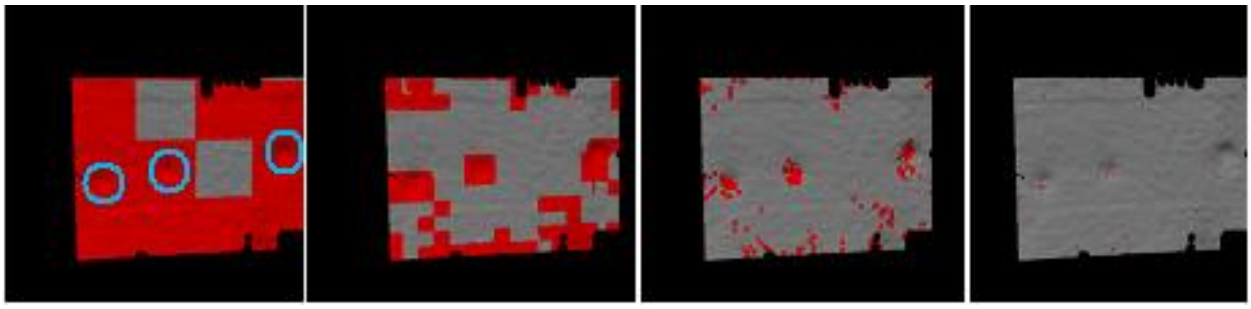

Figure 4.11. Original feature extraction on filtered CPU panel mesh at octree level 4, 6, 8, and 10.

It can be seen that the original feature extraction is not as effective in isolating the deformations of interest, and extracts areas all over the mesh at levels 6 and 8. By level 10, none of the features are extracted.

The enhanced algorithm is applied to the unfiltered high resolution car door mesh, with the threshold, *a*, set at 0.45, and the results are shown in Figure 4.12.

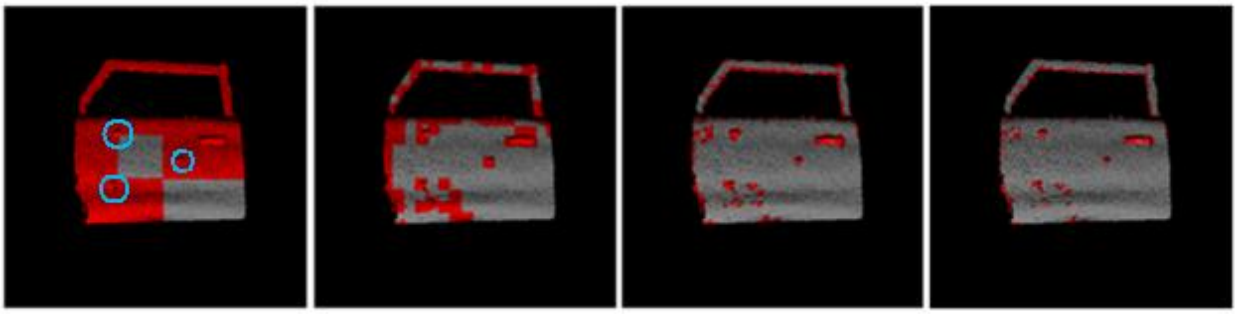

Figure 4.12. Enhanced feature extraction on unfiltered car door mesh at octree level 4, 6, 8, and 10.

The enhanced algorithm is applied to the filtered high resolution car door mesh, with the threshold, *a*, set at 0.3, and the results are shown in Figure 4.13.

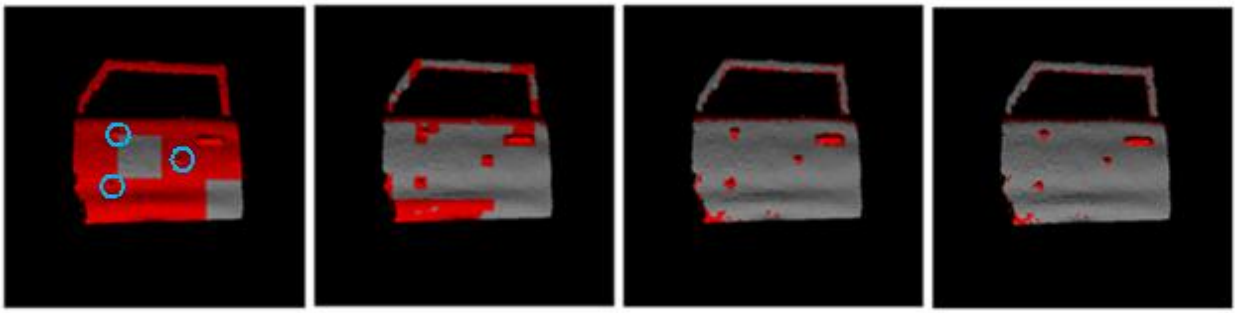

Figure 4.13. Enhanced feature extraction on filtered car door mesh at octree level 4, 6, 8, and 10.

The octree technique with the non-uniform weighting of surface normals enhancement does not remove much of the mesh, filtered or unfiltered, by the 4th level. In the 6th level, more of the mesh is removed and the extracted areas start to resemble the deformations. By the 8th level of the octree on the door mesh, the algorithm highlights the deformations of interest. The door handle is also extracted clearly. Some surface variation near the deformations is also extracted in the unfiltered mesh, but this is remedied in the filtered mesh. The algorithm also highlights minimal extra features along the boundaries of the mesh as well as some transient features along the body curvature. The large surface variations that are extracted along the boundaries are due to limitations of the acquisition system, which has difficulty acquiring the actual surface along the edges of the mesh. The 10th level, just as in the CPU panel mesh, provides no improvement and it becomes unnecessary to generate the tree that far since there is no benefit. In both the evaluated meshes, filtering gives an improvement in the results provided that the threshold, *a*, is reduced sufficiently, as the filtered meshes have a lower value of standard deviation overall.

Again, to compare the proposed enhancements to the original technique proposed by Woo *et al.* [40], the original technique is applied on the unfiltered high resolution car door mesh using a threshold of 0.2. The results are shown in Figure 4.14.

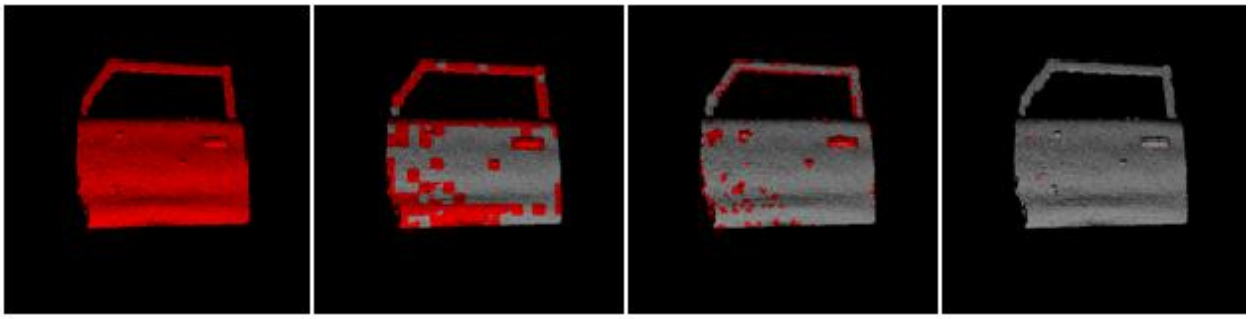

Figure 4.14. Original feature extraction on unfiltered car door mesh at octree level 4, 6, 8, and 10.

In the car door model, it can be seen that the original feature extraction is not as effective in isolating the deformations of interest, and extracts more non-deformation areas than the enhanced method at levels 6 and 8. By level 10, none of the features are extracted, whereas the enhanced method still extracts features accurately at that level.

Though it is capable of providing appropriate results, this enhanced feature extraction algorithm can also have very drastically different results because of slight threshold changes. An example of this is shown in Figure 4.15, where the enhanced algorithm is applied to the unfiltered medium resolution CPU panel at an optimal threshold of 0.223, and compared to the same algorithm applied with a threshold of 0.224.

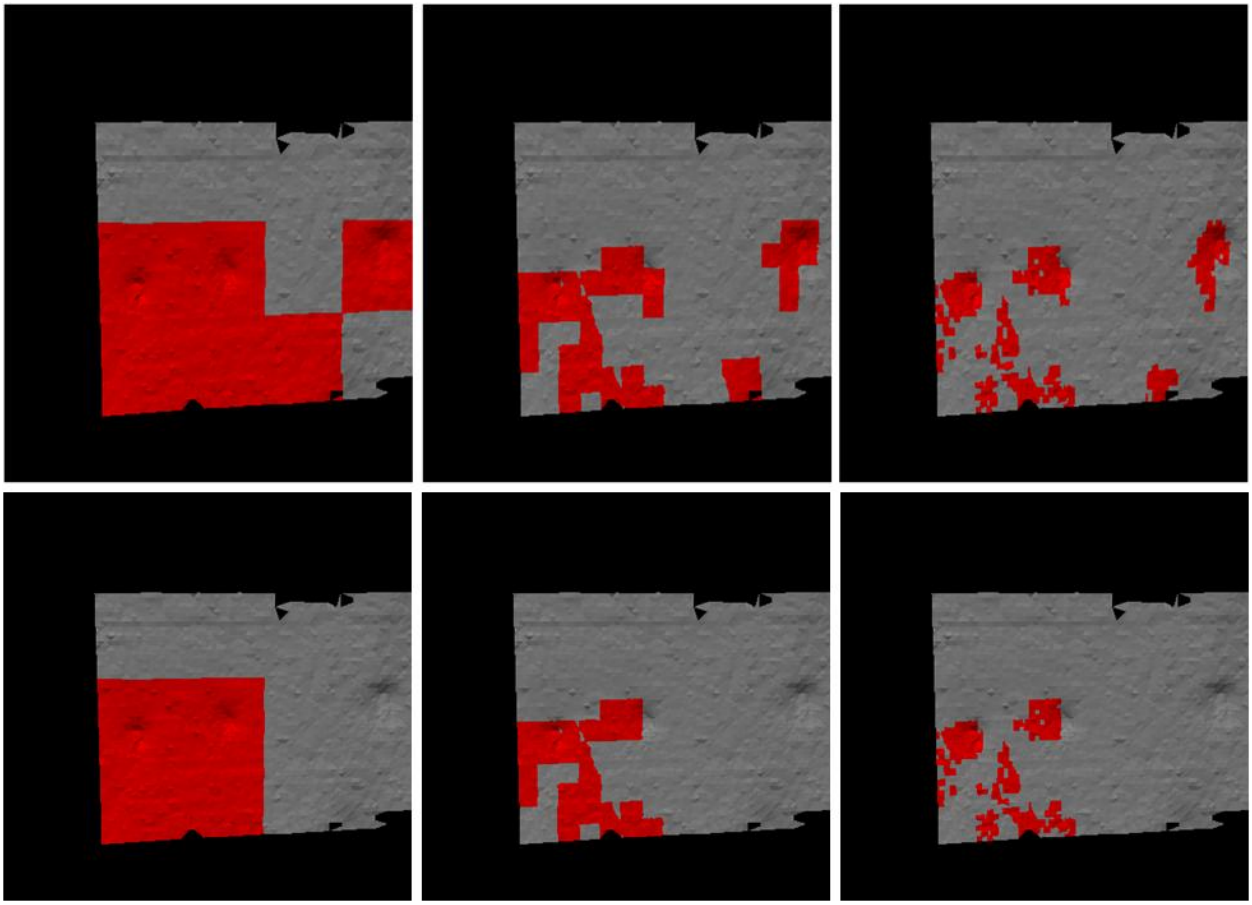

Figure 4.15. CPU panel mesh with enhanced feature extraction, threshold=0.223, at octree level 4, 6, and 8 (Top). CPU panel mesh with enhanced feature extraction, threshold=0.224, at octree level 4, 6, and 8 (Bottom).

The results show that a slightly higher threshold can significantly alter the results of the feature extraction. Since the right half of the mesh, at the low resolution of octree level 4, has a σ value that is lower than the threshold, that area is never evaluated at the higher octree resolutions. Because of this, even if the right-most dent in the CPU panel contains higher surface variation than the other deformations, that area cannot be extracted as a deformation due to its removal at a low resolution even though the other deformations will be extracted. This shows the unpredictability of the algorithm, and how the threshold may not always intuitively correlate to the results.

Overall, provided that good thresholds are selected, deformations can be extracted clearly using this enhanced octree-based technique. However, it also shows that sometimes no threshold can be found to select only the deformations of interest while excluding other features. This dependency on selecting a threshold specific to the characteristics of the mesh is not good for robustness. Using an automated method to determine the threshold is very difficult, since

the threshold depends on the distribution of standard deviation of surface normals at all resolutions. Sometimes the lower resolutions are not at all indicative of the characteristics at a higher resolution, so a good threshold cannot be selected without analyzing all levels of the octree as a whole without prematurely removing nodes via thresholding. However, creating the tree as a whole and then determining an appropriate threshold defeats the purpose of selectively removing nodes during the generation of the octree. This results section also demonstrates that the enhancements provide similar or better results for this application than the original feature extraction technique proposed by Woo *et al*.

4.3 Incremental Thresholding

The third strategy to enhance the performance is a modification of the original algorithm [40] to use a variable threshold for different levels of the tree, as reported in [59]. When higher threshold values are used at depths closer to the root, small features tend to be ignored given their small contribution to the σ value compared to the entire volume of the object. At lower threshold values, nodes farther from the root have more children because variations in smaller volumes give large standard deviation values beyond the threshold, resulting in excess features, such as noise and smooth curves being detected, as well as a larger tree. To preserve the detection of small variations while reducing unnecessary features, and to minimize the complexity of the tree, the threshold should increase at greater depths of the tree, such that the smallest threshold remains at the root, and the largest threshold appears at the leaves. The relationship between the depth of the node in the tree structure and the threshold can be optimized using various functions, depending on the thresholding model.

This thesis models the relationship between the depth of the node in the tree structure and the threshold as a linear relationship:

$$t = a + ix \tag{4.5}$$

where t is the threshold at tree level x, a is the threshold at the root node, and i is the desired depth increment.

This modification provides more precise results in extracting features of different scales when compared to a fixed threshold throughout the entire structure as proposed in [40]. It is also very effective at reducing the size of the tree, since more non-feature nodes can be removed while retaining the important information in nodes that contain features.

Just like in section 4.2, the selection of the threshold is important, and most importantly, it must be able to extract as much of the surface area of the deformations as possible while

minimizing the extraction of non-deformation features and ensuring that the areas between adjacent deformations are not extracted such that there is separation.

4.3.1 Results

The incremental threshold enhancement is to improve the algorithm's performance on deformations that are hard to isolate, so tests were not run on the artificial meshes since their deformations were extracted clearly in the previous section. The deformations that are hard to isolate were in the real-world meshes containing artifacts, noise, and significant surface curvature which the unenhanced algorithm may also extract as features. Optimal thresholds were experimentally determined to meet the same criteria described in section 4.2.1.

This algorithm is applied to the unfiltered medium resolution CPU panel mesh, with the thresholding parameters, *a*, set at 0.2, and *i*, set to 0.011, and the results are shown in Figure 4.16.

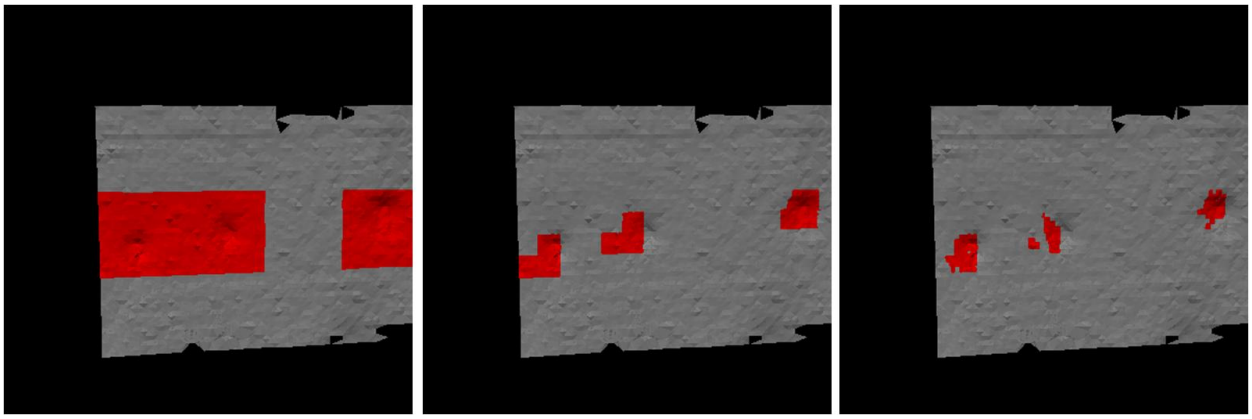

Figure 4.16. Feature extraction on unfiltered CPU panel mesh at octree level 4, 6, 8.

This algorithm is also applied to the filtered high resolution CPU panel mesh, with the thresholding parameters, *a*, set at 0.18, and *i*, set to 0.008, and the results are shown in Figure 4.17.

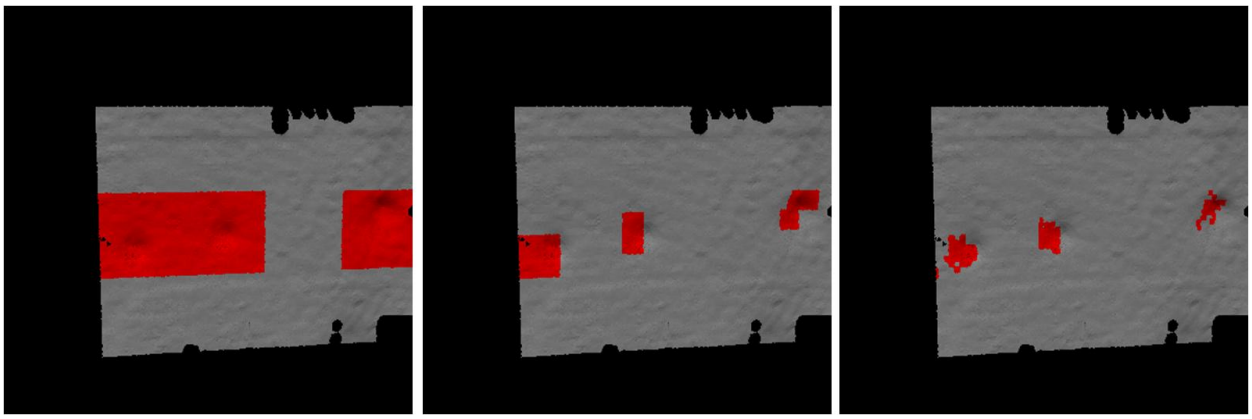

Figure 4.17. Feature extraction on filtered CPU panel mesh at octree level 4, 6, 8.

In both examples of the CPU panel mesh, the results show that the algorithm with incremental thresholding can extract the deformations of interest. The deformations of interest were extracted, with good separation, without extracting any other features, which is an improvement over the results achieved with uniform thresholding on the CPU panel mesh. Also, the deformations are clearly extracted and separated by level 6 of the octree using the enhancement, where the uniform thresholding was only able to extract the deformations at level 8, resulting in a much smaller tree.

The enhanced algorithm is applied to the unfiltered medium resolution car door mesh, with the thresholding parameters, *a*, set at 0.23, and *i*, set to 0.03, and the results are shown in Figure 4.18.

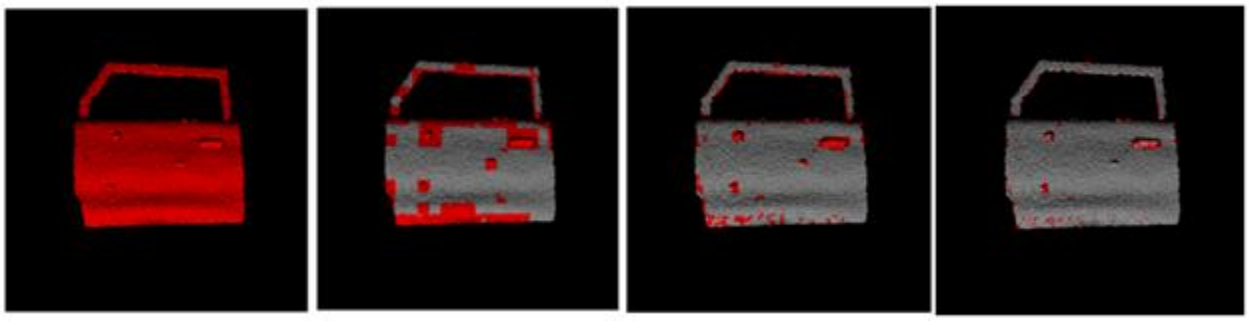

Figure 4.18. Feature extraction on unfiltered car door mesh at octree level 4, 6, 8, and 10.

This algorithm is also applied to the filtered high resolution car door mesh, with the thresholding parameters, *a*, set at 0.2, and *i*, set to 0.0175, and the results are shown in Figure 4.19.

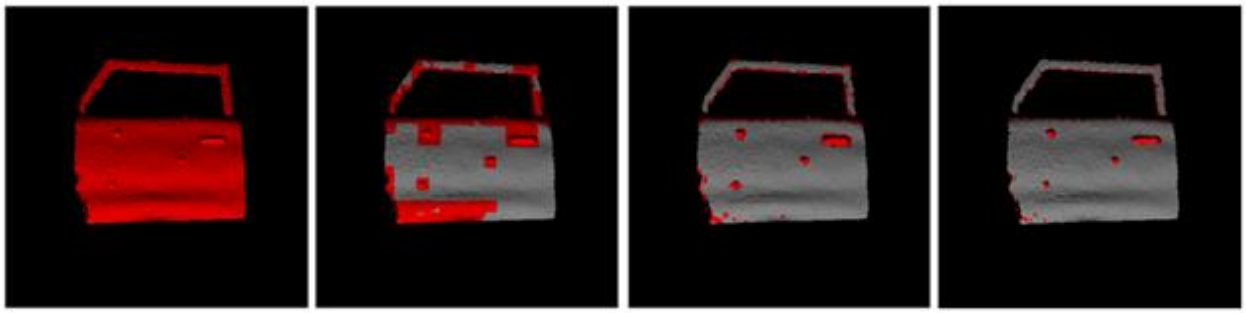

Figure 4.19. Feature extraction on filtered car door mesh at octree level 4, 6, 8, and 10.

When applied on the car door mesh, in both the filtered and unfiltered versions, the results show less noise and transient features being detected. None of the mesh is removed by the 4th level, as the large volumes represented by the nodes contain enough surface variation to warrant further inspection, based on the threshold selected. By the 6th level of the octree, the extracted areas are beginning to isolate the deformations, but not sufficiently enough to say the deformations of interest are clearly extracted by that level. All the deformations of interest are extracted by the 8th level of the octree, and they are extracted cleanly, with minimal noise and surface variations also being extracted, which will greatly aid in the segmentation and classification methods.

In the filtered mesh, the 10th level produces a slight reduction in noise and boundary areas being extracted, but nothing significant. In the unfiltered mesh, the 10th level actually removes some of the deformations, and a significant amount of the door handle, actually giving worse results than the 8th level. Because of this minimal improvement in one mesh, and substantial degradation in the other, it can be said generating the tree past the 8th level should not be done as it does not provide any real improvement in the feature extraction, for the resolution of the 3D sensor used in these experiments.

The filtered version of the mesh extracts fewer areas along the boundaries of the mesh, as well as less noise and non-deformation surface variations than the unfiltered version. The door handle is extracted in both versions, because it is composed of areas of very high surface variation, sometimes more than the deformations themselves, so it is difficult to remove it using just this feature extraction technique. The areas of variation along the boundaries, just as with the CPU panel mesh, are due to the limitations of the 3D acquisition system and its unreliable ability to acquire accurate 3D points along the boundaries of objects. It should be noted that the unfiltered versions of both the CPU panel and the door mesh contain a significant amount of noise, so it is difficult to isolate the deformations as their surface variation is only slightly greater than the noise. Picking an accurate threshold does a very good job of separating deformations

from noise, however a more accurate 3D acquisition system should be able to provide more accurate meshes with much less noise, allowing the thresholds to be more robust without being catered for the specific noise characteristics of the structured light sensor.

A serious limitation of incremental thresholding enhancement is that experimentation is required to determine the threshold parameters. In the original method, only one threshold must be selected. With the latter enhancement, two values must be selected to target the deformations of interest. This gives more flexibility for the algorithm to hone in on certain deformations, but at the same time requires more experimentation to determine an appropriate threshold.

Due to the nature of this technique requiring more thresholds to focus on deformations of interest more specifically, it is assumed that the results of picking poor thresholds will provide results similar to, or worse than, those in section 4.2.1. This enhancement is significantly less robust, but is more efficient and accurate in extracting features of interest.

4.4 Aggregate Standard Deviation

Even with the aforementioned improvements of incremental thresholding and non-uniform surface normal weighting, feature extraction using an octree requires trial and error to determine the appropriate thresholds. Its robustness remains questionable, as different 3D meshes may have a different distribution of variations amongst different resolutions. It is possible that no selection of thresholds could be found to properly isolate certain features, as shown in section 4.2.1. Incremental thresholding, as described in section 4.3, alleviates this problem by changing the threshold throughout the tree such that more low resolution nodes remain. That enhancement may be sufficient for very predictable scenarios, but when the distribution and appearance of features over multiple resolutions is unknown, determining appropriate threshold parameters that are robust to varying meshes may be difficult. Based on the thresholds alone, it is difficult to predict what features will remain and what features will be removed by the time the deeper levels of the octree are reached. As shown in section 4.2.1, a slight threshold change can produce drastically different results. In general, a more robust technique is required to deal with meshes with varying characteristics, with a more intuitive relationship between the threshold and the results.

The aggregate standard deviation variation of the octree-based technique allows the generation of the tree in its entirety, without limiting subdivision to only nodes that meet a certain threshold. Then, all nodes at selected resolutions of the fully generated tree can be extracted and nodes with low surface variation can be removed. Though this would be less efficient, as

the tree would be large and occupy more memory, the threshold selection would be easier and would ensure that the features of interest are still available to be extracted at the desired tree level, solving the problem of areas being prematurely removed as discussed in section 4.2.1. A new metric will be introduced to measure multi-resolution surface variation, however, the updated technique can be applied using σ as the main metric also.

Before the new metric is introduced, it is important to show how the algorithm can be applied using σ as the main metric, and to show the limitations of σ to justify the introduction of the new metric. Using the CPU panel as an example, Figure 4.20 represents the σ values on the mesh as intensity values, along with their respective histograms at different levels of the octree. The histograms shown separate the range between the lowest σ and highest σ at the evaluated octree level into 256 equally divided bins. The x-axis represents the bins, where bin 1 is the lowest range of σ values and bin 256 is the highest, and the y-axis represents the number of nodes with σ values in each bin. In the images, black represents bin 1 while white represents bin 256.

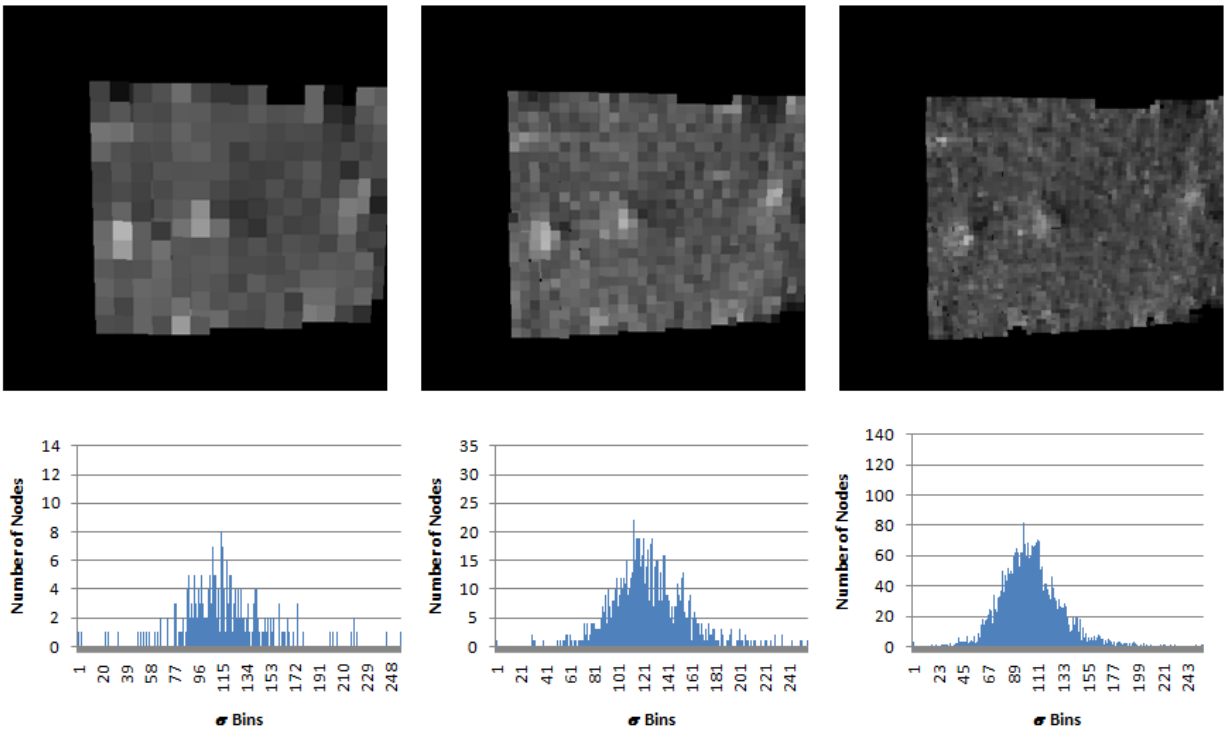

Figure 4.20. Intensity corresponding to σ at octree level 5, 6, 7 (Top). σ histogram at Level 5, 6, 7 (Bottom).

Looking at the histogram is helpful for threshold selection. The feature nodes are in the higher bins, which contain large σ values, with the highest σ value being in the 256th bin. Also,

the feature nodes will be few in number. The majority of the nodes are going to be non-feature nodes, so the bulk of the normal distribution will also be classified as non-feature nodes. Nodes with low σ values, which are in the lower bins, are definitely non-feature nodes. This information can aid in selecting a suitable threshold. Using the same CPU panel mesh as an example, Figure 4.21 shows how evaluating the σ values at a single resolution, using the histogram, can be successful in determining a threshold to isolate areas of interest. The nodes with σ values above the threshold are extracted as features.

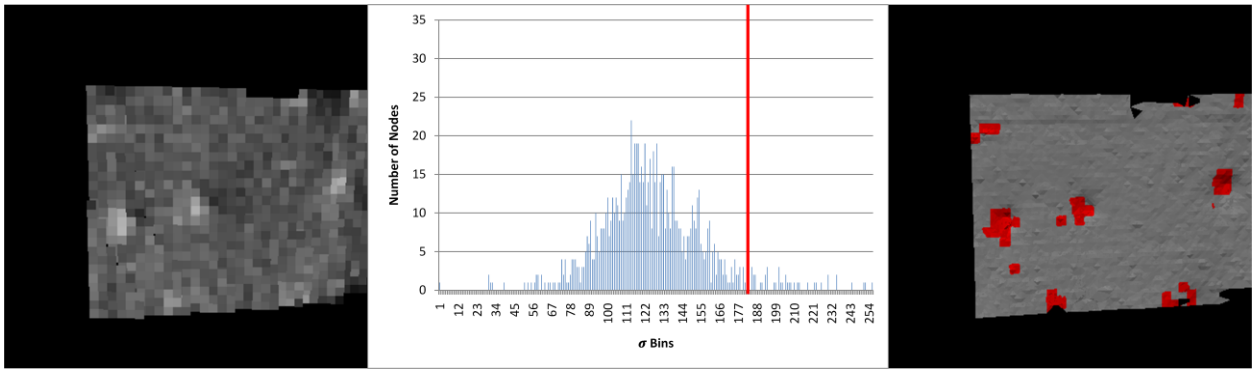

Figure 4.21. Intensity corresponding to σ in CPU panel at level 6 (Left). σ Histogram with threshold (Center). Extracted features (Right).

Unfortunately, that on its own would not make use of the multi-resolution capabilities of the octree and puts a lot of emphasis on the σ values of the selected resolution. As reviewed in section 2.3.2, Pauly *et al.* [39] described the idea of multi-resolution feature persistence, where a strong feature can be determined if it is persistently detected across multiple adjacent scales. In order to combine some of the elements of multi-resolution feature persistence with the octree-based feature extraction technique, it is proposed in this work that the characteristics of nodes are accumulated across multiple levels of the octree.

Each node should take into account the characteristics of its hierarchical neighbours, by accumulating the information of its parent and children. In the case of this system, the standard deviation of surface normals, σ, is being used as the main metric for determining features, so that will serve as the primary descriptor of a node's surface variation. A new metric, known as *s*, must be developed that takes into account σ values of several resolutions of the mesh, to describe the level of multi-resolution surface variation there is at a given area. It is important that areas with significant variation over multiple resolutions produce a higher *s* value than areas with low variation over multiple resolutions. Also, it is important that areas with significant variation over multiple resolutions produce a higher *s* value than areas with significant variation

in just a single resolution. Finally, it is important that using this new metric, *s*, provides better extraction and robustness results to justify the extra calculation.

It is possible to accumulate such a value over several resolutions, but it is important to note that a strong feature may not necessarily appear over many resolutions, so the *s* calculation should not produce a low *s* value for that type of feature, causing them to be overlooked. Considering that each level of the octree provides a significant resolution change, as it doubles the resolution in each of the axes, it is only important that a feature appears over two or three resolutions to determine that it is not a transient feature or noise, but in fact a consistent surface deformation in multiple scales. For this reason, three levels of resolution are accumulated to determine the *s* value, the current resolution of the node along with one resolution higher and lower. It is thought that a product of the surface variation in an area between the three resolutions would be sufficient to meet the above mentioned criteria. The equation to calculate the accumulated standard deviation of the surface normals for an octree node is shown in Eq. 4.6.

$$\sigma_{\text{children}} = \sum_{\text{i}=0}^{\text{n}} \frac{\sigma_{\text{i}}}{n}$$

$$s = (\sigma)(\sigma_{\text{parent}})(\sigma_{\text{children}}) \tag{4.6}$$

where s is the aggregate standard deviation value, σ is the standard deviation value of surface normals in the current node, σ_{parent} is the standard deviation value of surface normals in the parent node, σ_{i} is the standard deviation value of surface normals in the i^{th} child, and *n* is the number of children that are not empty. σ_{children} is calculated to be the average σ values of the node's non-empty children. Note that here a triangle-based analysis is used as described in section 4.1 and that σ values are calculated using non-uniform weighting as detailed in section 4.2. At any given scale, each node will contain a value representing the accumulated standard deviation, *s,* of a certain volume of the mesh.

Using the same CPU panel as an example, Figure 4.22 shows *s* values on the mesh along with the respective histograms at different levels of the octree, while Figure 4.23 shows how thresholding the *s* values at a single resolution can be successful in isolating areas of interest. The histograms shown are in the same format as the histograms shown in Figure 4.20, but use *s* values instead of σ values.

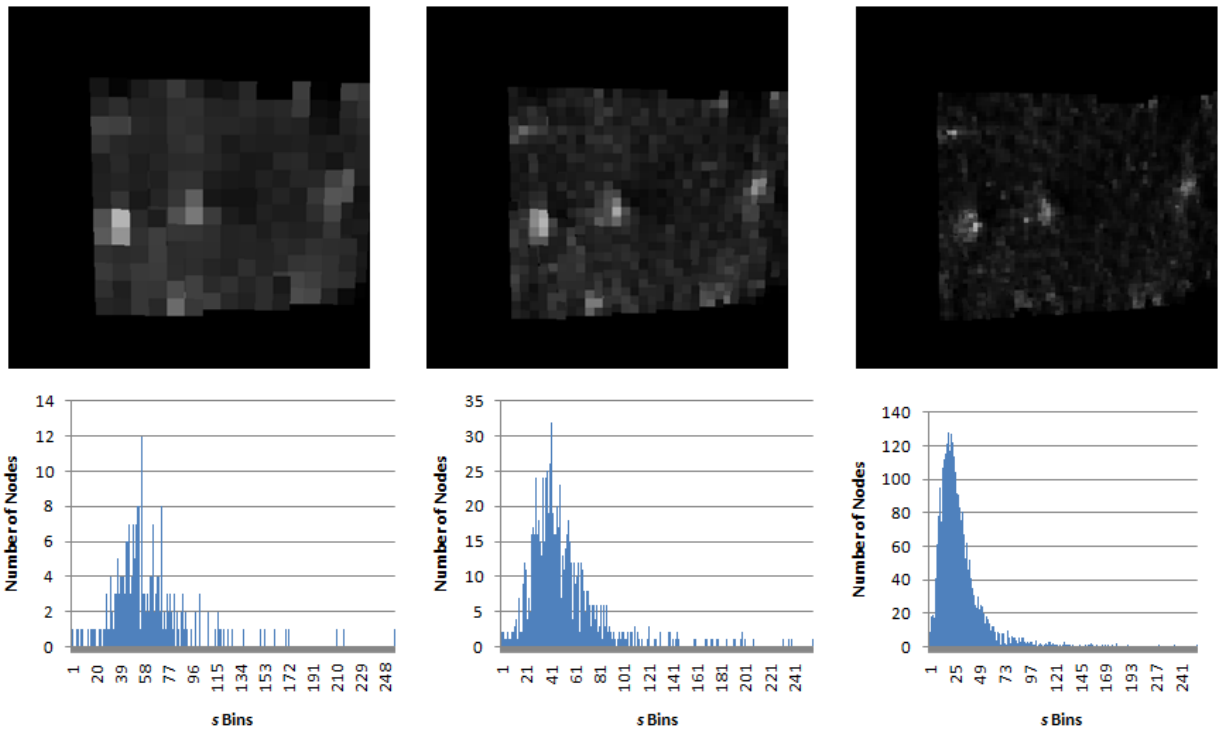

Figure 4.22. Intensity corresponding to *s* at octree level 5, 6, 7 (Top). *s* histogram at Level 5, 6, 7 (Bottom).

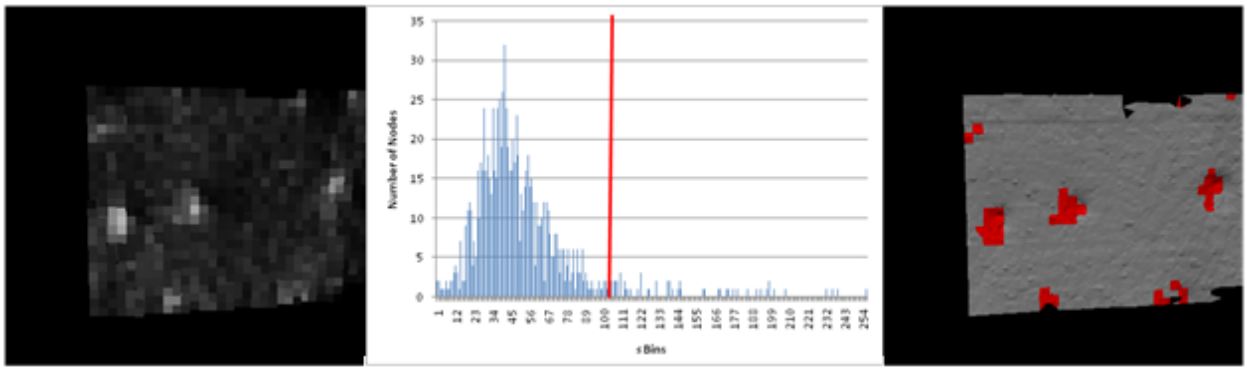

Figure 4.23. Intensity corresponding to *s* in CPU panel at level 6 (Left). *s* histogram with threshold (Center). Extracted features (Right).

Though thresholding the nodes based only on the σ value may give good results, the *s* value is expected to provide a greater separation between feature nodes and non-feature nodes. A better separation also adds tolerance to non-optimal thresholds. The difference in characteristics can be seen in Figure 4.24, where a histogram is used to compare σ distribution and *s* distribution in several levels of the octree when applied on the unfiltered high resolution CPU panel mesh.

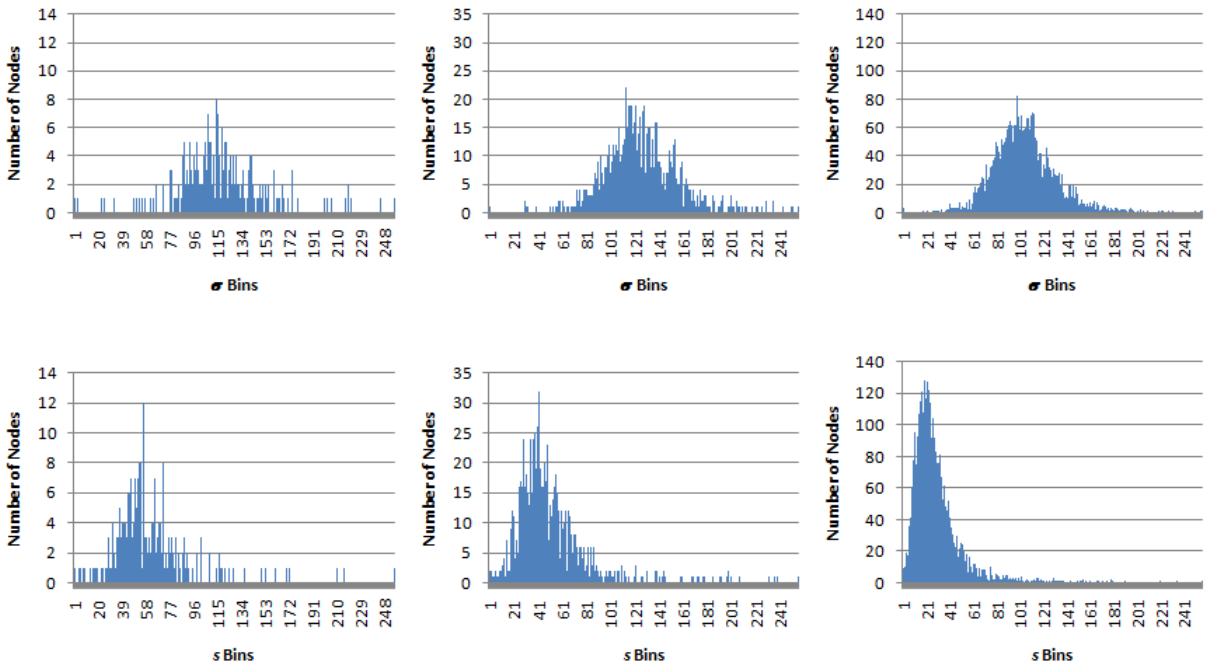

Figure 4.24. σ histograms of CPU panel at octree levels 5, 6, 7 (Top). *s* histograms of CPU panel at octree levels 5, 6, 7 (Bottom).

The histogram using the *s* value as a metric shows the non-feature nodes in a smaller amount of bins along the x-axis, and a sharper transition from non-feature nodes to feature nodes, when compared to using σ as the metric. Since all of the non-feature nodes are in a smaller quantity of the smaller bins, this allows more of the bins to describe feature nodes and further separates the bins that contain non-feature nodes from the bins that contain feature nodes. This type of distinction is important to find a good threshold, and it is also important to provide robustness if the perfect threshold is not found.

To compare the tolerance to non-optimal thresholds when using the σ value against using the *s* value for feature extraction, the algorithm is applied on the filtered CPU panel mesh with both metrics at octree level 5. A suitable threshold was found such that the results are comparable between using the σ value and the *s*. Then, using the histogram, thresholds which are 50 bins in either direction of the selected threshold are used to determine how that affects the extraction results. For using the σ metric, a threshold of 0.223 is used, which lies in bin 163. Then, a threshold of 0.157, lying in bin 113, and a threshold of 0.289, lying in bin 213 are also applied. For using the *s* metric, a threshold of 0.295 is used, which lies in bin 176. Then, a threshold of 0.214, lying in bin 126, and a threshold of 0.377, lying in bin 226 are also applied.

Figure 4.25 shows the results using σ for feature extraction, and Figure 4.26 shows the results using *s* for feature extraction.

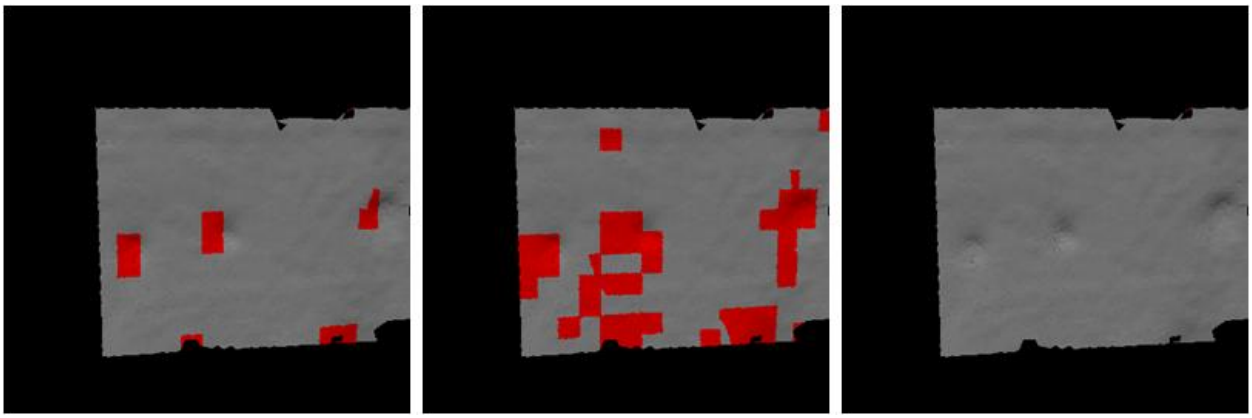

Figure 4.25. Optimal σ threshold (Left). Optimal threshold minus 50 bins (Center). Optimal threshold plus 50 bins (Right). Results are from octree level 5.

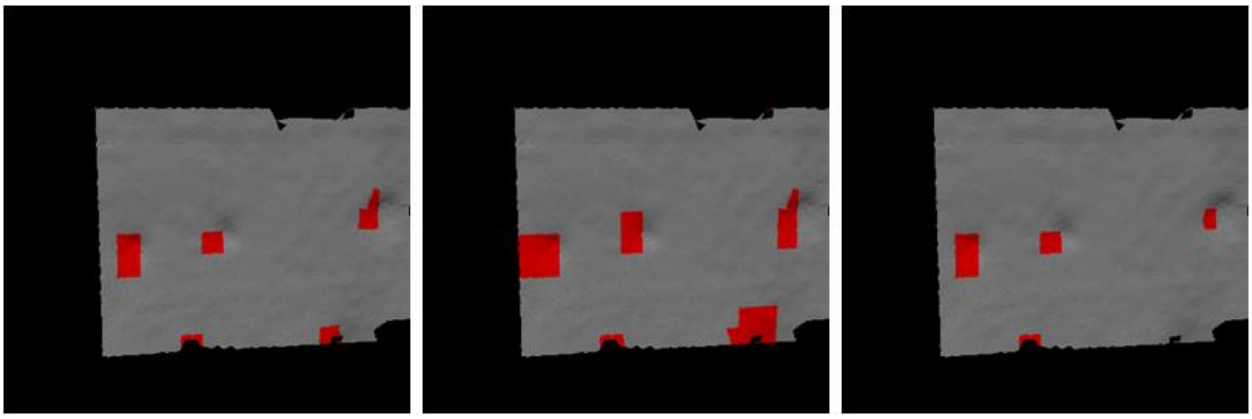

Figure 4.26. Optimal *s* threshold (Left). Optimal threshold minus 50 bins (Center). Optimal threshold plus 50 bins (Right). Results are from octree level 5.

A change in threshold affects the method when using the σ value as a metric much more than when using the *s* value as a metric. When using the σ value as a metric, the technique introduces more noise and transient features when the threshold is lowered, and removes all of the deformations when the threshold is increased. When using *s* as a metric, the method does not change that much with the different thresholds, as the deformations are all still present and not much additional surface variation is introduced. The only thing that changes is the amount of the surface variation extracted along the bottom boundary of the mesh. When the same threshold is used for several different meshes with slightly different deformation characteristics, that threshold will likely be non-optimal for some of them. As shown above, the *s* metric will be

better suited for that than the σ metric as the results of extracted deformations from the mesh stay similar in spite of significant threshold changes. When using experimentation to select thresholds, the results will be much more intuitive and much less sensitive to slight changes in thresholds than the methods described in sections 4.2 and 4.3. The criterion for threshold selection remains the same as it was in sections 4.2 and 4.3.

As stated above, the generation of the whole tree would be less memory efficient than using the original octree-based method with the enhancements described in sections 4.2 and 4.3. However, since only one level of the octree is going to be analyzed for thresholding, the generation of the entire octree is not necessary. Only the level of the octree in which the nodes will be extracted and thresholded, as well as the level above and below it, must be generated. By selectively generating only the important levels of the tree, the efficiency of the algorithm is increased significantly.

The increase in robustness to non-optimal thresholds, and significant separation between non-deformation areas and deformation areas in the *s* metric, justify the use of *s* over σ as a metric. The increase in intuitiveness of the thresholding parameter and efficiency over the original octree method, which thresholds nodes during tree generation as opposed to after tree generation, justify the development of this algorithm overall.

4.4.1 Results

The requirements for the surface shape analysis may change from application to application, and the preferred results will differ. This enhancement can be applied under two different requirements.

The first requirement is that the deformations must be extracted in full, using a lower threshold. This requirement may be due to the possibility that the feature extraction technique may extract different areas of high surface variation, where only a subset of which will be deformations of interest. Because of this, an additional step, discussed in chapter 6, will be required to distinguish the extracted areas into deformations and non-deformation areas, and a maximum amount of the deformation must be extracted to gain sufficient information for that additional classification step. With a low threshold, the deformations can be extracted in full, however other surface variations will also be extracted, at the risk of the deformations not being completely isolated. An example of the ideal situation, and non-ideal situation, when implementing this requirement, is shown in Figure 4.27.

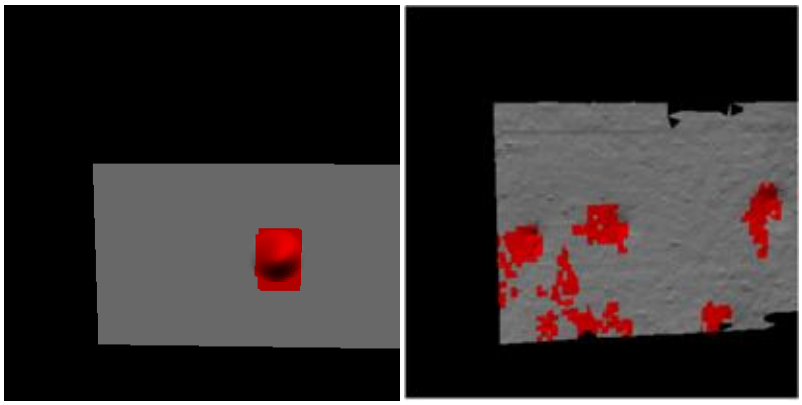

Figure 4.27. Deformation extracted in full (Left). Deformations extracted in full, with other surface variation also extracted (Right).

For some applications, where the feature extraction can be trusted with extracting only areas of interest, a higher threshold can be set. This would not normally extract the entire deformation, but just part of it. This higher threshold will also be more successful in removing other areas of high variation that do not belong to the deformation. However, in the case that non-deformation areas of high variation are also extracted, there will be no way to separate non-deformation areas from deformation areas. An example of the ideal situation, and a non-ideal situation, when implementing this requirement, is shown in Figure 4.28.

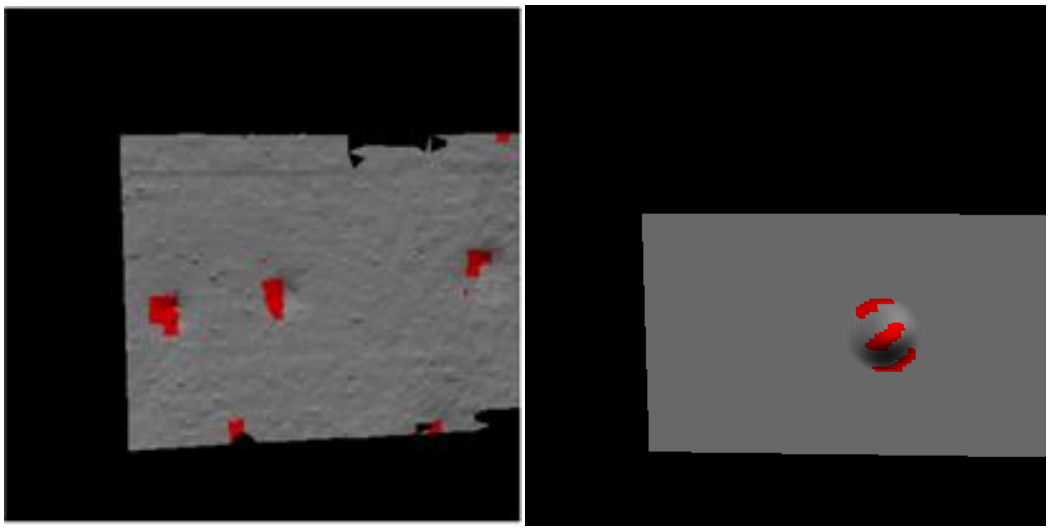

Figure 4.28. Partial deformations extracted, with no other surface variation extracted (Left). Deformation and areas of non-interest also extracted, with minimal surface area in each extraction to obtain meaningful information about which is a deformation and which is an area of non-interest (Right).

With these different requirements in mind, the results below will show how the incremental thresholding enhancement can be applied in either situation, with promising results. The discussed enhancement of the octree algorithm was applied to the experimental data, and the results are presented below. Optimal thresholds for the 3D meshes were experimentally determined.

This algorithm is applied to the artificial curved mesh with a small deformation and also to the artificial curved mesh with a large deformation. A threshold for *s* is set to 0.002 for both meshes and the results are shown in Figure 4.29.

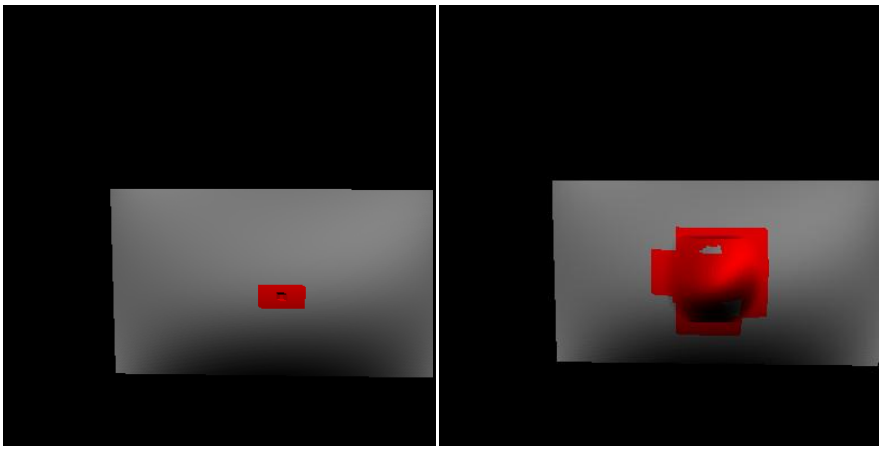

Figure 4.29. Feature extraction on curved surface with small dent (Left). Feature extraction on curved surface with large ding (Right). Both results are using a threshold of 0.002 at level 5 of the octree.

It is visible that, in the artificial meshes, the features are easily isolated at a shallower level of the tree, regardless of the size of the deformation, compared to the results in section 4.2.1. Also, the threshold is the same for both, since all of the other characteristics of the panel are similar, such as having no noise, no artifacts, and the same body curvature. When there is no noise and no other major surface deformations the algorithm is robust to different types and scales of deformations when using the same threshold. The deformations are extracted clearly by the 5^{th} level of the octree, which is an improvement over the results shown in section 4.2.1. This is because the *s* values of the nodes in the 5^{th} level show a clear separation between deformation and non-deformation areas, so a threshold can be applied to isolate only the deformations.

Unfortunately, real-world meshes may contain noise, surface variation other than the deformations to be extracted, and acquisition artifacts, so the limits of the algorithm must be tested for these more complex scenarios. To test a situation where two different requirements

will require two different threshold, the proposed algorithm with aggregate standard deviation is applied to the unfiltered medium resolution CPU panel mesh. It is applied once with the threshold set at 0.020 and once with the threshold set at 0.035, and the results are shown in Figure 4.30.

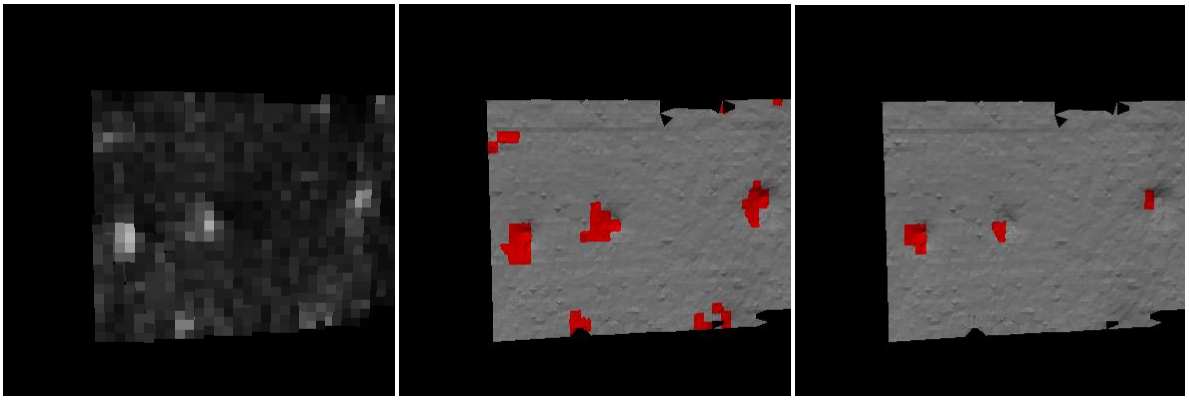

Figure 4.30. Intensity corresponding to *s* on unfiltered CPU panel at octree level 6 (Left). Feature extraction on unfiltered CPU panel, threshold = 0.020, at octree level 6 (Center). Feature extraction on unfiltered CPU panel, threshold = 0.035, at octree level 6 (Right).

This algorithm is applied to the filtered medium resolution CPU panel mesh, once with the threshold set at 0.011 and once with the threshold set at 0.025, at level 6, and the results are shown in Figure 4.31.

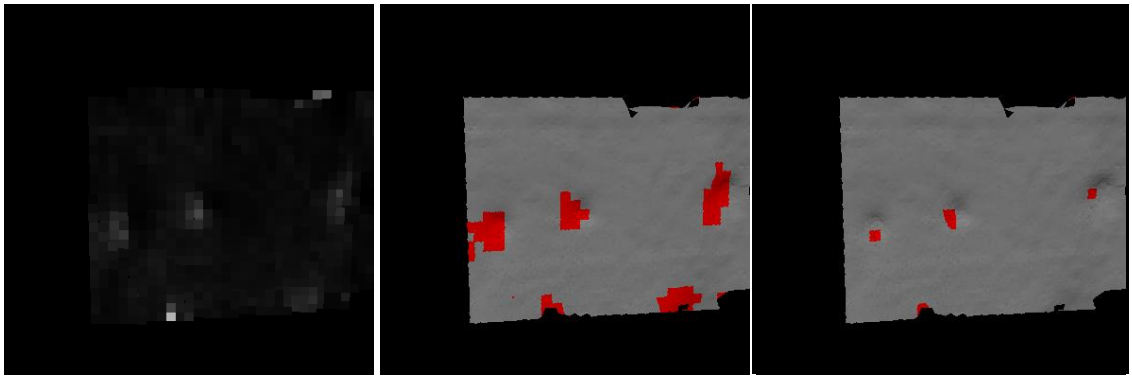

Figure 4.31. Intensity corresponding to *s* on filtered CPU panel at octree level 6 (Left). Feature extraction on filtered CPU panel, threshold = 0.011, at octree level 6 (Center). Feature extraction on filtered CPU panel, threshold = 0.025, at octree level 6 (Right).

Here is an example of where two different requirements may require two different thresholds. If the requirement is that deformations must be extracted in full, the algorithm can

extract the deformations in full from the unfiltered mesh, using a threshold of 0.020. It also extracts areas along the boundaries of the mesh, due to the acquisition system limitations, however these areas can be removed during the classification phase since enough surface area of the features are extracted to determine which are deformations and which are not. If the requirement is that just a portion of the deformation needs to be extracted, the algorithm can extract a piece of each deformation from the unfiltered mesh, using a threshold of 0.035. It clearly identifies each deformation, and also avoids the extraction of any other surface variations. Depending on the requirements of the system, this algorithm is capable of performing well, provided an acceptable threshold is selected. All of the results are from level 6 of the octree, as the tree does not need to be generated any further since the results are sufficient by that level.

These results are comparable to the results achieved using the incremental thresholding algorithm, in that sufficient extraction of deformations are achieved by octree level 6. The trade off is that this algorithm can either extract the deformations in full along with extra surface variation, or extract the deformations in part without the extraction of any surface variation, all by using one threshold, whereas the incremental thresholding enhancement can isolate the deformations in full without extracting any other surface variation, but requires the setting of two parameters.

This algorithm is applied to the unfiltered high resolution car door mesh, with the threshold set at 0.155, and the results are shown in Figure 4.32.

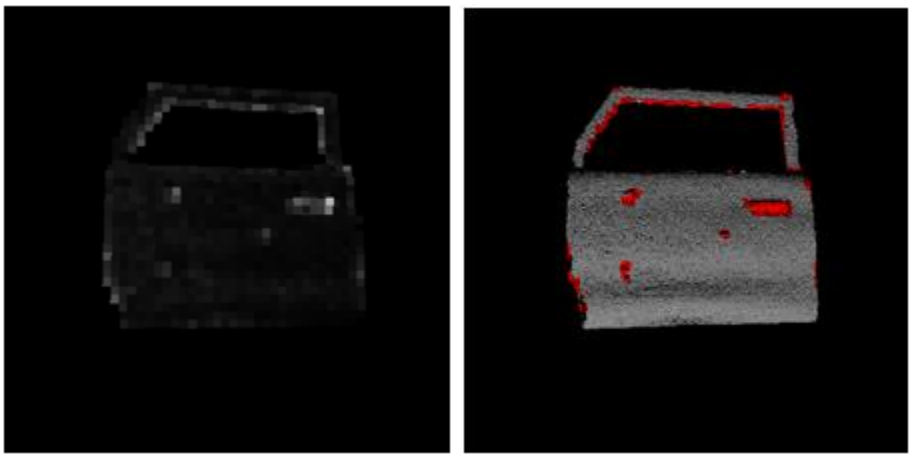

Figure 4.32. Intensity corresponding to *s* on unfiltered car door at octree level 6 (Left). Feature extraction on unfiltered car door at octree level 6 (Right).

This algorithm is applied to the filtered high resolution car door mesh, with the threshold set at 0.044, at level 6, and the results are shown in Figure 4.33.

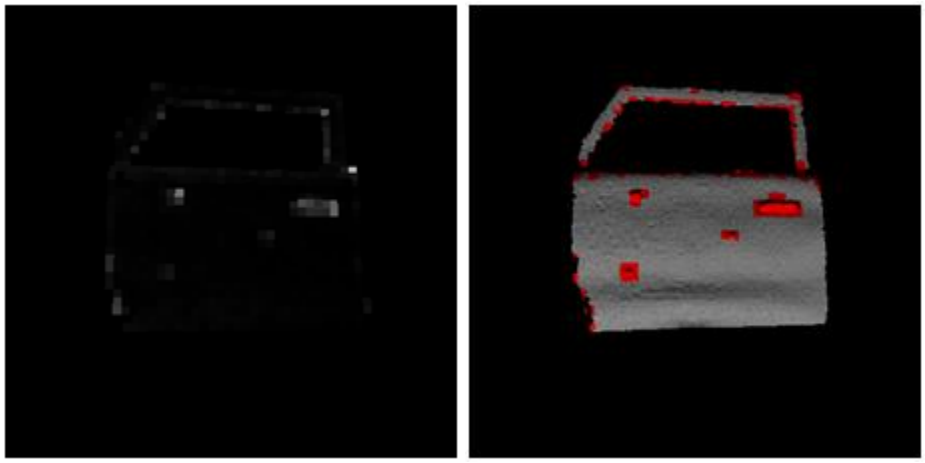

Figure 4.33. Intensity corresponding to *s* on filtered car door at octree level 6 (Left). Feature extraction on filtered car door at octree level 6 (Right).

In terms of its effectiveness in extraction, provided the correct threshold is selected, the algorithm extracts the deformations of interest with equal effectiveness in both the filtered and unfiltered car door meshes. It also extracts many of the edges around the door and window frame. These edges are very rough areas in the mesh due to the limitations of the acquisition system, and cause high *s* values where they are, resulting in them being extracted.

No threshold could be found to eliminate all of the extracted non-deformation areas. Even if application only required that a portion of each deformation is required, with this mesh, it was impossible to isolate only the deformations. For situations such as this, it is important that a classification stage exists to determine which of the extracted areas are deformations and which are not. Chapter 6 will discuss how to remove extracted features, which are not deformations, such as the rough edges of the mesh and the door handle.

The aggregate standard deviation enhancement, in some ways, outperforms the incremental thresholding enhancement on this particular mesh, in both the filtered and unfiltered versions. By the 6^{th} level of the octree, this method extracts the deformations of interest. And, similar to the incremental thresholding enhancement results, it also extracts the door handle and some areas along the boundary of the mesh, which is due to the limitations of the acquisition system generating high variation along the mesh boundaries, as discussed earlier. And though it extracts more of the boundary areas than the incremental thresholding enhancement, it does not extract additional surface variation near the deformations. It also does all of this with the

selection of a single threshold, as opposed to the experimental selection of two thresholding parameters required by the incremental thresholding enhancement.

4.5 Conclusion

In this chapter, several enhancements to the original octree-based feature extraction technique, by Woo *et. al* [40], were proposed. The use of the standard deviation of surface normals as a method to measure local surface variation is very effective. The non-uniform weighting of surface normals enhancement produced results that were at least as accurate as the original method. The incremental thresholding demonstrated a significant improvement in extracting deformations of interest and ignoring non-deformation features, however its additional parameter, *i*, can be difficult to set accurately. Overall, the proposed enhancements demonstrated that they can extract deformations of interest with a greater accuracy than the original method.

The proposed aggregate standard deviation enhancement alters the principles of the original method, since it does not require the generation of the entire tree. The proposed *s* metric applies the principle of feature persistence, and demonstrated that it can measure local surface variation over multiple scales for the purpose of detecting areas of strong variation, which are deformations in this case. The aggregate standard deviation enhancement requires that a voxel representation of the mesh is only generated for the resolution that the feature extraction is being conducted at, as well as the two adjacent resolutions, making it more computationally efficient than the original method and the other described enhancements. Also, its threshold parameter produces more predictable results when varied, as opposed to the parameters for the other enhancements, such as *a* and *i*, which must be more carefully selected and behave more unpredictably.

Overall, the presented enhancements show that deformations of multiple scales can be extracted with accuracy. Each enhancement shows some improvement, either in the accuracy of extracting deformation features, accuracy of ignoring non-deformation features, or in its parameter tuning. The results presented in this section will be compared to the contour analysis technique, which will be presented in Chapter 5, to determine which feature extraction technique is best suited to extract surface deformations over 3D meshes.

Chapter 5 Surface Shape Analysis Method Based on Contour Analysis

An alternative technique to the surface shape analysis discussed in chapter 4 is the contour analysis technique. The contour analysis technique analyzes the surface of the mesh and determines certain patterns of surface curvature in the mesh. Similar to the way Vosselman *et al.* [44] analyzed the scan lines of the original ordered point scans for patterns to fit geometric shapes, this algorithm analyzes cross-sections of the mesh, known as slices, for patterns of interest. The surface of the mesh is broken up into slices, resembling scan lines of ordered point data. The slices are converted into a 2-dimensional representation, and are analyzed for patterns resembling the desired deformations to be identified, resulting in several variable-length pieces describing cross-sections of the deformation of interest. Just like the octree-based surface shape analysis, the goal of contour analysis surface shape analysis is to break up the mesh into pieces and determine which of those pieces contain part of a deformation, as shown in Figure 5.1.

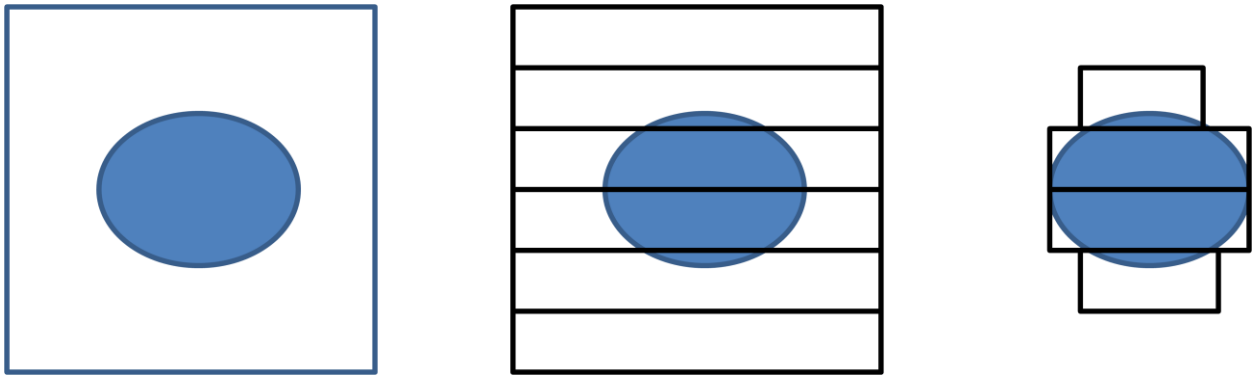

Figure 5.1. Mesh with oval feature (Left). Mesh broken into pieces (Center). Contour analysis surface shape analysis results (Right).

This technique is designed with the specific application of detecting deformations over surfaces, which makes it less of a general solution to feature extraction than the method discussed in chapter 4, but a more accurate and robust method of detecting deformations on a surface. Similar to many 3D feature extraction techniques, this approach looks for areas of high variation from an approximated ideal surface. For the current application, there are some notable limitations in existing techniques, where a surface is fitted to the data, and the deviation of points from this fitted surface is evaluated. Neighbourhood size directly influences what scale of deformations is detected and often multiple passes, at different resolutions, are required to determine deformations of different scales. Also, the entire mesh has to be available before

these techniques are applied. With laser range finders, scans are often performed one line at a time. But such techniques cannot be optimized to analyze one scan line at a time. Finally, most feature extraction algorithms do not take into account the shape of the areas of high variation, therefore all areas of high variation are extracted with little information to aid in further segmentation and classification.

The proposed technique breaks the surface into pieces and classifies the deformation as they are extracted. It also does not require the full mesh to detect deformations, as it can operate on individual scanlines as they are acquired. Also, it allows for selective multi-resolution scans, where multi-resolution evaluation is performed only where necessary because of the recursive component that will be discussed in section 5.5.1. Features of any scale can be determined, but maximum size is limited by the neighbourhood size of the curve-fitting, which will be discussed in section 5.2. Also, the proposed technique provides a lot of information about the curvatures that are extracted, simplifying the segmentation and classification process. The block diagram for the proposed contour analysis technique is shown in Figure 5.2.

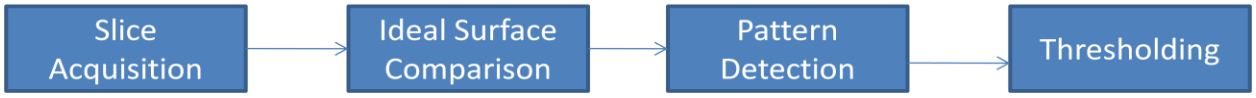

Figure 5.2. Contour Analysis System Diagram

The details will be discussed in the following sections. Slice acquisition will be discussed in section 5.1. Ideal surface comparison will be discussed in section 5.2, and pattern detection in section 5.3. Thresholding will be discussed in section 5.4. Also, several enhancements to the basic contour analysis method are discussed in section 5.5. Results are discussed in section 5.6.

5.1 Slice Acquisition

The first step in analyzing the contours of the mesh is to partition the mesh into slices, which will be analyzed individually. A slice is a cross-section of the mesh, and should represent the shape of a surface along a certain line traced across the surface of the mesh in the form of a series of ordered 3-dimensional points along the surface of the mesh. The characteristics of the slice are that it should represent the shape of a surface along a relatively constant direction. Some examples of acceptable and unacceptable slices are shown in Figure 5.3.

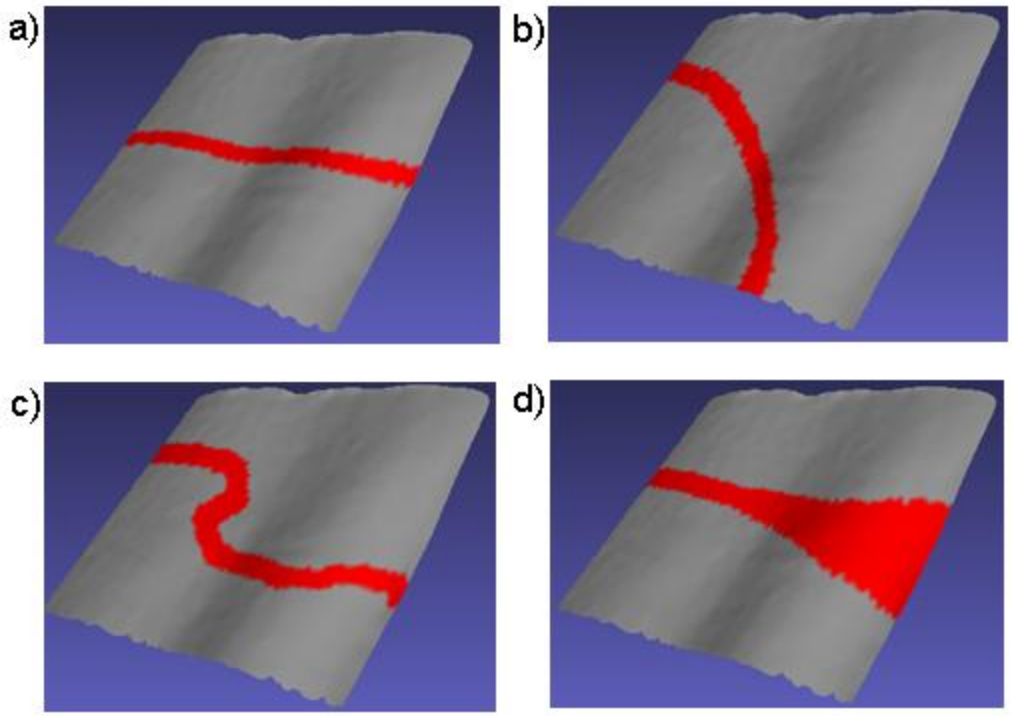

Figure 5.3. a) Acceptable slice. b, c, d) Unacceptable slices.

It is important to define how the mesh is divided into slices. The simplest method is if the data is in the form of a 2.5D range image or an ordered point cloud, a scan line of points can be easily extracted as a slice. This slice meets the criterion of being a cross-section of the mesh, since a scan line of points from a 2.5D range image or ordered point cloud represents the intersection of a plane with the object. Knowledge of which points lie along a line over the surface of the mesh as well as the knowledge of the order of the points going from one end of the line to the other is available in an ordered point cloud, which are the important criteria for a slice. Since many industrial 3D acquisition tools acquire 3D models by scanning lines across the surface of the object, such as [7], each of those lines can be a slice and this algorithm would be well-suited for this scenario.

This section deals with the case where the data is in the less desirable format of an unordered point cloud. As discussed in chapter 2, not all data will be in the format of a 2.5D range image or ordered point cloud, either due to the characteristics of the scanner, or the inability of the mesh to be accurately represented by such a format. In the case of the source data being in the format of an unordered point cloud, the data must be organized such that rows of points can be analyzed as if the data was in the form of an ordered point cloud. With unordered point clouds, determining a slice is difficult as the adjacency of the points is unknown. After surface reconstruction, the data will be in the form of a mesh, so the process of acquiring a

slice is simpler, since adjacency is known. If only points from the point cloud are used, generating a slice of constant direction may be still difficult as illustrated by Figure 5.4, since points may not lie on the same line traced across the surface of the mesh. This is not a significant problem, because the point cloud is expected to be dense enough to describe the surface, even if slight deviations from the line are taken. However, it is important to find which points on the point cloud must be selected to follow the direction of the slice. Since a slice is to represent a cross-section of the mesh, a plane can be intersected with the mesh to determine the cross-section, and points which lie near that plane can be selected.

A simple extension of evaluating the cross-section plane intersecting with the mesh is to determine which points along the mesh perfectly fit the cross-section plane. Points can be interpolated based on sampling the mesh at intervals across its surface where the cross-section plane intersects it, as illustrated by Figure 5.5, and the problem of generating a slice of constant direction will be solved.

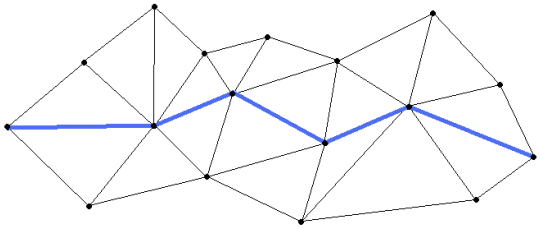

Figure 5.4. Acquiring slice by only using points from the point cloud.

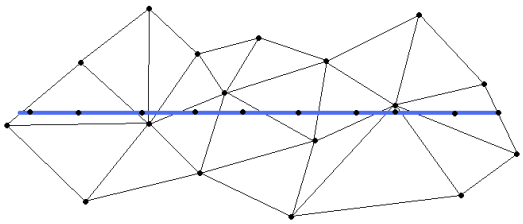

Figure 5.5. Acquiring slice by sampling points through interpolation.

Selecting the appropriate plane to intersect the mesh such that a cross-section can be extracted is difficult to do without some knowledge about the object or the acquisition tool. This solution makes the assumption that cross-sections of the mesh, taken by intersecting a plane parallel to the x-z plane, will provide sufficient information to extract deformations of interest. This assumption implies that the surface of the object is generally facing the z-axis, as shown in Figure 5.6.

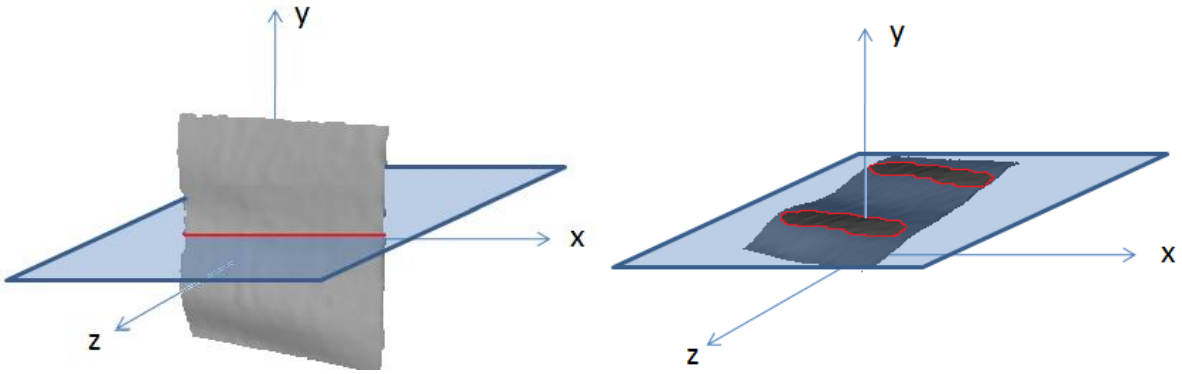

Figure 5.6. Object surface generally facing the z-axis, with a plane parallel to the x-z plane intersecting it, generating a highlighted cross-section of the mesh (Left). Object surface generally facing the y-axis, with the same plane intersecting it, not being able to generate a good cross-section (Right).

The limitation of the above mentioned techniques involving ordered and unordered point clouds is that the slice analysis would be tied closely to the resolution of the scanner. This would require the algorithm to always be run at maximum resolution using all of the data points, without the ability to increase execution speed by using lower resolution version of the mesh. Sometimes the resolution used by the scanner is much higher than required to extract deformations, so the option to operate in a lower resolution and work with less redundant data should be available. Altering the spacing and quantity of the samples along the cross-section of the mesh, as well as the spacing and quantity of the cross-section planes that divide the mesh, can provide this option, however it does so by selecting a subset of the points in the point cloud, instead of making use of the information provided by all of the points. And most importantly, this algorithm must be flexible to the output format of the scanner, so the assumption of using an ordered point cloud cannot be made for a robust solution, as it was in [44, 45].

Under the assumption that slices will be parallel to the x-z plane, a simpler technique to extract the requisite points would be to represent the points of the mesh in a 3-dimensional grid of voxels, with each voxel consisting of several points. In that way, depending on the size of the voxels, variable resolution analysis is supported, and the extraction of points along a plane parallel to the x-z plane can be determined by simply extracting all voxels at a certain y-coordinate, instead of the calculation of a cross-section plane intersecting the mesh. To facilitate the selection of various resolutions, but still using the concept of a grid of voxels, the octree data structure can be used.

The octree data structure has been used with great success in the surface deformation analysis technique proposed in chapter 4. Such a data structure can also be adapted to this new technique to provide some of the adjustments required for robustness and using an

unordered point cloud, while maintaining the benefits provided by an ordered point cloud. The octree data structure facilitates many neighbouring points to be used in the calculation for interpolation, providing a variable resolution capability and being able to suppress noise or outliers affecting points along the slice. Also, this allows the algorithm to easily be run at various resolutions, just by determining which level of the octree the cubes are extracted from, behaving as a method to change the sampling density of the mesh to acquire slices.

The voxel representation provided by the octree data structure is particularly useful for robustness because each node can represent a segment of the mesh, independent of that mesh's sampling density and noisiness. That node can provide a summary of a cluster of points, represented by characteristics such as a centroid, the orientation of the contained surface, the amount of variation within the surface, as well as other characteristics. This summary allows algorithms to be run in spite of the problems associated with uneven sampling density and noise, since the characteristics of a group of points can be analyzed as if corresponding to a single point on the surface.

Surface reconstruction cannot always be trusted, as the mesh provided to the system may not be correctly triangulated as a 2-manifold surface. Sometimes bad triangulation can lead to dead ends in the mesh during slice acquisition when a path is chosen, as shown in Figure 5.7. Inadequate slice acquisition would provide erroneous data to the algorithms following it, such as ideal surface comparison and pattern detection, resulting in poorly detected deformations. The voxel representation tolerates a certain amount of unfavourable triangulations where, even if there are dead ends in the triangulation, the adjacency of voxels in the grid will still allow the correct path to be taken due to the inclusion of multiple points in a voxel as shown in Figure 5.8.

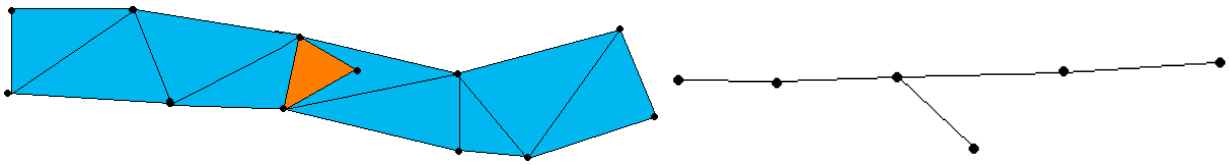

Figure 5.7. Not a 2-manifold surface, because of the highlighted triangle overlapping the surface (Left). Top down view of the same surface, showing the highlighted triangle sticking out (Right).

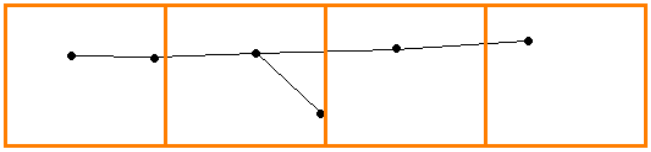

Figure 5.8. Voxel representation allows some tolerance to bad triangulation, preventing an incorrect path to be chosen for the slice acquisition.

An assumption is also made that the size of the voxels is sufficiently large such that several points are taken into account for the robustness purposes discussed above, while being sufficiently small such that each voxel only contains one element of surface with relatively low variation within. This requires an appropriate resolution selection depending on the sampling density of the scanner.

The octree is generated in a similar fashion as it was in the previously proposed technique, in chapter 4. However, during subdivision, it only eliminates nodes that contain no points, as opposed to nodes that fall beneath a certain standard deviation threshold. This results in several 3-dimensional grids, where each grid represents a single resolution and is composed of voxels of a certain size. Also, these 3-dimensional grids remove all voxels that contain no part of the mesh. If the exact resolution that the deformation detection will be carried out at is known, only a 3D grid of voxels of the appropriate resolution needs to be generated, instead of the entire octree. The octree is used for an easy way to switch between resolutions for testing purposes, but often a single resolution will be appropriate for deformation detection.

A resolution of the octree must be selected by the user, which represents the resolution that the contour analysis will be conducted at. This resolution essentially determines how many cross-section planes will be used, the spacing between those planes, and the spacing between the samples taken per plane. This directly translates to how many slices the mesh will be broken up into, how many points will be in each slice, and how much of the mesh is represented by one point in the slice. The maximum resolution is reached when one point in the slice represents one point in the mesh.

After a resolution of the octree is selected, all nodes at that resolution are retrieved, which is a set of 3-dimensional voxels that represent the mesh at that resolution. The slice acquisition takes the 3-dimensional voxels as input, and outputs a list of 3-dimensional points which represent a slice. The flowchart of the slice acquisition for a set of voxels, represented by nodes of the octree, is shown in Figure 5.9.

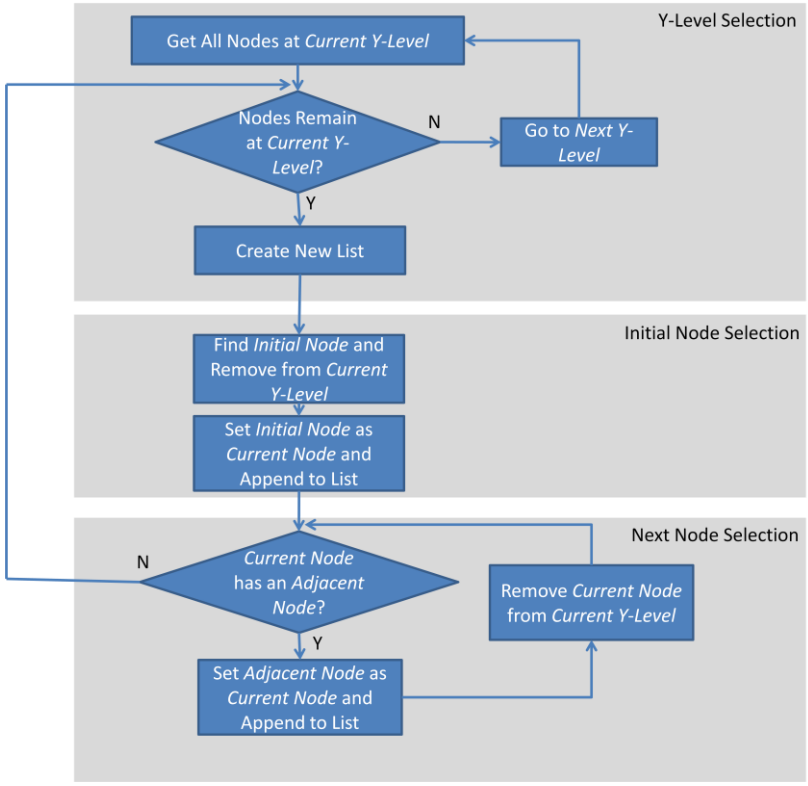

Figure 5.9. Slice acquisition flowchart.

Section 5.1.1 will discuss the Y-Level Selection, section 5.1.2 will detail the Initial Node Selection, and section 5.1.3 will present the Next Node Selection in further detail.

5.1.1 Y-Level Selection

The y-level represents the y-coordinate of the cross-section plane that will be used to generate the slices. The slices that are extracted will lie on planes parallel to the x-z plane, so the algorithm will start with all of the voxels with coordinates at a certain y level. The algorithm starts by selecting a y-level of 0.

If all voxels at the current y-level have been extracted, through Initial Node Selection or Next Node Selection, the algorithm moves on to the next y-level. If no y-levels remain, the Slice Acquisition stage is complete.

5.1.2 Initial Node Selection

Now that only voxels at a certain y-level are being evaluated, an initial node must be selected to start the slice. A suitable initial node will be at the boundary of a slice.

If the models being evaluated are relatively flat and are of consistent orientation, the initial node of a slice can be consistently selected using predefined criteria. The automotive body parts being evaluated for this thesis are models that are relatively flat. Also, because the scanning station and automotive panel are put in the same location every time, the orientation of the resulting mesh is constant between scans of different panels. As shown in chapter 3, in the prototype developed for this thesis, the mesh is facing dominantly in the z-direction, and is relatively parallel to the x-y plane, which means the nodes containing the boundaries of the mesh along the y-axis are the nodes with the lowest and highest x value. Therefore the predefined criterion used in this implementation is that slices start with the node at the lower x boundary, which means the node with the lowest x value boundary. An example of this is shown in Figure 5.10.

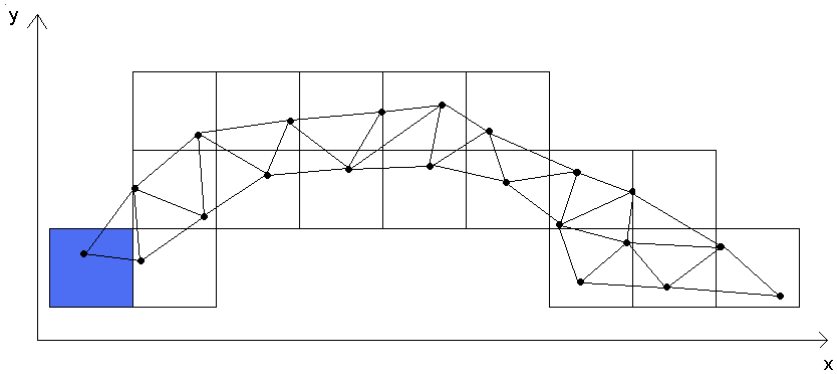

Figure 5.10. Voxel with lowest x-boundary is selected as initial node.

For robustness purposes, a more general solution is also proposed for when the characteristics of a mesh are not so consistent. As shown in Figure 5.11, the criteria defined for the meshes being evaluated in this thesis may not be sufficient for other meshes.

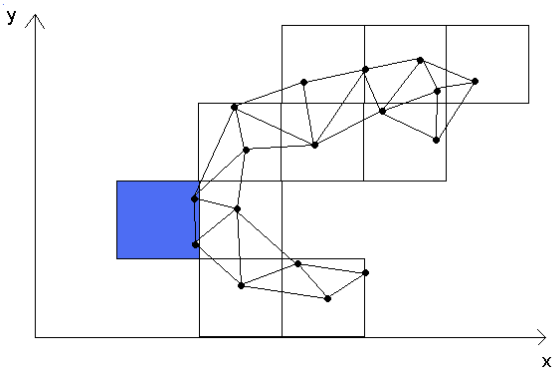

Figure 5.11. Voxel with lowest x-boundary is not always the best initial node.

A slice can either be continuous or discontinuous, as shown in Figure 5.12. If the slice is discontinuous, a search for the boundary to the mesh at the current y-level can be executed, and the voxel that contains that boundary would make an ideal initial node. Each point in each voxel at the current y-level is evaluated to determine if it satisfies either of these conditions: it is adjacent to fewer than three points of the mesh or it belongs to fewer triangles than the number of points it is adjacent to. If at least one of the two conditions is satisfied, the point is on a boundary. Examples of boundary points are depicted by circles in Figure 5.13. The voxel containing that boundary point will be used as an initial node. If no boundary point is found, the slice is continuous and any voxel can be selected as the initial node.

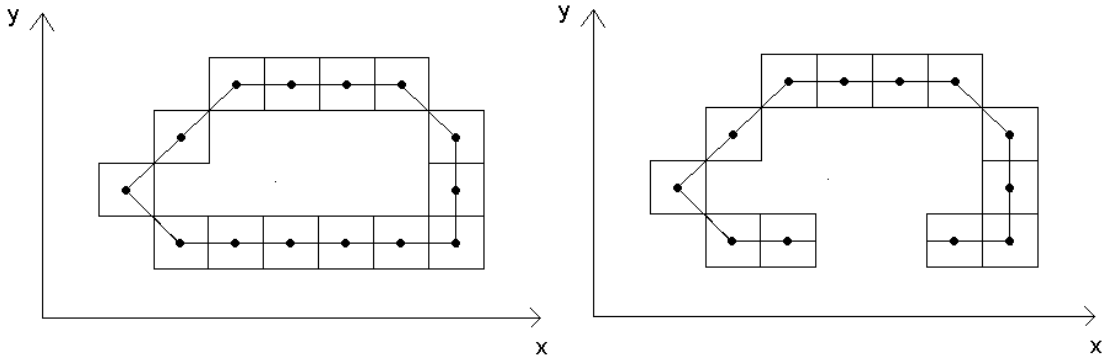

Figure 5.12. Continuous slice (Left). Discontinuous slice (Right).

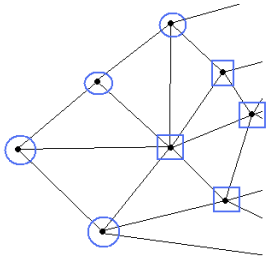

Figure 5.13. A segment of a mesh, where circles highlight boundary points and acquire boxes highlight non-boundary points.

Away from the top and bottom borders of the mesh, it is realistic to only find two boundary nodes. However, near the top and bottom borders of the mesh, multiple nodes contain boundary points, since many meet the criteria. This is shown in Figure 5.14.

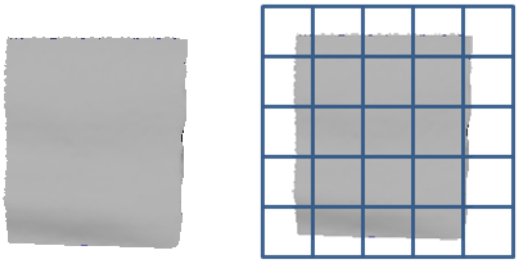

Figure 5.14. Original surface (Left). Grid representation shows only two boundary cells at each row, except at the top and bottom row where all the nodes are boundary cells (Right).

There are a few extra steps that can aid in determining an appropriate initial node. We look for an adjacent node to each of the possible nodes. And if there is a node that is adjacent to only one other node, that node can be determined as an initial node. If still no node meets this criterion, we can search for the two points that are the furthest apart, and either of the nodes containing those points can be set as an initial node. Even after following these steps, it is possible that an adequate initial node is still not selected. Since the likelihood of selecting an incorrect initial node when evaluating the boundaries of the mesh that are parallel to the x-z plane is high, the top and bottom y-levels are ignored for slice acquisition.

When an initial node is determined, it is extracted from the y-level. A list that represents the new slice is created, and the centroid of the initial node is added to that slice.

Again, in this thesis, as shown in chapter 3, the mesh is facing dominantly in the z-direction, and is relatively parallel to the x-y plane, so it is sufficient to do the initial node selection by finding the lowest X value node.

5.1.3 Next Node Selection

After the centroid of the initial node is added to the slice, the next node to be added to the slice must be selected. This selection could be performed by using just the octree data structure. Since each node in the octree data structure represents a voxel, it is possible to acquire spatially adjacent voxels. In an ideal scenario, when evaluating only the y-level nodes, only one node will be spatially adjacent to the initial node and final node, and only two nodes will be spatially adjacent to all the other nodes. However, this is often not the case, as illustrated in Figure 5.15. Determining which of the spatially adjacent nodes to pick is a difficult task. Because of this, the next node selection must be done in a more robust manner.

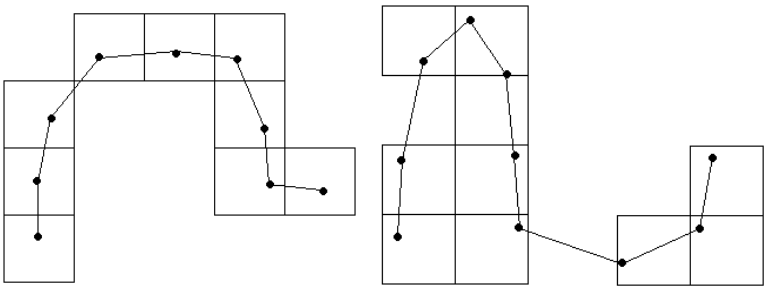

Figure 5.15. Ideal grid adjacency, where using grid cell adjacency can determine the slice (Left). Non-ideal grid adjacency, where grid cell adjacency cannot easily determine the slice (Right).

All points in the current node are evaluated to see which points are adjacent to them in the mesh. When one of the adjacent points lies outside the current node, it is evaluated for two criteria: it does not belong to a node that has already been added to the slice, and it does not belong to a node that is on a different y-level. If the criteria are met, the node which that adjacent point belongs to is the adjacent node. If it is not, the search continues. If no point in the node meets these criteria, there is no adjacent node, and the slice concludes.

Since the thickness of the slice is the range of the y values represented by the voxel, many points will likely be contained in the node. Because of this, and the uncertain shape of the mesh, it is possible that multiple nodes can be determined as adjacent nodes. Based on the proposed technique, the first adjacent node to be determined will be pursued. This may lead to concluding the slice early due to incorrect determination of the boundary, or putting points on the slice out of order. To remedy this problem, when an adjacent node is found, it is added to a temporary list. The search continues with the other points in the node, until all points have been exhausted. All of the adjacent nodes in the temporary list are locally merged into a single node, which we will call the supernode, to contain all of those points. That supernode is treated as the adjacent node as shown in Figure 5.16.

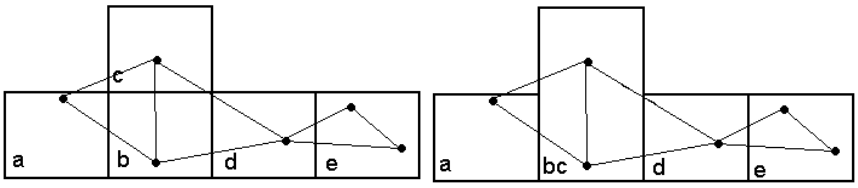

Figure 5.16. Node *a* is adjacent to both node *b* and node *c* (Left*).* Nodes *b* and *c* are joined to make supernode *bc*, so node *a* is adjacent to only node *bc* (Right).

Starting with the initial node, a spatially adjacent node which exists among the same y-level nodes can be selected. The centroid of that node is added to the slice. Now, starting with

that node, the next node is selected, which is a spatially adjacent node which exists among the same y-level nodes, which is not already part of the slice and has not already been evaluated. The centroid of that node is added to the slice. This process repeats until no next node can be found, thus concluding the slice. The flowchart of this whole process is shown in Figure 5.17.

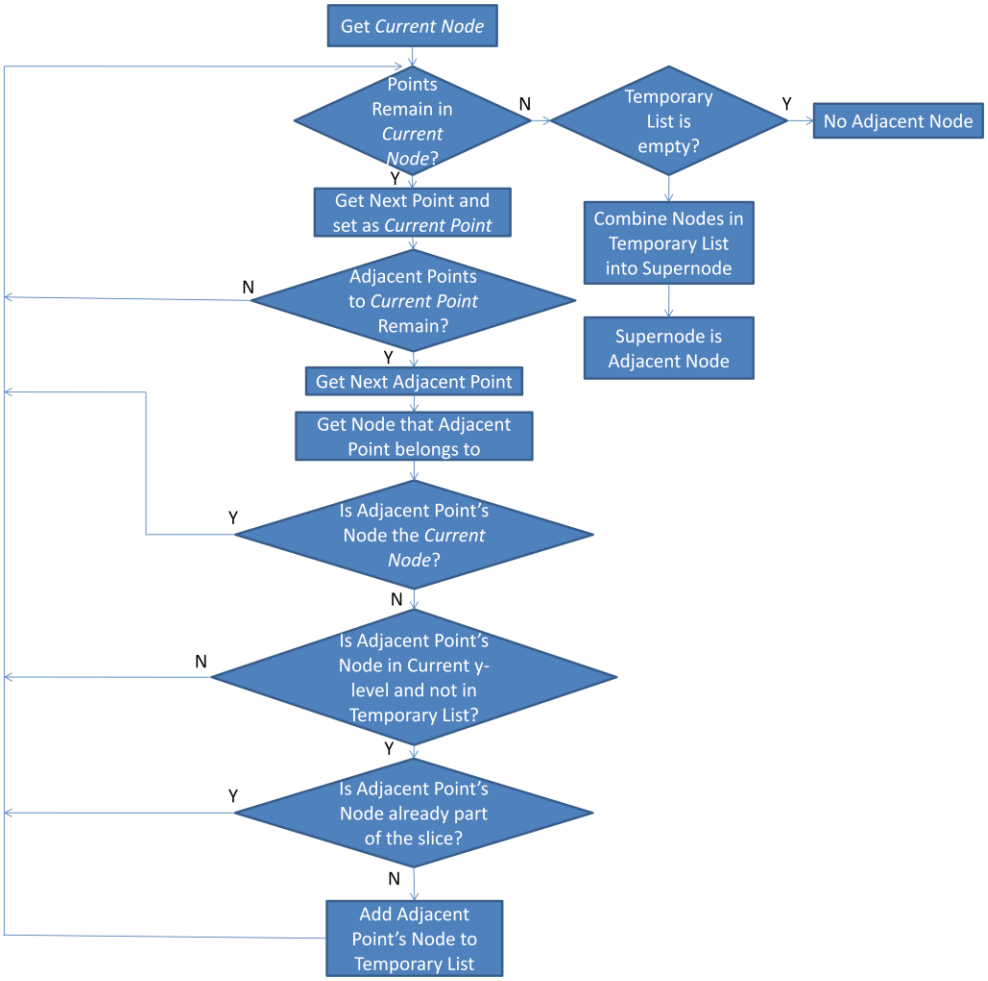

Figure 5.17. Algorithm to determine Next Node.

The result of this process is a set of ordered 3-dimensional points which is passed on to the ideal surface comparison method.

5.2 Ideal Surface Comparison

When a slice is acquired, it consists of a series of ordered 3-dimensional points. The goal of the contour analysis algorithm is to analyze the shape of each slice in 2 dimensions, as shown in Figure 5.18.

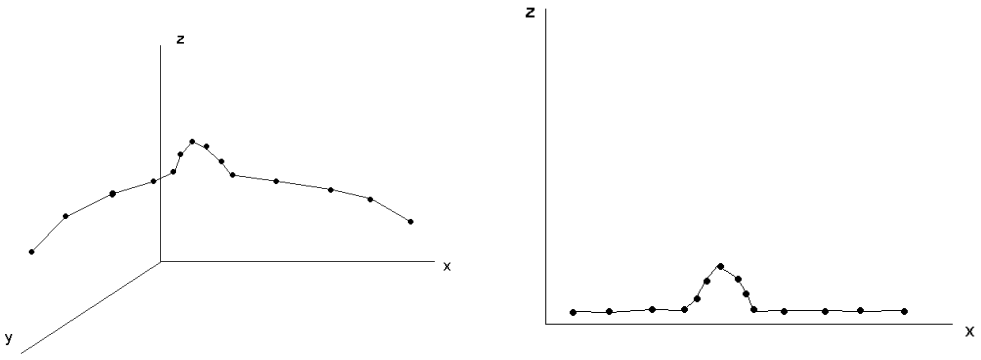

Figure 5.18. Ordered 3-dimensional points of a slice (Left). 2-dimensional difference curve (Right).

The ideal surface comparison stage converts the 3-dimensional points into a 2-dimensional difference curve which represents the difference between an approximated ideal surface and the actual surface. This process is shown in Figure 5.19.

Figure 5.19. Ideal surface comparison block diagram.

5.2.1 Flattening

The 2-dimensional difference curve represents the difference between the 3-dimensional contour and an estimated ideal 3-dimensional contour. This ideal contour can be determined by a best-fit curve to the 3-dimensional set of points. Since working in three dimensions is more computationally complex than working in two dimensions, the 3-dimensional coordinates will be *flattened* into 2-dimensional coordinates. Since the slice lies on a plane parallel to the x-z plane, and the size of the volume represented by each node in the y-direction is small compared to the dimensions of the entire model, the y-coordinate of the points will be the least important as they are similar amongst all the points. The points are all projected on to the x-z plane, and recorded as a set of ordered 2-dimensional coordinates. This set of ordered points will be referred to as the 2-dimensional slice. This slice represents the cross-section of the mesh. This process is shown in Figure 5.20.

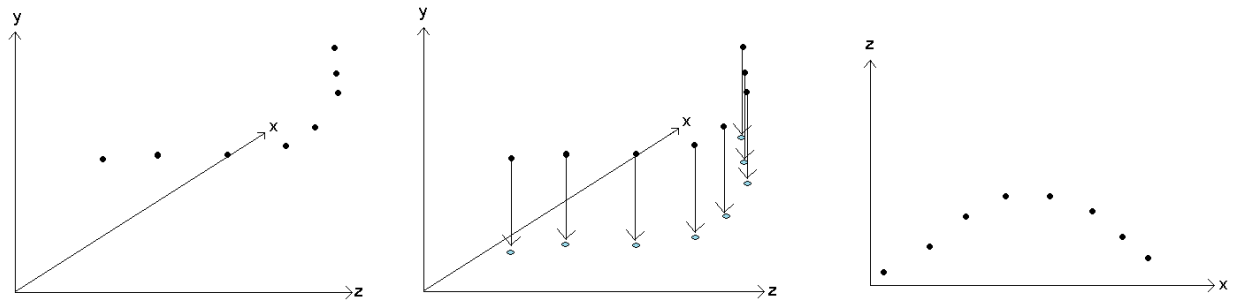

Figure 5.20. 3D slice (Left). 3D slice projection onto x-z plane (Middle). 2D slice (Right).

5.2.2 Curve-fitting

In theory, the 2-dimensional difference curve is to represent the difference between the 3-dimensional slice and an ideal surface which contains no noise or deformations. Since there is no *a priori* knowledge of the surface and no ideal model that is available for comparison, this ideal surface is unknown. The goal of curve fitting is to approximate an ideal surface from the points contained in the slice, similar to the way the data is processed in [3].

To approximate the ideal surface at each point, an ordinary least-squares 1st order polynomial, also known as a line-of-best-fit, is generated for each point in the slice. This moving line-of-best-fit comprises the ideal surface of the 2D slice. A higher order polynomial could be used, but the 1st order polynomial is preferred for several reasons. Using the same neighbourhood size, higher order polynomials tend to fit the deformation and result in the deformation being incorrectly considered a part of the ideal surface, where a 1st order polynomial will not be affected as greatly. Also, the highly-curved deformations lie on surfaces that have a much lower relative curvature, meaning that a line is sufficient to represent the surface in the area surrounding the deformation.

Since the remainder of the technique uses a 2D coordinate system, the z-axis of the slice is mapped to the y-axis, such that the slice lies on the x-y plane. The resulting line-of-best-fit equation is represented by the equation $ax + by + c = 0$. Thus the 2D difference curve is also located on the x-y plane.

The least-squares fitting uses the points in a certain neighbourhood to help estimate an ideal surface around that point. If too small of a neighbourhood is used, the approximated ideal surface used for comparison would fit the actual data too strongly. The difference between the approximated ideal surface and actual surface would result in a 2D difference curve that would not highlight the deformations, as shown in Figure 5.21. If too large of a neighbourhood is used, the approximated ideal surface will not fit the curvatures at all. The difference between the approximated ideal surface and the actual surface would result in a 2D difference curve that

would describe all curvatures, including broad curvatures which may be the actual curvature of the surface, as shown in Figure 5.22. A small enough neighbourhood for the least-squares fitting must be selected to eliminate undesired broad curvatures, but a large enough neighbourhood must be selected such that the fitting is not influenced much by deformations and noise, as shown in Figure 5.23. The neighbourhood, *r*, must be selected by the user to optimize performance for the characteristics of the scanner and the type of surfaces being analyzed.

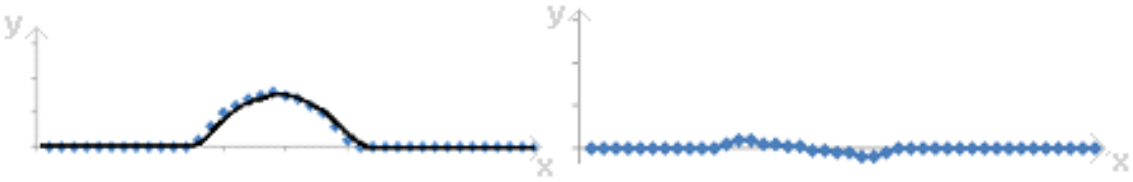

Figure 5.21. Small neighbourhood for fitting can result in an ideal surface that too closely resembles the actual surface (Left). The difference between the actual surface and the ideal surface is minimal and does not highlight the deformation (Right).

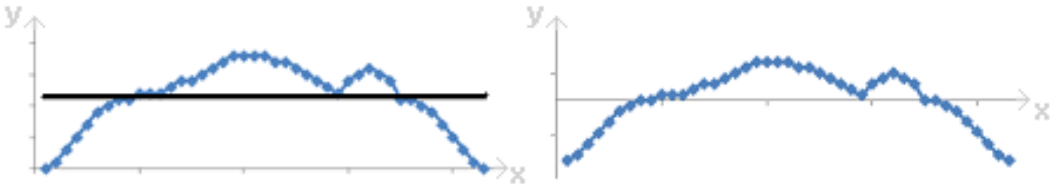

Figure 5.22. Large neighbourhood for fitting can result in an ideal surface that does not resemble the surface at all, and instead resembles a flat surface (Left). The difference between the actual surface and the ideal surface does not remove the broad curvature of the surface (Right).

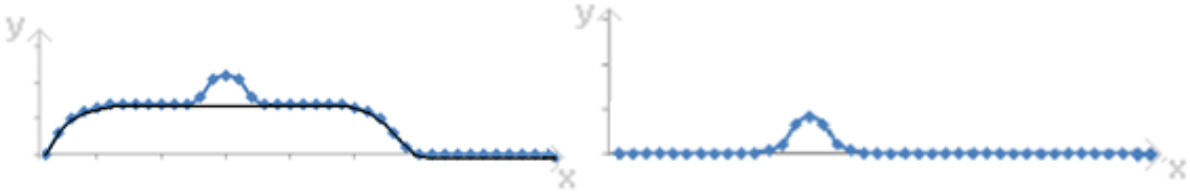

Figure 5.23. Properly fitted ideal surface conforms to surface, but not deformation (Left). The difference between actual surface and the ideal surface correctly shows only the deformation (Right).

5.2.3 Projection

The 2D difference curve is supposed to represent the difference between the actual surface and the ideal surface. The ideal surface is generated as shown during the curve-fitting step in section 5.2.2, and is composed of a line-of-best-fit for each point. The projection step generates the 2D difference curve, by evaluating the position of each point with respect to its line-of-best-fit. The signed distance from the point to its respective line-of-best-fit is the value of the corresponding point on the 2-dimensional difference curve, as shown in Eq. 5.1.

$$d = \frac{(au+bv+c)}{\sqrt{a^2+b^2}} \tag{5.1}$$

where d is the signed distance from the actual point, (u, v), to the line-of-best-fit represented by the equation $ax + by + c = 0$. The 2-dimensional difference curve is an ordered set of values, and is composed of the *d* value for each point. The difference curve will represent the difference between the actual 2-dimensional slice and an ideal 2-dimensional slice, which in turn represents the difference between the 3-dimensional slice and an ideal surface.

5.3 Pattern Detection

The 2-dimensional difference curve must be analyzed for certain patterns, such that deformations of importance can be identified. The goal of this component is to generate a list of all deformations, meeting the criteria of the deformations of interest, in the surface of the mesh.

Since the deformations of importance in the present application are dents and dings, in a 2-dimensional difference curve, the deformations will have certain predictable patterns. A dent is a deformation that goes into the surface as shown in Figure 5.24. In the 2-dimensional difference curve, a dent can be seen as a distinctive valley, which is defined by the contour decreasing and then increasing. The lowest point in this pattern will represent the magnitude of the dent. A ding is a deformation that comes out of the surface as shown in Figure 5.25. In the 2-dimensional difference curve, a ding can be seen as a distinctive peak, which is defined by the contour increasing and then decreasing. The highest point in this pattern will represent the magnitude of the ding.

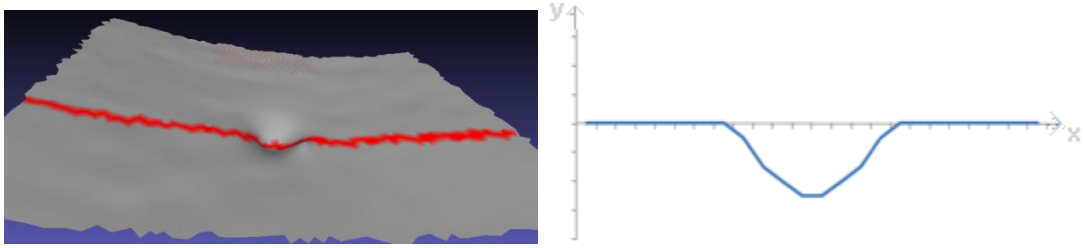

Figure 5.24. A slice taken across a dent, which is a deformation going into the surface (Left). The 2D difference curve of a dent on a surface (Right).

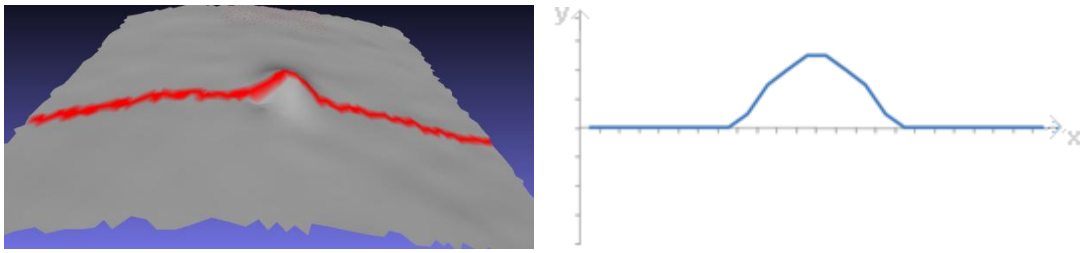

Figure 5.25. A slice taken across a ding, which is a deformation coming out of the surface (Left). The 2D difference curve of a ding on a surface (Right).

Along with the researched deformations, several undesired patterns will be identified during this analysis. Patterns such as noise or deformations that are not of interest will appear also, but can be removed during the thresholding stage, which will be further detailed below.

The path of the difference curve is followed, point to point, until a certain pattern representing a ding or dent is found. This pattern detection can be extended to looking for other possible patterns than the ones considered here, however for the purposes of this application, the very general definitions of a dent and ding are used as the patterns of interest.

When a point on the difference curve lies on the x-axis there is no difference between the 2D slice at that point and the ideal surface, therefore any deviation of the difference curve from the x-axis indicates some sort of variation. A set of consecutive points that lie above the x-axis in the difference curve can be classified as ding deformation pattern, while a set of consecutive points that lie beneath the x-axis can be classified as a dent deformation pattern. An example is shown in Figure 5.26.

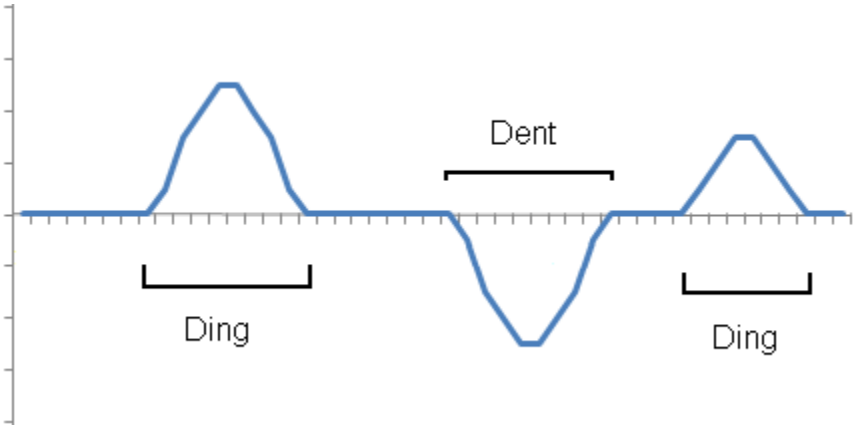

Figure 5.26. Deformation patterns in 2D difference curve.

When this process has been done for all of the slices, the end result is a set of deformation patterns contained in the mesh. These deformation patterns will be segmented

from the rest of the mesh and classified to leave only deformations of interest, and that will be discussed in chapter 6. Note that the ideal surface is generated for each slice, without using any knowledge of the slices adjacent to it. When selecting an octree resolution that is too low, each node contains a large amount of the mesh relative to the size of the deformation. Since the centroid of each node is used to represent a point on the 2D slice, the deformation will not exert as much influence on the centroid as the non-deformation areas also contained in the node. As a result, the ideal surface comparison stage will generate a 2D difference curve which does not contain the actual deformation, which causes the pattern detection to fail in extracting the deformation.

5.4 Thresholding

The extracted deformation patterns have certain characteristics that can be useful in determining if they are areas of interest or not. The two most important characteristics are *magnitude* and *span*. Magnitude is the distance between the highest point and the x-axis in a ding deformation pattern, or the distance between the lowest point and the x-axis in a dent deformation pattern. Span is the distance between the first and last point in the deformation pattern. These values use the same units as the original data.

The centroid of the deformation pattern is also a useful metric. A broad curve that extends over all of the points in the deformation pattern will have a centroid where the y-component is closer to the peak or valley of the deformation pattern. A sharp curve that extends over a small part of the deformation pattern will have a centroid with a y-value closer to the starting elevation. Examples of these are shown in Figure 5.27.

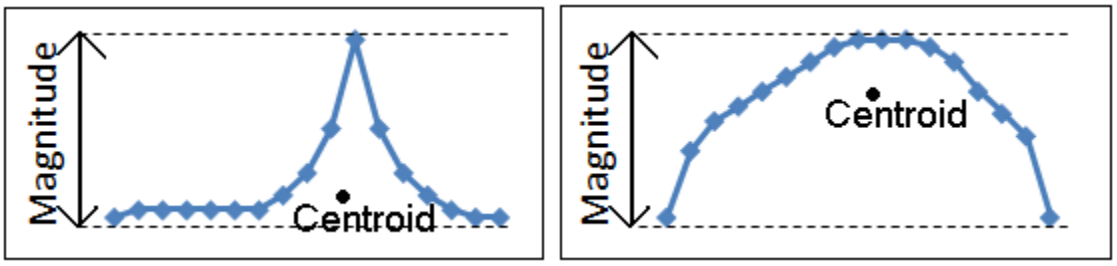

Figure 5.27. A sharp ding has a centroid closer to the starting elevation (Top Left). A broad ding has a centroid that is closer to the peak of the deformation pattern (Top Right).

For the purposes of the current application, a more useful metric that is considered is sharpness, as defined by Eq. 5.2.

$$\bar{p} = \sum_{i=1}^{n} \frac{p_i}{n}$$

$$sharpness = 1 - \left| \frac{\bar{p}_y}{magnitude} \right| \quad (5.2)$$

where n is the number of points in the deformation pattern, p_i is the point at index i in the deformation pattern, $\bar{p}_y$ is the y-component of the $\bar{p}$, and *magnitude* is the magnitude of the deformation pattern as discussed in section 5.3. Sharpness values closer to 1 indicate a sharp curve, and values closer to 0 indicate a broad curve.

The final list of all deformation patterns in the mesh can be analyzed to determine which contours meet the criteria of the dings or dents we are looking for. By thresholding the characteristics exhibited by these deformation patterns, certain types of features can be selected or removed. To remove noise, a simple threshold of the span and magnitude factors can be used. Since noise will be detected as deformation patterns of very small magnitude, they can be removed by eliminating curves with less magnitude than a certain threshold. Also, noise will have very short spans, as they will affect the surface with very high frequency. If noise is significant, it will make up the majority of the deformation patterns, thus significantly reducing the number of potential dents and dings after applying this thresholding process.

Actual dents and dings can be separated from body curvature by thresholding the span attribute. Curves that are long are likely to not be dents or dings, as the dents and dings should be smaller relative to the overall body of the automotive part. A large curve detected as a dent or ding is more likely to be an aesthetic curvature. Also, an evaluation of the sharpness parameter can determine whether the curve in question is a dent/ding, or a less distinct curvature which may be an aesthetic feature of the automotive part. These thresholds will be selected to isolate deformation patterns of interest before the segmentation and classification phase, which will be discussed in chapter 6.

5.5 Enhancements

The proposed contour analysis technique can function as is, but there are some short-comings that can be addressed with two enhancements.

Due to the complex nature of meshes in general, some deformations might not be detected with the basic contour analysis technique. Some deformations lie on top of curvatures or on top of other deformations, and may need to be extracted. This is often accomplished by multiresolution feature extraction, but a novel technique is proposed as an enhancement to the

contour analysis technique, using recursion, to solve this problem. It is discussed in section 5.5.1.

Also, the ordinary least-squares curve-fitting might work well, but poorly fitted curves are always a possibility. Noise, outliers, and especially deformations can affect the curve-fitting, and influence it more than other points, generating an ideal surface estimation which is far from optimal. The weighted least-squares curve-fitting enhancement provides a solution to poor curve-fitting, and is discussed in section 5.5.2.

5.5.1 Recursion Enhancement

The described method works well when the neighbourhood, *r*, is optimally selected such that no other curvature but the deformation is noticeable on the difference curve. However, real world data is not necessarily this simple. A potential problem is that there are often curvatures on an automotive body part that are part of the design, and there will be deformations of interest on top of such curvatures which are only slightly smaller in magnitude. An example of a problematic slice is shown in Figure 5.28, where the desired deformation is relative to a broad curve. Assuming that the ideal surface comparison is imperfect due to a large neighbourhood selection for *r*, the burden is on the pattern detection algorithm to extract both curvatures, and the algorithm described in the previous section would not detect such a deformation, since it would detect the large curvature instead.

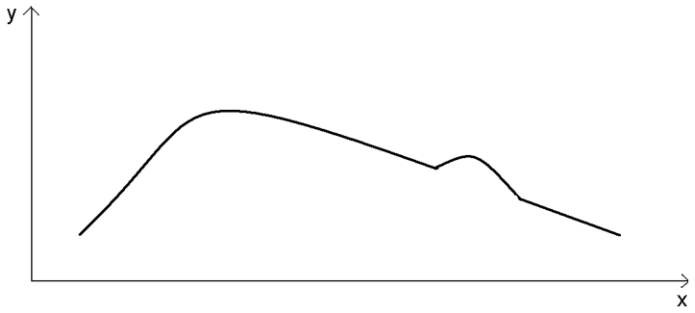

Figure 5.28. Deformation superimposed on a smooth curve.

If the curve-fitting uses too large of a value for *r*, the large curvature will show in the 2-dimensional difference curve and the large curvatures would be detected at the expense of the deformations that occur relative to that large curvature instead of relative to the original ideal surface. If the curve-fitting uses too small of a value for *r*, as explained in section 5.2.2, the ideal surface used for comparison would fit the actual data too strongly, resulting in a set of points that would not display the deformation. An important enhancement to the proposed

technique is to add a recursive component in order to detect deformations of different orientations and scales that may or may not be located on a curve instead of a flat line in the 2D difference curve. The adjustment to the main block diagram is shown in Figure 5.29.

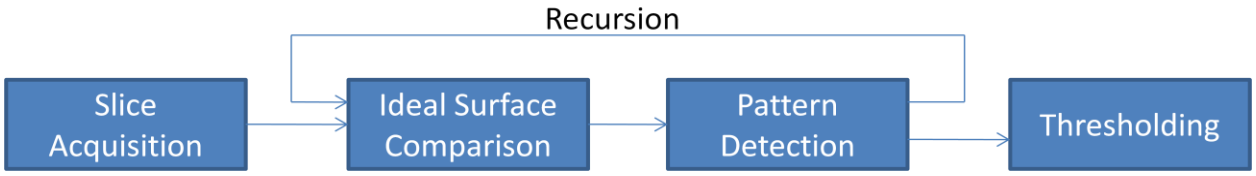

Figure 5.29. Contour Analysis System Diagram with Recursion Enhancement

This recursion enhancement is not necessary if the neighbourhood size is selected optimally, resulting in only the deformation being visible in the 2D difference curve. However, for robustness purposes, this enhancement ensures that small deformations are not overlooked because they lay on other surfaces with deformation-like characteristics.

After a deformation has been detected and the points belonging to that deformation define a deformation pattern, the curve-fitting and projection subsystems of the ideal surface comparison step, described in section 5.2.2 and 5.2.3 respectively, is performed with the points contained in the deformation pattern points as the data. The neighbourhood size, *r*, is altered to account for the scale of the deformation pattern relative to the 2-dimensional difference curve, using Eq. 5.3.

$$r_{new} = \frac{n}{m} r \tag{5.3}$$

where r_{new} is the neighbourhood size for the next ideal surface comparison step, *r* is the neighbourhood size that was used for the ideal surface comparison step, *n* is the number of points in the extracted deformation pattern and *m* is the number of points in the 2-dimensional difference curve.

The results of the curve-fitting and projection subsystems on the extracted deformation pattern, which is a subset of points on the original 2-dimensional difference curve, is a new 2-dimensional difference curve which represents the points on the deformation pattern relative to a newly fitted ideal curve of smaller neighbourhood size, indicating higher resolution. It is a higher resolution analysis because the neighbourhood size is adjusted to be smaller, meaning the ideal surface comparison stage is fitted more closely to the points and larger curvatures are ignored. Pattern detection for dings and dents are applied upon this new 2-dimensional difference curve to extract a new set of deformation patterns. This recursive process is applied to each new deformation pattern that is found and occurs for a finite number of recursions, until the neighbourhood size r_{new} falls beneath a certain threshold indicating that there is not enough

data to perform accurate ideal surface comparison, or until all of the projected points fall beneath a certain threshold indicating that the curve fits the deformation well and that there are likely no further deformations to be found through recursion. The relationship between the ideal surface comparison step and the pattern detection step is shown in Figure 5.30. An example of the recursive process is shown in Figure 5.31.

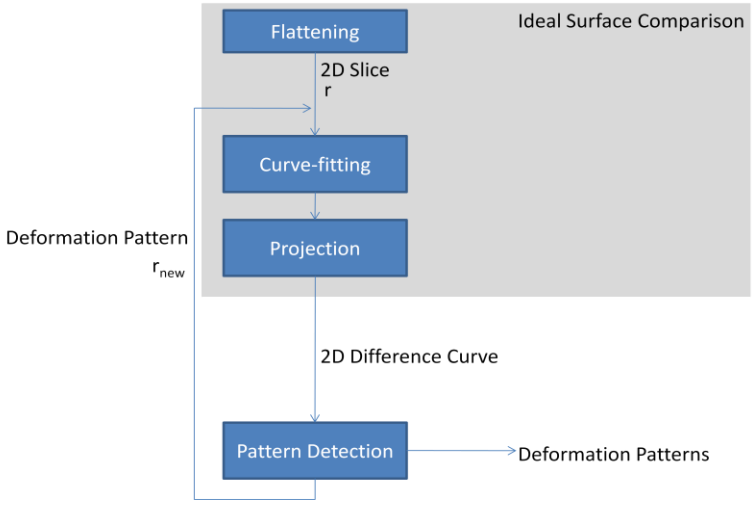

Figure 5.30. Recursion between the ideal surface comparison stage and pattern detection stage, resulting in an output of deformation patterns.

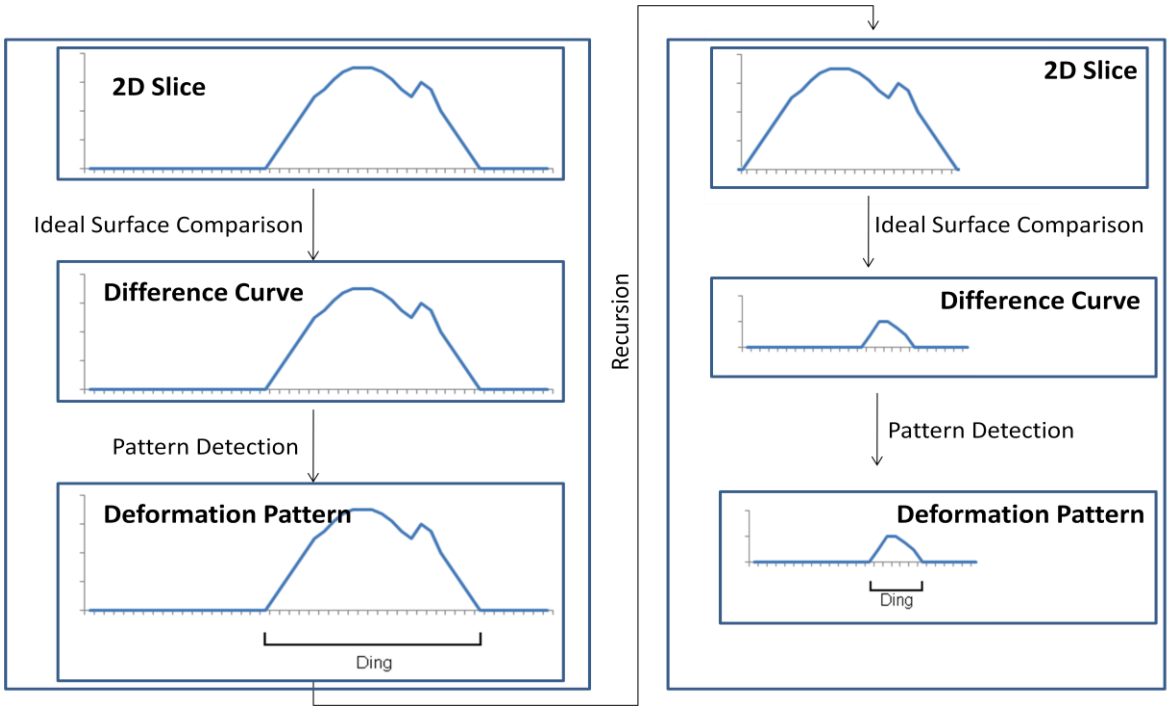

Figure 5.31. Example of Recursion.

It is important to note that for more levels of recursion, the accurate extraction of deformations is less and less likely, depending on the resolution and the ability to fit an appropriate curve to the deformation, since the deformations will be small and defined by fewer and fewer points.

5.5.2 Weighted Least-Squares Fitting Enhancement

This enhancement deals with the ideal surface comparison step, specifically the curve-fitting step, discussed in section 5.2.2. Though ordinary least-squares fitting can provide very good results, the results can be further improved. Ordinary least-squares regression can be heavily influenced by outliers, and in the case of this system, the deformations in question would influence the curve-fitting significantly. An example of this is shown in Figure 5.32, where there is a dent in the 2D slice, the curve fitting is influenced significantly by the dent thus giving an inaccurate representation of an ideal surface, which is reflected by showing extra curvature in the difference curve that does not actually exist. Using the ordinary least-squares regression for fitting will detect actual deformations to be smaller in magnitude than they actually are, while erroneously detecting non-deformation areas with a larger magnitude than they actually are.

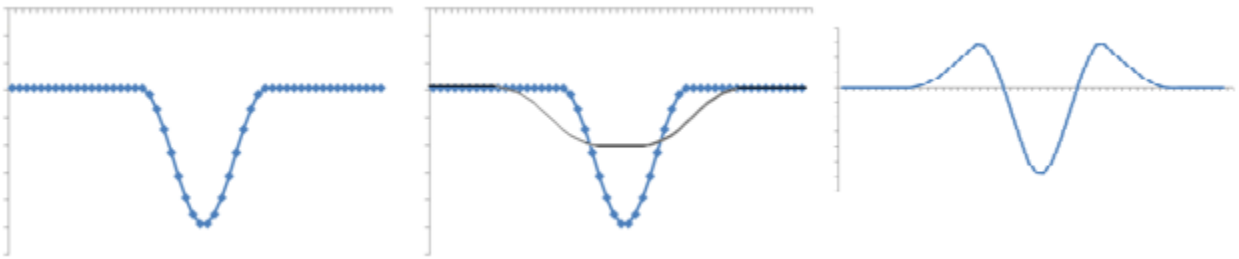

Figure 5.32. 2D Slice with deformation (Left). 2D Slice with least-squares ideal surface (Center). Difference between 2D slice and least-squares ideal surface (Right).

Though a larger neighbourhood can reduce the impact of these deformations, large neighbourhoods introduce other problems as explained above. The ordinary least-squares fitting technique cannot determine what parts of the data would belong to an ideal curve and what parts would not, so it weights all the points equally, hence the major problem in using this technique for ideal surface fitting. In chapter 4, the octree-based techniques provided very useful metrics of the deformation in the surface. σ represents the standard deviation of surface normals contained in a node and s represents the aggregate σ over three resolutions in the octree. Though they were not totally accurate in extracting the deformations, the σ values and s values gave an indication of what areas had strong surface changes. Since each point in the

2D slice represents the contents of one octree cube, the σ value and *s* value corresponding to that octree cube can be used to determine how much each point in the 2D slice influences the curve-fitting calculation. Points representing areas with minimal surface changes, reflected by low σ and *s* values, should be used in the least-squares calculation while reducing the impact of areas with significant surface changes, such as deformations, that are reflected by high σ and *s* values. It is favourable to use the *s* value instead of the σ value, because of the larger disparity between non-feature nodes and feature nodes, as shown in section 4.4. This larger disparity will aid in reducing the impact of areas that would be extracted as feature nodes, in the curve-fitting. The *s* metric can be calculated without the execution of the octree feature extraction by generating the voxels at only the selected resolution for slice acquisition.

The weighted least-squares method is a modification of the standard least-squares method to assign weights to each point involved in the calculation. These weights determine the influence that a given point has on the least-squares fitting calculation. By setting the weights for each point as some function of the σ or *s* value of the octree node it represents, the ideal curve fitting becomes less influenced by the deformations and more closely resembles the actual ideal surface. The selection of this function will determine how much of an improvement can be achieved in the results over using the ordinary least-squares regression. Note that the weights, before use, must be normalized such that the sum of the weights equals 1.

For this thesis, three functions were evaluated. The first function is $w_i = \frac{1}{s_i}$. The second function is $w_i = s_{max} - s_i$. The third is a hard threshold function $w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$.

In an ideal situation where the surface is flat everywhere but over the deformation, the first function, $w_i = \frac{1}{s_i}$, would be useful, since areas with no deformations will have σ values of 0, while areas with deformations would have positive σ values. Even though some s_i will be 0, resulting in w_i of infinity, upon normalization, all of the non-deformation areas will have equal weights, while areas of deformations will have weights of 0. This would give a perfect ideal curve, which is not at all affected by the deformation, as shown in Figure 5.33. In non-ideal circumstances, none of the σ values will be 0, and the ratio of the σ value in one cube to the σ values in other cubes will be a determining factor in how much variation in influence there will be between different points. For example, if the variation between the flat areas and non-flat areas are 0.5 and 0.55, that would not give much variation when using the first function, because the reciprocal values will be very close to each other. But if the variation between flat areas and non-flat areas are 0.01 and 0.95, the reciprocal values will be much more varied. As a result,

the effectiveness of the weighted least-squares method over the ordinary least-squares method could vary significantly and unpredictably from mesh to mesh indicating a lack of robustness. An example of this variation is shown in Figure 5.34.

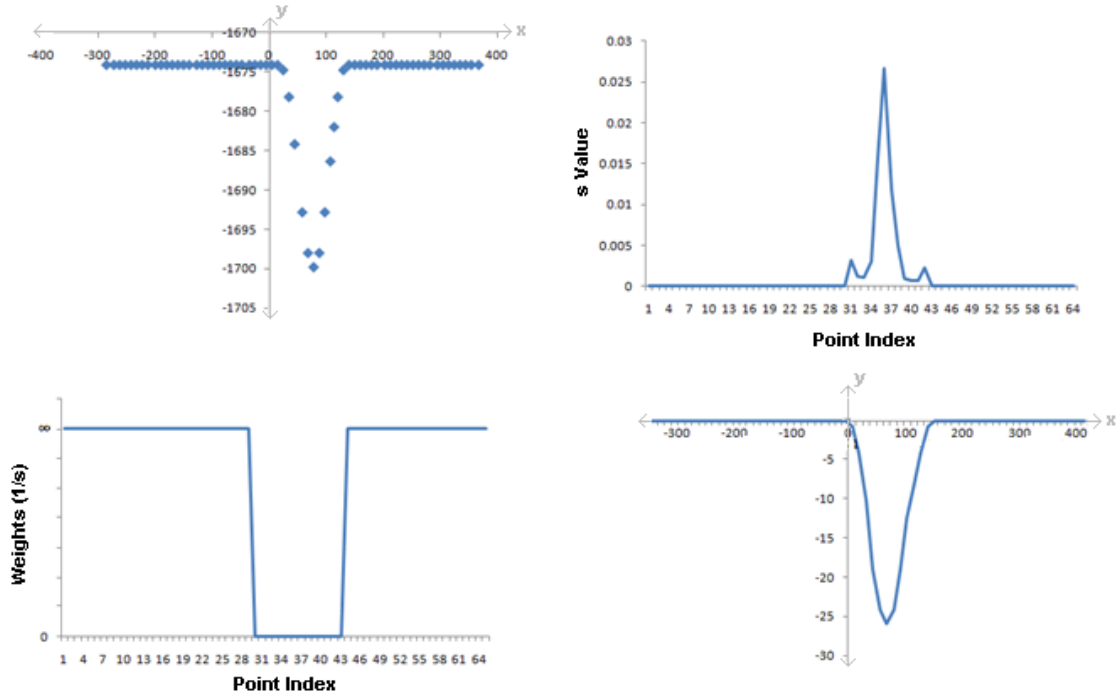

Figure 5.33. 2D slice of ideal flat surface, simple deformation, and no noise (Top Left). s_i plot of the octree nodes represented by the slice (Top Right). Weights of each point using $w_i = \frac{1}{s_i}$ (Bottom Left). 2D difference curve using weights (Bottom Right).

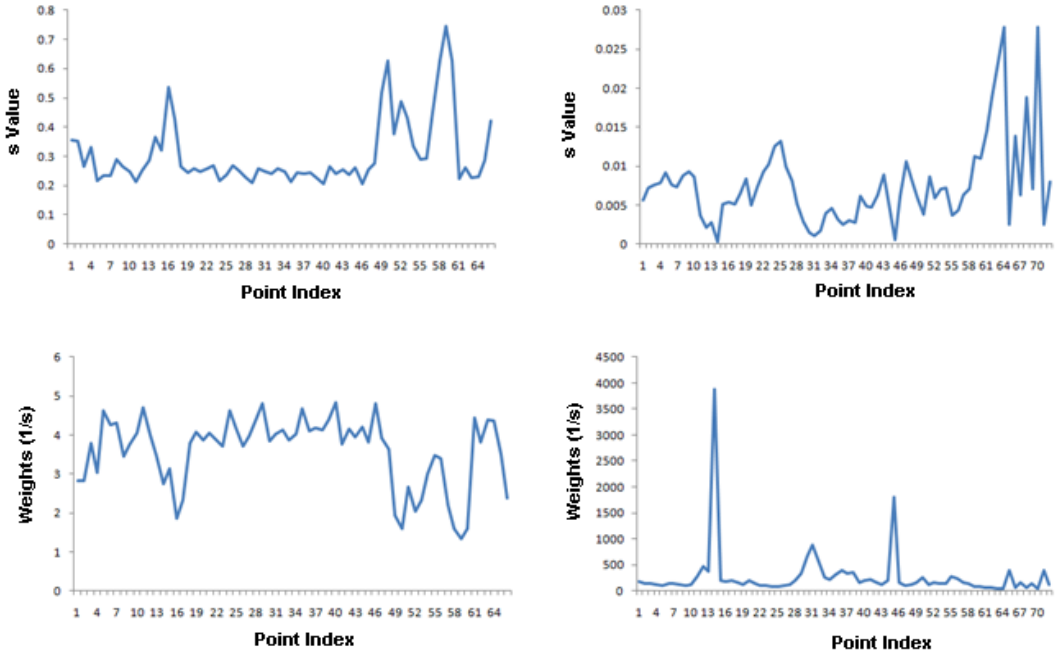

Figure 5.34. s_i plot of the octree nodes represented by a 2D slice (Top Left) and corresponding weights of each point using $w_i = \frac{1}{s_i}$ (Bottom Left). s_i plot of the octree nodes represented by a different 2D slice (Top Right) and corresponding weights of each point using $w_i = \frac{1}{s_i}$ (Bottom Right).

The second function, $w_i = s_{max} - s_i$, is better suited to non-ideal scenarios, where the variation in standard deviation values is unpredictable. By evaluating *s* relative to the maximum *s* value in the mesh, which will be denoted as s_{max}, the variation between ranges of *s* values can be better regulated and w_i will be more predictable. *s* values that are farther away from s_{max} will use large weights and *s* values closer to s_{max} will use small weights. An example of this is shown in Figure 5.35.

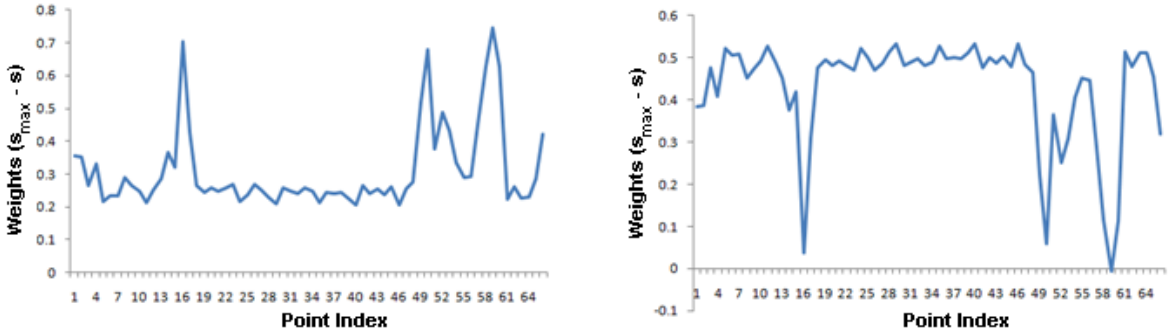

Figure 5.35. s_i plot of the octree nodes represented by a 2D slice (Left). Corresponding weights of each point using $w_i = s_{max} - s_i$, with $s_{max} = 0.74$ (Right).

The first problem with this technique is the assumption that there are deformations present in the mesh, and that values close to s_{max} are unreliable areas to be used for curve-fitting calculations. In all likelihood, on the assembly line, there will be automotive parts which contain no deformations of interest. s_{max} will still be chosen, but it will not be part of a deformation. As a result, most of the voxels containing the mesh may have many values near s_{max} and the line-of-best-fit calculation may reduce the influence of many points that it should be using, and increasing the possibility of providing incorrect estimates of the ideal surface. An example of this is shown in Figure 5.36.

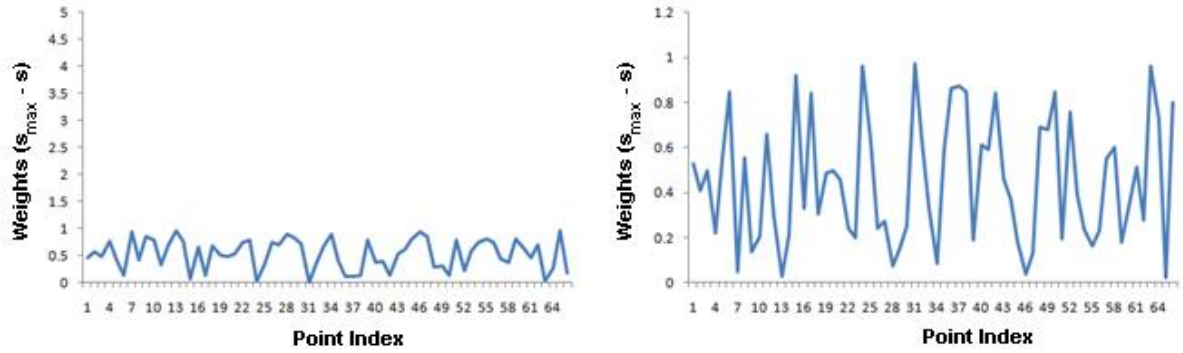

Figure 5.36. s_i plot of 2D slice with no deformations in the mesh (Left). Corresponding weights of each point using $w_i = s_{max} - s_i$, with $s_{max} = 1$ (Right).

The second problem with this technique is that not all unreliable areas will have *s* values near s_{max}, resulting in unreliable areas being weighted more than they should be. There may be a range of high *s* values that are unreliable and should not contribute to the line-of-best-fit calculation, as shown in an example in Figure 5.37.

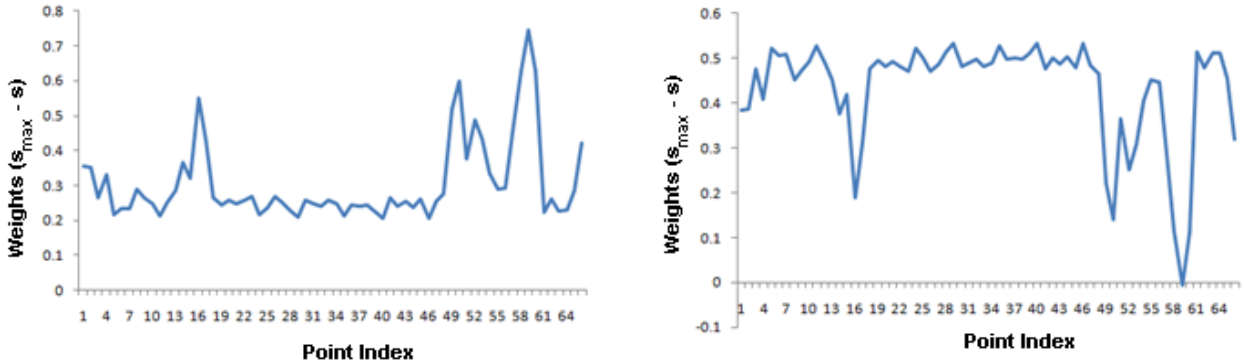

Figure 5.37. s_i plot of 2D slice with deformations of varying *s* values in the mesh (Left). Corresponding weights of each point using $w_i = s_{max} - s_i$, with $s_{max} = 0.74$ (Right).

To remedy the two problems discussed with the second function, the third function requires that a threshold, $s_{threshold}$, is set. In the same way that appropriate thresholds were set to get results in section 4.4.1, the same threshold can be used to improve the curve-fitting. Areas that would be extracted as deformations, because of their *s* value exceeding $s_{threshold}$, will be excluded from the ideal surface calculation, while all other areas are weighted equally. It is important to note, in the unlikely scenario that too many of the points that are being evaluated have 0 as a weight, the least-squares fitting will not have enough points available to make a good estimate of the line-of-best-fit. For this reason, it is easier to use a very small value instead of 0 as w_i. The updated function used for the experiments in this thesis is $w_i = \begin{cases} 0.1, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$, and an example is shown in Figure 5.38.

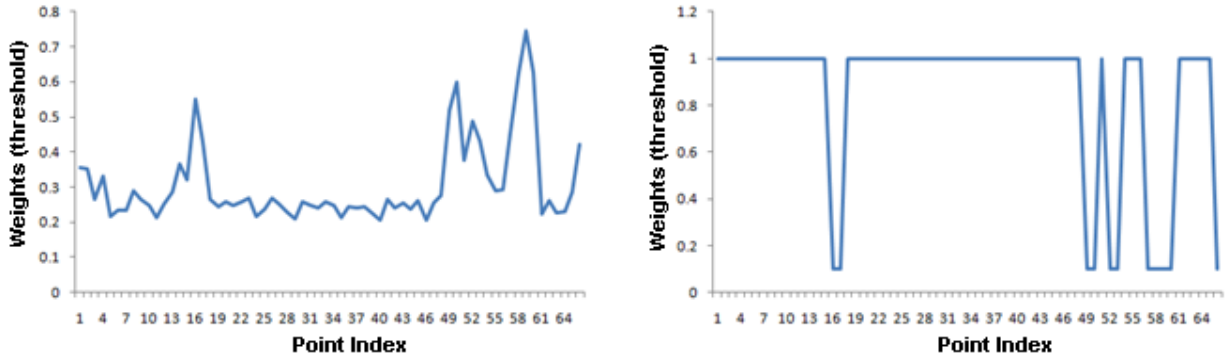

Figure 5.38. s_i plot of 2D slice with deformations of varying *s* values in the mesh (Left). Corresponding weights of each point using $w_i = \begin{cases} 0.1, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$, with $s_{threshold} = 0.4$ (Right).

The limitation is that an appropriate threshold must be selected, adding an extra parameter to be determined. Fortunately, this parameter can be set by using a test mesh and determining what s values often correspond to deformations or other outliers in the same way as it was done in section 4.4, using histograms or experimentation on a given set of panels to be inspected.

This improvement will undoubtedly increase the performance of the algorithm, and reduce effects such as the ones shown in Figure 5.32. It is important to note that it is risky to use the weighted least-squares technique for the curve-fitting in the recursion enhancement, discussed in section 5.5.1, because with less data, and more varied high *s*-values, the weighting can be unpredictable.

5.6 Results and Discussion

The contour analysis technique is applied to the meshes introduced in chapter 3. The basic technique, along with the two discussed enhancements, are tested to determine their effectiveness. The results are divided into three sections. The first section tests the basic contour analysis, without any enhancements. The second section tests the contour analysis with the recursion enhancement. The third section tests the contour analysis with the weighted least-squares enhancement and the ability to extract deformations with characteristics faithful to the actual deformations.

5.6.1 Basic Contour Analysis Results

To test the basic contour analysis methods's effectiveness in extracting deformations of interest, the algorithm is applied to several meshes.

The only parameters required to be set for the algorithm are the octree level used for slice acquisition, and the neighbourhood size used for curve-fitting. Ordinary least-squares is used, so no additional parameters need to be set for the curve-fitting. The resolution used to acquire the test results was at the 7^{th} level of the octree, and the neighbourhood size used for curve-fitting, r, is 10. This means that 10 points on either side of the current point, along with the current point itself, is used for curve-fitting, resulting in a total of 21 points used.

The algorithm is applied to the artificial flat mesh with a small deformation. The deformation can be isolated in various ways. It can be isolated by thresholding the magnitude to display deformations greater than 0.5 mesh units in size, or by thresholding the span to display deformations less than 15 mesh units in size. The results are shown in Figure 5.39.

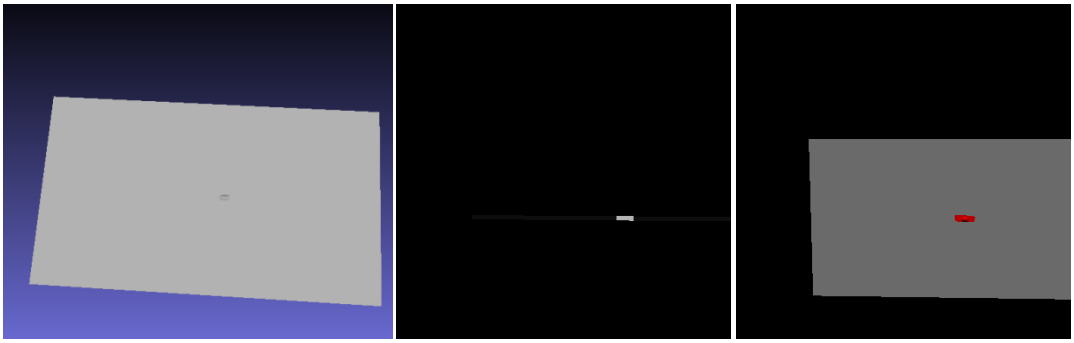

Figure 5.39. Artificial flat mesh with small dent (Left). Intensity corresponding to magnitude of deformation patterns (Center). Extracted deformation is highlighted (Right).

The deformation is detected as a dent, in a single slice, with a magnitude of 0.57 mesh units, and span of 10.37 mesh units. Its actual specifications are a magnitude of 0.91 mesh units and span of 10.49 mesh units. The deformation is clearly extracted, even though its characteristics are not similar to the actual deformation in the mesh. The magnitude of the deformation is slightly diminished because it influences the ideal surface estimation, thus reducing the separation between the ideal surface and the actual surface in the area containing the dent. For the same reason, there is a ding of a small magnitude detected on either side of the deformation in the intensity image, since the estimated ideal surface is inaccurate in the area containing the dent. These inaccurately detected deformations on either side of the dent can be called *residual deformation patterns*. Though the deformation can still be extracted, the characteristics of the extracted ding will be closer to the actual characteristics if the curve-fitting is improved by the enhancement described in section 5.5.2. Results for that enhancement will be shown in section 5.6.3. Regardless, the residual deformation patterns do not interfere very much with the extraction of the deformation of interest.

Also, the resolution of the octree is not the maximum resolution available from the original point cloud, meaning that the algorithm was operating at a lower resolution than the actual mesh. Because of resampling, and using fewer points to describe the mesh, the peaks of some of the deformations are not fully captured. Though, the extracted deformation characteristics are not exact, it is still easily separable from the rest of the mesh, with performance at least as accurate as the techniques described in chapter 4.

Next, the algorithm is applied to the artificial flat mesh with a large deformation. It is isolated by thresholding the magnitude to display deformations greater than 3 mesh units in size. The results are shown in Figure 5.40.

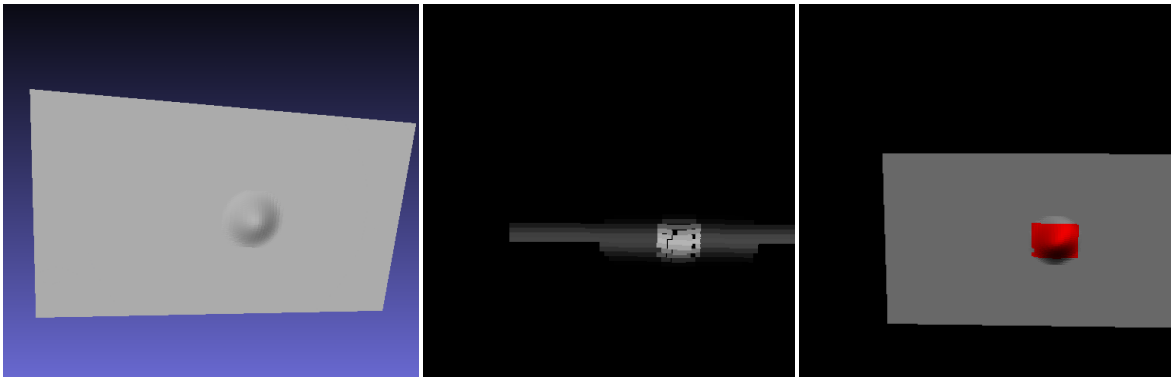

Figure 5.40. Artificial flat mesh with large ding (Left). Intensity corresponding to magnitude of all deformation patterns (Center). Extracted deformation is highlighted (Right).

The deformation is detected, in 6 slices, with the maximum magnitude deformation pattern being a magnitude of 6.45 mesh units, and span of 29.96 mesh units. The deformation's actual specifications are a magnitude of 9.5 mesh units and span of 43.78 mesh units. The deformation is detected clearly, and can be isolated using simple parameters corresponding to the expected size of the deformation. It can be seen that several slices are used to describe the deformation, and these will be combined to be one segment during the segmentation and classification phase, which will be discussed in chapter 6. It is expected, and is visible, that each slice will have a different magnitude because the deformation is a spherical bulge, meaning cross sections will have varying deformation magnitudes and spans. Fortunately, thresholding the magnitude and span parameters can still separate all slices of the deformation of interest from the other slices describing deformations that we are not interested in.

Again, there are some residual deformation patterns as a result of the curve-fitting being influenced by the deformation, but it does not interfere with the extraction of the deformation of interest, as the residual deformation patterns are areas of large span, low magnitude and thus low sharpness.

For the following experimentation, the algorithm is applied to the high-resolution unfiltered CPU panel mesh. The deformations are isolated by thresholding the magnitude to display deformations greater than 1.1 mesh units in size. The results are shown in Figure 5.41.

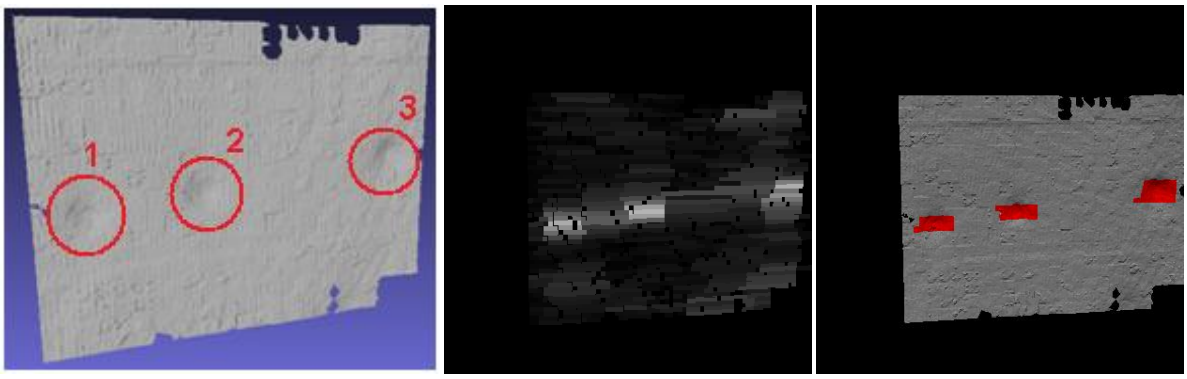

Figure 5.41. Unfiltered high-resolution CPU panel mesh with deformations labelled (Left). Intensity corresponding to magnitude of deformation patterns (Center). Extracted deformations are highlighted (Right).

The three deformations are extracted as dents. The intensity image shows various potential deformations, due to the significant amount of noise and variation in the surface. However, these noisy areas can easily be ignored. By looking for deformations that are greater than 1.1 mesh units, only the deformations of interest are extracted without any additional

areas. It can be seen how clearly the contour analysis technique can characterize noise, surface variation, and deformations with different characteristics such that simple and intuitive thresholds can separate one from the others. Realistically, these thresholds can be estimated from the knowledge of the dings and dents that a given manufacturer wants to identify. The approximate magnitude of deformations that require repair will be known, while deformations with magnitudes beneath that can be neglected as they will be masked by painting.

To demonstrate that this algorithm can extract deformations even when subtle changes in the surface are only slightly greater than the noise, the algorithm is applied to the low resolution unfiltered CPU panel mesh. Using the basic contour analysis technique, defects can be isolated by thresholding the magnitude to display deformations greater than 0.75 mesh units in size and the span to display deformations less than 30 mesh units in size. The results are shown in Figure 5.42.

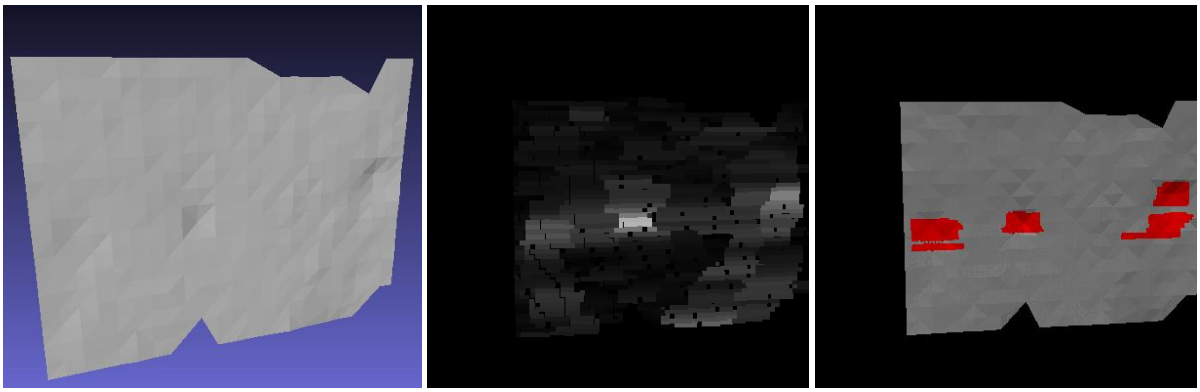

Figure 5.42. Unfiltered low-resolution CPU panel mesh (Left). Intensity corresponding to magnitude of deformation patterns (Center). Extracted deformations are highlighted (Right).

It is important to note that the low-resolution CPU panel has deformations that are very difficult to extract, mostly due to the amount of noise and the limited amount of variation in the deformations themselves. The algorithms discussed in chapter 4 could not extract any deformations from this model, but the contour analysis technique easily isolates the dents and prevents the extraction of broad curves.

To test the ability of the contour analysis technique to extract a large scale deformation, it is applied to the artificial curved mesh with a large deformation. The latter can be isolated by thresholding the magnitude to display deformations greater than 7 mesh units in size. The results are shown in Figure 5.43.

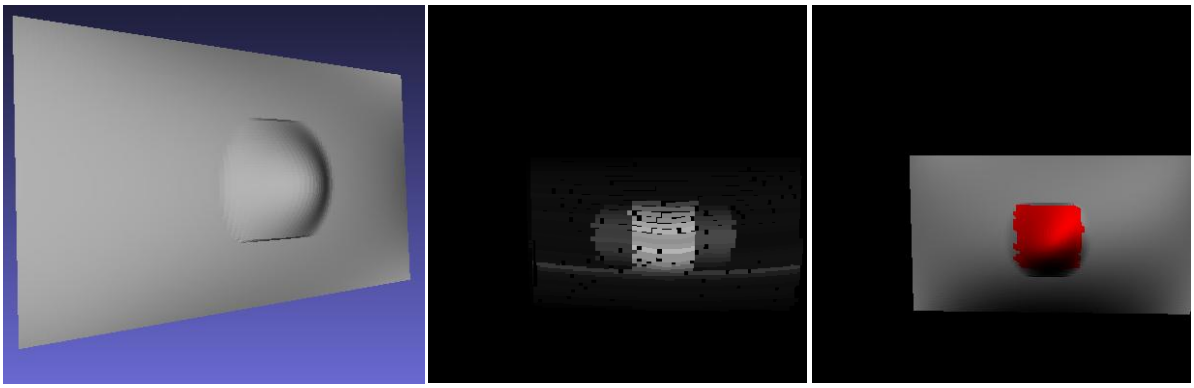

Figure 5.43. Artificial curved mesh with large ding (Left). Intensity corresponding to magnitude of deformation patterns (Center). Extracted deformation is highlighted (Right).

In the intensity image, it is visible that there are a lot of deformation patterns. The curvature of the mesh is detected as a set of broad deformation patterns, mostly because the neighbourhood size is large. The ding itself, which lies on top of the broad curvature is detected as a deformation with larger magnitude than the broad curvature, which is important for its extraction. The areas to the left and right of the deformation, indicate residual deformation patterns because of inaccurate curve-fitting and could potentially be extracted as dents. There is a lot of burden on the curve-fitting because the curvatures in this mesh are complex, but knowing the scale of the deformation is important to set the thresholds. Due to its spherical shape, it shares characteristics similar to the deformation extracted in the artificial flat mesh with a large curve, in that the slices will be of varying magnitude and span in the deformation. But looking for deformations that are larger than 7 mesh units in magnitude easily isolates the deformation of interest from the other broad curvatures and residual deformations that are found.

Finally, the algorithm is applied to the high resolution unfiltered car door mesh. The deformations can be isolated by thresholding the magnitude to display deformations greater than 5 mesh units in size, and span between 40 mesh units and 60 mesh units in size. The results are shown in Figure 5.44.

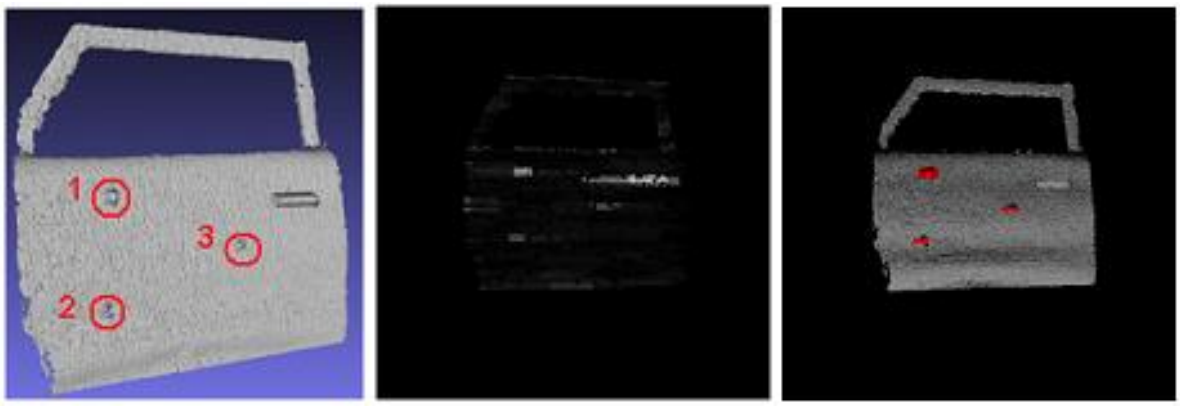

Figure 5.44. Unfiltered high-resolution car door mesh with deformations labelled (Left). Intensity corresponding to magnitude of deformation patterns (Center). Extracted deformations are highlighted (Right).

There is a lot of noise and many broad body curvatures in this high resolution car door model. They are visible in the intensity image, but the deformations of interest stand out in magnitude. By isolating deformations with magnitudes larger than 5 mesh units, most of the curvature and noise is removed, however the door handle and residual curvatures would remain. The residual curvatures around the door handle seem to have magnitudes of at least the same size as the deformations of interest. The span attribute was useful to remove residual curvatures and the door handle, which are very long in span, and to remove potential small noisy areas that might be extracted that are very short in span. The three dings were clearly extracted using the basic intuitive parameters of span and magnitude.

A limitation of this algorithm is that curvatures can only be thresholded on their characteristics along the x-axis of the model. For example, if the door handle were rotated 90 degrees, the door handle could not be removed using the contour analysis thresholding alone, because it would be broken up into smaller slices. This is why the supplementary segmentation and classification phase is important, discussed in chapter 6, to remove areas of the mesh that may have slipped by the contour analysis thresholding phase.

5.6.2 Recursion Enhancement Testing

The enhanced algorithm is applied to the high-resolution artificial recursion test mesh. Using the regular method, as shown in Figure 5.45, only the large deformation can be isolated because the small deformation is on top of it.

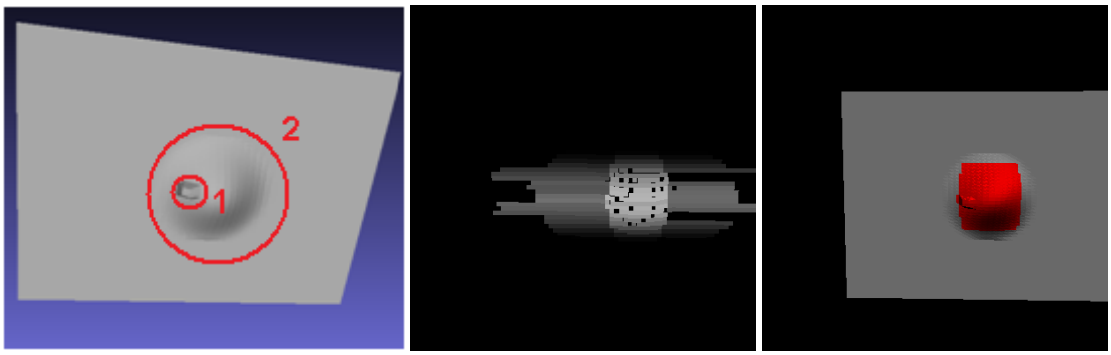

Figure 5.45. Artificial recursion mesh with small ding (1) on top of large ding (2) (Left). Intensity corresponding to magnitude of deformation patterns (Center). Extracted deformation is highlighted (Right).

Because of the nature of the small deformation being on top of the large deformation, it is impossible to isolate it from the rest of the mesh since it is labelled as belonging to the large deformation. For this reason, the recursion enhancement is necessary. After one level of recursion, the small deformation can be extracted because it lies on top of the large deformation.

The recursion will look for a new deformation only within the points defining an existing deformation, so it is already limited in the amount of data it can use. To provide the algorithm with more data in order to get more accurate results, the algorithm with recursion is run on the nodes extracted from the 8^{th} level of the octree instead of the 7^{th} level. Also, because there are twice as many points describing the same physical range, the neighbourhood size, r, is also doubled from 10 to 20 to remain proportional to the resolution change.

The large deformation can be isolated by thresholding the magnitude to display deformations greater than 3 mesh units in size. The results for extracting the large deformation are shown in Figure 5.46.

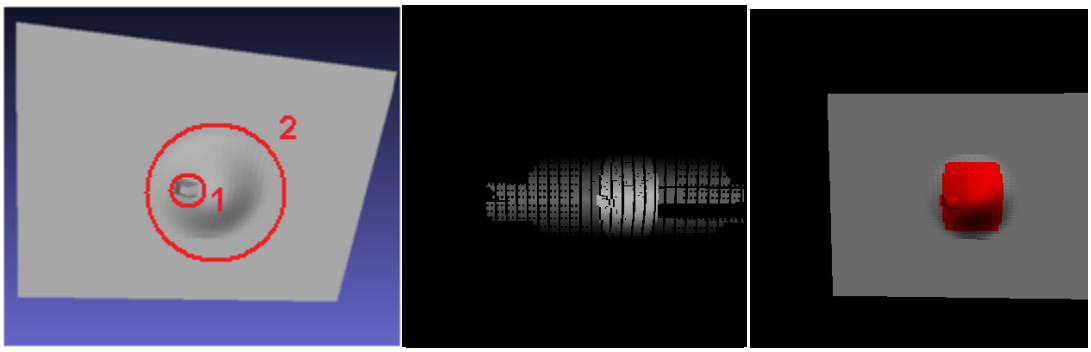

Figure 5.46. Artificial recursion mesh with small ding (1) on top of large ding (2) (Left). Intensity corresponding to magnitude of deformation patterns after the initial pass (Center). Extracted deformation is highlighted (Right).

The large deformation is detected, in 10 slices, with the maximum magnitude deformation pattern being a magnitude of 5.09 mesh units, and span of 23.02 mesh units. Its actual specifications are approximately a magnitude of 9.05 mesh units and span of 43.75 mesh units. Again, the spherical characteristics of the deformation show varying magnitudes and spans for each of the deformation patterns describing it, but the parameters can easily isolate it from the rest of the mesh. Though it seems trivial to isolate it from the rest of the mesh, the residual deformation patterns due to the deformation's effect on the curve fitting adds some complexity. But these residual deformations can be excluded in the same way that they are in the other meshes, primarily by thresholding the span to exclude long broad curves. The sharpness parameter could also be used. Regardless, here the magnitude characteristic was sufficient to have the large deformation isolated clearly.

The small deformation is extracted by thresholding the magnitude to display deformations greater than 1 mesh unit in size, and span to be less than 10 mesh units in size. The results for extracting the small deformation are shown in Figure 5.47. Note that the extraction of the small deformation does not need to be limited to thresholding deformations only found after the recursion, as it can be isolated from all detected deformations over the original pass and the recursion just by using the thresholding parameters specified.

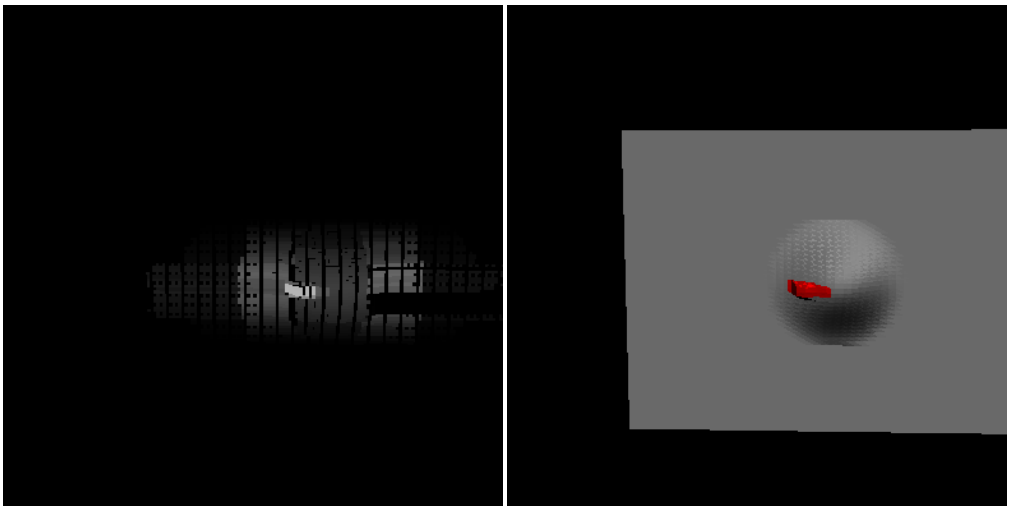

Figure 5.47. Intensity corresponding to magnitude of deformation patterns detected after one recursion (Left). Extracted deformation is highlighted (Right).

The deformation is detected, in 2 slices, with the maximum magnitude deformation pattern being a magnitude of 1.60 mesh units, and span of 8.29 mesh units. Its actual specifications are approximately a magnitude of 3.65 mesh units and span of 6.76 mesh units.

The intensity image after recursion shows lots of residual curvature extracted relative to the deformation patterns found in the initial pass, along with the small deformation of interest. Since the generation of the difference curve uses a line-of-best-fit, instead of a curve-of-best-fit, no matter how many recursions are done, any curvature will produce residual curvature after the recursion because fitting a straight line perfectly to the points over a curve will always produce error.

The results show that even though the deformation is small, it can still be isolated using the magnitude and span parameters, and the residual curvatures do not represent much of an obstacle. The large deformation is a significant curvature and the small deformation can still be detected relative to that curvature. Areas that are actually deformations will be more distinct than the residual curvatures, will be limited in span, and will have a significantly larger magnitude also. In the results it can be seen that, though residual curvatures have an increasing effect after each recursion, deformations of interest can still be extracted clearly. However, the magnitude of the extracted small deformation is very different from the actual magnitude. This can be attributed again to the lack of sufficient data points and the complexity in fitting an appropriate curve to the data points of the large deformation. It is important to note that it is still clearly distinguishable using the magnitude and span parameters, even though the exact attributes are not the same.

Though getting results with this model were trivial with the techniques in chapter 4, the small deformation over a large broad curvature proves a challenge for the initial contour analysis technique. The algorithm, with recursion, is applied to the artificial curved mesh with a small deformation. It is possible to do this without recursion, however the neighbourhood size would have to be smaller to avoid detecting the large curvature, and to find an appropriate neighbourhood size, it would require further experimentation. To show the algorithm's robustness, this deformation is detected without setting custom parameters for the model. Just as the results were acquired for the previous model, the algorithm with recursion is carried out at the 8^{th} level of the octree, with neighbourhood size at 20. The deformation is isolated by thresholding the magnitude attribute to display deformations greater than 1 mesh unit in size, and the span attribute to display deformations less than 10 mesh units. The results are shown in Figure 5.48.

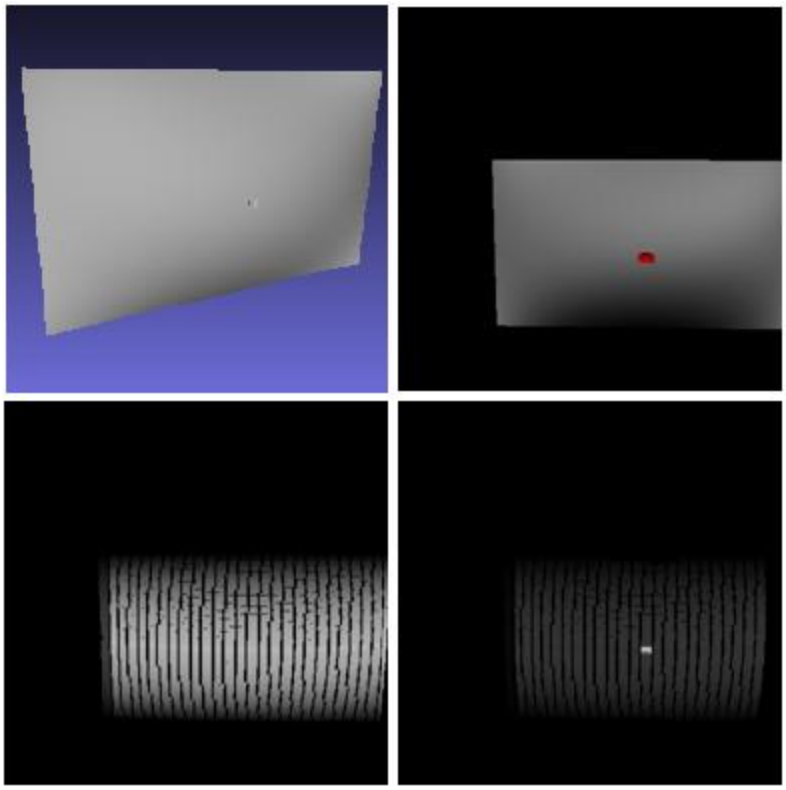

Figure 5.48. Artificial curved mesh with small dent (Top Left). Extracted deformation is highlighted (Top Right). Intensity corresponding to magnitude of deformation patterns (Bottom Left). Intensity corresponding to magnitude of deformation patterns after one recursion (Bottom Right).

The deformation is detected, in 1 slice, at magnitude of 1.32 mesh units, and span of 6.91 mesh units. Its actual specifications are approximately a magnitude of 1.50 mesh units and span of 6.91 mesh units.

The size of the deformation of interest is very small relative to the broad curvature of the mesh. The curvature of the mesh is so large that only broad curvatures are found in the initial pass. After the first recursion, the small deformation is found along with residual curvature of the broad curves found in the first pass. This is another example of why recursion is needed, in that large curvatures and a non-optimal neighbourhood size can result in small deformations of interest being overlooked. The recursion ensures that the deformation of interest is found, and can be isolated using the basic parameters of magnitude and span.

It is important that enough data is available for the algorithm to do this type of selective multiresolution deformation detection. If the resolution of the scanner is not high enough, or the selected octree resolution is not high enough to acquire distinct points from the smaller deformation, and separate it from the noise, the algorithm will not be able to find it.

5.6.3 Deformation Characteristics and Weighted Least-Squares Enhancement Testing

The contour analysis technique's ability to isolate the deformations of interest was demonstrated to be very good in sections 5.5.1 and 5.5.2. Though the isolation of the deformation is the primary interest, for classification purposes, the algorithm's ability to also extract deformation characteristics accurately must be tested.

There are two important sources of error that can be a factor in not extracting deformations or extracting them without their actual characteristics. The first is the octree resolution, which dampens the finer details of the mesh by resampling it at a lower resolution. The second is the curve-fitting, which does not always estimate the ideal surface properly due to interference of noise, body curvature, and the deformations themselves.

Both of these sources of error are put to the test to examine how much influence they have on extracting the characteristics of the deformation. The visual results of these tests are very similar to the results in section 5.5.1 and 5.5.2, and the deformations can be extracted using the same or similar parameters. However, the improvement comes in the faithfulness of the extracted deformation characteristics to the actual deformation characteristics.

The magnitude of the extracted deformations using the basic technique is evaluated by using octree level 7 and comparing the results to using octree level 8 for the slice acquisition. Also, though the ordinary least-squares (OLS) technique in the basic contour analysis algorithm provided good results, the fact that the deformations are not of the correct size and that there are so many residual curvatures is explained from the shortcomings of the curve-fitting component using the ordinary least-squares technique. The weighted least-squares enhancement, discussed in section 5.5.2, provides some improvement in those areas.

The maximum magnitude deformation patterns for each of the deformations in each of the models are compared. They are compared against the actual measured magnitude of the deformation. The deformation patterns are extracted using the ordinary least-squares technique, and with the three proposed weighting functions for the weighted least-squares technique. Also, they are computed at the octree level 7 and 8, except for the meshes that were processed with recursion where only octree level 8 was used due to a higher resolution being required.

The results on the artificial flat mesh with small deformation are presented in Table 5.1 and Table 5.2.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = 0.001$
Deformation	0.91	0.57	0.61	0.60	0.61

Table 5.1. Maximum magnitude deformation pattern in artificial flat mesh with small deformation at octree level 7. All measurements are in mesh units.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = 0.0$
Deformation	0.91	0.86	0.91	0.89	0.91

Table 5.2. Maximum magnitude deformation pattern in artificial flat mesh with small deformation at octree level 8. All measurements are in mesh units.

The artificial mesh contains no noise, which aids in generating an accurate ideal surface through curve-fitting. In octree level 7, the results do not come very close to the actual magnitude. This is due to the slice acquisition sampling at octree level 7 not acquiring the peak of the dent. If the slice does not contain the peak of the dent, it is impossible for the remainder of the algorithm to extract a deformation of accurate magnitude. The weighted least-squares enhancement can only do so much to get the deformation magnitude closer to the magnitude, as shown by the results by the reciprocal function and the thresholding function. But that is the maximum possible improvement given the poorly sampled slice acquisition.

Slice acquisition sampling at octree level 8 clearly acquires the peak of the dent, and it is reflected in the findings. Even the ordinary least-squares technique provides very good results, close to that of the actual deformation. The difference function produces a slight improvement, while the reciprocal function and the thresholding function produce perfect results in this case. Since there is no noise, all the areas that do not contain the deformation are assigned *s* values of 0. Because of this, the reciprocal function and the threshold function can give equal weights to all of the flat areas, and give 0 weights to the deformation area, allowing a curve that is only fitted to the flat areas. Because of this, the deformation does not influence the curve-fitting at all, allowing a perfect extraction of the deformation characteristics.

The results on the artificial flat mesh with large deformation are presented in Table 5.3 and Table 5.4.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = 0.001$
Deformation	9.50	6.45	9	7.18	9

Table 5.3. Maximum magnitude deformation pattern in artificial flat mesh with large deformation at octree level 7. All measurements are in mesh units.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = 0$
Deformation	9.50	7.05	9	7.5	9

Table 5.4. Maximum magnitude deformation pattern in artificial flat mesh with large deformation at octree level 8. All measurements are in mesh units.

There is no difference in the sampling between octree level 7 and 8, as they happen to acquire a similar amount of information regarding the peak of the deformation. This is shown by the similarity in results between the reciprocal function and the thresholding function in both level 7 and 8. However, those results are significant improvements over the ordinary least-squares technique and the difference function weighting.

The reason that the actual magnitude is not achieved by the reciprocal function and the thresholding function is that the neighbourhood size is smaller than the width of the ding. This means that there are times when the curve-fitting uses only information from the ding to generate an ideal surface. In a perfect situation, none of the deformation should be used for the ideal surface generation, such that the comparison is free from the effects of the deformation in the curve-fitting step. That is not always possible as large deformations can occur, reducing the magnitude of the deformation in the difference curve and ultimately a diminished magnitude in the final deformation pattern. The algorithm still provides acceptable results, which can be improved if the neighbourhood size is increased, but the ultimate setting is dependent on the deformation characteristics which cannot be entirely predicted in practice.

The results on the artificial curved mesh with large deformation are presented in Table 5.5 and Table 5.6.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold}$ =0.00016
Deformation	18-22	10	14.3	10.1	14.3

Table 5.5. Maximum magnitude deformation pattern in artificial curved mesh with large deformation at octree level 7. All measurements are in mesh units.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold}$ =0.00001
Deformation	18-22	12.5	20.5	12.7	21.1

Table 5.6. Maximum magnitude deformation pattern in artificial curved mesh with large deformation at octree level 8. All measurements are in mesh units.

The artificial curved mesh with large deformation is a challenging test for the contour analysis technique in that there are different curvatures, which makes the curve-fitting inaccurate in determining an ideal surface. The actual magnitude of the deformation is anywhere between 18 and 22, as it is a curvature on top of another curvature. So any value in that range is an acceptable value for the deformation.

Because the neighbourhood size is not optimized to remove the large curvature, the isolation of the ding may provide characteristics that are not close to its actual characteristics. The 8th level of the octree provides better estimates than the 7th level of the octree judging by the improved results. Again, the reciprocal function and thresholding function produce results closer to the actual deformation magnitude, compared to the ordinary least-squares method and the difference function, in their respective octree levels.

The results on the artificial recursion mesh are presented in Table 5.7.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold}$ =0.001
Large Deformation	9.05	5.09	9.05	5.12	8.98
Small Deformation	3.65	1.60	2.22	1.59	2.73

Table 5.7. Maximum magnitude deformation patterns in the recursion test mesh at octree level 8. All measurements are in mesh units.

The recursion mesh was only tested at octree level 8, because lower resolutions would not provide sufficient data to extract the smaller deformation. The reciprocal weighting function

is able to get perfect results for the large deformation, while the thresholding weighting function gets a very close value. Again, due to the lack of noise, the reciprocal function is capable of perfectly extracting the large deformation because the flat areas contain *s* values of 0, assuring that the flat areas have equal weight, while the areas that contain *s* values greater than 0 will be weighted as 0. This ensures that only the flat areas are used for the curve-fitting, completely uninfluenced by the large deformation.

The reciprocal and thresholding functions also provide better results for the small deformation, but this is more likely to properly extracting the large deformation before passing it on to the recursion. It is difficult to gauge the effectiveness of extracting the small deformation because it is inherently reliant on the quality of extraction of the large deformation. Regardless, the reciprocal and thresholding functions provide the best results.

The results on the artificial curved mesh with small deformation are presented in Table 5.8.

	Actual	**OLS**	$\boldsymbol{w_i = \frac{1}{s_i}}$	$\boldsymbol{w_i = s_{max} - s_i}$	$\boldsymbol{w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}}$ $\boldsymbol{s_{threshold} = 0.017}$
Deformation	1.50	1.32	0.63	0.87	0.88

Table 5.8. Maximum magnitude deformation pattern in artificial curved mesh with small deformation at octree level 8. All measurements are in mesh units.

Just as for the recursion mesh, the artificial curved mesh with small deformation was only tested at octree level 8, because lower resolutions would not provide sufficient data to extract the deformation.

The ordinary least-squares technique provided the best results, but this is not a very good indicator of the success of that technique. Because the dent is only detected after one recursion, the extraction of the curvature that the dent resides upon has an important influence on the extracted deformation characteristics. The neighbourhood size is significantly smaller than that of the broad curvature, so it is difficult to extract those curves with exact characteristics. Also, the ideal surface that is generated through the recursion must be accurate to extract the deformation relative to it. Since the broad curve is a large curvature, and a line-of-best-fit cannot perfectly fit it, there will always be some residual curvature. The residual curvature can influence the fitting such that the deformation of interest is not extracted with characteristics similar to the actual deformation. The characteristics of the extracted small deformation in the recursion mesh are inaccurate for the same reason.

The results on the unfiltered high-resolution car door mesh are presented in Table 5.9 and Table 5.10.

	Actual	**OLS**	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = \mathbf{0.155}$
Deformation 1	18-20	14.02	15	14.5	15.4
Deformation 2	10-13	5.72	5.3	5.3	5.3
Deformation 3	10-13	6.53	7.03	6.6	6.7

Table 5.9. Maximum magnitude deformation patterns in unfiltered high resolution car door mesh at octree level 7. All measurements are in mesh units.

	Actual	**OLS**	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = \mathbf{0.119}$
Deformation 1	18-20	10.50	10.8	10.7	10.8
Deformation 2	10-13	8.95	12.6	9.0	12.1
Deformation 3	10-13	9.15	10.4	9.6	10.1

Table 5.10. Maximum magnitude deformation patterns in unfiltered high-resolution car door mesh at octree level 8. All measurements are in mesh units.

It can be seen in the results that in this real world model, which contains a significant amount of noise and deformations that are only slightly larger in amplitude than the noise, there is some difficulty in getting deformation magnitudes close to the actual magnitude. Compared to the 7th level of the octree, the 8th level of the octree provides results that are much closer to the actual values on deformation 2 and deformation 3 overall. Deformation 1 is an anomaly, where the spacing of the slice acquisition sampling caught the tip of the ding at resolution 7 but not at resolution 8. If the tip of the ding is not even in the points making up the slice, it is impossible for the rest of the algorithm to get an accurate magnitude for the deformation.

Weighting using the reciprocal function and the thresholding function shows a significant improvement in the results over the ordinary least-squares method, as the deformation magnitudes are closer to the actual magnitudes.

The results on the unfiltered high-resolution CPU panel mesh are presented in Table 5.11 and Table 5.12.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = 0.011$
Deformation 1	2	2.2	1.7	2.1	2.1
Deformation 2	2	2.2	2.37	2.1	2.3
Deformation 3	2	1.8	2.1	2.1	2.1

Table 5.11. Maximum magnitude deformation patterns in unfiltered high resolution CPU panel at octree level 7. All measurements are in mesh units.

	Actual	OLS	$w_i = \frac{1}{s_i}$	$w_i = s_{max} - s_i$	$w_i = \begin{cases} 0, & s_i > s_{threshold} \\ 1, & s_i \leq s_{threshold} \end{cases}$ $s_{threshold} = 0.016$
Deformation 1	2	1.95	4.3	2.1	3.2
Deformation 2	2	2.15	2.4	2.2	2.3
Deformation 3	2	1.95	2.2	1.9	2.2

Table 5.12. Maximum magnitude deformation patterns in unfiltered high-resolution CPU panel at octree level 8. All measurements are in mesh units.

It is very difficult to get an accurate approximation of the actual deformation magnitudes on the CPU panel because of the noise and the many curvatures over the surface, so further accuracy could not be achieved beyond the size 2 magnitudes measured. All of the results are quite close to the actual magnitudes, so it is difficult to show that one technique is better than the other. One noticeable error is in the Deformation 1, at octree level 8, using the reciprocal weighting function. As it was explained in section 5.5.2, the weighted least-squares enhancement can provide improved results, but in real world scenarios, certain types of weighting functions can be unpredictable. This is a case of that unpredictability where the weighting actually produces worse results than the ordinary least-squares technique. It is likely that noise and the uneven characteristics of the mesh generated some outlier *s* values in that area. Those outliers cause the weighting of the points to generate a lopsided curve to fit the points, ultimately resulting in a noticeably poor deformation magnitude.

Overall, the weighted least-squares enhancement provides better results than the ordinary least-squares curve-fitting. The reciprocal function is the best selection for a weighting function, as it provides good results on the real world meshes and excellent results on the artificial meshes. The results show that, if the acquisition system is more accurate and can minimize noise, the reciprocal weighting function for weighted least-squares enhancement will provide much better results for curve-fitting than the ordinary least-squares curve-fitting.

5.7 Conclusion

It can be seen that real world meshes offer more obstacles in extracting accurate deformation characteristics. However, the characteristics are still sufficient to isolate the deformation from its surroundings, and provide enough information to distinguish one deformation from another. Also, the characteristics of the deformation patterns, though imperfect, can be very useful for the extraction and classification of the deformations, and the weighted least-squares curve-fitting used to enhance the algorithm can be very helpful.

The use of deeper levels in the octree produce higher quality slices, and provides less hindrance to the rest of the contour analysis in extracting the deformations accurately. However, the use of shallower levels can also provide sufficient data for extraction, with less computational complexity, as shown in section 5.6.1.

The use of the weighted least-squares enhancement, specifically the reciprocal weighting function, consistently provides better results. There are some examples where the enhancement provides slightly worse results than the ordinary least-squares fitting, but those instances are often not poor enough to offset the benefits gained. The recursion enhancement provides selective multi-resolution analysis to find deformations of different scales, even if the neighbourhood, *r*, is incorrectly selected.

In comparison to the octree-based feature extraction, the contour analysis provides much more accurate results. The contour analysis algorithm precisely extracts all deformations and extracts none of the other curvatures or artifacts, while the octree-based technique cannot claim the same result. Realistically, only two parameters need to be set, and both are related to the size of the deformation to be extracted. The neighbourhood size, *r*, must be set to a value that is larger than the width of the deformation and preferably smaller than the width of other curvatures that may appear on the body. The magnitude threshold must be set to a value smaller than the magnitude of the deformation to be extracted and preferably larger than curvatures that are not of interest, including noise. These two parameters can easily be determined based on the known characteristics of dings and dents as identified by the manufacturer. The octree-based technique requires parameters that are less intuitive, such as the *s* threshold and the *a* and *i* parameters for $\boldsymbol{\sigma}$ thresholding. These parameters cannot be directly determined from the manufacturer specifications of dings and dents, and require trial and error testing to be set. This leads to contour analysis, along with its weighted least-squares and recursion enhancements, as the recommended feature extraction technique. However, the performance of the feature extraction in the context of the entire system must still be evaluated to determine its effectiveness. That discussion will be presented in chapter 6.

Chapter 6 Segmentation and Classification of Detected Deformations

After features have been extracted by the techniques presented in chapter 4 or chapter 5, an algorithm must be run to combine those isolated feature areas into a consistent segment. When using the octree-based surface shape analysis technique described in chapter 4, each node has information representing if it belongs to a feature or not, and to what degree. Nodes that contain pieces of the deformation must be grouped together to segment the deformation from the rest of the panel surface scan, as shown in Figure 6.1.

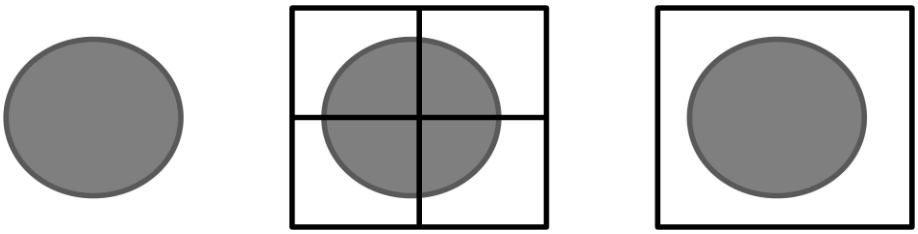

Figure 6.1. Original deformation (Left). Octree-based surface shape analysis results (Center). Octree-based segmentation (Right).

When using the contour analysis technique described in chapter 5, instead of voxels, several deformation patterns represent cross-sections of a deformation. These deformation patterns contain information regarding the shape and location of pieces of the deformation. These deformation patterns, representing pieces of the deformation, must be grouped together to segment the deformation from the rest of the model also, as shown in Figure 6.2.

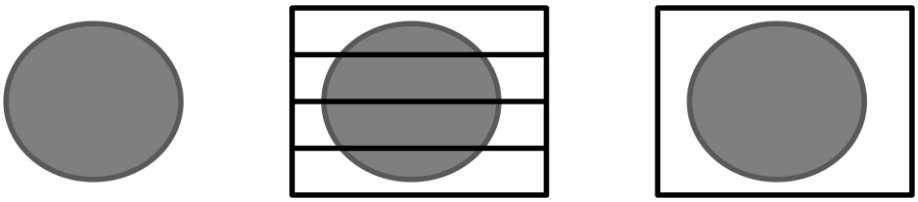

Figure 6.2. Original deformation (Left). Contour analysis surface shape analysis results (Center). Contour analysis segmentation (Right).

Finally, before the entire system outputs the segments that contain deformations, they must be classified as dings or dents as that was one of the primary objectives of the system. Contour analysis, and the corresponding segmentation, already classifies the deformations as dings or dents. However, the octree-based feature extraction requires the classification component to handle this task by receiving the segments containing the deformations of interest

as an input and labelling them as dings or dents for the output. The classification component also provides additional tools to remove segments that might contain non-deformations, and can be used for both the octree-based feature extraction segments as well as the contour analysis segments.

Since there are two proposed techniques to provide results from the surface shape analysis component, octree-based extraction and contour analysis, there is a separate segmentation method for each of those techniques. Section 6.1 and 6.2 discuss single-resolution and multi-resolution segmentation, respectively, dealing with the output of the octree surface shape analysis method. Section 6.3 discusses segmentation dealing with the output of surface shape analysis based on the contour analysis method. Section 6.4 discusses the classification of segments generated by the methods in sections 6.1, 6.2, and 6.3. Section 6.5 will compare the two deformation detection pipelines of the octree-based deformation detection system and the contour analysis deformation detection system. Finally, section 6.6 summarizes the different techniques and the findings of the system comparison.

6.1 Single-Resolution Segmentation based on Octrees

If the scale of the desired deformations is known, an appropriate resolution of the octree can be selected to extract those deformations. Since different depths of the octree represent different resolutions, selecting all nodes at a certain depth will provide a voxel representation of the object at that scale. Each of those voxels will contain important information about the piece of the mesh contained in that voxel, and this information can be used to aid in the segmentation.

The desired resolution may be different than the scale used for the octree surface shape analysis. The latter is the resolution to which voxels belong to deformations, however the resolution for segmentation must be selected based on connectivity. The resolution to analyze the surface shape must be high such that pieces of the deformation are detected. Due to the real-world nature of the data, there may be discontinuities in the grid of voxels where not all voxels belonging to the deformation are detected. Because of this possibility, sometimes the resolution to segment the deformation must be lower than the surface shape analysis resolution such that small discontinuities in the deformation can be overlooked because of the connectivity of the lower resolution version of that deformation. An example of discontinuities causing poor segmentation at high resolution is shown in Figure 6.3. An example of low resolution segmentation avoiding discontinuities is shown in Figure 6.4.

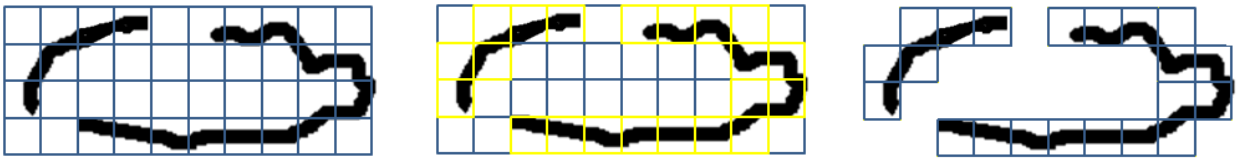

Figure 6.3. High resolution representation of a feature (Left). High resolution voxel connectivity shows discontinuities (Center). Segmented areas based on high resolution representation divide the feature into multiple segments due to discontinuities (Right).

Figure 6.4. Low resolution representation of a feature (Left). Low resolution voxel connectivity shows no discontinuities (Center). Segmentation at low resolution gets entire feature in spite of discontinuities (Right).

However, the segmentation resolution must be sufficiently high to avoid deformations being grouped with non-deformations, and to reduce the size of small segments defining things such as noise to avoid confusion with the actual deformations during the classification phase. An example of two features being incorrectly grouped together at low resolution is shown in Figure 6.5. An example of high resolution segmentation avoiding incorrect grouping is shown in Figure 6.6.

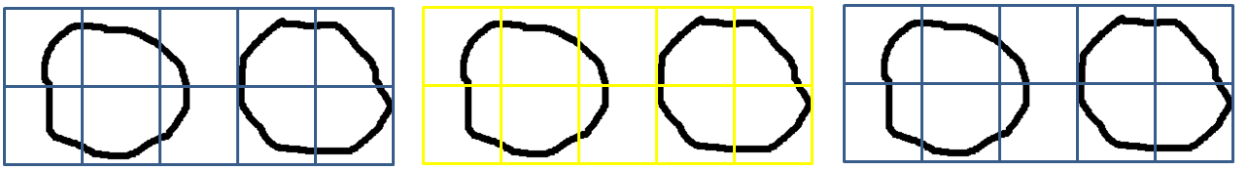

Figure 6.5. Low resolution representation of two features (Left). Low resolution voxel connectivity incorrectly shows connectivity between two features (Center). Segmented area based on low resolution representation combines two deformations together incorrectly (Right).

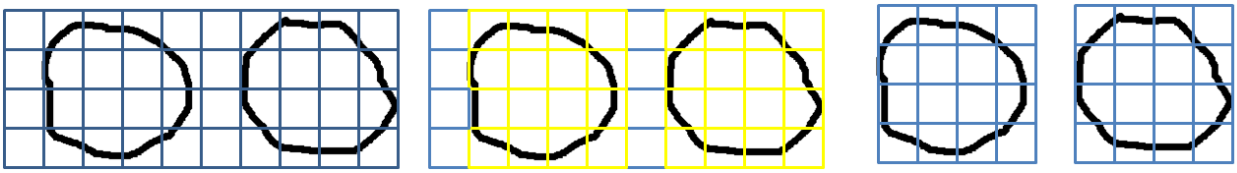

Figure 6.6. High resolution representation of two features (Left). High resolution voxel connectivity correctly shows disconnect between two features (Center). Segmentation at high resolution correctly separates features (Right).

After all voxels at the desired resolution are extracted, two operations are performed on those voxels to partition the mesh into segments and separate deformation-like areas from the rest of the mesh: Thresholding and Grouping, discussed in section 6.1.1 and 6.1.2, respectively.

6.1.1 Thresholding

The set of voxels must first be thresholded to remove voxels that do not likely belong to a deformation. This thresholding is only for the enhanced octree-based method, which refers to the technique and enhancements described in section 4.1, 4.2, and 4.3. It is required because the method provides some additional information that can be used to remove areas that are not removed in the octree-generation itself. For the output from the aggregate standard deviation octree method, discussed in section 4.4, no thresholding needs to be done, as the maximum amount of area that could be removed confidently was removed during the thresholding phase of that method.

There are two important characteristics of the voxels in the enhanced octree-based method: occupancy and standard deviation of surface normal orientation. All voxels of low standard deviation have already been removed during the tree generation, provided that the selected resolution for segmentation is not the deepest level of the tree. At the deepest level of the tree, nodes of low standard deviation, which includes empty nodes, are not removed.

Another important characteristic is the occupancy characteristic. After the tree has been generated and non-feature nodes are removed, only some nodes, which contain the highest resolution features, will remain at the deepest level of the tree. The pieces of the mesh that are not found in the deepest level of the octree are also removed in all nodes at the shallower levels of the tree, altering the tree to represent a multi-resolution characterization of the features found in the deepest level of the tree. Voxels that do not contain any of the features found at the deepest level of the octree are removed. Sometimes some areas containing noise or transient features can also remain at the deepest level of the tree, therefore it would be beneficial to not only remove voxels that contain none of the highest resolution features but also to remove voxels that contain small isolated features. An example is shown in Figure 6.7.

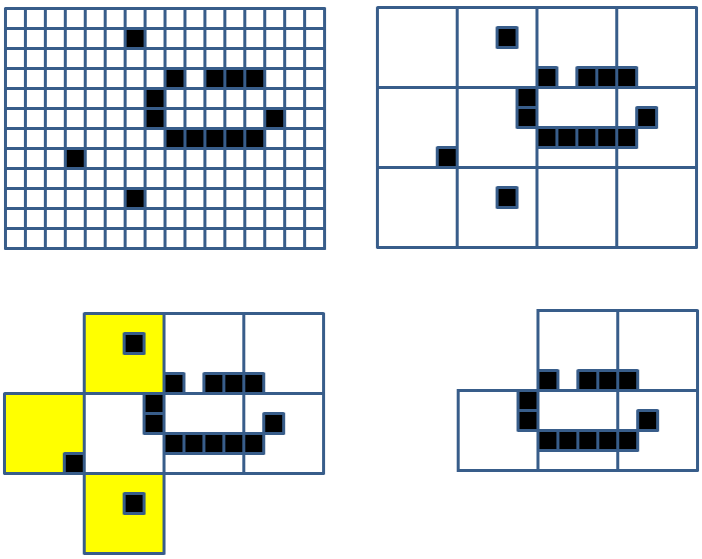

Figure 6.7. High resolution representation of features results in discontinuities that can make segmentation unsuccessful (Top Left). Lower resolution can improve connectivity and help segmentation despite discontinuities (Top Right). Sometimes better connectivity puts unwanted features in the segment (Bottom Left). If small isolated features are removed, a correct segment can be determined (Bottom Right).

The proposed occupancy characteristic, which represents the area of the surface mesh contained in the voxel relative to its volume, is useful to eliminate such isolated features. It is calculated by Eq. 6.1.

$$\rho = \sum_{i=0}^{n} \frac{a_i}{v} \tag{6.1}$$

where ρ is the occupancy, v is the volume of the voxel, n is the number of triangles contained in the voxel, and a_i is the area of the ith triangle in the voxel. The ρ value is re-calculated for each voxel at the resolution the segmentation is being conducted at.

Small isolated features will be noticeable at lower resolution because the voxels that contain them will have very low ρ values. As a result, a threshold is used to remove voxels with low ρ values to maintain voxels that contain deformation features which will be of higher ρ values.

6.1.2 Grouping

When thresholding is complete, only important voxels remain. These remaining voxels are the only voxels that will be considered for the remainder of this method, and will be denoted as *feature voxels*. Sets of feature voxels are to be grouped together to define a deformation, since each voxel will contain a piece of a deformation. The most important factor in grouping

these voxels is adjacency. Since the voxels are just cells of a 3-dimensional grid, adjacent voxels can be determined based on their coordinates in the 3-dimensional grid and looking for voxels at adjacent coordinates to the current one. If those voxels are feature voxels they are determined as adjacent voxels.

By extending the idea of blob extraction, which is a well-known 2-dimensional image processing algorithm covered in chapter 2, to three dimensions, adjacent voxels can be grouped together. An empty list is created to represent the segment. A feature voxel is selected as the seed voxel and added to the segment. Any feature voxel can be selected for this. All adjacent feature voxels to that feature voxel are added to the segment. For each of those voxels, their adjacent feature voxels are added to the segment, provided that those adjacent voxels are not already part of the segment. This process is repeated until no more voxels can be added to the segment. When this happens, the segment is complete, and all the voxels in that list define a deformation-like segment.

When one segment is completed, a new segment must be determined, so a new empty list is created and a new feature voxel is selected as the seed. The new seed voxel can be any feature voxel provided that it does not already belong to a segment.

This whole process repeats until no new seed voxel can be found. The result of the algorithm is a set of segments, each of which consist of one or more voxels. The pieces of the mesh contained in each segment represent a deformation.

6.1.3 Results

To test the single-resolution segmentation, the algorithm is applied on several meshes which had feature extraction performed on them using octree-based methods. In the images, the bounding box represents a segment generated by the segmentation algorithm, and the highlighted areas represent the results of the feature extraction. The occupancy threshold is set to 0.5% of the maximum occupancy in the mesh. This should remove small unnecessary areas, without affecting the connectivity of the deformations of importance.

The flat mesh with a small deformation, uses the octree-based method to extract the small deformation until octree level 8. The segmentation results are shown in Figure 6.8.

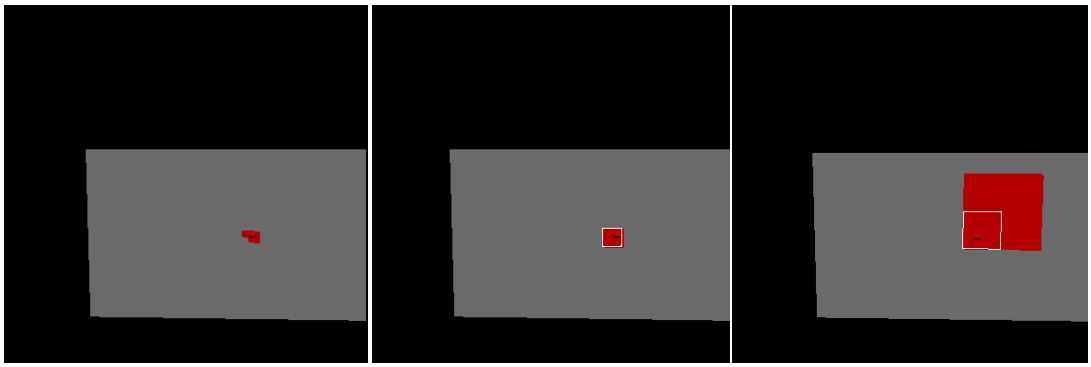

Figure 6.8. Octree-based feature extraction, at octree level 8, of flat mesh with small deformation (Left). Bounding box of segmented deformation at octree level 6 (Center). Bounding box of segmented deformation at octree level 4 (Right).

These results show that the segmentation can group the required voxels to define the deformation, provided that the deformation is successfully extracted by the feature extraction algorithm. Applying segmentation at level 6, segments the deformation clearly, and corresponds directly to the extracted features at level 6. The segmentation at octree level 4 shows that the deformation is still segmented clearly, but with larger dimensions. This is because the octree resolution is lower so the voxel containing the deformation is larger, and since segments are composed of a set of voxels, the segment at octree level 4 is the smallest possible segment that could contain the deformation, being one voxel in size. Also, the segmentation is smaller than the corresponding feature extraction results, because some areas are removed due to the occupancy thresholding discussed in section 6.1.1. Some of the areas highlighted by the feature extraction at level 4 are areas with 0 occupancy, and are removed since they contain no features that are still present at the maximum depth of the tree, which is octree level 8.

The curved mesh with a large deformation, uses the octree-based method to extract the large deformation until octree level 8. The segmentation results are shown in Figure 6.9.

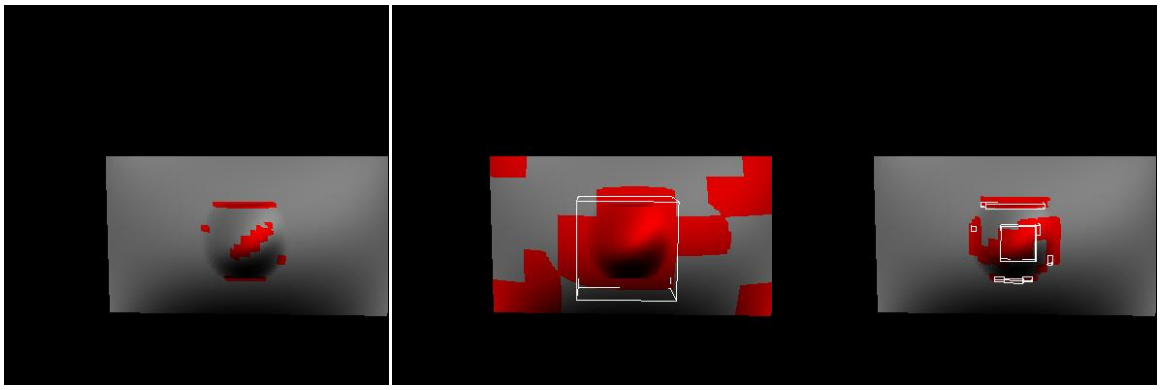

Figure 6.9. Extraction at octree level 8, of curved mesh with large deformation (Left). Bounding box of segmented deformation at octree level 5 (Center). Bounding boxes of segments at octree level 7 (Right).

These results show that the entire segment can be extracted, even if the feature extraction was imperfect in extracting all of the deformation. At octree level 8, only some of the deformation is extracted, leaving holes in places and causing a disconnect between some pieces of the deformation. This is an example of where using a lower resolution for segmentation can ignore the connectivity problems in the extracted deformation, as segmentation at octree level 5 allows the entire deformation to be extracted. However, running segmentation at the octree resolution of 7 shows the discontinuities, and many areas of the deformation are extracted separately. Also, areas with low occupancy are removed, resulting in sections of the mesh not being extracted at all. This shows the importance of selecting a good resolution for segmentation for connectivity purposes, provided that the feature extraction did extract the deformation clearly enough.

The unfiltered high resolution car door mesh uses the octree-based method to extract deformations until octree level 8. The segmentation results are shown in Figure 6.10.

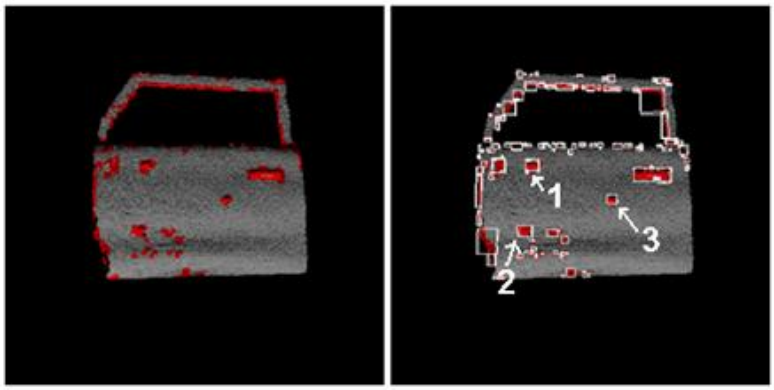

Figure 6.10. Extraction at octree level 8 on unfiltered high resolution door mesh (Left). Bounding boxes of segments at octree level 8 with deformation segments numbered (Center).

The segmentation is applied at octree level 8. In this situation, since there are no discontinuities in the extraction of the deformations, it would be best to apply the segmentation at the same resolution as the maximum depth of the feature extraction. A lower resolution is only required to bridge connectivity problems, and since there are none, a high resolution can be used to reduce the size of non-deformation segments, such as noise and acquisition artifacts along the borders of the mesh. The segmentation itself cannot remove these areas, but by reducing the size, they will be easier to remove during the classification phase. It is visible that the deformations and door handle are segmented clearly.

The filtered high resolution CPU panel mesh uses the octree-based method to extract deformations until octree level 8. The segmentation results are shown in Figure 6.11.

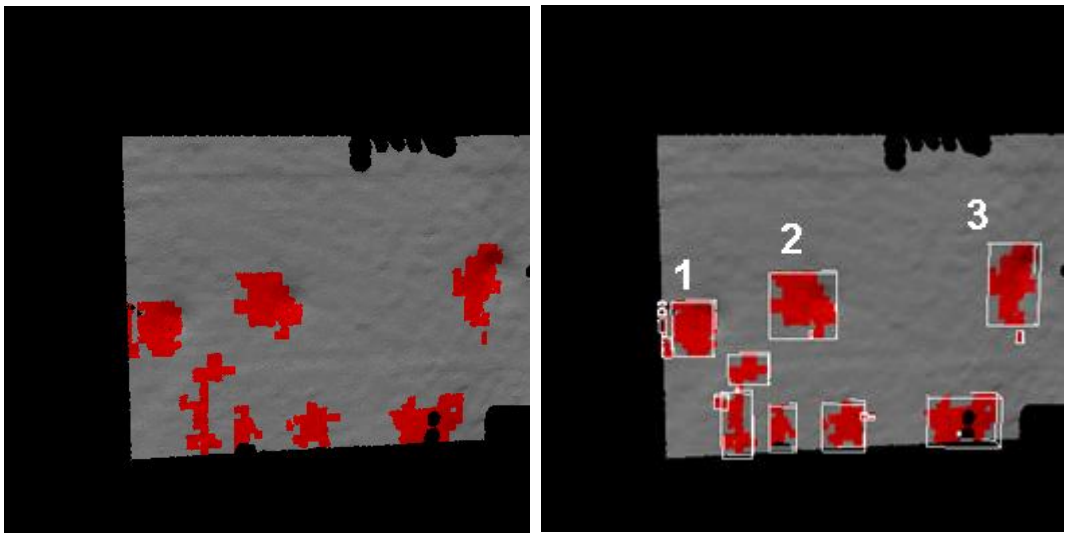

Figure 6.11. Extraction at octree level 8 on filtered high resolution CPU panel mesh(Left). Bounding boxes of segments at octree level 8 (Center).

Again, the segmentation is done at the same resolution as the maximum depth in the feature extraction, since there are no discontinuities. The segments are clear, and there are a few non-deformation areas extracted underneath the right-most deformation, and along the left-most boundary, which are due to noise and acquisition systems limitations along the borders. The bottom has a significant amount of variation that is segmented, but the blame must be placed on the feature extraction and not the segmentation for these segments.

The results above show the segmentation applied on problematic feature extraction results, but the segmentation algorithm can also work very well when the deformations are isolated more clearly through the feature extraction. An example of this is shown in Figure 6.12,

using incremental thresholding to isolate only the deformations on the filtered high resolution CPU panel mesh.

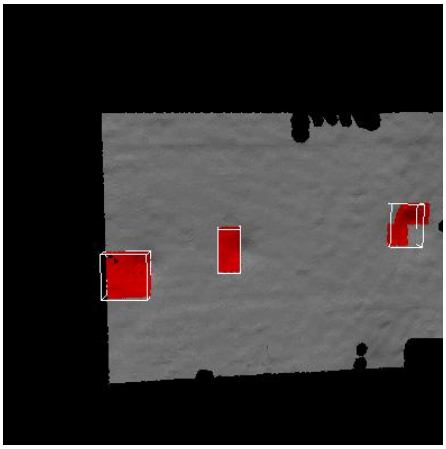

Figure 6.12. Bounding boxes of segments, from clearly extracted deformations on filtered high resolution CPU panel mesh, at octree level 6 (Center).

6.2 Multi-Resolution Segmentation based on Octrees

With the aggregate standard deviation octree enhancement, presented in section 4.4, multi-resolution information is taken into account during the tree generation, and as a result, the single resolution segmentation inherently uses multi-resolution information. Also, features are extracted in the aggregate standard deviation octree technique on only one resolution of voxels, meaning the other levels of the octree may not have even been generated, let alone thresholded to extract features. Because of these characteristics, multi-resolution segmentation cannot be applied onto the extracted features from the aggregate standard deviation octree technique.

For the original octree method, with the non-uniform weighting incremental thresholding enhancements that were discussed in section 4.2 and 4.3, there is no multi-resolution contribution to the segmentation. Also, all levels of the octree, until the maximum resolution, are generated, so they can be used as data for the multi-resolution segmentation. For this reason, multi-resolution segmentation can be applied on the original octree method and the discussed enhancements from section 4.1, 4.2, and 4.3.

The idea of feature persistence was introduced by Pauly and Gross [39], and reviewed in chapter 2. This idea inspired the accumulated standard deviation enhancement, proposed in section 4.4, and can be extended to the segmentation algorithm for the octree technique using the enhancements described in section 4.1, 4.2, and 4.3. This idea also inspires the proposed multi-resolution segmentation technique.

As feature persistence states that if a feature appears over multiple resolutions it is considered a strong feature, the idea can be extended to segments such that if the same segment exists over multiple resolutions then it is also a strong segment. Since feature information is represented in multiple resolutions of the octree, single-resolution segmentation is run on multiple levels of the octree. To apply the idea of persistence to segmentation, segments from a resolution can be grouped with segments from its adjacent resolutions to determine which deformations consistently appear over several successive resolutions. An example of this is shown in Figure 6.13.

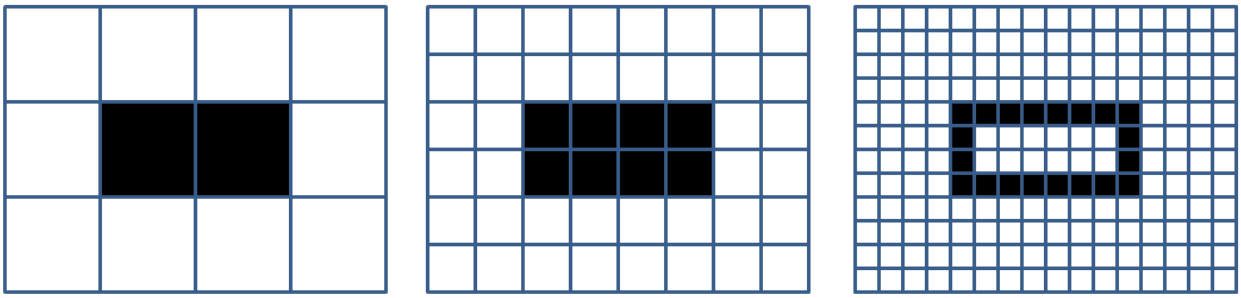

Figure 6.13. Progressively higher resolution versions of the same feature.

Typically, the approximate size of the deformations of interest will be known by the operator, in the same way that they are required to be known for the thresholding in the contour analysis, in section 5.4. Though their exact shape, distribution, locations on the mesh are not known, the size range will allow the operator to determine which resolutions the deformations will lie in. Therefore, that range of resolutions must be selected. As in the single-resolution feature segmentation, these resolutions are selected based on connectivity. Only levels of the octree that represent the selected resolutions are evaluated. A multi-resolution segment is represented by a list, where each element of the list represents that segment at a certain resolution. A multi-resolution segment consists of only segments that are in consecutive resolutions.

First, the single-resolution segmentation is executed on the voxels of each of the resolutions in the defined range. The first resolution in the defined range is used as the *starting resolution*. The algorithm starts by creating a multi-resolution segment for each segment in the starting resolution, and the segments are added to their respective lists.

Beginning with any existing multi-resolution segment, a matching segment is searched for in the next resolution. This matching, which will be discussed below in further detail, is done by comparing the last segment which was added to the multi-resolution segment with each of

the segments in the next resolution. When a match is found, that segment is added to the multi-resolution segment, and a matching segment to that is searched for in the next resolution. This process is repeated until there is no match at a certain resolution or until all resolutions in the range have been exhausted for matching, thus completing the multi-resolution segment. When a multi-resolution segment is complete, the next multi-resolution segment is evaluated for matching. The matching process is repeated, searching for matches in each successive resolution, beginning with the starting resolution, excluding segments that already belong to another multi-resolution segment. If no multi-resolution segment remains to be evaluated, all multi-resolution segments that start at the starting resolution have been completed.

It is possible that some multi-resolution segments will not start in the first resolution of the range, therefore this algorithm must be re-started for each of the resolutions in the range. The algorithm is re-started by incrementing the starting resolution and creating a new multi-resolution segment for each segment at the new starting resolution, excluding segments that already belong to other multi-resolution segments. When all multi-resolution segments have been completed and all starting resolutions have been exhausted, the multi-resolution segmentation is complete and results in a set of lists that represent multi-resolution segments that appear in the model. Lists with only one element contain segments that were only found at a single resolution, while lists with multiple elements contain segments that were found at multiple resolutions, indicating persistence. Removing multi-resolution segments that contain segments in fewer resolutions can remove noisy areas, transient features, and even acquisition artifacts. Examples of this will be shown in section 6.2.3.

The matching process is a fundamental part of extracting multi-resolution segments. Segments can change size and shape between resolutions. Also due to the nature of evaluating a feature at multiple resolutions, it is possible that a segment at one resolution might be broken up into several segments at a finer resolution, as shown in Figure 6.14. To account for the inexactness of segments that represent the same segment at different resolutions, flexible matching techniques are required. Two possible techniques are proposed and evaluated. The first technique for matching segments in successive resolutions is based on hierarchical relationships with the octree, and is discussed in section 6.2.1. The second technique attempts to match segments of successive resolutions based on shared points, and is discussed in section 6.2.2.

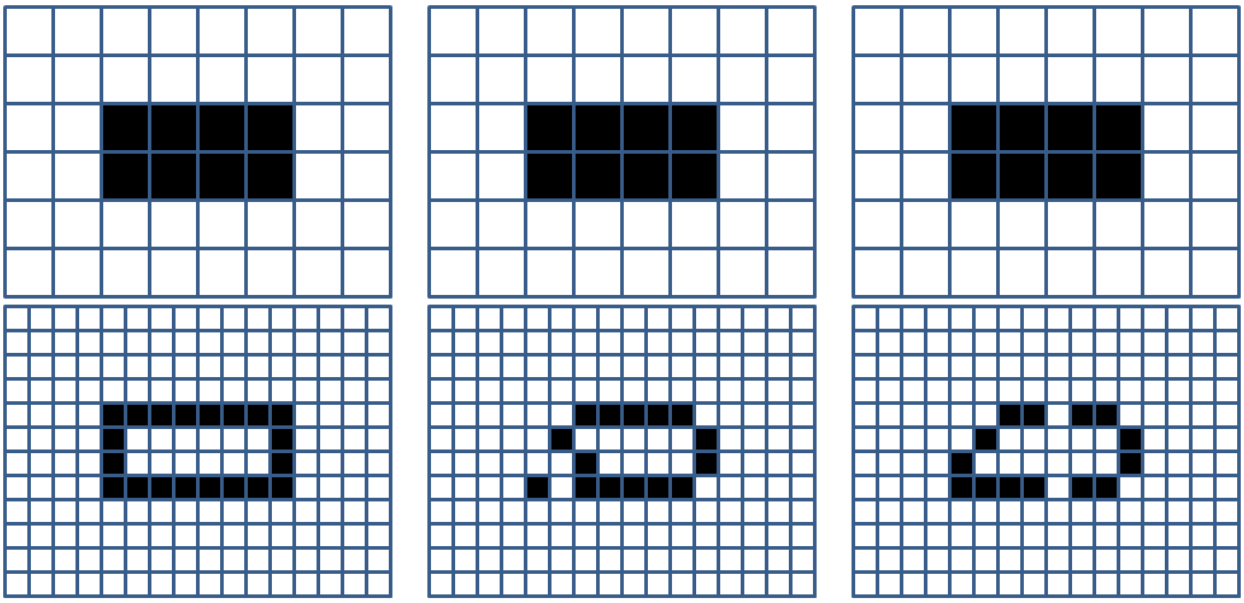

Figure 6.14. a) An ideal scenario where a single high resolution segment matches to a single low resolution segment. b) A scenario where the low resolution segment is slightly broken up in the higher resolution, but can still be matched to the large high resolution segment while ignoring the outlying high resolution segment. c) A scenario where the low resolution segment is broken up such that it should not be matched to either of the two high resolution segments.

6.2.1 Matching based on Hierarchy

This technique attempts to match one segment to a segment in the next resolution using the hierarchy of the octree as guidance. The hierarchy of the octree provides information as to which nodes represent higher resolution versions of other nodes.

Let *a* represent a segment composed of a set of voxels. Let *b* represent a segment in a resolution one step higher. If some set of voxels in *a* are parents of some set of voxels in *b*, some part of segment *a* matches segment *b*. Due to the nature of the multi-resolution representation, it is likely that all voxels from *b* are children of *a*, yet not all voxels of *a* are parents of voxels from *b*. If all voxels in *b* are children of voxels in *a*, it is either a good match or it is possible that *a* has been broken up into multiple segments. If all voxels in *a* have children in *b*, it is either a good match or it is possible that *b* contains voxels that have parents that do not belong to *a*.

A proposed measure of this is to determine how many nodes in *b* have parents in *a*, and how many nodes in *a* have children in *b*. Some metric must be used to represent this hierarchical consistency, and some threshold must be set to determine how much hierarchical

consistency there must be between two segments such that they can be considered the same feature. A proposed metric is shown in Eq. 6.2.

$$h = \frac{1}{2}\left(\frac{p}{r} + \frac{q}{s}\right) \tag{6.2}$$

where *h* is the hierarchical consistency metric, *p* is the number of voxels in a segment *a* that has at least one child in segment *b, r* is the number of voxels in segment *a, q* is the number of voxels in a segment *b* that is a child of a voxel in segment *a*, and *s* is the number of voxels in segment *b*. The metric *h* is compared to some threshold to determine whether segment *a* and segment *b* are the same segment based on hierarchical consistency.

This metric is good in theory, but in practice it is rare that $\frac{q}{s}$ is equal to anything other than 0 or 1, which unnaturally influences the *h* metric. This is because a higher resolution segment will usually either belong entirely to a single lower resolution segment, or not at all. It is very rare that a higher resolution segment belongs only partly to one low resolution segment. For this reason, it is better to use only $\frac{p}{r}$ as a metric, as that will give a more realistic measurement of the hierarchical consistency. The updated hierarchical consistency metric equation is shown in Eq. 6.3.

$$h = \frac{p}{r} \tag{6.3}$$

6.2.2 Matching based on Shared Points

This technique attempts to match one segment to a segment in the next resolution using geometric properties as guidance. The geometric properties of the contained mesh in a segment can be compared to the geometric properties of the mesh contained in another segment to determine if they represent the same part of the mesh.

In the same way as in the preceding section, let *a* represent a segment composed of a set of voxels. Let *b* represent a segment in a resolution one step higher. If some portion of the mesh contained in *a* is also contained in *b*, then some parts of segment *a* match segment *b*.

Again, due to the nature of multi-resolution representation, it is unlikely that the size and centroid will be the same for two features at different resolutions. A more appropriate matching scheme is to see how much the bounding boxes of the segments overlap. However, this would result in skewed representations of how much overlap there is, since the surface contained may not be parallel to the x-y, x-z, or y-z plane, resulting in bounding boxes with large volume, even though the surface is not large in area.

Since there is knowledge available about which 3D points of the mesh lie within each voxel, that information can be used for comparison. Each point in segment *a* is checked to see

if it is also in segment *b*. The segments may not always be exact versions of each other, as segment *a* may be broken up further in the subsequent resolution as shown in Figure 6.14.

A metric is used to represent this comparison, and a threshold must be set to determine how much of the model must be shared between two segments such that they can be considered the same feature. That proposed metric is shown in Eq. 6.4.

$$g = \frac{t}{v} \tag{6.4}$$

where *g* is the shared points metric, *t* is the number of points that are shared by being in both segment *a* and segment *b,* and *v* is the number of points that are in segment *a*. The metric *g* is compared to a threshold to determine whether segment *a* and segment *b* are the same segment based on shared points.

This metric of shared points, *g*, is a more accurate representation of whether a segment is a higher resolution version of another segment than the hierarchical consistency metric, *h* introduced in Eq. 6.3, but it is also more computationally expensive. The need to iterate through each individual 3D point in the mesh, instead of through each voxel, causes the added processing time since there are many more 3D points than voxels.

6.2.3 Results

The multi-resolution segmentation is applied to several meshes, to test its effectiveness. Both the hierarchical matching metric and the point-based matching metric were used, and they provided very similar results with no discernible difference other than a slightly longer execution time for the point-based metric, as shown in Table 6.1. As a result of there being no visually displayable difference, the hierarchical matching metric is used for all the results, with a similarity threshold of 0.85.

Model	Hierarchical Matching	Point-based Matching
CPU Panel	1.03s	1.3s
Car Door	5.2s	8.4s
Flat Mesh with Small Deformation	0.04s	0.04s
Flat Mesh with Large Deformation	1.7s	1.73s
Curved Mesh with Small Deformation	0.08s	0.09s
Curved Mesh with Large Deformation	0.5s	0.6s

Table 6.1. Computation time comparison between hierarchical matching and point-based matching for multi-resolution segmentation.

Also, for simplicity, the bounding box of the lowest resolution version in a multi-resolution segment is displayed for each multi-resolution segment. To show the usefulness of multi-resolution segmentation, segments that are represented in only one resolution are not displayed. The multi-resolution segmentation algorithm is applied on the results of feature extraction on the flat mesh with a small deformation. The feature extraction was done until octree level 9, and multi-resolution segmentation was executed at a starting resolution of 4 and ending resolution of 9. The results are shown in Figure 6.15.

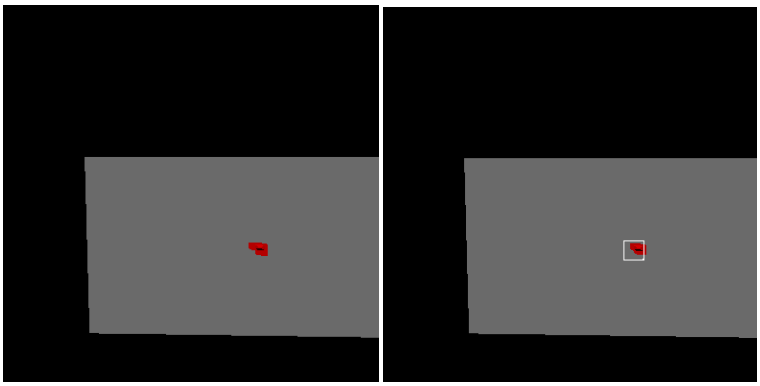

Figure 6.15. Feature extraction results at octree level 9 on flat mesh with small deformation (Left). All multi-resolution segments with bounding boxes (Right).

Only one multi-resolution segment is created, and it highlights the only deformation in the mesh. That multi-resolution segment spans 4 resolutions, making it a strong segment.

The multi-resolution segmentation algorithm is applied on the results of feature extraction on the filtered high resolution CPU panel mesh. The feature extraction was done until octree level 8, and multi-resolution segmentation was executed at a starting resolution of 3 and ending resolution of 8. The results are shown in Figure 6.16.

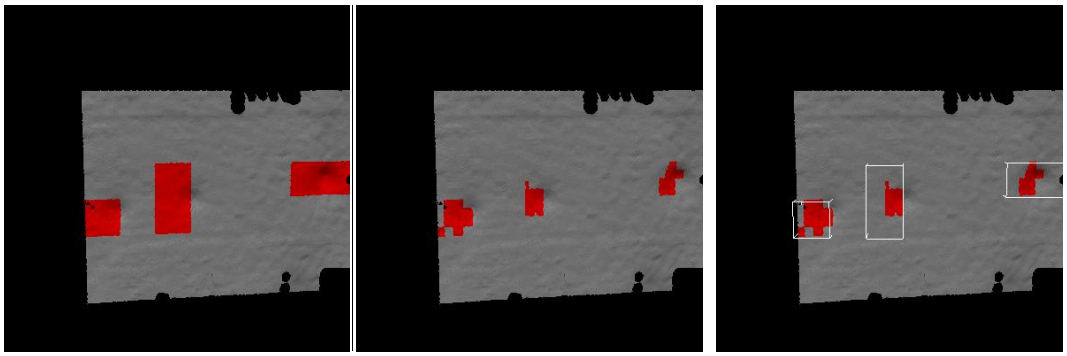

Figure 6.16. Feature extraction results at octree level 5 on the filtered high resolution CPU panel mesh (Left). Feature extraction results at level 7 (Center). All multi-resolution segments with bounding boxes (Right).

The only segments created are multi-resolution segments representing each of the three deformations. Each of those multi-resolution segments contain 4 resolutions, meaning those features are very strong across multiple resolutions. Also, the bounding boxes are larger in the displayed image of the segments. This is because, as stated earlier, the bounding boxes are the dimensions of the lower resolution version in the multi-resolution segment, and it can be seen that the bounding box sizes correspond to the extracted features at octree level 5. This shows that with good feature extraction results, exact multi-resolution segmentation can occur where the same deformations over multiple resolutions are grouped together.

With good feature extraction results, such as the ones shown above, the multi-resolution segmentation excels. To test its performance in situations that are less than ideal, the algorithm must applied on much less desirable feature extraction results. The multi-resolution segmentation algorithm is applied on the results of feature extraction on the filtered high resolution car door mesh. The feature extraction was done until octree level 9, and multi-resolution segmentation was executed at a starting resolution of 4 and ending resolution of 9. The results are shown in Figure 6.17.

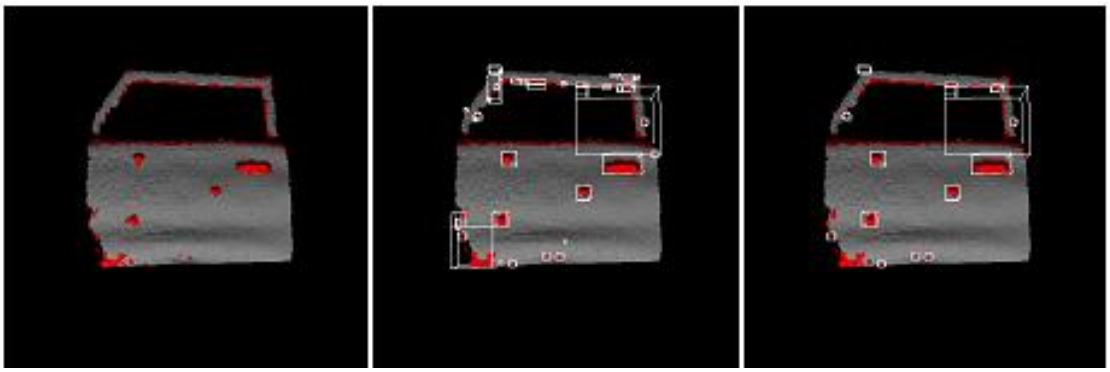

Figure 6.17. Feature extraction results at octree level 9 on filtered high resolution car door mesh (Left). All multi-resolution segments with bounding boxes (Center). Multi-resolution segments of 3 resolutions or greater (Right).

The door mesh feature extraction is quite good, but has several noisy areas and acquisition artifacts along the boundaries that are extracted as features. The multi-resolution segmentation is applied, and produces better results than the single-resolution segmentation, as fewer areas are segmented, while keeping the deformations of interest. When only multi-resolution segments that are 3 resolutions or greater in span are selected, even fewer non-deformation areas are segmented, while keeping the deformations of interest. This reduction shows the power of multi-resolution segmentation in identifying features that are strong over

multiple resolutions. The reduction of non-deformation areas brings the system closer to the goal of extracting only deformations of interest. The final step, classification, is relied on to remove all non-deformation areas and will be discussed in section 6.4.

The multi-resolution segmentation algorithm is applied on the results of feature extraction on the filtered high resolution CPU panel mesh. The feature extraction was done until octree level 9, and multi-resolution segmentation was executed at a starting resolution of 4 and ending resolution of 9. The results are shown in Figure 6.18.

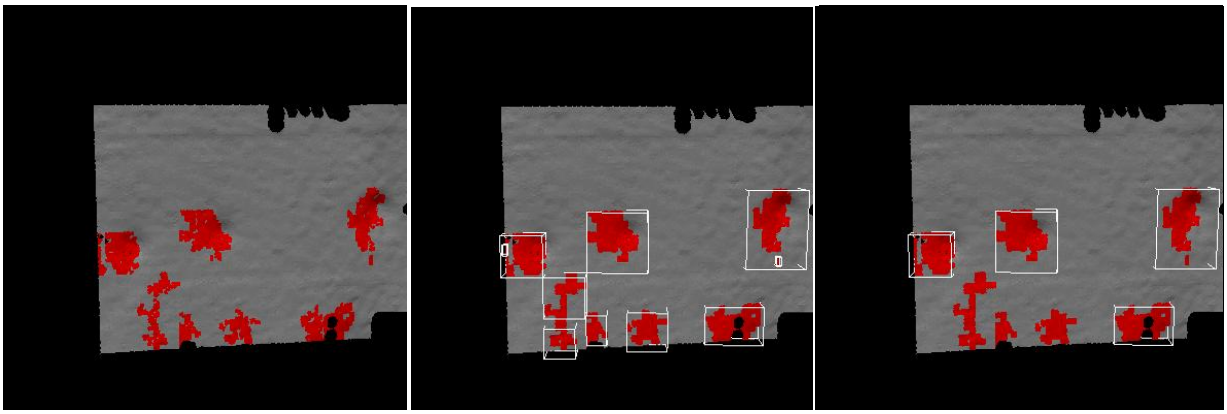

Figure 6.18. Feature extraction results at octree level 9 on filtered high resolution CPU panel mesh (Left). All multi-resolution segments with bounding boxes (Center). Multi-resolution segments of 4 resolutions or greater (Right).

In this case, the feature extraction data with is not very clean, and contains areas of surface deformation and boundary acquisition artifacts which are also extracted alongside the deformations. The multi-resolution segmentation provides better results than single-resolution segmentation, in keeping the deformations of interest while reducing other non-deformation areas from being segmented. When multi-resolution segments of 4 resolutions or greater are isolated, even more non-deformation areas are removed. Only the deformations of interest, and a significant surface variation on the bottom of the mesh, remain. This shows that, though more of a complex algorithm, the idea of feature persistence can help differentiate important segments from transient segments of no importance.

The multi-resolution segmentation algorithm is applied on the results of feature extraction on the curved mesh with a large deformation. The feature extraction was done until octree level 7, and multi-resolution segmentation was executed at a starting resolution of 2 and ending resolution of 7. The results are shown in Figure 6.19.

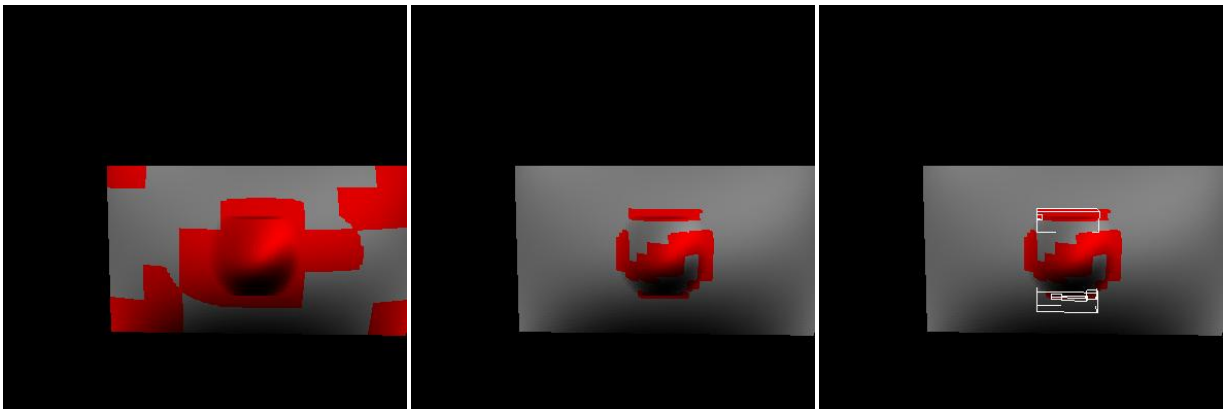

Figure 6.19. Feature extraction results at octree level 5 on curved mesh with large deformation (Left). Feature extraction results at level 7 (Center). All multi-resolution segments with bounding boxes (Right).

This final set of results shows the shortcomings of the multi-resolution segmentation. If the feature extraction itself is not strong, and there is significant difference in the extraction of a deformation over several resolutions, the multi-resolution segmentation will not be able to properly extract the deformation. In the extraction, there is significant difference in the extraction results over multiple resolutions, as shown by the differences between octree level 5 and octree level 7. This causes a segment that contains the entire deformation to fail in matching segments at higher thresholds, and the matching results never exceed the matching threshold due to significant variation and discontinuities between each resolution. Several multi-resolution segments are created in the areas which are consistently extracted over multiple resolutions, however these are not the actual deformation but pieces of the deformation. In such a situation, it is better to select an appropriate resolution and conduct single-resolution segmentation to properly segment the deformation, as shown in section 6.1.3.

Overall, the multi-resolution segmentation shows that, provided consistent data from the octree-based feature extraction technique, the segmentation can isolate areas of interest further. By implementing the idea of feature persistence, across multiple resolutions, this algorithm successfully creates segments for areas of importance, while further eliminating non-deformation areas.

6.3 Segmentation based on Contour Analysis

The contour analysis technique results in lots of useful information that can be beneficial for segmentation. At the worst, this additional information can be ignored and segmentation can occur in much the same way as it does for the octree-based segmentation technique, and that is discussed in section 6.3.1. At best, this additional information can be used to segment areas of

the mesh more accurately, including the added benefit of classification of deformation type, and that is discussed in section 6.3.2. Finally, results for both are presented in section 6.3.3.

6.3.1 Contour Analysis Segmentation without Deformation Pattern Information

After the thresholding stage in the contour analysis technique, from section 5.4, and provided that the octree data structure was used for slice acquisition, only deformation patterns of interest remain. Each point in that deformation pattern belongs to a node in the octree, which represents a voxel. Only voxels that belong to the thresholded deformation patterns remain, and these will be the feature voxels. Single resolution segmentation is performed on the results of the contour analysis technique, as it was in section 6.1. All of the voxels contained in the deformation patterns outputted by the contour analysis technique are used as feature voxels. From there, the single resolution segmentation can be applied on them.

The method of determining adjacency, when dealing with the contour analysis segmentation, is slightly different than when dealing with the octree method. Adjacency is determined by evaluating each point contained in a voxel to determine, using the mesh triangulation, which points are connected and which voxels those connected points belong to. This process was detailed in the contour analysis technique, during the next node selection step of slice acquisition, discussed in section 5.1.3. In the next node selection step of slice acquisition, the selection of adjacent nodes were limited to nodes along the same y-level. For determining adjacency in general, that limitation does not exist. To reiterate this technique for determining adjacency, all points in the current feature voxel are evaluated to see which points are adjacent to them in the mesh. Each one of the adjacent points that lies outside the current voxel has a voxel that it belongs to, and if that voxel is a feature voxel, it is determined as an adjacent voxel. This technique is slower than the cell-addressing technique used in section 6.1.2 for single resolution grouping, but it is more accurate in determine which voxels contain an adjacent portion of the mesh and consistent with the contour analysis technique. Such a technique cannot be applied in the octree-based technique since some of the points contained in each node will have been removed due to thresholding occupancy and removal of non-feature points that do not exist in the highest resolution of the tree.

It is important to note that, if ordered lines of data points were used for slice acquisition, instead of the voxels, as briefly discussed in section 5.1, then each point in the deformation pattern belongs to a point in the mesh, and these mesh points are the feature points. Single resolution segmentation can be applied on those points in the same manner, except adjacency between points will be based only on the connectivity of points in the mesh, as opposed to cell

addressing. Also, the outputted segment of the segmentation technique will be a group of points and not a group of voxels. This extension of segmentation from a voxel-based domain to a point-based domain would be useful if the data acquisition outputted data in the form of an ordered point cloud, and the slice acquisition acquired lines of ordered points instead of using voxels. Though such an extension can easily be applied, this thesis is concerned only with data acquired in the form of unordered point data, so all explanations and results are with respect to using a voxel-based system.

6.3.2 Contour Analysis Segmentation with Deformation Pattern Information

The contour analysis technique provides very useful information about the characteristics of the extracted deformation patterns. Based on these characteristics, certain educated decisions can be made about which areas to connect for segmentation. Also, because contour analysis inherently classifies the deformation patterns as dings or dents, the segments will also have this attribute. This means that the additional classification step does not need to be executed to determine whether each segment is a ding or dent, as that attribute can be directly copied from the deformation patterns contained in the segment. The classification step is still useful to remove non-deformation areas that may have been extracted in the contour analysis technique through non-optimal thresholding.

There are two types of segmentation that can be done when using deformation patterns. The first type makes use of only the deformation patterns that were outputted by the contour analysis technique after the thresholding, and is discussed in section 6.3.2.1. The second type is proposed for a very limited scenario. It makes use of all deformation patterns calculated by the contour analysis technique and is discussed in section 6.3.2.2.

6.3.2.1 Matching using Only Thresholded Deformation Patterns

Given that each deformation pattern is a cross-section of a deformation, it only makes sense that to segment an entire deformation, all the cross-sections of the deformation must be grouped together. Starting with a single deformation pattern, only deformation patterns that are at an adjacent y-level should be connected to form a segment. Also, an additional constraint is that connected deformation patterns must be of the same type, meaning that dings can only connect to dings, and dents can only connect to dents. This is so, because the assumption of this thesis is to extract only deformations of those two types, and therefore each cross-section of a ding will resemble a ding, and the same is true for dents.

The single-resolution segmentation algorithm is still applied, with the same extension from deformation patterns to voxels and point-connectivity-based adjacency discussed in section 6.3.1, but the mentioned constraints are added as modifications to that algorithm. The voxels that belong to the deformation patterns after the thresholding phase of contour analysis will be the feature voxels.

The algorithm begins in the same fashion, by selecting any feature voxel as the seed voxel. Adjacency is determined in the same way as the single-resolution segmentation. However, the adjacent voxel must meet certain criteria before it can be added to the segment. The adjacent voxel must either belong to the same deformation pattern as the seed voxel or it must belong to a deformation pattern that is in an adjacent y-level to that deformation pattern. Also, the adjacent voxel must belong to a deformation pattern that is of the same type as the deformation pattern which the seed voxel belongs to, in that both deformation patterns must be dings or dents. Also, the original criterion that the adjacent voxel is not already part of the segment must still be met. If these criteria are met, that adjacent voxel can be added to the segment. This process is repeated for each voxel in the segment, as the new seed voxel, until no more voxels can be added to the segment. At this point, the segment is complete, and it represents all the voxels belonging to an actual deformation.

By simply adding criteria to the single resolution segmentation technique to make use of information provided by the deformation patterns, falsely connected voxels to a segment can be reduced, increasing the likelihood that a segment resembles the actual deformation. And since segments will only contain deformation patterns of one type, either dings or dents, the entire segment can be labelled with that deformation type as an attribute. This, as stated earlier, allows the deformation segment to be classified as a ding or dent without the need for the classification step.

6.3.2.2 Matching using All Deformation Patterns

Again, given that a deformation pattern is a cross-section of a deformation, one important consideration is whether the contour analysis thresholding allows the extraction of all cross-sections of all deformations. Since different cross-sections of a deformation might have different magnitude and span characteristics, in some situations, however rare, it might be difficult to select a threshold that isolates all cross-sections of that deformation. Therefore, it would be difficult to extract the entire deformation, using just the available cross-sections after thresholding. An example of this is shown in Figure 6.20, where only the large deformation

must be extracted, but by thresholding the magnitude and span attributes, only a small section of the large deformation can be extracted.

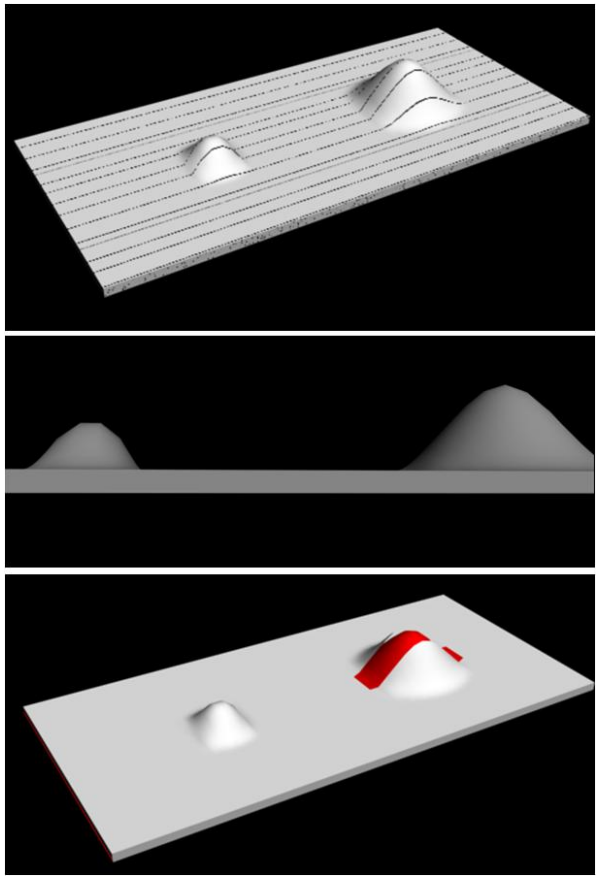

Figure 6.20. Large and small deformation, with direction of slice acquisition shown (Top). Profile view of deformation sizes (Center). By eliminating the small deformation using magnitude and span in contour analysis thresholding, only some cross sections of the large deformation can be extracted. (Bottom)

Thresholding should be able to isolate at least one cross-section of the deformation, and the deformation pattern representing that cross-section should be enough to aid in the search for all cross-sections that belong to the same deformation, instead of being able to extract only a piece of the deformation as a segment. This technique is useful in the case where all the cross-sections of a deformation are required in a segment, but the contour analysis thresholding cannot isolate all the cross-sections of a deformation. This process is similar to the single resolution segmentation, but uses deformation patterns instead of voxels and a different set of criteria to determine adjacency. Also, it uses all the information available from the contour analysis, not just the thresholded deformation patterns, and employs something similar to a

seeded region growing approach, the 2-dimensional segmentation technique explained in chapter 2.

The primary goal is to combine a set of deformation patterns into a segment, which is slightly different from the techniques discussed earlier, where segments are composed of a set of voxels. However, the last stage of this algorithm converts a segment composed of deformation patterns into a segment composed of voxels, to remain consistent with the definition of a segment described above, as well as the output of the segmentation module, also described above.

Each extracted deformation pattern from the contour analysis technique, after thresholding, is labelled as a potential seed pattern. Each of these seed patterns will belong to a deformation, and will behave as a starting point in the search for other deformation patterns that belong to the same deformation. A seed pattern must be selected from the potential seed patterns as the largest magnitude deformation pattern that does not already belong to a deformation pattern segment. Then all deformation patterns that were extracted, before the thresholding phase of the contour analysis technique, are available to be adjacent deformation patterns. This allows adjacent patterns to be connected, even if they did not meet the thresholding criteria of the contour analysis technique, hence the motivation for this method.

A seed pattern is selected and is added to a blank list which will represent a segment. This segment represents the deformation, and contains its own characteristics of magnitude and span. This segment's magnitude will be the magnitude of the initial seed deformation segment that was added to it, and the starting span will be the span of that same deformation segment. These values will be useful to determine which deformation patterns should be added to the segment and which should not. A seed pattern is chosen as the deformation pattern with the largest magnitude that has not already been added to a segment, ensuring that the deformations are segmented in the order of largest magnitude to smallest magnitude.

All adjacent deformation patterns must be found to connect it to the seed pattern, and add it to the deformation pattern segment. All deformation patterns extracted in the contour analysis technique, before the thresholding phase, are available to be potential adjacent deformation patterns, but they cannot be seed patterns. An adjacent deformation pattern is determined as a pattern that meets certain criteria. It must be in an adjacent y-level to the seed pattern. It must not already belong to the deformation pattern segment. It must be the same type as the seed pattern, in that both have to be dings or dents. At least one of the voxels, represented by a point in the deformation pattern, must be adjacent to at least one of the voxels represented by a point in the seed pattern. And finally, the deformation pattern cannot already

belong to a deformation pattern segment. If these criteria are met, the deformation pattern is determined to be adjacent.

Region growing thresholding criteria are required to ensure that the segment does not include various adjacent deformation patterns that do not actually belong to the deformation in question. Thresholds are used to determine whether an adjacent deformation pattern is to be added to the segment or not, just as in the 2D region growing approach where segments would be concluded when no adjacent pixel meets the threshold. Noise and surface curvature may also appear as adjacent deformation patterns that meet all of the above criteria, so measures must be taken to ensure those patterns are not added to the segment. The thresholding criteria are that the magnitude must be at least some percentage of the segment's magnitude and the span must be within some range of the segment's starting span. If the adjacent deformation pattern meets the thresholding criteria, it can be added to the segment. In practice, only a magnitude threshold is required to be set for the region growing, and will be demonstrated in the results section.

When adjacent deformation patterns to the seed pattern have been found, and they meet the thresholding criteria, they are added to the deformation pattern segment. Adjacent deformation patterns are found to those deformation patterns, and so on so forth until no deformation pattern can be added, thus concluding the deformation pattern segment and resulting in a set of deformation patterns that define an actual deformation. Then a new seed pattern is found to start a new deformation segment, to repeat the entire process until no new seed pattern can be found, concluding the algorithm and resulting in a set of deformation pattern segments. The end result is a set of deformation pattern segments, which must be converted to voxel segments to remain consistent with the overall system.

The process to convert deformation pattern segments to voxel segments is simple. All the voxels contained in each deformation pattern for a given deformation pattern segment are taken, and a new segment is created composed of those voxels. This is performed for each deformation pattern segment, until the final result is a set of segments composed of voxels, just as in the other techniques described for segmentation. And just like the segmentation technique in section 6.3.2.1, since segments will only contain deformation patterns of one type, the deformation segment can be classified as a ding or dent without the need for the classification step.

A minor limitation of this segmentation technique is the requirement to set additional thresholds which aid in concluding a segment. This introduces more room for error if these are selected incorrectly. Also, it reintroduces deformation patterns that were thresholded in the

contour analysis technique, increasing the possibility of less accurate segmentation. Usually a simple magnitude threshold will suffice, which will be demonstrated in the results section. Ensuring deformation patterns have a magnitude that is at least a certain percentage of the segment's magnitude is a simple way of limiting the deformation patterns that can be grouped into a segment. This threshold is very simple, and the same threshold can be used for various meshes, depending on the desired accuracy.

6.3.3 Results

6.3.3.1 Single Resolution Segmentation for Contour Analysis

The single resolution segmentation is first applied on the contour analysis results of the unfiltered high resolution CPU panel mesh. The results are shown in Figure 6.21.

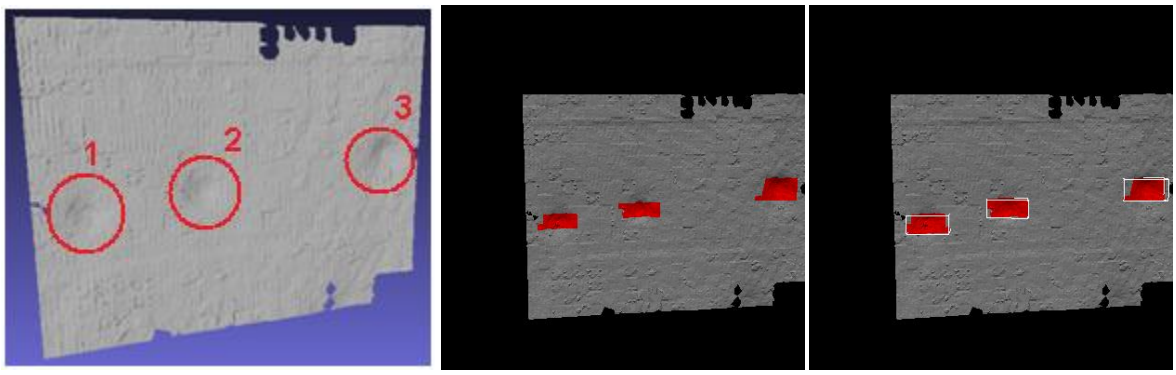

Figure 6.21. CPU panel with deformations circled (Left). Properly thresholded contour analysis results (Center). Properly segmented deformations (Right).

The results are as expected. Since the contour analysis clearly highlighted the deformations, the single resolution segmentation easily creates segments for each of the deformations.

To test how the segmentation responds to incorrectly thresholded contour analysis results, it is applied to the non-optimal results of the contour analysis on the curved mesh with large deformation. The results are shown in Figure 6.22.

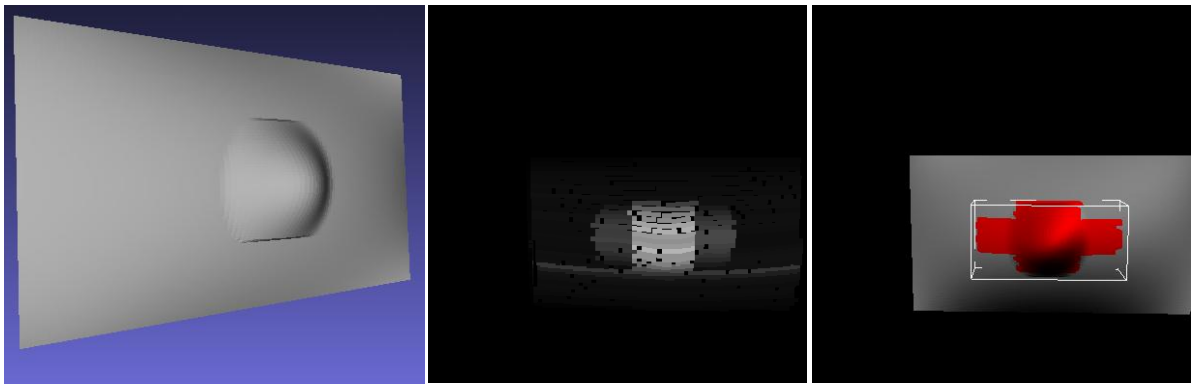

Figure 6.22. Curved mesh with large deformation mesh (Left). Intensity corresponding to magnitude of contour analysis deformation patterns (Center). Poorly thresholded contour analysis results causes incorrect segmentation (Right).

The non-optimal contour analysis thresholding results in the ding being extracted along with some residual curvature, which are dent deformation patterns, on the left and right sides of the actual ding. Without additional information from the contour analysis deformation patterns, the segmentation does not know which areas belong to which deformations. Because of this, it incorrectly ends up generating one large segment for the ding and the residual deformation patterns next to it.

Provided that the contour analysis thresholding only isolates the deformations of interest, the single resolution segmentation will be excellent in creating segments for those areas. However, in some scenarios, this thresholding might not be able to isolate the deformation, and additional knowledge would be required to create correct segments for the extracted deformations.

6.3.3.2 Segmentation using Information from Thresholded Deformation Patterns

Often, the single-resolution segmentation will be sufficient, as the contour analysis thresholding is very flexible and can often isolate areas of interest. However, in cases where it does not perform well, additional information from the thresholded deformation patterns can be useful. An area where the segmentation without using additional information failed was on the poorly thresholded curved mesh with large deformation, shown in the previous results section. Using deformation pattern information, the segmentation is applied with the results being shown in Figure 6.23.

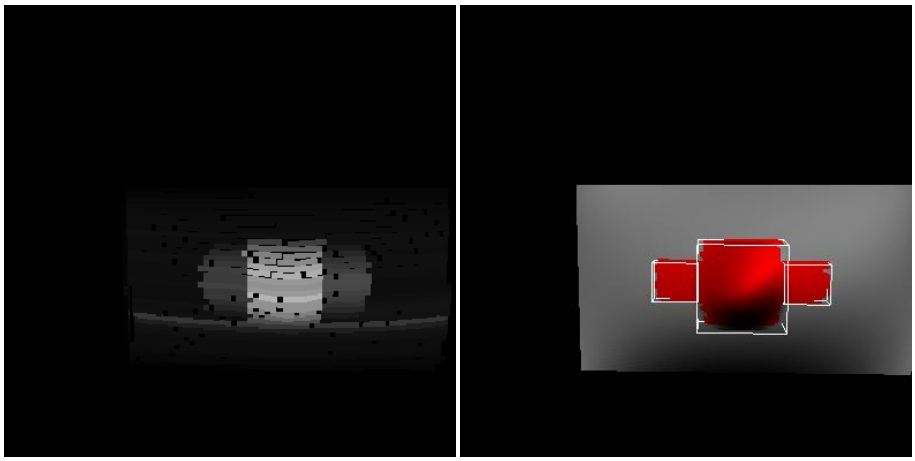

Figure 6.23. Intensity corresponding to magnitude of deformation patterns (Left). Poorly thresholded contour analysis results still generates properly created segments using deformation pattern information (Right).

In the area where the single-resolution segmentation without using additional contour analysis information failed, the addition of contour analysis information, clearly succeeds in segmenting the correct areas. The used contour analysis information, which was detailed in section 6.3.2.1, ensures that only deformation patterns at adjacent y-levels and deformation patterns of the same deformation type are grouped together. The thresholding was poorly chosen and resulted in the ding along with residual curvature extracted as dents on either side of it. As the additional information of whether the deformation is a ding or dent is used, it provides the segmentation with an understanding that dings should not be connected with dents in a segment. This leads to a correct segmentation. Also, the segment containing the deformation is classified as a ding, and the other two segmented areas are classified as dents.

To be tested on a real world model, where two deformations are connected by a residual deformation, the segmentation is applied on poorly thresholded contour analysis extraction of the unfiltered high resolution CPU panel mesh. The results are shown in Figure 6.24.

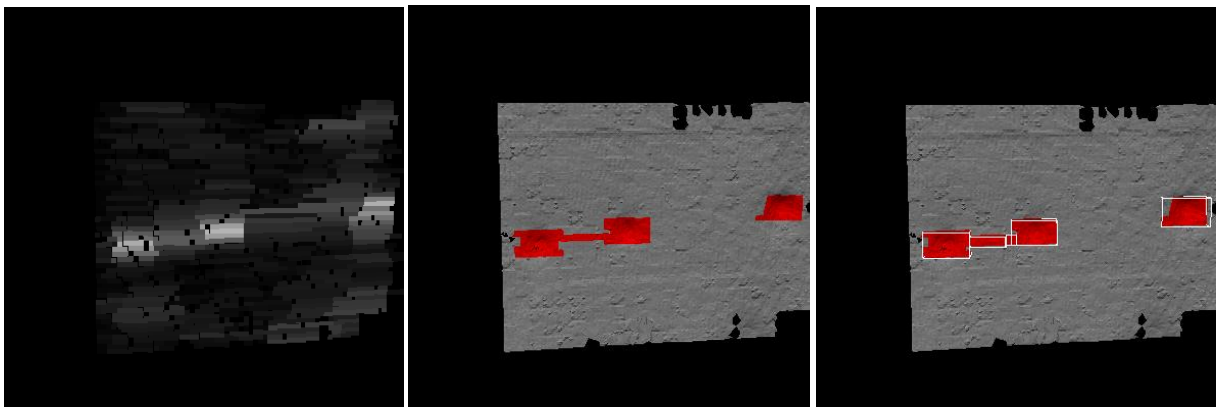

Figure 6.24. Intensity corresponding to magnitude of deformation patterns (Left). Poorly thresholded contour analysis results (Center). Segmentation generates properly created segments created using deformation pattern information (Right).

The poorly selected threshold results in the three deformations being extracted, but the left-most dent and the center dent are connected by a ding deformation. Without using contour analysis information, those two deformations would be segmented as one segment, similar to the results on the curved mesh with large deformation shown in Figure 6.22. Adding contour analysis information about the type of deformation found, the three deformations are clearly segmented, along with an additional segment for the residual deformation. The deformations are correctly classified as dents, and the residual deformation area is classified as a ding. At this stage, it cannot be known that the residual deformation is an area that is not a deformation and should not be extracted, but the correct segmentation of that area will allow it to be removed in the classification stage. Also, the correct segmentation of that area, by nature, allows the correct segmentations of the two deformations on either side of it, preventing them from being missed in the classification phase by being combined into one large segment.

When poorly selected thresholds cause deformation patterns which are not of interested to be extracted, due to the lack of separation from deformations of interest, segmentation can incorrectly group these areas together. Correct segmentation is crucial to extract only the deformations while avoiding non-deformation areas. If non-deformation areas happen to slip through the feature extraction phase, the correct segmentation of these areas will allow the classification to remove it, while maintaining the presence of the deformations of interest. This segmentation algorithm uses additional information to more accurately segment the extracted areas from the contour analysis technique.

Finally, since one of the primary goals of this system is to correctly identify whether each deformation is a ding or a dent, the deformation type attribute is shown for each deformation

segment in each tested mesh. The segmentation results are shown in Figure 6.25, and the classification is shown in Table 6.2.

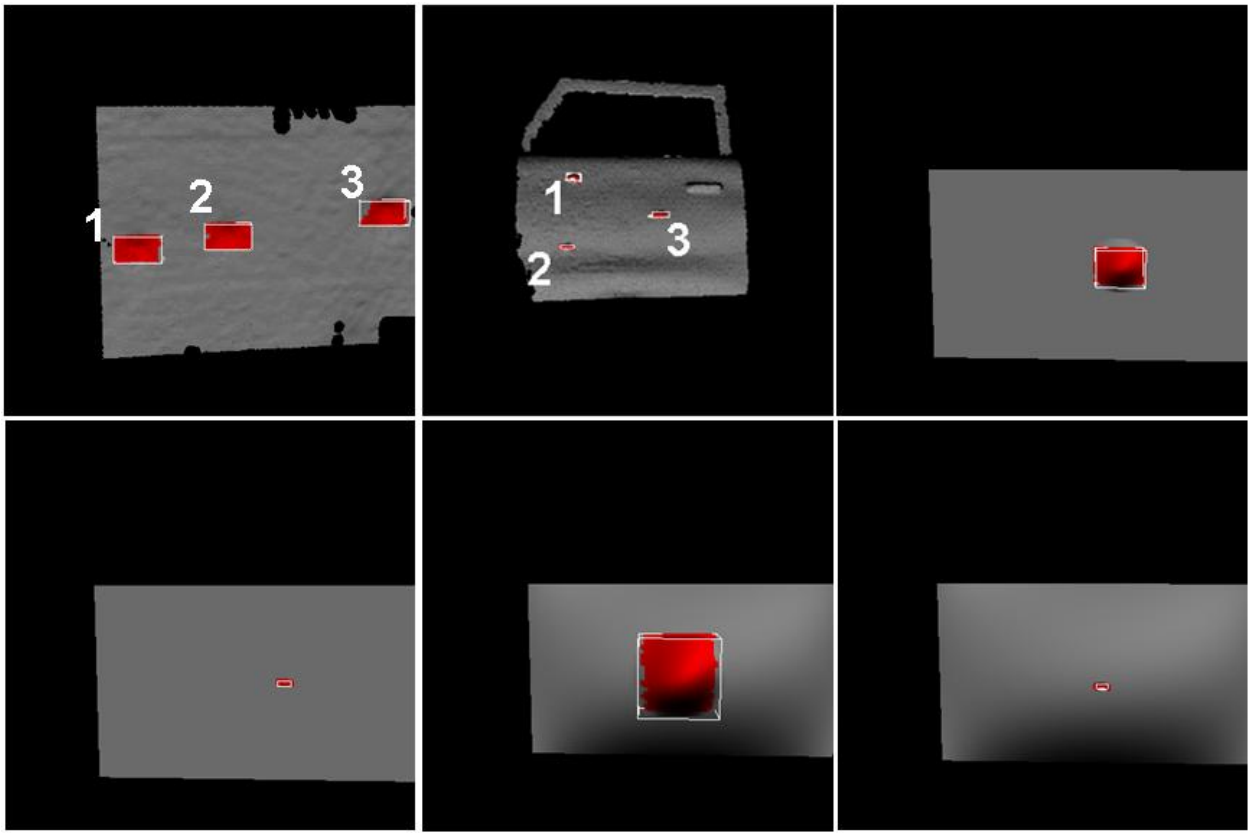

Figure 6.25. Using contour analysis feature extraction and segmentation using information from thresholded deformation patterns: CPU panel mesh with dent segments labelled (Top Left). Car door mesh with ding segments labelled (Top Middle). Flat mesh with large ding segmented (Top Right). Flat mesh with small dent segmented (Bottom Left). Curved mesh with large ding segmented (Bottom Center). Curved mesh with small dent segmented (Bottom Right).

Model/Deformation	Actual	Contour Analysis Classification
Car Door		
Def 1	Ding	Ding
Def 2	Ding	Ding
Def 3	Ding	Ding
CPU Panel		
Def 1	Dent	Dent
Def 2	Dent	Dent
Def 3	Dent	Dent
Flat Mesh		
Small	Dent	Dent
Large	Ding	Ding
Curved Mesh		
Small	Dent	Dent
Large	Ding	Ding

Table 6.2. Comparison of actual deformation characteristics and the results of classification using contour analysis segmentation.

All of the meshes were tested with optimal contour analysis thresholds, and the segmentation correctly identifies the deformation type of every deformation in each mesh. This 100% accuracy shows the effectiveness of the contour analysis technique in extracting deformations and classifying them.

6.3.3.3 Segmentation using Information from All Deformation Patterns

The segmentation is tested on non-optimal thresholding of the contour analysis results using the flat mesh with a large deformation. The results are shown in Figure 6.26.

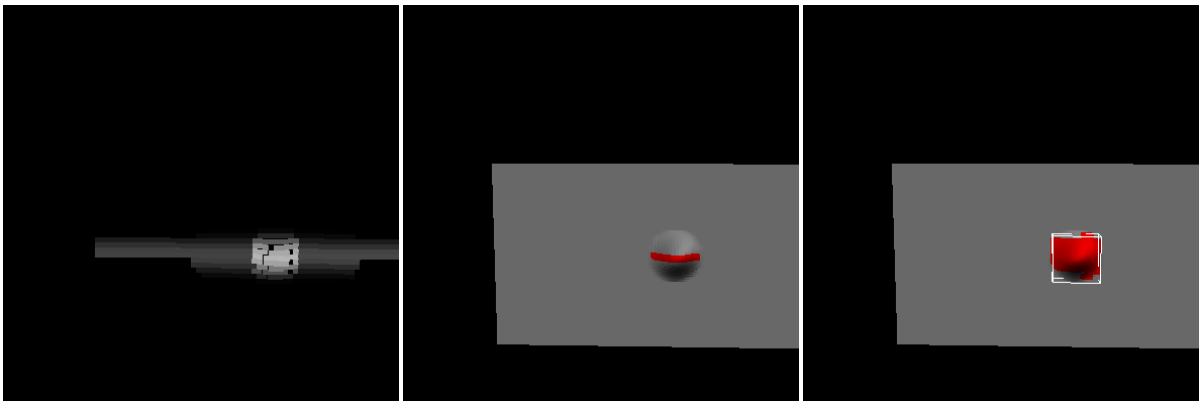

Figure 6.26. Intensity corresponding to magnitude of deformation patterns (Left). Only one cross-section is extracted by the contour analysis (Center). Segmentation generates proper segments by restoring thresholded deformation pattern information, using a region growing magnitude threshold of 0.05 (Right).

It can be seen that the non-optimal thresholding only extracts a single cross-section of the ding. By using all deformation patterns from the contour analysis, instead of just the ones that survive the thresholding, with a region growing magnitude threshold of 0.05, the segmentation is able to segment the entire deformation, and classify it as a ding. The region growing magnitude threshold of 0.05 means that only deformation patterns with magnitudes greater than 5% of the seed deformation pattern can be added to the segment. In this case, the seed deformation pattern is the only deformation pattern extracted from the ding. The results demonstrate that this algorithm can recover areas lost during feature extraction due to poor thresholding, and can still provide an accurate segmentation of the deformation.

To attempt this on a real world model, the segmentation algorithm is applied to the results of non-optimal thresholding with the contour analysis technique using the unfiltered high resolutions CPU panel mesh. The results are shown in Figure 6.27.

Figure 6.27. Only one cross-section of each deformation is extracted by contour analysis (Left). Segmentation, through restoring thresholded deformation pattern information, generates segments that do not fit the deformations tightly enough with a region growing magnitude threshold of 0.05 (Center). Segmentation generates proper segments by restoring thresholded deformation pattern information, using a region growing magnitude threshold of 0.25 (Right).

As shown on the left of Figure 6.27, only a single cross-section of each of the deformations is extracted with the contour analysis technique. Using a region growing magnitude threshold of 0.05, the deformations are segmented, however a larger area than the actual deformations is extracted for each segment. This is because the deformation patterns of non-deformation areas also have similar curvature but with less magnitude. When a region growing magnitude threshold of 0.25 is used, the deformations are segmented with more accuracy. In both results, regardless of the region growing magnitude threshold, the CPU panel deformations are correctly classified as dents. In real world data, the deformation patterns are less predictable. However this algorithm can still segment the dents on the panel even if the contour analysis thresholding proves inadequate. This shows the quality of both the segmentation algorithm and the contour analysis technique in acquiring accurate analysis of the cross-sections of the mesh.

The segmentation is applied on the curved mesh with large deformation with non-optimal contour analysis thresholding results. The results are shown in Figure 6.28.

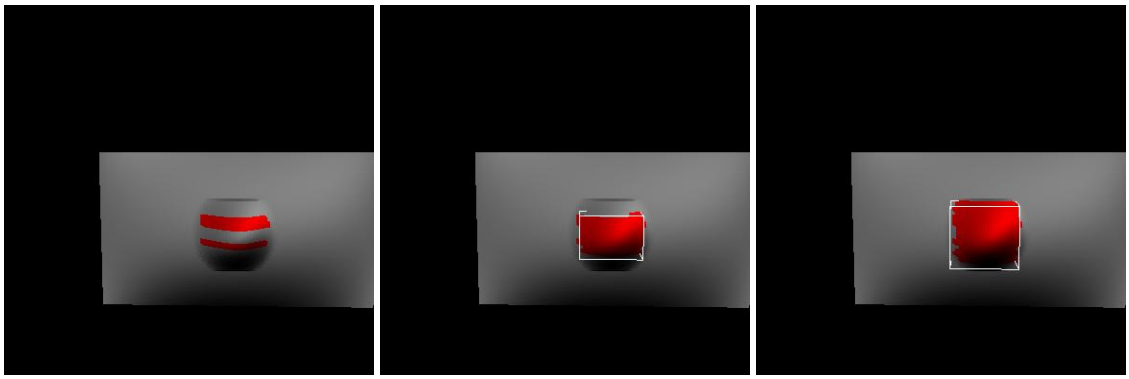

Figure 6.28. Several disjoint cross-sections are extracted by contour analysis (Left). Segmentation, through restoring thresholded deformation pattern information, generates a correct segment using region growing magnitude threshold of 0.25 (Center). Segmentation, through restoring thresholded deformation pattern information, generates a correct segment using region growing magnitude threshold of 0.05, and includes more of the deformation (Center).

The contour analysis results shown in Figure 6.28 exhibit several cross-sections of the ding being extracted. The segmentation algorithm, using a region growing magnitude threshold of 0.25, is able to segment the deformation including the disjoint cross-sections, instead of creating a new segment for each of them. When the threshold of 0.25 is decreased to 0.05 a segment of a larger area of the deformation is extracted because more adjacent deformation patterns are allowed to be part of the pattern. Regardless of the threshold selection, the deformation is correctly classified as a ding.

To make sure such an algorithm can be applied even with optimal thresholding cases, it is applied on the high resolution unfiltered car door mesh with optimal contour analysis results. The results are shown in Figure 6.29.

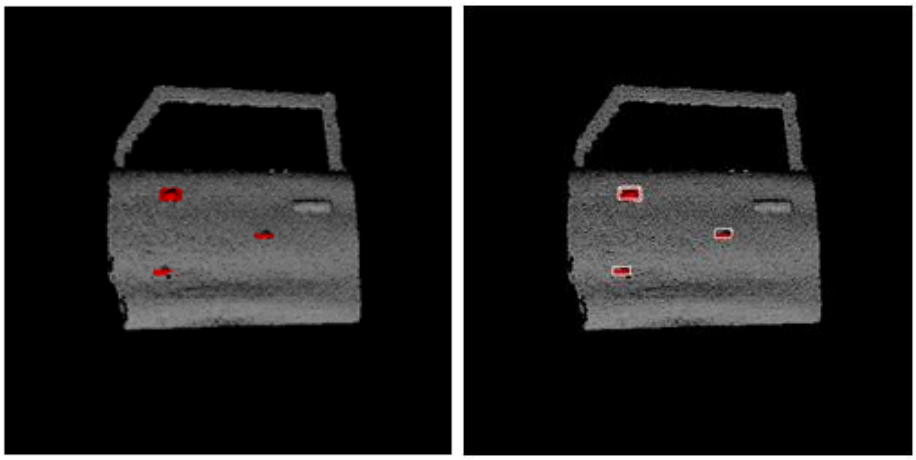

Figure 6.29. Properly extracted deformations by contour analysis (Left). Segmentation generates correct segments using region growing magnitude threshold of 0.25 (Right).

Using the region growing magnitude threshold of 0.25, the deformations are still segmented properly without segmenting any additional areas and without the deformations being broken up into multiple segments. They are all classified as dings. These results show that, though the segmentation using all deformation patterns is designed for non-ideal conditions, it can be used without detriment even in ideal situations where the contour analysis thresholding works well.

Though an additional parameter must be set, the latter does not need to be fine tuned for different scenarios. A simple assessment of the tolerable range can be done with very predictable results upon threshold selection. On real world models, it is better to be conservative with the threshold and to not allow very small magnitude deformation patterns to be added, as they may be in abundance due to surface variation and noise, resulting in a segment that does not fit the deformation tightly. Also, since the principle in classifying deformation type is the same as the segmentation in section 6.3.3.1, all deformations in all meshes are correctly classified as the appropriate deformation type, providing this segmentation with a 100% accuracy in classification. Overall, this segmentation demonstrates to be an effective solution when non-optimal thresholding is applied. It performs as well as any of the other discussed segmentation methods under ideal circumstances, and outperforms them in non-ideal circumstances.

6.4 Classification

Classification is the final phase of the deformation detection process to determine whether the segments are dings or dents. It also provides abilities to ensure that the extracted segments are indeed deformations of interest, though this is not its primary function. Ideally, the previous steps of surface shape analysis and segmentation have already removed non-deformation areas, but if they did not, the classification phase provides some last resort tools to remove those areas. Complex classification methods, such as using a neural network technique as proposed by Döring *et al.* [3] or spin image signatures as attempted by Assfalg *et al.* [52], could be used in this module. However this thesis focuses on simpler solutions as accurate results have already been obtained by the surface shape analysis and segmentation components.

6.4.1 Deformation Type Classification

The primary goal of the classification component is to determine whether each segment is a ding or a dent. If the contour analysis technique is used, the classification of the segments

has already been done during the segmentation phase. If octree-based techniques are used, then the classification step is the only way to determine whether the segment is a ding or a dent. That is one of the requirements of the overall system. Ideally, each segment will contain the entirety of a deformation with minimal non-deformation areas surrounding it, therefore accurate information can be obtained about the surface shape. Also, the deformations are assumed to be simple dings and simple dents, where all of the points in the deformation will either be protruding from the surface or be sunk into the surface. More complex deformation types would require a more sophisticated classification system and is beyond the scope of this thesis.

To obtain shape characteristics of the segments, a basic understanding of the orientation of the segment must be determined. A least-squares plane-of-best-fit fitted to the points contained in a segment can be used to determine the orientation of the shape represented by that segment. There are two ways to fit the plane to the points in a segment. The first way is to use all of the points in the segment, which provides plenty of data to generate an accurate plane-of-best-fit. But this can also allow uneven point distribution and significant surface variation to negatively affect the plane fitting. The second way uses only boundary points. Since they are on the outside edges of the segment, they would more likely belong to the regular surface of the automotive panel than to the deformation. This results in a plane fitted to the surface of the automotive panel, and accurately determines the orientation of the shape contained in the segment. However, in general there are not many points along the boundaries. As a result, the plane fitting may not make use of optimal information. If the feature extraction detects the entire deformation, a plane-of-best-fit using only the boundary points provides an accurate orientation. Also, the plane-of-best-fit using only the boundary points provides an approximation of the ideal surface such that the direction of the surface variation can be found, aiding in determining whether the surface contained inside the segment is in the shape of a dent or a ding.

The plane-of-best-fit could face in one of two directions, with the each of the normal vectors being opposite to each other. To ensure that the plane-of-best-fit is facing in the same direction as the mesh, a similar technique to 3D hidden surface removal is used. 3D hidden surface removal, in video games and graphics rendering, is used to avoid rendering polygons that are not facing the camera [60]. In much the same way, the average surface normal of the segment is compared to the normal vector of the plane-of-best-fit by computing a dot product between the two vectors. If the dot product is positive, the plane-of-best-fit is facing the same way as the average surface normal and no change needs to be made. If the dot product is negative, the normal vector of the plane-of-best-fit must be flipped by multiplying it by -1. The

orientation of this normal vector is important, and serves as a basis to determine the direction of the surface variation in order to determine whether the surface variation direction corresponds to a dent or a ding. The final least-squares plane-of-best-fit equation is in the form of:

$$ax + by + cz + d = 0 \tag{6.5}$$

With the orientation of the deformation determined, as *(a, b, c)* is a normal to the plane, the plane-of-best-fit represents the ideal surface had the deformation not been present. Several descriptors are proposed to determine whether the surface contained in the segment is a dent or ding.

The first descriptor, the point-count descriptor, uses the quantity of points that share a similar positional relationship to the plane-of-best-fit in estimating the direction of variation of the surface contained in the segment. If most of the points contained in the segment are above the plane-of-best-fit, in the direction of the normal vector, the deformation is classified as a ding. If most of the points are below the plane-of-best-fit, in the direction of the normal vector, the deformation is classified as a dent. In any classification, a certainty measure is important. The percentage of the points that are above the plane-of-best-fit, in the case of a ding, or below the plane-of-best-fit, in the case of a dent, is the certainty measure in the classification. This way, if there are a similar amount of points that are above and below the plane-of-best-fit, the certainty measure is closer to 50%, indicating uncertainty.

The second descriptor, the magnitude descriptor, uses the maximum distance from the plane-of-best-fit as a descriptor of whether the segment is a ding or a dent. The distance of each point from the plane-of-best-fit is determined, representing how far each point is displaced from the estimated ideal surface. The distance can be calculated using Eq. 6.6.

$$d = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}} \tag{6.6}$$

$$d_{abs} = |d| \tag{6.7}$$

where d is distance from the point (x_0, y_0, z_0) to the plane-of-best-fit in the form defined by Eq. 6.5, and d_{abs} is the magnitude corresponding to that point. The d value corresponding to the largest magnitude determines whether the segment contains a ding or a dent. If the corresponding d value is less than 0 it is a dent, and if it is greater than 0 it is a ding. The certainty measure for this metric is shown in Eq. 6.8.

$$cm = \frac{m}{d_{largest} - d_{smallest}} \tag{6.8}$$

where cm is the certainty measure, $d_{largest}$ is the largest d value in the segment, $d_{smallest}$ is the smallest d value in the segment, and m is the calculated magnitude of the deformation. Note that m is equal to $d_{largest}$ if the deformation is classified as a ding, and equal to $d_{smallest}$ if it is

classified as a dent. This certainty measure will be 50% if there are points that are equally far from each other on either side of the plane-of-best-fit, meaning it is hard to determine based on the furthest point from the plane-of-best-fit whether it is a ding or a dent. Note that the certainty measure will never be less than 50% in distinguishing between dings and dents, because if it is, the certainty measure of the opposite classification will be higher, resulting in that classification being selected.

In an ideal situation, both of these descriptors would produce equally effective results. However, the limitations of the system require a decision on which descriptor should be used and must be based on certain factors. If there is high accuracy with the feature extraction and segmentation, such that each segment is composed of a deformation with only minimal non-deformation surface area, then the point-count descriptor would be very accurate. It would be accurate because the majority of the points would describe the deformation, and they would all likely fall above or below the plane-of-best-fit. It is also robust to outliers, in that they will not affect the classification significantly because they will only be a few points compared to the entire segment. If the feature extraction and segmentation are not reliable and may extract significant amounts of non-deformation areas along with the deformation, then the magnitude descriptor may be useful. It will look for the largest displacement from the ideal surface, and use that as the main way of describing the deformation type. However, this is heavily dependent on a single point, and may not always be reliable as a classifier. The performance of these descriptors, in the context of this thesis, is analyzed in section 6.4.3.

6.4.2 Additional Classification

Though it is not the primary goal of the classification stage, the classification stage is an appropriate place to allow fine-tuning of certain parameters by the operator as a final effort to ensure that only deformations of interest are outputted as segments. In the process of analyzing descriptors to determine whether a segment is a ding or a dent, segments that do not meet the characteristics of either can be removed. Certain well-known characteristics, which can slip past the surface shape analysis and segmentation phases, can be taken into account to remove possible non-deformation areas that remain. A set of descriptors can be extracted from the segments outputted by the segmentation phase, and operator-defined parameters can select which segments are deformations and which are not based on these descriptors. Unlike the deformation type classification, this additional classification may be required to be applied on contour analysis results, since the contour analysis thresholding may not have been successful in removing all non-deformations.

The number of voxels contained in a segment can be a good descriptor, and can be useful in determining whether a segment is an actual deformation or not. From the octree-based surface shape analysis techniques, discussed in chapter 4, even after segmentation, noise and acquisition artifacts remain as segments. These are often found to be composed of a few voxels. The elimination of such segments with few voxels can eliminate those unwanted areas. The exact quantity of voxels describing noise and acquisition artifacts are variable, therefore a threshold must be set that does not interfere with actual deformations, but removes small non-deformation areas.

A segment is composed of a set of voxels, and each of those voxels contain a set of triangles that belongs to the mesh. The combined surface area of the mesh contained in those voxels can also be a good descriptor. Thresholding surface area can be effective in removing noise and acquisition artifacts, as those extracted segments will contain a very small surface area. Thresholding surface area can also be effective in removing large surface features that are not deformations, such as door handles or aesthetic curves, as those objects have relatively large surface areas, while deformations are small in surface area relative to them.

After the orientation of the shape contained in the segment is determined and the plane-of-best-fit provides the shape its own local coordinate system, characteristics such as the deformation size in the x, y, and z directions can be measured relative to the shape's local coordinate system. The difference between the furthest points along those directions provides an estimate of the shape's length, width, and height. The magnitude, which is calculated for the magnitude descriptor in section 6.4.1, can be used to determine the magnitude of the deformation, similar to the magnitude characteristic of a deformation pattern in the contour analysis technique. Noise typically has a small magnitude, while features like door handles have a larger magnitude relative to the deformations of interest.

For multi-resolution segments, the number of resolutions that a segment is detected through can be another good descriptor of whether the segment is a transient feature or an actual strong feature in the mesh. The similarity between resolutions, judged by the output of the matching metrics described in section 6.2.1 and section 6.2.2, can also be a useful descriptor in determining how strong a feature is over multiple resolutions.

The combination of these descriptors can give a lot of information about shapes contained in the extracted segments, but it is heavily dependent on tuning parameters. By thresholding these descriptors, the classification module can ensure certain shapes are not classified as deformations while others are. The accuracy of this output will be strongly determined by which descriptors are used, and which ranges are selected for those descriptors.

6.4.3 Classification Results

Both the deformation type classification and the additional classification options are tested in section 6.4.3.1 and 6.4.3.2, respectively.

6.4.3.1 Deformation Type Classification Results

To test the classification technique's ability to determine whether a deformation is a ding or a dent, it is applied on deformation segments of each mesh using the point-count descriptor and the magnitude descriptor, and the results are compared. The magnitude descriptor is also used to compare the estimated magnitude of the deformation to the actual magnitude.

The extraction and segmentation results are presented in Figure 6.30, while the resulting classification results are presented in Table 6.3. Note that the extraction and segmentation results shown here are based on the octree method, as the contour analysis technique and segmentation inherently classifies the deformations and has already been tested in the previous chapter. Also, the octree-based extraction and segmentation uses optimal thresholds to test the classification system.

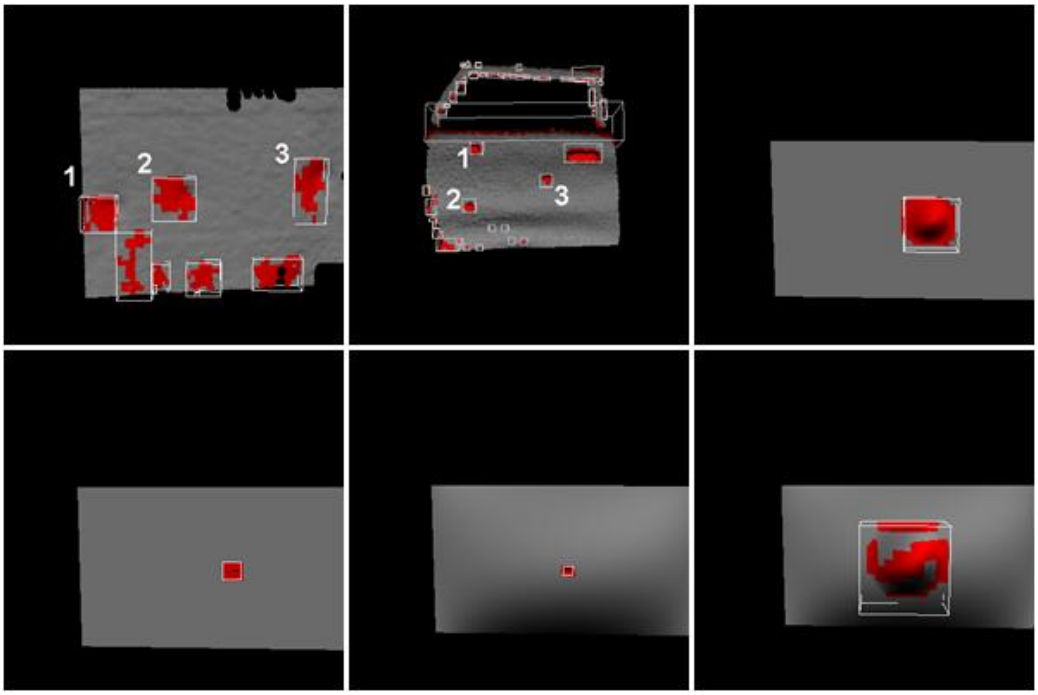

Figure 6.30. Using octree-based feature extraction and single-resolution segmentation: CPU panel mesh with dent segments labelled (Top Left). Car door mesh with ding segments labelled (Top Middle). Flat mesh with large ding segmented (Top Right). Flat mesh with small dent segmented (Bottom Left). Curved mesh with small dent segmented (Bottom Center). Curved mesh with large ding segmented (Bottom Right).

Model/ Deformation	Actual		Point-Count Descriptor Estimates		Magnitude Descriptor Estimates		
	Type	*Magnitude (mu)*	*Type*	*Certainty*	*Type*	*Certainty*	*Magnitude (mu)*
Car Door							
Def 1	Ding	18-20	Ding	0.636	Ding	0.753	17.13
Def 2	Ding	10-13	Ding	0.644	Ding	0.655	8.87
Def 3	Ding	10-13	Ding	0.609	Dent	0.514	8.40
CPU Panel							
Def 1	Dent	2	Ding	0.502	Ding	0.568	6.06
Def 2	Dent	2	Dent	0.511	Ding	0.562	3.83
Def 3	Dent	2	Dent	0.546	Dent	0.506	6.13
Flat Mesh							
Small	Dent	0.91	Dent	0.583	Dent	0.997	0.91
Large	Ding	9.50	Ding	0.896	Ding	0.989	9.07
Curved Mesh							
Small	Dent	1.50	Dent	0.667	Dent	0.946	1.39
Large	Ding	18-22	Ding	0.558	Ding	0.694	19.0

Table 6.3. Comparison of actual deformation characteristics and the results of classification using the point-based descriptor and the magnitude descriptor on octree-based feature extraction and segmentation results.

Over the artificial flat and curved meshes, it can be seen the classification results are very accurate. Dings and dents are classified in the proper category, using both descriptors. Also, the magnitudes estimated by the magnitude descriptor are very close to the actual measured values. These results show that the classification can behave well on artificial models, corresponding to an acquisition system with minimal noise and acquisition artifacts. The certainty measures of the magnitude descriptor are generally high, as there are not many outliers in artificial meshes and the plane-of-best-fit is relatively accurate as there is no excessive surface curvature. This indicates that the classification is quite reliable. The certainty levels of the point-based descriptor are lower because some of the results in Figure 6.30 for artificial meshes show non-optimal segments, notably the flat mesh with small deformation and the curved mesh with large deformation. The segmentation of the flat mesh with small deformation is non-optimal because it results in a segment that contains the deformation as well as a large amount of surrounding non-deformation areas. The segmentation of the curved mesh with large deformation is non-optimal because it results in a segment where pieces of the

deformation are missing. Since the dual criteria of maximizing the deformation while minimizing non-deformation areas in the segment were not sufficiently met, they can greatly influence the point-count descriptor, as there are more non-deformation points to be used for measurement.

For the real world meshes of the CPU panel and car door, the results are less accurate overall. The descriptors accurately classify each of the dings on the car door except for one, where the magnitude descriptor incorrectly classifies one of the dings as a dent. This is because the segment does not wholly contain the deformation. In the feature extraction of the car door, some of the boundary nodes in segment 3 are part of the deformation itself, which results in an ineffectively calculated plane-of-best-fit. The certainty measure reflects the inaccuracy of the classification by being significantly lower than the other certainty levels. The CPU panel provides poor results because of the poor feature extraction. It can be seen in Figure 6.30, in the results of the feature extraction on the CPU panel, that the segments do not contain the entire deformation, and often contain just half of the deformation grouped along with large amounts of non-deformation areas. Because of this unreliable extraction of the deformation, the classification cannot obtain accurate results. This consistent inaccuracy over both descriptors is also indicated in the certainty measures by being just slightly over 50%, meaning that the classification technique is only slightly more certain that the deformation is one type instead of the other type.

The magnitude descriptor provides an additional parameter which gives the estimated magnitude of the deformation. Estimated magnitudes of the deformations in the artificial models are quite accurate. The deformations in the real world models of the car door and the CPU panel are estimated with less precision. The disparity in the accuracies of estimated magnitudes between the real world models and the artificial models are due to complex characteristics in the real world models. Due to noise and surface variation that is only marginally less curved than the deformations, as well as segmentation results that do not accurately isolate the deformations of interest, the descriptor's plane-of-best-fit calculation is inaccurate in both orientation and position. This results in the magnitude calculation not being representative of the actual deformation characteristics. The results show a potential weakness of the magnitude descriptor for deformation type classification, as it performs well in its classification but is not robust in determining the actual magnitude of the deformation.

Overall, it can be seen that the point-count and magnitude descriptors provide similarly accurate classification results when the prior feature extraction and segmentation results are accurate. As the criterion of maximizing deformation area and minimizing non-deformation area in the segmentation is more closely met, the descriptors provide more accurate results with

higher certainty values. When the feature extraction and segmentation are inaccurate, the poor fitting of the plane-of-best-fit serves as the basis for the error more than the descriptors themselves and the certainty measures show the inaccuracy.

The results here can be compared to the results from Table 6.2 in section 6.3.3.2 to evaluate the respective techniques' performance in classifying the deformations. It can be seen that the octree-based feature extraction and segmentation, even when using optimal thresholds, often extract and segment non-deformation areas as well as deformations of interest. Also, the inaccurate extraction of the complete deformation, especially on the CPU panel mesh, can cause the classification to provide poor results. With the contour analysis results, the deformations are isolated clearly with no other non-deformation areas being segmented, and the deformations in the test results were all accurately classified. This comparative analysis provides strong evidence that the contour analysis is capable of much more accurate results than the octree-based techniques for the criteria of this application.

6.4.3.2 Additional Classification Results

As demonstrated in the previous sections, there are situations where the deformations cannot be isolated using only the techniques for feature extraction and segmentation. Though the classification phase is intended to separate dings from dents, it is also an opportune area to determine characteristics of those segments. Those characteristics can aid in separating deformations from non-deformations. Some cases of poorly thresholded feature extraction resulting in non-deformation areas being created into segments are used to test the classification system's ability to distinguish between deformations and non-deformations. Note that the ranges for the parameters selected are non-optimal, and are not set based on the exact known sizes of the deformations present in the mesh. They are selected to be in a general range of the deformations present in the mesh to show some robustness.

First, the additional classification is applied to some non-optimal contour analysis results that have been segmented. With contour analysis, the most likely non-deformation areas that are segmented are residual deformations due to imperfect curve-fitting. An example is shown using the curved mesh containing a large deformation in Figure 6.31. By thresholding certain attributes of the segments, those that do not contain the deformation of interest can be removed. The result is also shown in Figure 6.31.

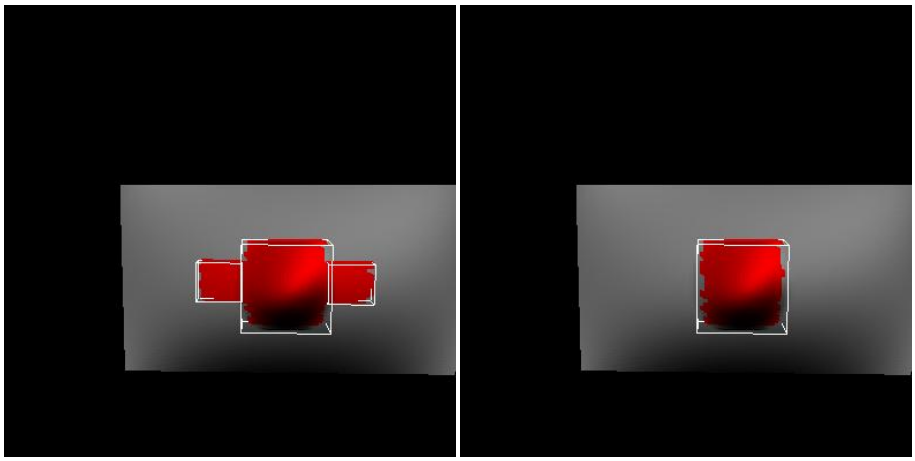

Figure 6.31. Deformation and non-deformation areas are segmented on the curved mesh with large deformation (Left). Additional classification removes the non-deformation areas (Right).

By removing segments containing surface area less than 2000 mesh units squared, magnitude less than 7 mesh units, or a deformation type classification certainty value lower than 70% will allow the residual deformation segments to be removed. The deformation type classification is not usually run on contour analysis results because they are already classified at an earlier stage, however the certainty value from the classification can provide insight into the likelihood that a segment contains a deformation.

Another example containing a residual deformation segment is with the CPU panel mesh shown in Figure 6.32. The results after using additional parameters to classify the segments as deformations or non-deformation areas are also shown.

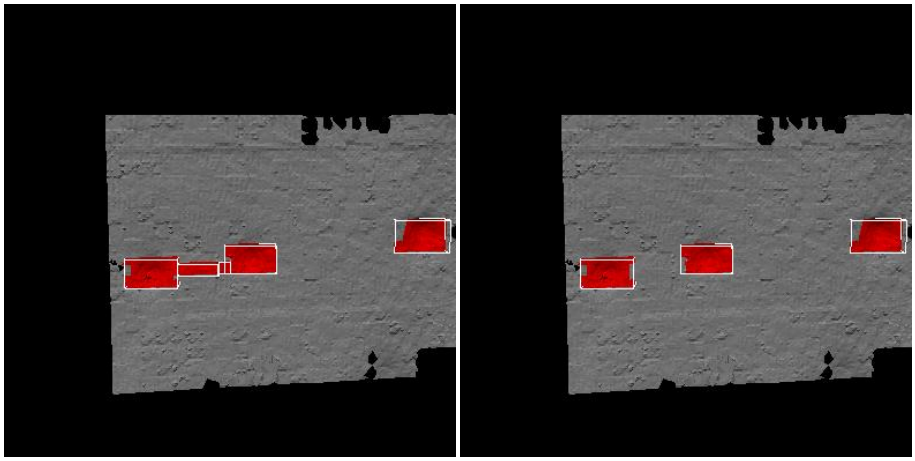

Figure 6.32. Deformation and non-deformation areas are segmented on the CPU panel mesh (Left). Additional classification removes the non-deformation areas (Right).

The non-deformation areas are clearly removed by evaluating the attributes of the surface and the deformation type classification certainty measure. By removing segments with surface area less than 500 mesh units squared or by removing segments with a deformation type classification certainty measure of less than 60%, the non-deformation segments are removed.

The octree-based feature extraction technique detects non-deformation areas quite frequently in the tested real world models. Sometimes the segmentation cannot do much to remove those areas, but the additional classification provides some solutions. Some non-optimal feature extraction results on the CPU panel mesh are shown in Figure 6.33, along with the results after the classification removes non-deformation segments.

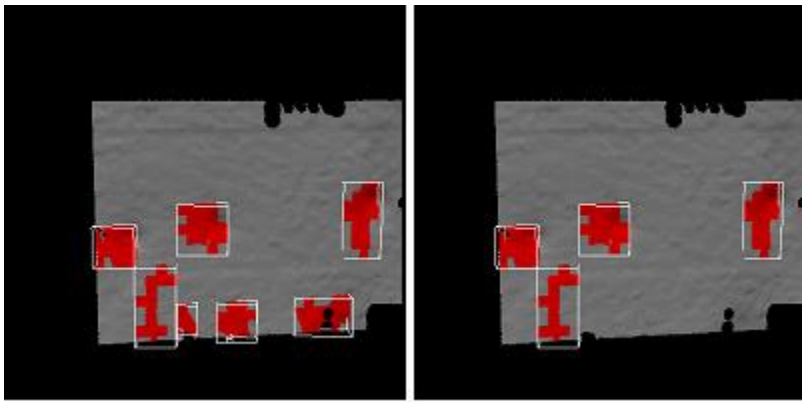

Figure 6.33. Deformation and non-deformation areas are segmented on the CPU panel mesh (Left). Additional classification removes the non-deformation areas (Right).

In this case, segments with surface area less than 90 mesh units squared or deformations less than a magnitude of 3 mesh units are removed. Of course, if the exact parameters corresponding to the attributes of the deformations are selected, all non-deformation areas can be removed. However, as stated above, to show robustness the parameters are set to be in a wider range. Even with these criteria, not all non-deformation segments are removed. The deformations of interest remain, but a surface variation near the bottom also remains. More complex classification methods are required to deal with this.

The octree-based feature extraction, applied on the car door mesh, can extract deformations of interest, but it also extracts noise, surface variation, and acquisition artifacts. The segmentation creates segments for those non-deformation areas. The classification must then be able to remove them. The segmentation results and the results of the additional classification to remove non-deformation areas are shown in Figure 6.34.

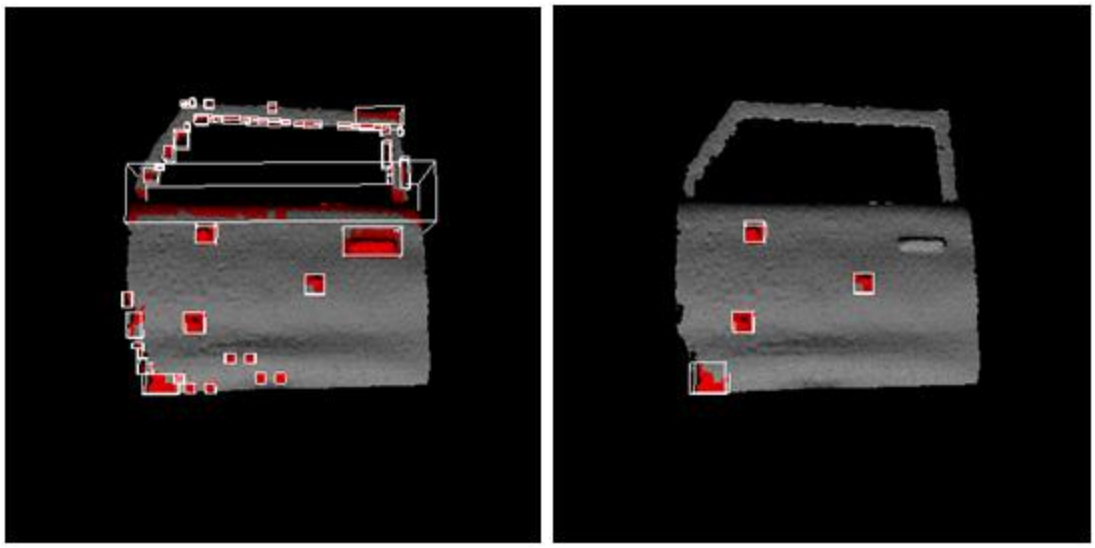

Figure 6.34. Deformation and non-deformation areas are segmented on the car door mesh (Left). Additional classification removes the non-deformation areas (Right).

By setting the parameters of the additional classification to remove magnitudes less than 8.5 mesh units or greater than 17 mesh units, most of the broad curvatures, the door handle, and the small noisy areas are removed. Also, setting the parameters to remove segments with surface area less than 2800 mesh units squared or segments with surface area greater than 5200 mesh units squared results in further non-deformation areas being removed. Even with those aforementioned settings, the additional classification still cannot identify and remove all of the non-deformation areas, as some portion of the door in the lower left corner is still classified as a deformation.

The additional classification provides some tools to serve as a last resort in removing non-deformation areas in situations where the feature extraction and segmentation are inaccurate. The results show that it is possible to remove many of the non-deformation areas, in both artificial meshes and real world meshes, however the results are not always perfect. Even though the thresholds were selected to not specifically isolate the deformations, a certain amount of robustness is still lost with such a technique. If the operator is sure of the characteristics of the deformations likely to be present, those characteristics can be used to remove non-deformation areas, but the parameters cannot be easily generalized to panels and deformations of significantly different qualities. This section shows that it is possible to remove non-deformation areas, but not all non-deformation areas can be removed with certainty nor is the technique particularly robust as the operator must carefully select the parameters.

6.5 Performance Analysis and Recommendations

By comparing the recommended systems of surface shape analysis and segmentation, a recommendation can be made on which system is best for automated deformation detection. The final recommendation relies on the overall computation time required to process one entire scan, given the requirement for real-time operation in factory settings, as well as on the occurrence of false positives and false negatives in the detection of deformations.

To evaluate computation time, the deformation detection systems, composed of feature extraction and segmentation, are applied on several test meshes with optimal parameters. To eliminate unnecessary processing, the aggregate standard deviation enhancement of the octree-based technique is applied directly on a voxel representation with spatial characteristics of the 7^{th} octree level without generating the entire octree. The contour analysis technique, with the weighted least-squares enhancement, is also applied on a voxel representation with spatial characteristics of the 7^{th} octree level without generating the entire octree. It is important to note that none of the techniques are optimized for speed, and the contour analysis is not optimized for the parallel processing it is capable of. Regardless, the computation times give an indication of their relative speeds. Computation times are calculated through execution on an Intel Pentium 4 dual-core 2.0GHz machine with 4GB of RAM and are shown in Table 6.4.

Model	Contour Analysis with Segmentation using Thresholded Deformation Patterns	Octree-based Feature Extraction using Incremental Thresholding with Single Resolution Segmentation and Deformation Type Classification	Octree Aggregate Standard Deviation Enhancement with Single Resolution Segmentation and Deformation Type Classification
Car Door	7.8s	3.5s	0.97s
CPU Panel	7.3s	2.1s	0.12s
Flat Mesh with Small Deformation	2.1s	0.29s	0.01s
Flat Mesh with Large Deformation	2.1s	0.50s	0.22s
Curved Mesh with Small Deformation	2.1s	0.32s	0.01s
Curved Mesh with Large Deformation	2.1s	0.52s	0.21s

Table 6.4. Comparison of computation times between the proposed deformation detection systems.

The table shows that the system using the aggregate standard deviation enhancement of the octree technique executes in the shortest amount of time overall. The system using the octree-based technique with incremental thresholding takes slightly longer, but is still two to four times faster than the contour analysis system. It is expected that with optimization for speed and parallelization, the execution times of the contour analysis system and the octree-based systems can be improved by a magnitude of two or three.

For this application, false positives are much less important than false negatives. This is due to the fact that an inspector will still evaluate the marked locations to determine how to fix them. If the inspector sees no deformation, it will not be fixed even if the area is marked. But if deformations are missed, the inspector will not get a chance to even evaluate those areas in order to fix them. All of the proposed systems, when applied on the test meshes, do not generate any false negatives, meaning that all deformations are detected. In that sense, they are all equally satisfactory, however to distinguish one system over the other, false positive rate can be evaluated. A comparison of false positive detections is presented in Table 6.5. Note

that the additional classification to remove non-deformation segments is not considered part of either system, as its robustness is very limited and it requires an excessive amount of custom parameters specifically set for each mesh.

Model	Contour Analysis with Segmentation using Thresholded Deformation Patterns	Octree-based Feature Extraction using Incremental Thresholding with Single Resolution Segmentation and Deformation Type Classification	Octree Aggregate Standard Deviation Enhancement with Single Resolution Segmentation and Deformation Type Classification
Car Door	0	45	25
CPU Panel	0	0	2
Flat Mesh with Small Deformation	0	0	0
Flat Mesh with Large Deformation	0	0	0
Curved Mesh with Small Deformation	0	0	0
Curved Mesh with Large Deformation	0	0	0

Table 6.5. Comparison of false positive detections between the proposed deformation detection systems.

The table indicates that there are no false positive detections over the artificial meshes. In the real world mesh results, the contour analysis system produces no false detections while the octree-based systems detect quite a few non-deformation areas. The contour analysis system also inherently classifies the deformation type with a very high accuracy, as shown in Table 6.2, where the octree-based system requires the additional classification step that is significantly less accurate, as shown in Table 6.3.

Though computation times are important, it is expected that the amount of time that the system has available to estimate the deformation locations are significantly longer than any of the measured computation times, that is on the order of a few minutes. This is especially true if the robotic marking station is located at some distance from the panels' range scanning station

along the assembly line. Based on its detection rates, classification accuracy, as well as the simplicity and intuitiveness of the parameter selection, the contour analysis technique with segmentation using thresholded deformation patterns is therefore the recommended solution for automated deformation detection on automotive surface panels.

6.6 Conclusion

Several segmentation techniques were reviewed in this chapter specific to each feature extraction method, considered in both chapter 4 and chapter 5. For the octree-based surface shape analysis, the single resolution segmentation and multi-resolution segmentation techniques were proposed. The single resolution segmentation, which is an extension of blob extraction, provided reasonable results, applied on the octree-based feature extraction results, without much computation time. Multi-resolution segmentation was able to remove a lot of transient features and non-deformation segments automatically, providing much better results than the single resolution segmentation, but requires additional parameters to be set and a longer amount of computation time. Based on this comparison, the single-resolution segmentation is recommended for both the octree-based surface shape analysis technique and its aggregate standard deviation enhancement due to it not requiring any additional parameters.

For the contour analysis surface shape analysis, the following techniques were proposed: segmentation without deformation pattern information, segmentation using only thresholded deformation patterns, and segmentation using all deformation patterns. Segmentation using thresholded deformation patterns provided the best results while providing tolerance to non-optimal contour analysis thresholding. Also, it requires no parameters to be set. Segmentation using all deformation patterns requires additional parameters to be set, however it provides added robustness to non-optimal contour analysis thresholding. Based on the models used and the expected complexity of industrial automotive panel scans, segmentation using thresholded deformation patterns for contour analysis feature extraction is recommended due to its independence from additional operator-set parameters and its robustness to non-optimal contour analysis thresholding.

Finally, after describing the segmentation and classification components, the comparison of the entire octree-based deformation detection system and entire contour analysis deformation detection system was performed. Based on detection rates, classification accuracy, and its basic parameters, the contour analysis technique with segmentation using thresholded deformation patterns was recommended as the best solution for automated deformation detection on automotive surface panels.

Chapter 7 Conclusion

7.1 Summary

In this thesis, a system is proposed that is able to meet the requirements of an automated deformation detection system for use in the context of an automotive panel quality control station over an assembly line. The requirements in place are that such a system must be able to detect deformations of interest, using 3D analysis, without knowledge of the ideal surface and without any CAD model. The deformations must also be classified as dings or dents. Assuming that the operator has knowledge about the approximate size and scale of these deformations, the system can use this additional information to aid in isolating deformations from design features that typically exist over an automotive panel.

A variety of techniques were reviewed for the deformation detection pipeline. An octree-based technique and contour analysis techniques were designed for the surface shape analysis component. Segmentation methods specific to each surface shape analysis were also proposed to refine the location of deformations. Finally, a classification approach was introduced and a complete experimental evaluation was performed on every stage of the proposed inspection station. Given the recommendations made about which techniques should be selected, it is demonstrated that the proposed system, that is the contour analysis using weighted least-squares and recursion enhancements with segmentation using thresholded deformation patterns, is effective in identifying the locations of deformations of interest, and classifying them as dents or dings when presented with a mesh of a surface. Furthermore, its limited set of two intuitive parameters, r and the deformation magnitude threshold, allow an operator to easily tune the system to extract the deformations of preferred size and scale, without dealing with more complex parameters from other non-custom feature extraction techniques.

Experiments were conducted on both artificial and real-world test data, offering a set of meshes encompassing characteristics of varied situations. Experiments demonstrated that the system can be used in both ideal circumstances, such as finding a large deformation over a flat, noiseless mesh, as well as in more complex circumstances, such as finding a small deformation over a noisy, continually varying surface, with acquisition artifacts and holes. The experimental results show that the proposed system is effective and robust to meshes with noise and acquisition artifacts, along with non-ideal surfaces containing shape variation other than the deformations of interest, making it a suitable candidate for a complete deformation detection system in a quality control station for automotive panels.

7.2 Contributions

This thesis provides many contributions in the field of 3D surface deformation detection. The first set of contributions of this thesis are several novel enhancements detailed in chapter 4, using the octree-based technique proposed by Woo *et al. [40]* as a foundation. By means of triangle-based analysis, non-uniform weighting of triangles for calculating the standard deviation of surface normals, and incremental thresholding, the original technique was improved for the extraction of deformations. An additional contribution is the proposed aggregate standard deviation method, which extracts features using a proposed s-metric to represent the variation of a surface over consecutive resolutions of the octree after the octree is fully generated.

An important contribution of this thesis, which is also the recommended method amongst the presented options, is the contour analysis procedure, described in chapter 5. By combining and improving existing mathematical surface approximation concepts, range image-based geometric analysis techniques, and multi-scale feature extraction, the contour analysis procedure was developed to separate the mesh into cross-sections and to characterize and locate deformations on those cross-sections. Where range image-based geometric analysis is limited to having the data in the form of a 2.5D image, contour analysis uses the octree data structure to extend such an analysis for data that is in the form of an unordered point cloud or mesh. The improvement over standard surface approximation resides in the use of weighted least-squares fitting, and the use of s-values, proposed in chapter 4, as weights to indicate the reliability of areas for approximating a surface. To set itself apart from other surface approximation-based techniques, contour analysis allows for selective multi-resolution feature extraction through a proposed enhancement using recursion. The procedure's built-in ability to determine whether the extracted feature is a dent or a ding is also useful in satisfying one of the initially presented requirements on the classification of the deformations.

Finally, chapter 6 also contains some notable contributions through multi-resolution segmentation and contour analysis segmentation. Relating to an octree-based method, the multi-resolution segmentation innovatively extends the idea of feature persistence to combining single-resolution segments, in a voxel representation, over consecutive resolutions to determine multi-resolution segments. A matching metric is also proposed for this technique to facilitate the identification of segments found in multiple resolutions of an octree. Contour analysis segmentation borrows from region growing and blob extraction techniques, but benefits from using additional information in creating the segments. Using additional information such as the

magnitude of extracted deformation pieces, or the knowledge that they are a ding or a dent, aids in connecting the correct areas to determine a segment.

Some parts of the work in this thesis were published in [59] and [5].

7.3 Future Work

Though the deformation detection system produces favourable results over both artificial meshes and real-world meshes, some future work can still be done to refine the technique.

Some work was performed to achieve proper classification of dings and dents, however more sophisticated techniques can be explored to provide better results in separating deformations from non-deformation features while still avoiding the use of a CAD model. Refinements can contribute to the reduction of the number of operator-set parameters, and provide the ability to accurately separate deformation features from non-deformation features using knowledge from training data similar to Döring *et al.* [3]. The lack of appropriate training data, and even test data for the overall system, limited such a system from being tested and applied for this thesis.

Additional testing on many more meshes, both artificial and real-world, would be required to further refine the analysis of the effectiveness of this system. To fully prove the system's accuracy and robustness, it would need to be applied on more test cases, which would contain difficult situations, such as varying levels of noise, body curvature, and design features. Test data from a higher resolution scanner, which could acquire considerably smaller deformations, is suitable to validate the scalability of the approach, even though it was already demonstrated to be able to detect small deformations regardless of the unit size. Also, the use of meshes from different acquisition systems would be appropriate to show the success of the system without having to change any parameters. Though some real-world meshes were used and the artificial meshes attempted to imitate real-world scenarios, real automotive panels must be scanned to truly characterize how effective this software would be in an industrial environment.

References

[1] T. Lilienblum, P. Albrecht, R. Calow, and B. Michaelis, "Dent Detection in Car Bodies," *Proceedings of the 15th International Conference on Pattern Recognition (ICPR)*, vol. 4, pp. 775-778, Barcelona, Spain, 2000.

[2] T. S. Newman and A. K. Jain, "A System for 3D CAD-Based Inspection using Range Images," *Pattern Recognition*, vol. 28, pp. 1555-1574, 1995.

[3] C. Döring, A. Eichhorn, D. Girimonte, and R. Kruse, "Improving Surface Detection for Quality Assessment of Car Body Panels," *Mathware & Soft Computing*, vol. 11, pp. 163-177, 2004.

[4] H. Chen, "Automatic Dent Detection on Car Bodies," Masters Thesis, Hong Kong University of Science and Technology, 2008

[5] V. Borsu, A. Yogeswaran, and P. Payeur, "Automated Surface Deformations Detection and Marking on Automotive Body Panels," *Proceedings of the 6th IEEE International Conference on Automation Science and Engineering,* pp. 551-556, Toronto, Ontario, Canada, 2010.

[6] V. Borsu, "Pose and Motion Estimation of Parts Exhibiting Few Visual Features for Robotic Marking of Deformations," Masters Thesis, University of Ottawa, 2010

[7] F. Blais, J. Taylor, L. Cournoyer, M. Picard, L. Borgeat, G. Godin, J. A. Beraldin, M. Rioux, and C. Lahanier, "Ultra High-Resolution 3D Laser Color Imaging of Paintings: the Mona Lisa by Leonardo da Vinci," *Proceedings of the 7th International Conference on Lasers in the Conservation of Artworks,* pp. 435-440, Madrid, Spain, 2007.

[8] V. Sequeira, J. Goncalves, and M. Ribeiro, "3D Environment Modelling Using Laser Range Sensing," *Robotics and Automation*, vol. 16, pp. 81-91, 1995.

[9] S. Parthasarathy, J. Birk, and J. Dessimoz, "Laser Rangefinder for Robot Control and Inspection," *Proceedings of SPIE Robot Vision 1982*, vol. 336, pp. 2-11, 1982.

[10] J. Marszalec and R. Myllyla, "Shape Measurements Using Time-Of-Flight-Based Imaging Lidar," *Proceedings of the SPIE Conference on Three-Dimensional Imaging and Laser-based Systems for Metrology and Inspection III*, vol. 3204, pp. 14-15, 1997.

[11] S. B. Gokturk, H. Yalcin, and C. Bamji, "A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions," *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop*, vol. 3, pp. 35, Washington D.C., USA, 2004.

[12] O. Faugeras, *Three-Dimensional Computer Vision*: MIT Press, 1993.

[13] D. Murray and J. J. Little, "Using Real-Time Stereo Vision for Mobile Robot Navigation," *Autonomous Robots*, vol. 8, pp. 161-171, 2000.

[14] S. Se, D. G. Lowe, and J. Little, "Vision-Based Mobile Robot Locatization and Mapping Using Scale-Invariant Features," *Proceedings of the International Conference on Robotics and Automation 2001*, pp. 2051-2058, 2001.

[15] D. Murray and C. Jennings, "Stereo Vision Based Mapping and Navigation for Mobile Robots," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '97),* pp. 1694-1699, Albuquerque, New Mexico, USA, 1997.

[16] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno, "A Low Cost 3D Scanner Based on Structured Light," *Computer Graphics Forum (Eurographics 2001 Conf. Issue)*, vol. 20, pp. 299-308, 2001.

[17] L. Zhang, B. Curless, and S. M. Seitz, "Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming," *Proceedings of Int. Symposium on 3D Data Processing Visualization and Transmission,* Padova, Italy, 2002.

[18] P. Payeur and D. Desjardins, "Structured Light Stereoscopic Imaging with Dynamic Pseudo-random Patterns," *Proceedings of the International Conference on Image Analysis and Recognition (ICIAR 2009)*, vol. 5627, pp. 687-696, 2009.

[19] R. J. Woodham, "A Cooperative Algorithm for Determining Surface Orientation from a Single View," *Proceedings of IJCAI-79,* pp. 971-977, Tokyo, Japan, 1977.

[20] R. J. Woodham, "Photometric Method for Determining Surface Orientation from Mulitple Images," *Optical Engineering*, vol. 19, pp. 139-144, 1980.

[21] D. F. Watson, "Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes," *The Computer Journal*, vol. 24, pp. 167-172, 1981.

[22] A. Bowyer, "Computing Dirichlet Tessellations," *The Computer Journal*, vol. 24, pp. 162-166, 1981.

[23] D. T. Lee and B. J. Schachter, "Two Algorithms for Constructing the Delaunay Triangulation," *Int. J. Comput. Inf. Sci.*, vol. 9, pp. 219-242, 1980.

[24] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The Ball-Pivoting Algorithm for Surface Reconstruction," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, pp. 349-359, 1999.

[25] J. D. Foley, A. v. Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics Principles and Practice*: Addison-Wesley Publishing Co., 1990.

[26] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, vol. 21, pp. 163-169, 1987.

[27] R. Shu, Z. Chen, and M. S. Kankanhalli, "Adaptive Marching Cubes," *The Visual Computer*, vol. 11, pp. 202-217, 1995.

[28] B. Mederos, L. Velho, and L. H. D. Figueiredo, "Moving Least Squares Multiresolution Surface Approximation," *Proceedings of SIBGRAPI 2003 - XVI Brazilian Symposium on Computer Graphics and Image Processing*, pp. 19, 2003.

[29] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva, "Computing and Rendering Point Set Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, pp. 3-15, 2003.

[30] N. Amenta, M. Bern, and M. Kamvysselis, "A New Voronoi-Based Surface Reconstruction Algorithm," *Proceedings of ACM SIGGRAPH 1998*, pp. 415-421, 1998.

[31] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*: Prentice Hall, 2008.

[32] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*: Springer Publishing, 2000.

[33] C. A. Glasbey, "An Analysis of Histogram-based Thresholding Algorithms," *Graph. Models Image Process.*, vol. 55, pp. 532-537, 1993.

[34] N. Otsu, "A Threshold Selection Method from Gray-level Histogram," *IEEE Trans. Syst. Man Cybern.*, vol. 9, pp. 62-66, 1979.

[35] K. Palagyi and A. Kuba, "Directional 3D Thinning Using 8 Subiterations," *Discrete Geometry for Computer Imagery*, vol. 1568, pp. 325-336, 1999.

[36] O. Schall, A. Belayev, and H. Seidel, "Robust Filtering of Noisy Scattered Point Data," *Proceedings of Point-Based Graphics, 2005, Eurographics/IEEE VGTC Symposium,* pp. 71-77, Stony Brook, New York, USA, 2005.

[37] E. Shaffer and M. Garland, "Efficient Adaptive Simplification of Massive Meshes," *IEEE Visualization '01*, 2001.

[38] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," *Proceedings of SIGGRAPH '92,* pp. 71-78, Chicago, Illinois, 1992.

[39] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient Simplification of Point-Sampled Surfaces," *Proc. of IEEE Visualization*, pp. 163-170, 2002.

[40] H. Woo, E. Kang, S. Wang, and K. H. Lee, "A new segmentation method for point cloud data," *International Journal of Machine Tools and Manufacture*, vol. 42, pp. 167-178, 2002.

[41] M. Pauly, R. Keiser, and M. Gross, "Multi-scale Feature Extraction on Point-Sampled Surfaces," *Computer Graphics Forum*, vol. 22, pp. 281-289, 2003.

[42] Y. Lee, S. Park, Y. Jun, and W. C. Choi, "A Robust Approach to Edge Detection of Scanned Point Data," *International Journal of Advanced Manufacturing Technology*, vol. 23, pp. 263-271, 2004.

[43] A. Hubeli and M. Gross, "Multiresolution Feature Extraction from Unstructured Meshes," *Visualization, 2001, Proceedings of Vis' 01*, pp. 287-294, 2001.

[44] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani, "Recognising Structure in Laser Scanner Point Clouds," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 46, pp. 33-38, 2004.

[45] X. Y. Jiang and H. Bunke, "Fast Segmentation of Range Images into Planar Regions by Scan Line Grouping," *Machine Vision and Applications*, vol. 7, pp. 115-122, 1994.

[46] G. Sithole and G. Vosselman, "Automatic Structure Detection in a Point-Cloud of an Urban Landscape," *Proceedings of the 2nd GRSS/ISPRS Joint Workshop on Data Fusion and Remote Sensing over Urban Areas,* pp. 66-70, Berlin, Germany, 2003.

[47] J. Koenderink and A. v. Doorn, "Surface Shape and Curvature Scales," in *Image and Vision Computing*, pp. 557-565, 1992.

[48] C. Dorai and A. Jain, "A Representation Scheme for 3D Free-Form Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, pp. 1115-1130, 1997.

[49] H. Ho and D. Gibbins, "Multi-scale Feature Extraction for 3D Models using Local Surface Curvature," *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications,* pp. 16-23, Canberra, Australia, 2008.

[50] A. Jagannathan and E. Miller, "Three-dimensional Surface Mesh Segmentation using Curvedness-based Region Growing Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, pp. 2195-2204, 2007.

[51] A. E. Johnson and M. Hebert, "Using Spin-Images for Efficient Multiple Model Recognition in Cluttered 3-D Scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, pp. 433-449, 1999.

[52] J. Assfalg, M. Bertini, A. D. Bimbo, and P. Pala, "Content-based Retrieval of 3-D Objects Using Spin Image Signatures," *IEEE Transactions on Multimedia*, vol. 9, 2007.

[53] A. Boyer, "Adaptive Structured Light Imaging for 3D Reconstruction and Autonomous Robotic Exploration," Masters Thesis, University of Ottawa, 2009

[54] A. Boyer, P. Curtis, and P. Payeur, "3D Modeling from Multiple Views with Integrated Registration and Data Fusion," *Proceedings of the 6th Canadian Conference on Computer and Robot Vision,* pp. 252-259, Kelowna, BC, Canada, 2009.

[55] V. Borsu and P. Payeur, "Pose and Motion Estimation of a Moving Rigid Body with Few Features," *Proceedings of the IEEE International Workshop on Robotic and Sensor Environments,* pp. 116-121, Lecco, Italy, 2009.

[56] J. Shi and C. Tomasi, "Good Features to Track," *Proc. of 9th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 593-600, 1994.

[57] J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker - Description of the Algorithm," *http://robots.stanford.edu/cs223b04/algo_tracking.pdf*.

[58] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. of the 1981 DARPA Imaging Understanding Workshop*, pp. 121-130, 1981.

[59] A. Yogeswaran and P. Payeur, "Feature Extraction from Point Clouds for Automated Detection of Deformations on Automotive Body Parts," *Proceedings of the IEEE International Workshop on Robotic and Sensor Environments,* pp. 122-127, Lecco, Italy, 2009.

[60] D. H. Eberly, "3D Game Engine Design: A practical approach to real-time computer graphics," Morgan Kaufman Publishers, 2007, pp. 66-72.