Edson Kenji Ueda

Injured Skull Zone Determination by Piecewise Bézier Curve Model

São Paulo
2020

Edson Kenji Ueda

Injured Skull Zone Determination by Piecewise Bézier Curve Model

Revised Version

Ph. D. Thesis presented at the Escola Politécnica da Universidade de São Paulo to obtain the degree of Doctor of Science.

Concentration area: Mechanical Engineering

Advisors:
Prof. Dr. Marcos de Sales Guerra Tsuzuki (Poli-USP)
Prof. Dr. Ahmad Barari (Ontario Tech University)

São Paulo
2020

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Ueda, Edson Kenji
Injured Skull Zone Determination by Piecewise Bézier Curve Model / E. K. Ueda -- versão corr. -- São Paulo, 2020.
100 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Curva de Bézier 2.Recozimento simulado 3.Ajuste de curvas 4.Lesão craniana 5.Prótese craniana I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

A minha família, amigos e professores

Acknowledgements

I express my sincere thanks to Prof. Dr. Marcos de Sales Guerra Tsuzuki and Prof. Dr. Ahmad Barari for the guidance throughout the development of this project and I specially thanks to Prof. Dr. Ahmad Barari for having me for two months in his laboratory.

To my colleagues from the Computational Geometry Laboratory, who helped me on this journey.

To CNPq for the scholarship (Grant 305.959/2016–6) and CAPES.

Last, but not the least, my parents for the support throughout these years.

Resumo

No projeto de uma prótese craniana é importante determinar como será o encaixe entre o crânio e a mesma. Desta forma, é importante determinar a superfície de contato entre a prótese e a lesão craniana. Os pontos que definem esta superfície são denominados de zona da lesão, e com estes dados é possível projetar uma prótese com maior precisão e avaliar o contato entre a prótese e o crânio, ao determinar as distância entre as duas partes. Nesta tese é proposto um método semi-automático iterativo para se determinar a zona da lesão, em que é utilizado um modelo de curvas como base para se determinar todos os pontos que compõem essa região. O modelo de curvas é determinado com o auxílio do recozimento simulado de vizinhança adaptativa, onde os parâmetros que determinam a curva são ajustados de forma que a curva se aproxime da região requerida. É utilizado uma curva cúbica de Bézier por trechos, em que cada trecho da curva aproxima somente uma parte da zona da lesão. A cada iteração pontos são adicionados e removidos na zona da lesão e um novo modelo de curva do contorno da lesão é determinada. Este processo de inclusão e remoção de pontos é realizado até que não haja mais buracos na superfície de contato. O método proposto foi testado em quatro exemplos; três com lesões criadas artificialmente e uma com uma lesão real; e em todos os testes o método conseguiu determinar a zona da lesão com sucesso.

Palavras-chave: Lesão craniana, prótese craniana, curva de Bézier, recozimento simulado, ajuste de curvas.

Abstract

It is highly important to determine the fitting between the injured skull and the fabricated prosthesis in the Cranioplasty surgical procedure. This requires a precise definition of the contact surface between the injured skull and the prosthesis.
A geometric representation that determines this surface is named as injured zone. It is possible, with the aid of the injured zone, to develop a more accurate prosthesis and evaluate the contact between the prosthesis and injured skull by determining the gaps between them. An iterative semi-automatic method is proposed to determine the injured zone, in which a curve model is used as reference to determine all the points of this area. The adaptive neighborhood simulated annealing algorithm determines the curve model, in which the curve parameters are adjusted to determine a curve that approximates the required zone. A piecewise cubic Bézier curve is used, in which each curve segment approximates only a part of the injured zone. Points are added and removed to the injured zone and a new injured zone curve model is determined in each new iteration. This procedure of adding and removing points is done until there are no more holes in the injured zone. The proposed method was tested in four examples; three artificially created injuries and one natural injury; and the method successfully determined the injured zone in all the four tests.

Keywords: Injured skull, skull prosthesis, Bézier curve, simulated annealing, curve fitting.

Contents

List of Figures

1 Introduction

Cranioplasty is a surgical procedure to repair an injured skull, that was damaged because of a trauma, tumor recession, infections or deformity of the skull. This procedure repairs the skull function which is protect the brain; however, due to aesthetic reasons and possibly other practically requirements the skull's shape may influence the patient's social and emotional life [TRAINOR; RICHTSMEIER, 2015]. The main difficulties of the surgery and neurosurgery teams are the complex anatomic geometry of the skull, each injury is unique, the presence of vital adjacent anatomical structures and the high risk of infection [PARTHASARATHY, 2014; HIEU et al., 2002; PARK et al., 2016].

The patient autologous bone is the best compatible material to be used in a cranioplasty. However, there are limitations in size and shape. It cannot be practical if the injury extension is too large [PARK et al., 2016]. That is why alloplastic materials are also used, that are not organic material used to manufacture the prosthesis. According to Park et al. [2016], the prosthesis material required characteristics must be: not magnetic, radio transparent, light, sterilizable, and must be easily fixable to the skull. The polymethyl methacrylate (PMMA), hydroxyapatite and titanium are the most used materials considering the required characteristics.

The cranial prosthesis manufacturing cost is very high, due to the high complexity of the geometry and mainly to the fact that each prosthesis is unique. Thus, there is no scalability to reduce the production cost. Due to this fact, the cost of this surgery procedure has been very high. In the literature, there are several methods to manufacture the prosthesis. Wehmöller et al. [1995], Eufinger et al. [1995] machined a prosthesis using a CNC milling machine, however due to the high complexity of the skull geometry, the machining time is very high. Another possibility is to machine a mold, and the prosthesis is manufactured by injecting PMMA in the machined mold [HIEU et al., 2002; LEE et al., 2002]. However, the manufacturing time and cost decreased with the advent of additive manufacturing. Initially, 3D printers were used to produce a mold [KIM et al., 2012], or to manufacture a silicon mold from a printed injured skull replica [ROTARU et al., 2012]. The manufacturing time is still high, once firstly it is necessary to create the mold and then fabricate the prosthesis. However, it is possible to print metallic objects through the sintering process with the improvement of additive manufacturing techniques. Thus, the prostheses began to be made of titanium by the sintering process [PARTHASARATHY, 2014; JARDINI et al., 2014; VOLPE et al., 2018] as shown in Fig. 1.1.

The additive manufacturing was not the only one that contributed to the cranioplasty improvement. The usage of CAD/CAM (Computed Aided Design/Computer Aided Manufacturing) techniques helped the cranioplasty development in two aspects. The first one is the capability to design the prosthesis digitally, and second is the capability to plain

Figure 1.1 – Example of a titanium prosthesis. Font[1]

the surgical procedure virtually [ZIELINSKI et al., 2015; JAISINGHANI et al., 2017]. There are two basics approaches to design a prosthesis in the literature. The prosthesis can be designed by mirroring the healthy skull side over the injured side [HIEU et al., 2002; JARDINI et al., 2014], or the prosthesis can be designed by interpolating a surface defining its geometry [HIEU et al., 2002; WEHMÖLLER et al., 1995; EUFINGER et al., 1995].

Only Hieu et al. [2002] studies the detail of the injured zone. According to them the injured zone is important because it is possible to design a better prosthesis, as well as to determine a better fit between the prosthesis and the skull.

1.1 Objectives

The main goal of this work is to develop a method to determine the injured skull zone. Three surfaces defines the injured skull; the inner surface, the outer surface and the surface that links these two surfaces. The linking surface is the injured zone. The determination of this zone enables a better prosthesis design.

1.2 Literature Review

Wehmöller et al. [1995] developed a 3D geometric model with surface parameters so that the defined geometry approximates a skull. The outer injured skull surface is parametrically

[1] Source: <https://www.polygonica.com/polygonica-blog/polygonica-software-new-features-at-tct-show-2017/?amp=&amp=&utm_campaign=TCT_Events_Page&utm_medium=Event_Click&utm_source=Events%20Page>

represented considering the surface curvature of the healthy skull portion. An offset of the outer surface defines the inner surface. However, both surfaces were defined with a gap of 0.5 to 1 mm between the prosthesis and the skull to ensure the contact between both parts. This is a limitation of the method, because it does not ensure a perfect contact between the skull and the prosthesis, once the injured zone is not determined.

Hieu et al. [2002] developed a method that creates three surfaces (internal, external and contact) to define the skull prosthesis. These surfaces are defined by using contour lines through the mirroring technique. A curve is defined in each computed tomography image slice, using the healthy mirror side of the skull and parameters to ensure the continuity between the skull and the injured part. This methodology has some limitations, once the curve parameters are separately defined in each slice and each curve modifies the whole surface. Thus, the complete tangential continuity is not ensured in the whole surface. The tangential continuity is only guaranteed in each curve, and not in the entire surface.

Lin et al. [2017] used a snakes algorithm to determine the inner border and outer border of each tomography slice. The stack of several curves determine the prosthesis surface. This approach has the same problem of the previous method, it does not ensure the continuity between the curves in each slice.

Carr et al. [1997], Sing et al. [2005] and Chen et al. [2006] projected the prosthesis using surface interpolation. Each paper used a different surface to define the geometry of the prosthesis; Carr et al. [1997] used radial basis function, Sing et al. [2005] used Bézier surfaces and Chen et al. [2006] used NURBS surfaces. This approach ensures the continuity between prosthesis and skull and it is possible to be used with injuries at any location on the skull. However, it cannot be used for large injuries, once there are information just in the border of the prosthesis (the tangent vector between the prosthesis and skull), and there are few information to define the best geometry in the middle of it.

Jardini et al. [2014] made a simpler approach, the prosthesis was designed through a Boolean operation by subtracting two solids, the injured skull region and the mirror side of the healthy skull. This is not an ideal solution, because the skull is not perfectly symmetric not being possible to ensure the continuity in the contact between the prosthesis and skull.

Other works [KIM et al., 2012; ROTARU et al., 2012; PARK et al., 2016; VOLPE et al., 2018] designed the prosthesis using proprietary software, such as Materialise Mimics® . There is a problem in using these software tools. The user does not have proper control over the geometric parameters, because the software algorithms are unknown to the user, the user does not know the limitations and the method of how the geometry is determined.

1.3 Work Structure

The proposed method determines the points from a skull point cloud which defines the injured skull zone. The input is a STL file from where the point cloud is extracted. The

user defines two clouds, one representing the healthy region and another representing the injured region of the skull. A symmetry plane is defined and the point cloud is mirrored creating a new point cloud. Both point clouds are compared using a distance map. By filtering the distance map, a first guess for the injured zone is found. The injured zone is defined by a curve approximating procedure.

This work is structured as follow. Chapter 2 presents some basic concepts on the curve fitting. Curve fitting is an important part of the thesis because the injured zone is represented by a curve. In Chapter 3, the optimization algorithm developed based on simulated annealing is presented. In Chapter 4, the developed application of simulated annealing to curve fitting is described. In Chapter 5, an integrated procedure based on the developed methodologies is presented. Chapter 6 presents the validation case studies and the corresponding results. And conclusions are drawn in Chapter 7.

2 Curve Fitting - Curve Approximation

In this chapter, the principles of Bézier curves and some basic concepts on curve fitting are presented.

2.1 Bézier Curve

The Bézier curve $\mathbf{P}(u)$ is a parametric curve given by

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u), \qquad u \in [0,1] \tag{2.1}$$

with $\mathbf{p}_i$ being the control points, $n+1$ is the number of control points, (n) is the polynomial degree and $B_{i,n}(u)$ is the Bernstein polynomial basis function defined by

$$B_{i,n}(u) = \binom{n}{i} u^i (1-u)^{n-i}, \qquad i = 0, ..., n. \tag{2.2}$$

with $\binom{n}{i}$ being the binomial given by

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \qquad \binom{0}{0} \equiv 1. \tag{2.3}$$

Fig. 2.2 shows a cubic Bézier curve and its four control points. The control points $\mathbf{p}0$, $\mathbf{p}1$, $\mathbf{p}2$ *and* $\mathbf{p}3$ define the curve, and it always passes through the first and last control points. The curve is always inside the control point convex-hull [CHIYOKURA, 1988, section 5.2.1]. Points $\mathbf{p}1$, $\mathbf{p}2$ are internal control points.

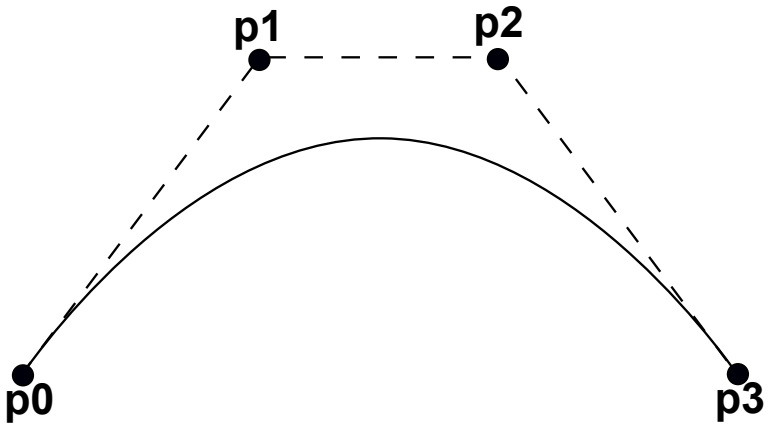


Figure 2.2 – Bézier curve, with control points $\mathbf{p}0$, $\mathbf{p}1$, $\mathbf{p}2$ *and* $\mathbf{p}3$.

2.2 Introduction to Curve Approximation

The objective of the curve fitting problem is to determine an approximating curve from a sequence of points. This curve can be an interpolation curve, in which the curve passes through all the points; or an approximation curve, in which a smooth curve approximates the sequence of points.

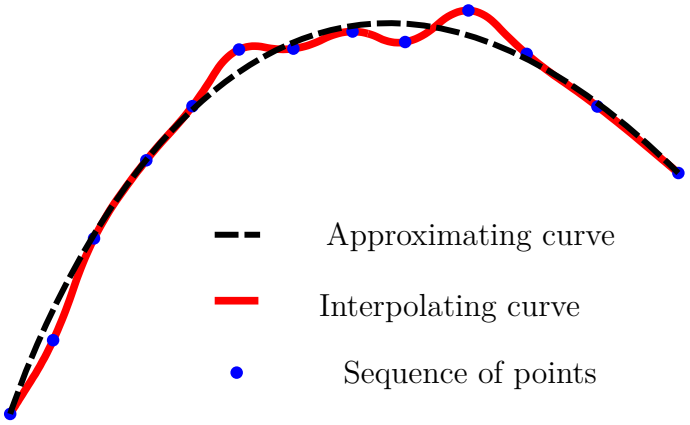


Figure 2.3 – Approximating curve (dashed) versus the interpolating curve (red).

Fig. 2.3 shows an example of the difference between these two curves, in which the dashed curve is the approximating curve and the red curve is the interpolating curve.

The procedure to find a curve that approximates a sequence $\{\mathbf{d}\}$ of m points can be translated into the determination of control points $(\mathbf{p_0}, ..., \mathbf{p_n})$ that defines the approximating curve. Thus, it is necessary to minimize the summation of the distances between each point to the curve, and the function that describes this summation is

$$f(\mathbf{p_0}, ..., \mathbf{p_n}) = \sum_{k=0}^{m} ||\mathbf{d}_k - \mathbf{P}(u_k)||^2, \tag{2.4}$$

with n being the number of control points. The expression $||\mathbf{d}_k - \mathbf{P}(u_k)||$ is the discrepancy or distance between point $\mathbf{d}_k$ and the curve. Several approaches to determine the distance between a point and a curve were proposed. Hasegawa et al. [2013] and Pandunata and Shamsuddin [2010] determined the distance with a predetermined parametrization, in which a parameter u_k obtained by the cord length parametrization determines the curve closest point to a point $\mathbf{d}_k$. This parameter is the ratio between the chord length of two consecutive points of the sequence $\{\mathbf{d}\}$ ($||\mathbf{d}_k - \mathbf{d}_{k-1}||$), and the summation of all the chords ($\sum_{i=1}^{m} ||\mathbf{d}_i - \mathbf{d}_{i-1}||$). An advantage of this method is its agility, since the calculation is done only once, and if an iterative algorithm needed to determine the distance, it would be quickly determined. However, this method does not ensure the correct distance, once this parameter changes by modifying the curve. Fig. 2.4 shows a Bézier curve with 4 control points and a sequence of 5 points, in which each point $\mathbf{d}_k$ of the sequence has a closest point $\mathbf{P}(u_k)$. It is possible to observe that at point $\mathbf{d}_3$ the distance to the curve is

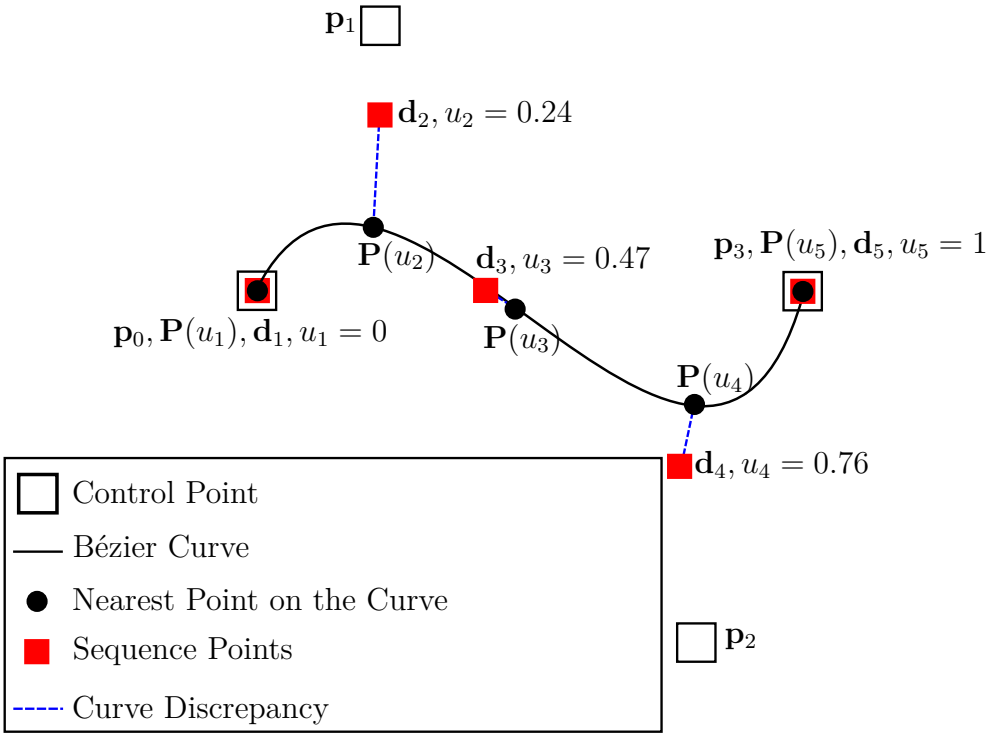


Figure 2.4 – Distance determined by a predetermined parametrization (Adopted from: Hasegawa et al. [2013, Figure 1]).

null, but the predetermined parametrization determined point $\mathbf{P}(u_3)$, and the distance between these two points are not null.

The second method to determine the distance between a point and a curve is an iterative method. Maekawa et al. [2007] and Gofuku et al. [2009] used the Newton-Raphson method to obtain the parameter u_k that determines the point $\mathbf{P}(u_k)$ as the closes point to point $\mathbf{d}_k$. This parameter is determined by solving

$$(\mathbf{d}_k - \mathbf{P}(u_k)) \cdot \dot{\mathbf{P}}(u_k) = 0. \tag{2.5}$$

Eq. 2.5 determines a parameter u_k in which a vector $(\mathbf{d}_k - \mathbf{P}(u_k))$ is orthogonal to $\dot{\mathbf{P}}(u_k)$ as shown in Fig. 2.5. Another iterative method to determine the distance of a point to a curve is the algorithm proposed by Hu and Wallner [2005], in which a first and second order algorithms are presented to obtain the orthogonal projection of a point over the curve or surface. Both iterative methods can be very accurate, if enough processing time is available. When the curve fitting method is iterative, the use of this method is not feasible. Eq. 2.5 may have more than one solution, because it looks for orthogonal lines. This equation returns multiple solutions if a curve has parallel or orthogonal parts, as can be seen in Fig. 2.6. Fig. 2.6(a) the point $\mathbf{d}_k$ (circle) has 3 points (squares) at the curve in which Eq. 2.5 is satisfied, and there are two solutions in Fig. 2.6(b).

A third method to determine the distance of a point to the curve is by approximation, in which the curve is discretized, and from the sampled points it is determined the closest

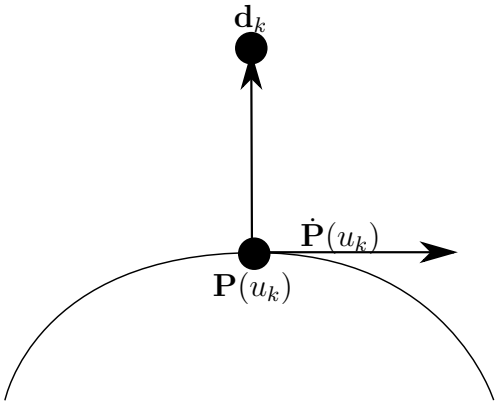


Figure 2.5 – Determination of the distance between a point and a curve.

point to $\mathbf{d}_k$. Tavares et al. [2011] and Ueda et al. [2016] used this method, which is the best balance between the accuracy and processing time. Ueda et al. [2016] improved the method by calculating the distance between point and curve as being the height of a triangle. The method is divided in three steps. The first step is the above described procedure, in which the curve is sampled by a sequence of discrete points $\{\mathbf{P}\}$. The search for the closed point in $\{\mathbf{P}\}$ happens for every point every point $\mathbf{d}_k$. The closest point is named as $\mathbf{P}_v$. Two triangles are defined using the next and the previous point to $\mathbf{P}_v$; respectively $\mathbf{P}_{v+1}$ and $\mathbf{P}_{v-1}$; and the points $\mathbf{d}_k$ and $\mathbf{P}_v$, as can be seen in Fig. 2.7. The second step is to define the triangle with smallest area, by comparing the length of $||\mathbf{P}_{v+1} - \mathbf{d}_k||$ and $||\mathbf{P}_{v-1} - \mathbf{d}_k||$. If $||\mathbf{P}_{v-1} - \mathbf{d}_k|| < ||\mathbf{P}_{v+1} - \mathbf{d}_k||$ the triangle determined by vertex $\mathbf{d}_k$, $\mathbf{P}_v$ and $\mathbf{P}_{v-1}$ has the smallest area, otherwise the other triangle has the smallest area. The third step is to verify whether the chosen triangle is acute, because if the chosen triangle is obtuse the height of the triangle will be smaller than the distance between point $\mathbf{d}_k$ and the curve, as can be seen in Fig. 2.8. Thus, if the chosen triangle is obtuse, the other triangle is chosen to calculate the height; and if both triangle are obtuse, the distance $|\mathbf{P}_v - \mathbf{d}_k|$ is used as the distance between the point and the curve.

2.3 Approach with Regularization and Statistic Approach

There are others approaches in the literature to solve the curve fitting problem. Because Eq. 2.4 just minimizes the discrepancy between the sequence of points and the approximating curve, it may result the generation of a family of curves with the same discrepancy. A regularization is necessary to distinguish these curves. Ueda et al. [2016] used the absolute difference between the sequence length and the approximate curve length as regularization, using

$$f(\{\mathbf{p}_i\}, \{\mathbf{d}\}, \{\mathbf{P}\}) = W \cdot \sum_{k=1}^{m-1} ||\mathbf{d}_k - \mathbf{P}_v||^2 + (1 - W) \cdot |(L(\mathbf{P}(u)) - L(\mathbf{d})|. \tag{2.6}$$

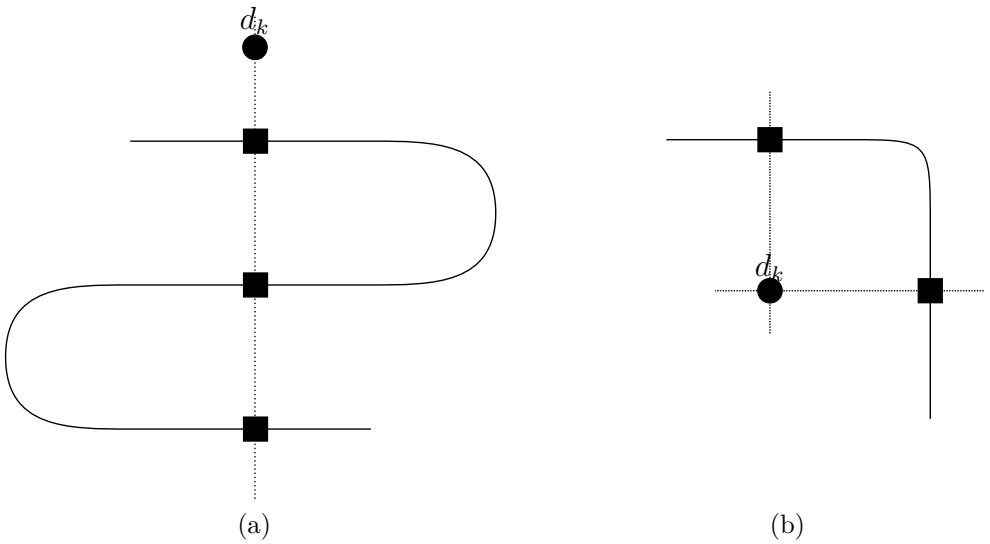


Figure 2.6 – Multiple solutions for Eq. 2.5; (a) curve with parallel segments; (b) curve with orthogonal segments.

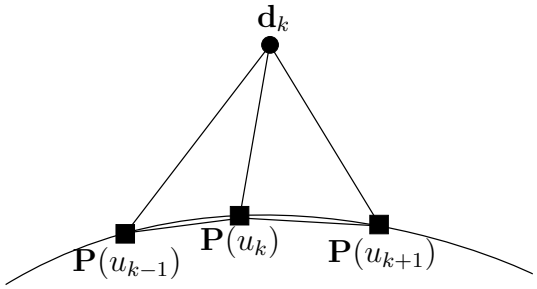


Figure 2.7 – The distance between $\mathbf{d}_k$ and the curve is approximated by the height of the triangle with smallest area. The sampled point $\mathbf{P}_v$ is the closest point to $\mathbf{d}_k$ in the sampled sequence of points. The previous $\mathbf{P}_{v-1}$ and next $\mathbf{P}_{v+1}$ sampled points are considered in the determination of the triangle with smallest area.

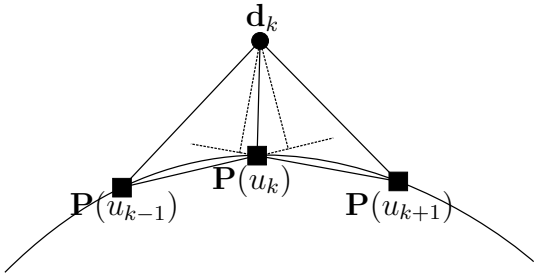
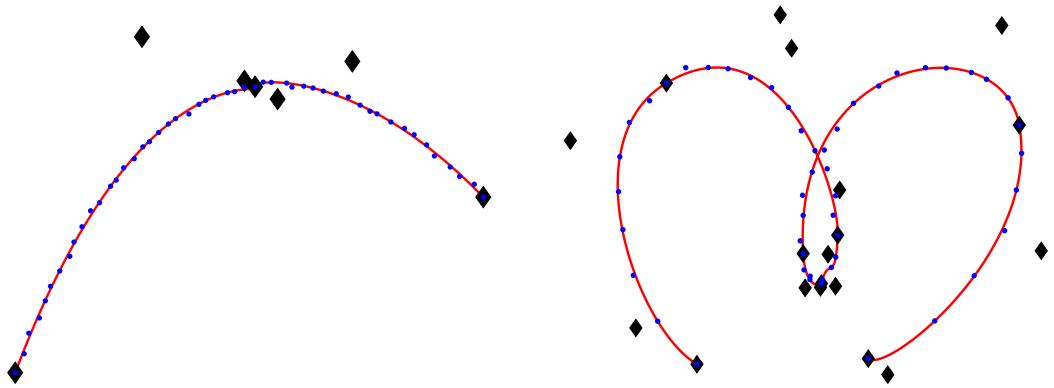


Figure 2.8 – A different approach happens if both triangles are obtuse. In this case the distance between $\mathbf{d}_k$ and the curve is the edge between $\mathbf{d}_k$ and $\mathbf{P}_v$.

with W being a weight that controls the regularization, $L(\mathbf{P}(u))$ is the approximating curve length and $L(\mathbf{d})$ is the length of the sequence of points.

The obtained curve using Eq. 2.6 is not smooth considering that a noisy sequence of points is used. The reason the equation does not minimize the approximate curve length. This equation minimizes the difference of length, i.e., it equates both lengths. However, the length of a noisy sequence of points defines a curve longer than a smooth curve, as can be seen in Fig. 2.9. The sequence of points is represented with blue points, the curve control point with black diamonds and the approximating curve in red line. Note that in both Fig. 2.9(a) and (b) the curve length is longer in some specific parts of the curve, in Fig. 2.9(a) at the junction of the curves and in Fig. 2.9(b) in the middle of the sequence and in the end of the sequence.



(a) (b)

Figure 2.9 – (a) The curve is not smooth near the concentration of control points; (b) This one has a similar behavior.

Another approach is a statistic method in which the Akaike Information Criterion (AIC) [AKAIKE, 1974] is used. This criterion evaluates just a part of the sequence to be approximated. Thus, it creates a model in which only a part of the sequence is used to calculate the discrepancy. Ravari and Taghirad [2016] used the following equation to solve the curve fitting problem applied to the signal treatment problem.

$$AIC = Nln\frac{1}{N}\Big(\sum_{i=1}^{N}(y(x_i) - \hat{y}(x_i))^2\Big) + 2k, \tag{2.7}$$

with N being the number of points to be used in the model, y is the function value measured in the x_i instant, $\hat{y}$ is the estimated value of the function and k is the number of parameters used in the model.

Gálvez and Iglesias [2011] used the same criterion with a different penalization. The criterion is composed by two parts, the first one is the model error estimation, and the second one is a model penalization. However, this approach has problems. It is known that if the number of parameters is very low overfitting happens, i.e., the penalization is low if the number of parameters is too low, and the determined curve is no longer smooth.

Thus, the number of parameter that describes the model influences the outcome, and it is difficult to determine the required minimum amount of parameters to make the fitting. A second problem is to determine how much and which points are used in the calculation of model error estimation.

In the following chapter it is presented the optimization algorithm called simulated annealing which is used to determine the best approximating curve for a noisy sequence of points. The simulated annealing is very flexible, it can simultaneously handle combinatorial parameters, integer parameters and continuous parameters.

3 Simulated Annealing

The simulated annealing is a probabilistic metaheuristic based on the Metropolis algorithm [METROPOLIS et al., 1953] that simulates the atomic re-crystallization of a metal in the annealing process. Kirkpatrick et al. [1983] modified the algorithm by adopting a cooling schedule to solve combinatorial optimization problems. Corana et al. [1987] proposed an adapted version to use it in problems with continuous variables.

During the annealing procedure, atoms migrate resulting in a configuration that always minimizes the energy. This is analogous to the process of solving an optimization problem, in which a function is minimized. In the simulated annealing, candidates are generated by making a random modification of the current solution, then the cost function is evaluated. If the new candidate is better, i.e., the new cost function value is smaller than the current value, the new candidate is accepted and the algorithm continues to the next iteration. However, if the new cost function increases, the candidate is discarded and a new iteration is performed. There is a possibility of accepting a worst candidate with a probability

$$P(\Delta E) = e^{-\frac{\Delta E}{k_b T}} \tag{3.1}$$

with k_b being the Boltzman constant; usually $k_b = 1$; ΔE is the cost function variation; and T is the temperature. If $P(\Delta E) > random(0, 1)$ the new candidate is accepted.

If the new solution is rejected, a new candidate is generated by applying a random modification to the current solution. In combinatorial problems, as the one in Kirkpatrick et al. [1983], the determination of a new candidate is not difficult. A classical example is the traveling salesman problem. In this problem, a simple variable permutation can generate the next solution. There are several strategies that can be adopted to continuous variable optimization problems. According to Bohachevsky et al. [1986], it is necessary to generate a random unitary vector $\mathbf{u}$ ($\|\mathbf{u}\| = 1$), multiply it to a step of a size Δr and add it to the current solution x_c to determine the next candidate $\mathbf{x}_n$. Thus,

$$\mathbf{x}_n = \mathbf{x}_c + \Delta r \cdot \mathbf{u}. \tag{3.2}$$

Unlike the method proposed by Bohachevsky et al. [1986] in which the step size is unitary and constant, Corana et al. [1987] proposed an anisotropic method, in which the number of accepted solution is tried to keep constant while the step size in each direction changes. Thus, the next candidate is generated by

$$\mathbf{x}_n = \mathbf{x}_c + v \cdot \Delta r_i \cdot \mathbf{e_i} \tag{3.3}$$

with Δr_i being the value of the step towards $\mathbf{e_i}$, $\mathbf{e_i}$ an unitary vector representing the direction of the i *th* parameter and v a random value between $[-1, 1]$. The value of Δr_i varies with the change of the temperature.

3.1 Adapting Neighborhood and Crystallization Factor

Algorithm 1 Adapting Neighborhood Simulated Annealing (ANSA)

Define T_0 and α /* T_0 initial temperature and α cooling coefficient */
$\mathbf{x}_0 \leftarrow$ <Random initial solution>
$j \leftarrow 0$
while < Global stop criteria> **do**
  $T_{j+1} \leftarrow T_j * \alpha$
  $j \leftarrow j + 1$
  while <Local stop criteria> **do**
    $i \leftarrow random(0, 1) * n$ /* n being the number of variables */
    $\mathbf{x}_n \leftarrow \mathbf{x}_c + \frac{1}{c_i} \sum random(-1/2, 1/2) \cdot \mathbf{e_i} \cdot \Delta r_i$
    $\Delta E = F(\mathbf{x}_n) - F(\mathbf{x}_c)$
    if $\Delta E < 0$ **then**
      $\mathbf{x}_c \leftarrow \mathbf{x}_n$
      $c_i \leftarrow$ <positive feedback>
    else
      if $random(0, 1) < e^{-\Delta E / k_b T_k}$ **then**
        $\mathbf{x}_c \leftarrow \mathbf{x}_n$
        $c_i \leftarrow$ <positive feedback>
      else
        $\mathbf{x}_n$ is rejected.
        $c_i \leftarrow$ <negative feedback>
      end if
    end if
  end while
end while

The strategy proposed by Corana et al. [1987] frequently converted to local minimum because the step size decreases with the temperature. Martins et al. [2012a] hereby proposed an Adaptive Neighborhood Simulated Annealing (ANSA) (Algorithm 1), in which the minimum step size is kept constant and the probability of each step size changes. It is considered that rejected candidates do not contribute to the advance of the simulated annealing algorithm. Thus, the probability distribution is adjusted according to the number of accepted candidates. Only one parameter is modified at each iteration, and the new candidate is given by

$$\mathbf{x}_n = \mathbf{x}_c + \frac{1}{c_i} \sum_{1}^{c_i} random(-1/2, 1/2) \cdot \mathbf{e_i} \cdot \Delta r_i, \tag{3.4}$$

with Δr_i being the value of the step towards $\mathbf{e_i}$, $\mathbf{e_i}$ an unitary vector representing the direction of the *i th* variable, *random* a uniform distribution and c_i being the crystallization factor corresponding to a continuous variable i that is modified in this iteration. The applied modification is a Bates distribution [JOHNSON et al., 1995, section 26.9] center on zero with amplitude of 1/2. Thus, the standard deviation of this distribution is

$$\sigma = \frac{1}{2\sqrt{3c_i}}. \tag{3.5}$$

Throughout the optimization procedure, each continuous variable is correlated to a correspondent crystallization factor, in which is adjusted favoring the acceptance of new candidates. The standard deviation corresponding the probability density of a variable decreases by rejecting a candidate, resulting a smaller density probability amplitude to the next modification. There is a standard deviation increase by accepting the solution, resulting a larger density probability amplitude to the next modification. There is a positive feedback each time a candidate is accepted, in which the crystallization factor c_i value corresponding to the variable i decreases. The standard deviation given by eq. 3.5 increases, resulting in an increase of probability to choose candidates further from the current solution. Otherwise, the crystallization factor c_i increases with the rejection of a candidate (negative feedback). There is a decrease of the standard deviation resulting in a decrease of the distribution probability to choose candidates further from the current solution. The increase of the crystallization factor cannot be greater than a limit, because this increase results in a numerical error in the summation of the Bates distribution.

The procedure of accepting and rejecting solutions can be observed in Fig. 3.10. The expected behavior of the crystallization factor with the temperature variation can be seen in Fig. 3.11. In this procedure, the crystallization factor tends to increase with the temperature decrease and there is a change of phase during this decrease. The two phases are an exploratory phase and a refinement phase. In the exploratory phase, the crystallization factor is smaller, the probability of choosing further candidates is greater and the temperature is higher. In the refinement phase the crystallization factor is greater, the probability decreases and the temperature is lower.

The ANSA accounts that the problem to be minimized has n continuous variables, a cooling schedule started with a initial temperature T_0 and it uses a cooling factor α. This algorithm has two stop criteria, a local criteria that controls the inner loop, and a global criteria that controls the outer loop. Usually, it is used as local stop criteria a maximum number of iterations or maximum number of accepted solutions; and it uses as global stop criteria a minimum temperature, a maximum number of iterations or a minimum number of accepted candidates. The decrease of the crystallization factor can be performed through several ways. It can be made by subtraction or division, with caution. Because, there is a possibility of freezing the algorithm if the decrease is very mild, resulting the

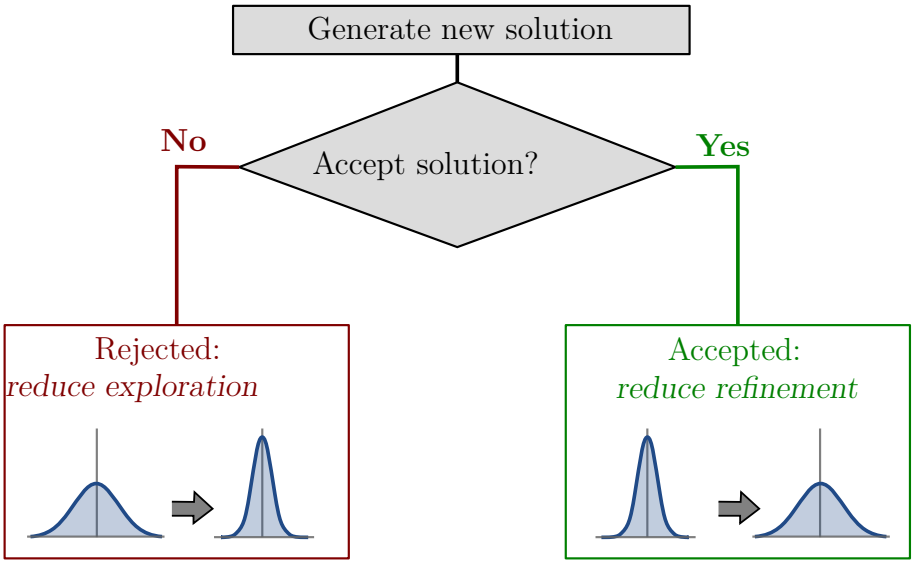


Figure 3.10 – Crystallization factor feedback.

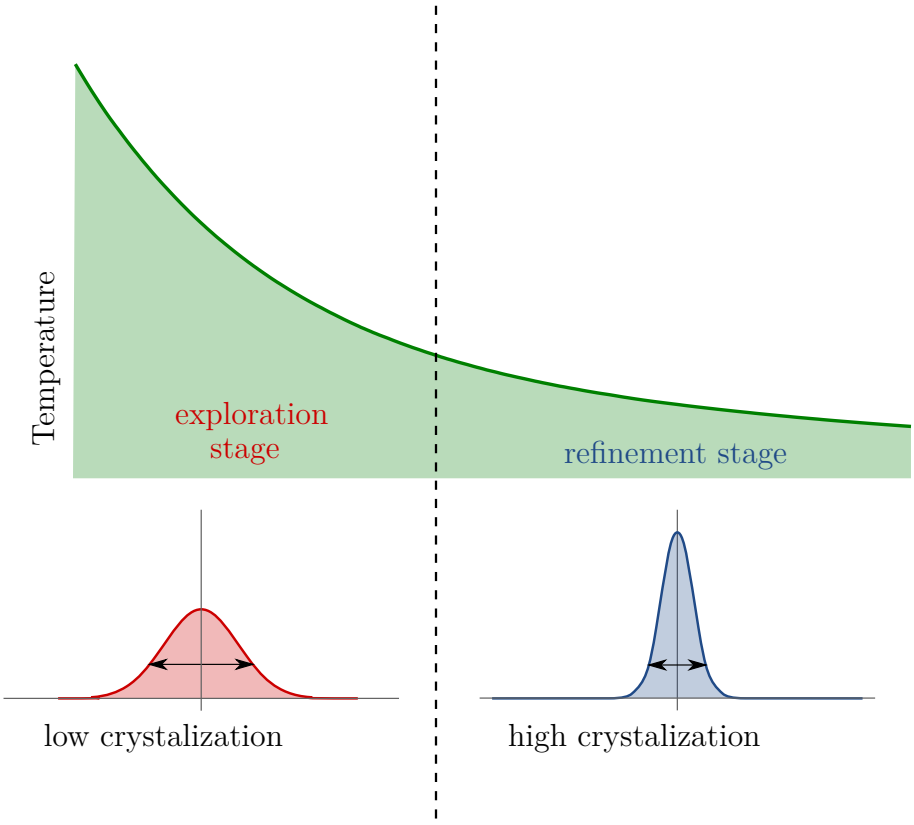


Figure 3.11 – Crystallization factor behavior with the temperature variation.

parameters in being trapped in a solution; if the decrease is very extreme (such as zeroing the crystallization factor), the algorithm takes longer time to achieve the global minimum.

This implementation was applied to several problems in the literature; Martins and Tsuzuki [2010a] and Martins and Tsuzuki [2010b] used this algorithm to solve the cutting and packing problem; and [MARTINS et al., 2012b] used it to solve the inverse problem of the electrical impedance tomography (EIT).

3.2 Cooling Schedule

The cooling schedule is the function that defines the temperature decrease throughout the algorithm. The chosen cooling schedule directly influences the quality of the final result, as shown in Fig. 3.11. There are two phases in Fig. 3.11; an exploration phase, in which the temperature is high and the probability to search further candidates is high; and a refinement phase with the decrease of the temperature, in which the probability of searching is lower. If the algorithm decreases the temperature in a very abrupt way, the algorithm does not perform a very long time in the exploration phase, resulting in a possibility of the algorithm to be trapped in a local minimum.

A possible cooling function is the geometric cooling given by $T_{k+1} = T_k * \alpha$, with $\alpha =]0, 1[$. The final result is better if α is closer to 1, however the processing time is higher. Smaller values for α turns the algorithm faster, but the final result is worst. This is because for lower values of α the faster the cooling occurs. As mentioned, the exploration phase is performed in higher temperatures; and the quality of the final result is worst if the algorithm perform less iteration in the exploration phase. It is not an easy task to define the right value for α, because it is possible that at some temperatures the sample distribution standard deviation is low, showing that the sample is already frozen to the current temperature and therefore the next temperature will be lower than the geometric decrease. The opposite effect is also possible, the sample distribution standard deviation is higher and the cooling need to be slower than the geometric decay. Thus, it will adopted an adaptive cooling, in which the value for α is correlated to the standard deviation $\sigma(T)$ of all the costs from a determined temperature T. α is determined by the following expression

$$\alpha = e^{-\frac{\gamma . T}{\sigma(T)}}. \tag{3.6}$$

with γ being an adjustable coefficient and $\sigma(T)$ the cost function distribution standard deviation in a determined temperature T.

The simulated annealing generates a lot of cost functions C_i, and these cost functions

standard deviation is calculate by the probability distribution

$$Prob_i(T) = \frac{e^{\frac{-C_i(T)}{k_B T}}}{\sum_j e^{\frac{-C_j(T)}{k_B T}}}. \tag{3.7}$$

with $k_B = 1$.

The mean cost for a determined temperature is calculated by

$$\langle C(T) \rangle = \sum C_i(T) Prob_i(T) \tag{3.8}$$

The quadratic mean cost is defined as

$$\langle C^2(T) \rangle = \sum C_i^2(T) Prob_i(T). \tag{3.9}$$

Thus, the variance; the squared standard deviation; is expressed by

$$\sigma^2(T) = \langle C^2(T) \rangle - \langle C(T) \rangle^2. \tag{3.10}$$

The following chapter describes the application of the ANSA in the curve fitting problem, focusing on how the cost is handled and explains the problem variables.

4 ANSA Application in the Curve Fitting Problem

In this work the ANSA is used to solve the curve fitting problem, in which a piecewise cubic Bézier curve approximates a sequence of points. The method is shown in Fig. 4.12. The method starts with a random initial solution, and only one parameter is modified in each iteration. After the modification, there is a verification of the cost function, that is the composition of the curve discrepancy and the curve length. It is necessary to sample the approximating curve to determine the discrepancy between the sequence $\{\mathbf{d}_k\}$ and curve. At the end of each iteration it is verified if the ANSA achieved the termination criteria.

4.1 Piecewise Bézier Curve

It is an undesired effect that the control points of a Bézier curve modify the entire curve, in the curve fitting. However, the Bézier curve is simpler than B-Spline and NURBS resulting in faster evaluation and shorter computational time. Thus, the usage of a piecewise Bézier

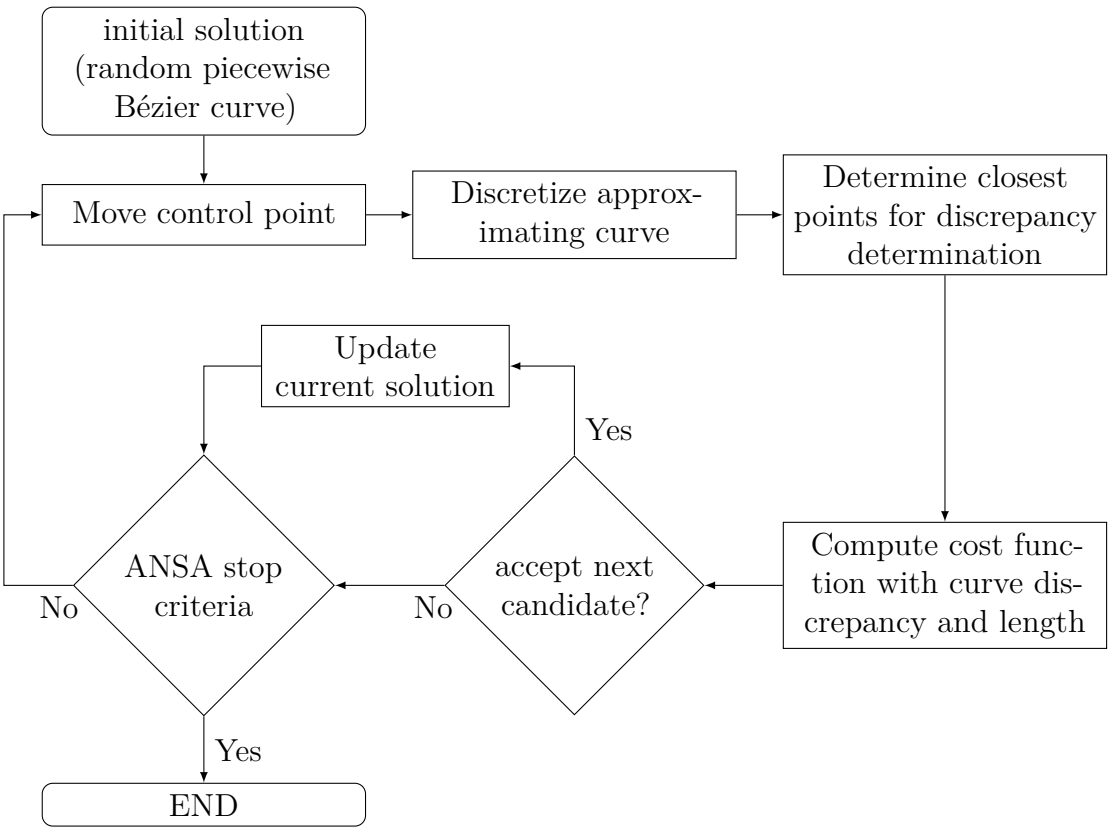


Figure 4.12 – Flowchart of the ANSA method to determine the approximate curve.

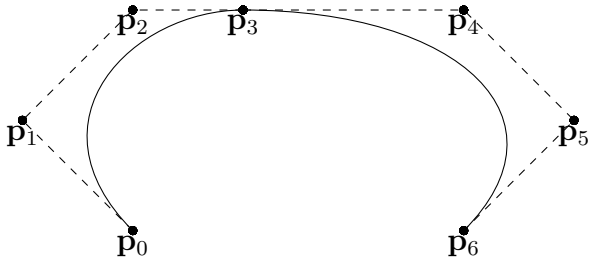


Figure 4.13 – Piecewise cubic Bézier curve with 2 curve segments, in which $\mathbf{p}_3$ is a connecting control point.

curve is a way to solve this problem. This curve is a sequence of Bézier curves with weak-G^1 continuity in which each control point modifies just adjacent curves. A piecewise cubic Bézier curve example is shown in Fig. 4.13, in which a composition of two curve segments is presented. The first segment is defined by control points **p0**, **p1**, **p2** and **p3** , and the second one is defined by control points **p3**, **p4**, **p5** and **p6**. Note that the last control point of the first curve segment is the first control point of the second segment, and it is necessary to determine the control point **p4** to ensure the weak-G^1 continuity in the junction of both curves, that is given by

$$\mathbf{p4} = \mathbf{p3} - \beta \cdot (\mathbf{p2} - \mathbf{p3}). \tag{4.1}$$

This equation makes the derivative of the first curve tail (defined by the control points **p0**, **p1**, **p2** *and* **p3**) the same as the derivative of the head of the second curve (defined by the control points **p3**, **p4**, **p5** *and* **p6**). β is a proportional ratio between the module of both derivatives. The weak-G^1 continuity relaxes the condition of continuity at point **p3**, in which the derivatives has the same direction, but not necessary the same module. Thus, the procedure to create a new Bézier curve segment is to define the second control point with eq. 4.1, and to adopt as first control point the last control point of the previous curve segment. Yamaguchi [1988, section 5.1.2] used this method to create a piecewise cubic Bézier curve with G^2 continuity, ensuring the equality of the first and second derivative in the junction of the segments.

4.2 Cost Function Determination

Previously presented cost functions cannot reach satisfactory results. Equation 2.4 was used by [HASEGAWA et al., 2013; PANDUNATA; SHAMSUDDIN, 2010; ADI et al., 2010]. As mentioned, this equation determines a family of curves with the same discrepancy, but with different lengths. The usage of this equation results in overfitting. Eq. 2.6 used by [UEDA et al., 2016] fails in cases of a noisy sequence of points leading to the determination of longer curves. Eq. 2.7 leads to an increase of variables to be adjusted, the number of points and which points are new variables.

Thus, it is used as cost function a similar equation to eq. 2.6. A minor difference is the regularization function, it will be used as regularization the approximating curve length instead of the difference of lengths. The adopted cost function is

$$f(\{\mathbf{p}_i\},\{\mathbf{d}\},\{\mathbf{P}\}) = W \cdot \sum_{k=1}^{m-1} ||\mathbf{d}_k - \mathbf{P}_v||^2 + (1-W) \cdot L(\mathbf{P}(u)). \tag{4.2}$$

with W being a weight that controls the regularization, and $L(\mathbf{P}(u))$ the approximating curve length.

4.3 Determination of the Curve Discrepancy and Curve Length

The method to calculate the discrepancy between point and curve is the method presented in Chapter 2. This method is the one in which the discrepancy between point and curve is the height of the acute triangle with smallest area.

As proposed by Ueda et al. [2016], it is possible to decrease the processing time in the search performed in the first step of the discrepancy determination. Once the chosen curve is a piecewise Bézier curve, each segment of this curve just approximates one part of the sequence of points, as shown in Fig. 4.14. Using as example the curve from Fig. 4.15, points $\mathbf{d}_5$ and $\mathbf{d}_{m-5}$ are junction points between two segments, and curve is sampled with v points and the curve is composed by j segments. As consequence, the search for the closest point for points $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_3$ and $\mathbf{d}_4$ are performed in the first segment $(\mathbf{P}_1, ..., \mathbf{P}_9)$, and the search to the points $\mathbf{d}_{m-4}$, $\mathbf{d}_{m-3}$, $\mathbf{d}_{m-2}$ and $\mathbf{d}_{m-1}$ are performed in the last segment $(\mathbf{P}_{v-9}, ..., \mathbf{P}_{v-1})$. According to this logic, for each point $\mathbf{d}_k$ there is only one section on which the search must be performed. The processing time decreases with the local search, once the search for the closest point is performed in a smaller set of points for each point $\mathbf{d}_k$. A second advantage of this method is the fact that the discrepancy is locally determined. This advantage can be observed in examples with self-intersecting curves, as shown in Fig. 4.16. In both examples, the curve length and curve discrepancy are close, however there is a self-intersection in Fig. 4.16(a), while in Fig. 4.16(b) the self-intersection is not presented. The points $\mathbf{d}_k$ approximate the wrong segment using the global search. The reason is the points that should approximate the red curve approximate the black one, and vice-versa. Fig. 4.17 shows another examples with self-intersection curves with different number of curve segments and points, each curve segment is shown in a different color and the sequence of points $\{\mathbf{d}_k\}$ is shown in black points.

The curve length is easily determined. The curve was already been sampled to determine the curve discrepancy, then the curve length is the summation of all the segments of the

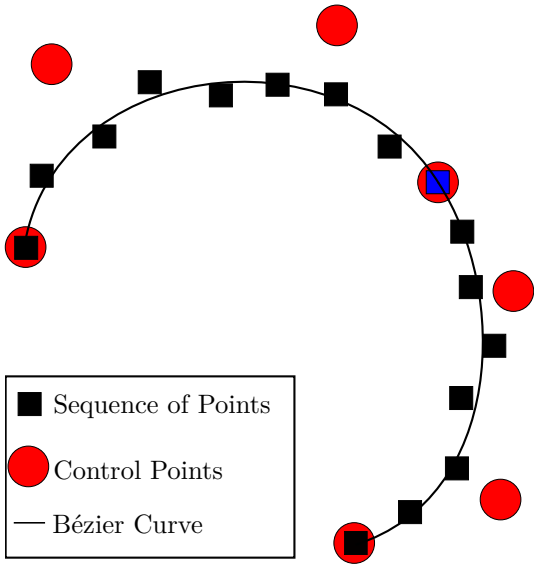


Figure 4.14 – Each Bézier curve segment approximates just one part of the sequence of points.

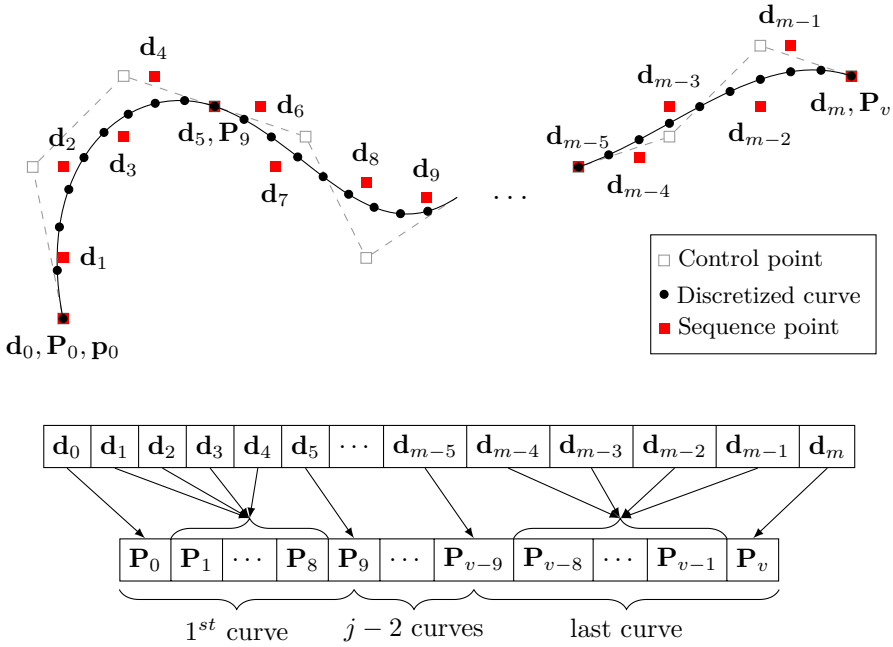
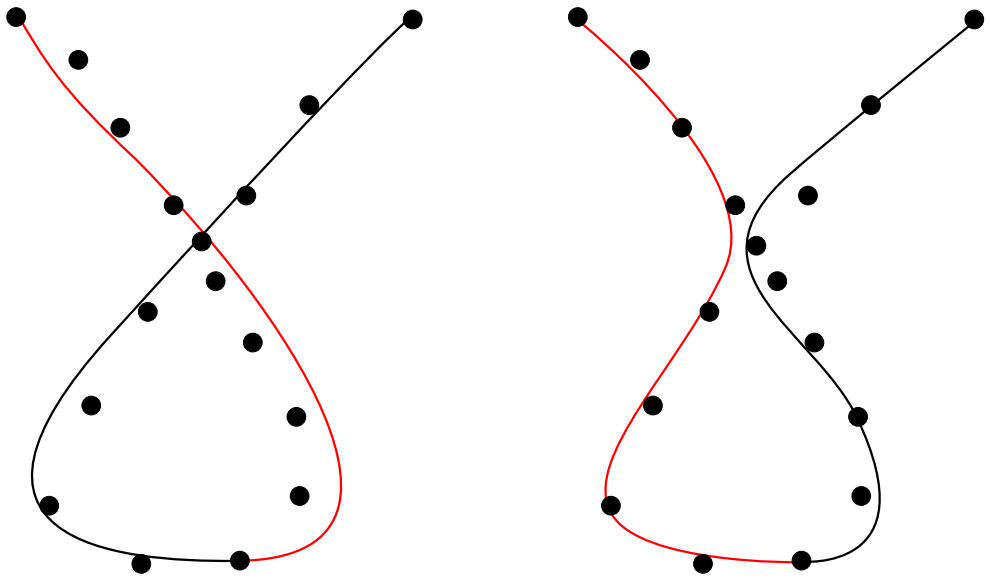


Figure 4.15 – Local search for the closest point, the search for each point $\mathbf{d}_k$ is performed in only one curve segment. The sequence $\{\mathbf{P}_v\}$ are sampled points from the approximating curve. The sequence $\{\mathbf{d}_k\}$ is the sequence of points.

sampled curve, and it is given by

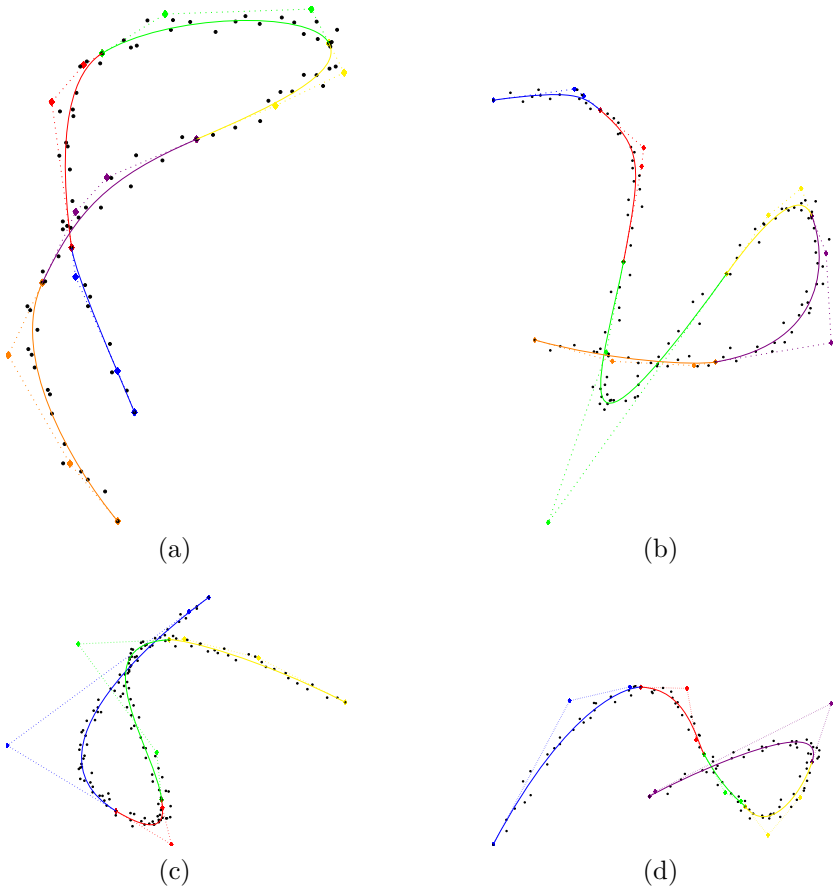
$$L(\{\mathbf{p}_i\}) = \sum_{k=1}^{v} ||\mathbf{P}_k - \mathbf{P}_{k-1}||, \tag{4.3}$$

with v being the number of sampled points, and j the number of Bézier curve segments.



(a) (b)

Figure 4.16 – Example of a two segment Bézier curve. (a) There is self-intersection; (b) There is no self-intersection.



(a) (b)

(c) (d)

Figure 4.17 – Curves with self-intersection. (a) 6 curve segments and 85 points; (b) 6 curve segments and 135 points; (c) 4 curve segments and 147 points; (d) 5 curve segments and 111 points.

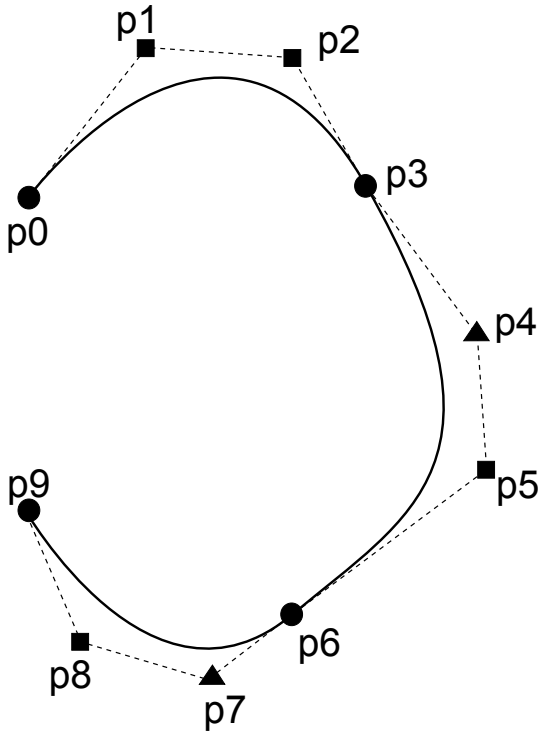


Figure 4.18 – Piecewise cubic Bézier curve, circles are point of the sequence of points, squares are the control points adjusted by the ANSA, and triangles are control points adjusted by ANSA thought the β proportionality coefficient.

4.4 Parametric Representation of the Piecewise Bézier Curve

The parameters to be adjusted are the control points coordinates and the coefficient β, the one that modifies the derivative module between two adjacent Bézier curve segments. It is relevant to notice that it is not required to directly adjust all control points. The first and last control point of a segment are points of the sequence of points, and Eq. 4.1 determines the second control point of a new segment.

Using Fig. 4.18 as example, in which a curve with three cubic Bézier segments is shown, the parameters to be adjusted by ANSA are:

1. Coordinate x of point $\mathbf{p}_1$ (p_1x)
2. Coordinate y of point $\mathbf{p}_1$ (p_1y)
3. Coordinate z of point $\mathbf{p}_1$ (p_1z)
4. Coordinate x of point $\mathbf{p}_2$ (p_2x)
5. Coordinate y of point $\mathbf{p}_2$ (p_2y)
6. Coordinate z of point $\mathbf{p}_2$ (p_2z)
7. Proportionality coefficient of point $\mathbf{p}_4$ (β_4)

8. Coordinate x of point $\mathbf{p}_5$ (p_5x)
9. Coordinate y of point $\mathbf{p}_5$ (p_5y)
10. Coordinate z of point $\mathbf{p}_5$ (p_5z)
11. Proportionality coefficient of point $\mathbf{p}_7$ (β_7)
12. Coordinate x of point $\mathbf{p}_8$ (p_8x)
13. Coordinate y of point $\mathbf{p}_8$ (p_8y)
14. Coordinate z of point $\mathbf{p}_8$ (p_8z)

Points $\mathbf{p}_0$ and $\mathbf{p}_9$ are the first and last points of the sequence, and points $\mathbf{p}_3$ and $\mathbf{p}_6$ are points of the sequence. Thus, the parameters to be adjusted are the x, y and z coordinate of points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_5$ and $\mathbf{p}_8$, and the β proportionality coefficient that determines the control points $\mathbf{p}_4$ and $\mathbf{p}_7$.

4.5 Parameter Update

The parameters values are updated using the method shown in Algorithm 1, in which it is used the crystallization factor to modify the probability distribution for each parameter throughout the temperature decrease. The parameter update is given by

$$x_n \leftarrow x_c + \frac{1}{c_i} \sum random(-1/2, 1/2) \cdot \mathbf{e_i} \cdot \Delta r_i. \tag{4.4}$$

5 Injured Skull Zone Determination

Using the developed methodologies and principles presented in the previous chapters, an integrated procedure is developed. In this Chapter the developed procedure as an algorithm is explained. The proposed method is presented in Fig. 5.19 and the Algorithm 2 describes this method. The input data is a STL file from which the point cloud is extracted. It has some manual operation to define the healthy and injured region and the symmetry plane and the initial number of clusters, these procedures correspond the initial section of the algorithm until the loop. After completing these tanks, the rest of the procedure is automated. The automatic part is an iterative method that uses the piecewise Bézier curve as reference to determine all the points of the injured zone.

Algorithm 2 PROPOSED METHODOLOGY

$i = 1$ and $\mathbf{P}_0 = null$ */ $\mathbf{P}_i$ is the injured zone curve model */
Input: STL file
Extraction of unique points from STL (Pc)
User defines healthy and injured regions
Definition of a symmetry plane
Healthy region mirroring creating a new point cloud (Pc')
Creation of the distance map ($Pc' \ominus Pc$)
User defines k /* k is the number of clusters */
Set $\mathbf{d}_i$ using a filter threshold t from Pc
while < Stop criteria> **do**
    Cluster $\mathbf{d}_i$ using K-means
    Order the clusters and set the junction points between the clusters
    Define the injured zone curve model ($\mathbf{P}_i$) from $\mathbf{d}_i$ using the curve fitting algorithm
    if < number of finished clusters decreases > **then**
        $k++$
    else
        $i++$
        Update $\mathbf{d}_i$, use the curve model $\mathbf{P}_i$ to classify $\mathbf{d}_i$ as inlier and outlier, eliminating the outliers
    end if
end while

5.1 STL File and Extraction of a Point Cloud

The STL file is widely used in rapid prototyping and 3D printing. There is a triangulation in this stereolithography CAD file, representing a geometric model of the object. The file consists of a list of a triangle vertex coordinates and its facet normal vector. Thus, using as example the triangle in Fig. 5.20, an example of a triangle representation is shown in

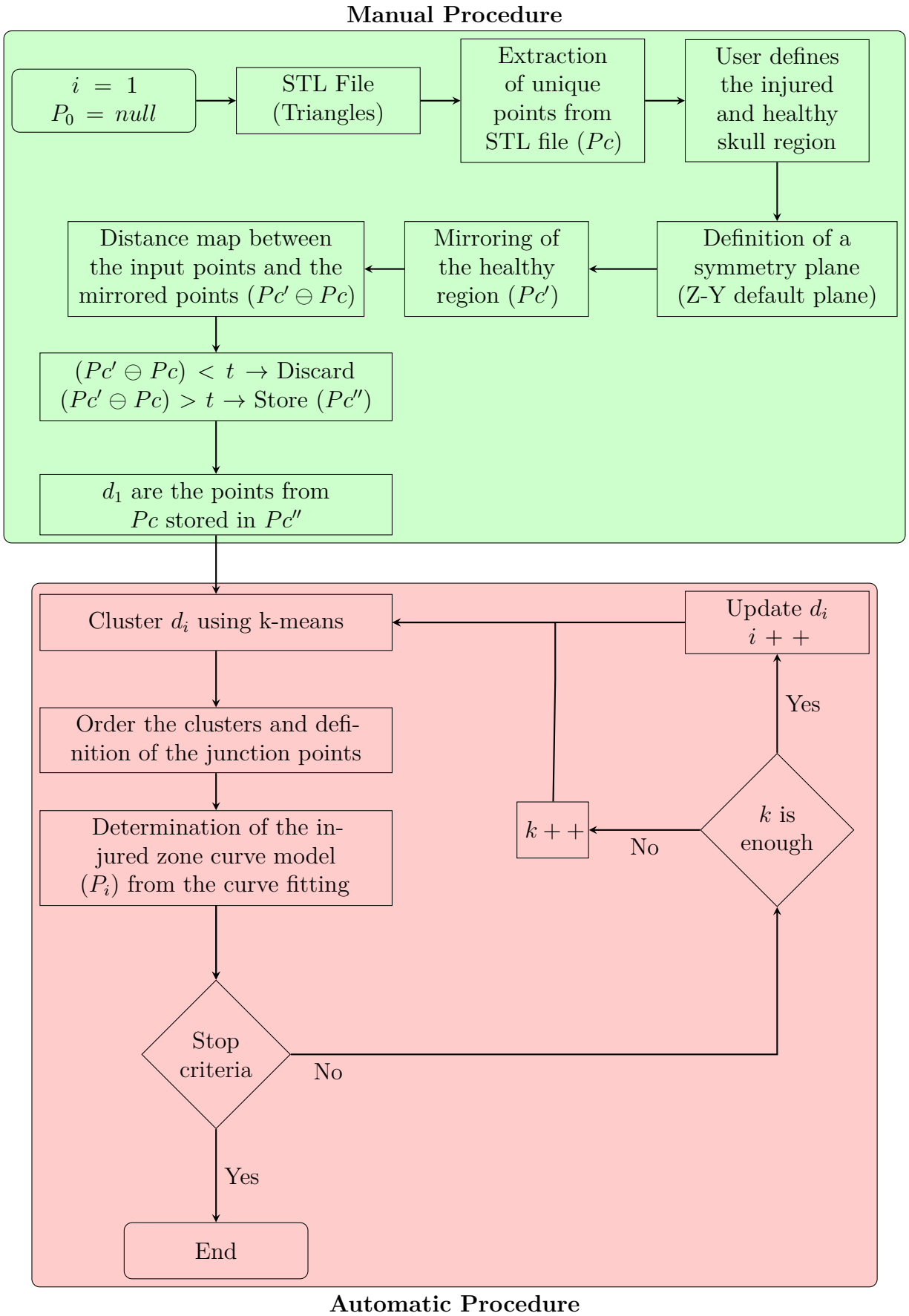


Figure 5.19 – Flowchart of the proposed method to determine the injured skull zone.

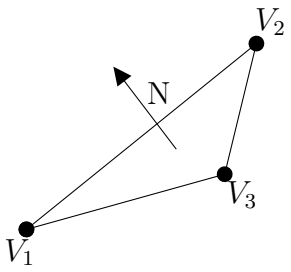


Figure 5.20 – Example of a triangle to be represented in the STL file.

```
1: solid name
2: facet normal Ni Nj Nk
3:       outer loop
4:             vertex V1x V1y V1z
5:             vertex V2x V2y V2z
6:             vertex V3x V3y V3z
7:       endloop
8: endfacet
9: endsolid name
```

Figure 5.21 – STL file example.

the file format example in Fig. 5.21. The solid name is defined in the first line, followed by the definition of a triangle in lines 2 to 8. The triangle facet normal vector is defined in line 2, and the vertex coordinate of the triangle are defined in lines 4 to 6. Thus, it is necessary to inform a normal vector and three vertexes coordinates to inform a new triangle in the file.

In the proposed algorithm, it is used only the triangle vertex coordinates; however not all the points represented in the STL file will be used, the number of points used in the method will be smaller. In the STL file, the same vertex is represented several times inside the file. This multiplicity can be observed in Fig. 5.22, in which the red vertex can be listed in 6 or 12 triangles; 6 triangles with a normal entering the paper plane and 6 triangles with the opposite normal. Thus, the initial point cloud to be used as initial data to the proposed algorithm are all the unique vertex presented in the STL file. Fig. 5.23 shows an example of an injured skull, Fig. 5.23(a) presents the skull STL file, and the respective extracted point cloud in Fig. 5.23(b).

5.2 Determination of the Reference Point Cloud and the Distance Map

The user manually determines the injured skull region, i.e., whether the injury is in the right or left side of the skull. Fig. 5.23(b) shows an injured skull point cloud and the

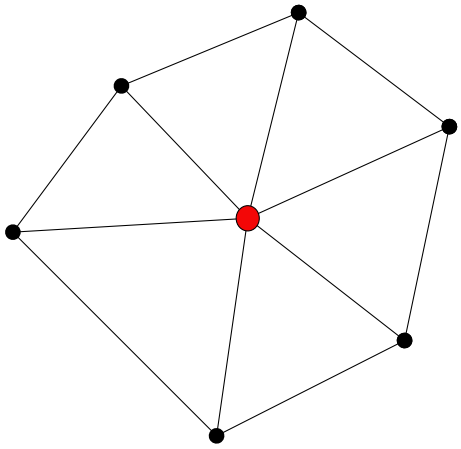
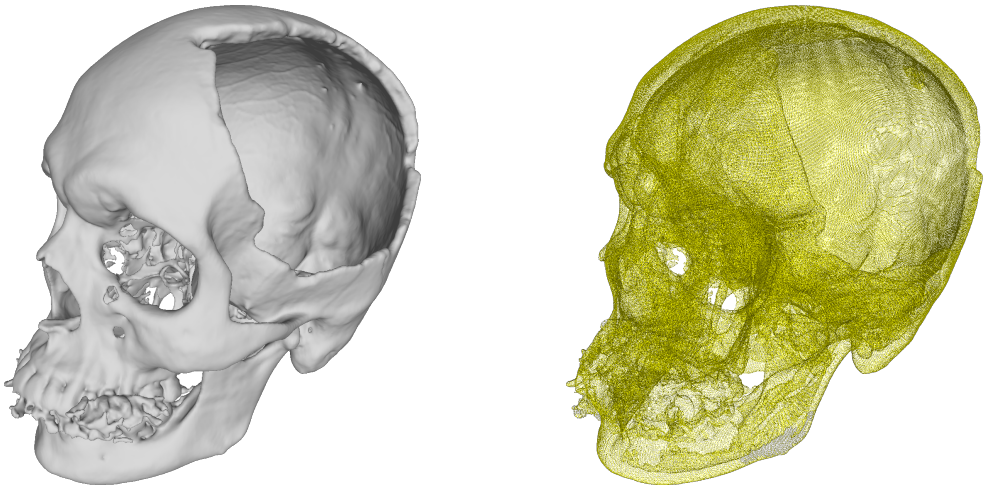


Figure 5.22 – Example of how the same vertex is presented in several triangles, the red vertex defines multiple triangles.



(a) (b)

Figure 5.23 – Injured skull example. (a) Rendered from the STL file; (b) Point cloud.

injured region is the left side of the skull. Once the injured region is determined, the next step is to determine the symmetry plane. Usually, it is used the sagittal plane Z-Y as default symmetry plane to define a symmetric point cloud. This procedure is simple. Just multiply the X coordinate of the points by -1, i.e., the signal of the X coordinate is inverted (positive coordinates will be negative and vice-versa). Fig. 5.24 shows the input injured skull (gray) overlapping the symmetrical or mirrored skull (red). Notice that both skulls are displayed with their triangulation to habilitate the observation of the overlapping. It is possible to observe that there is no perfect overlapping between both skull point clouds, due to the no perfect symmetry of the skull. If the skull were perfectly symmetrical the determination of the injured zone and the skull prosthesis would be easily performed. In this case, a simple Boolean operation would be enough to determine both features.

The next step is to create the distance map between both point clouds ($Pc' \ominus Pc$).

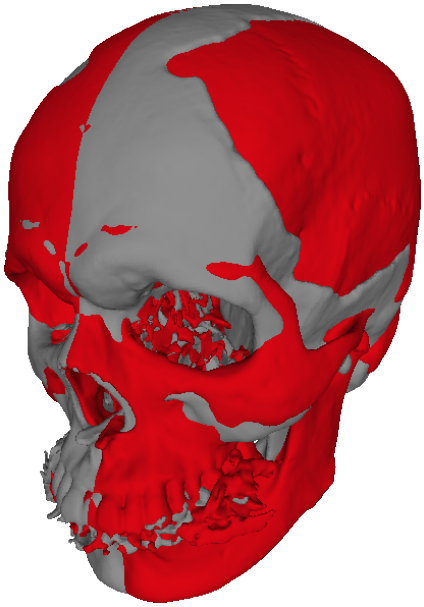


Figure 5.24 – Gray skull is the input data and the red skull is the mirrored one.

This map is not performed in the whole point cloud, it is just performed in a part of the point cloud. The user defined the injured region of the skull in the previous step, then just the injured region and a part of the mirrored point cloud are used. The distance map is a list of pairs of points and their respective distance, then there is a set of respective point from the input point cloud (Pc) for each point from the mirrored point cloud (Pc'). Thus, this step consists in the search for a set of correspondent points from Pc for each point from Pc', as shown in Fig. 5.25 in which a red point from the left skull (Pc') has a correspondent red point in the right skull (Pc). This group of points are the nearest neighbour or closest point and the points within a determined radius. The radius is the distance between the point of Pc' and its respective nearest neighbour plus 10% of this distance. This increase is limited by the average length size of the triangle edges from the STL file. Therefore, the number of corresponding points from Pc is different for each point from Pc'.

The search for the nearest neighbour and the points within a range is performed using the K-d tree data structure [BENTLEY, 1975]. There is a naive implementation to find the nearest neighbour, in which the distance between the reference point and every point in the other is calculated, and the closest point is the one with smallest distance. This implementation has a problem; if the number of points from Pc is too large, the processing time increases significantly, once this algorithm cost is $O(n)$. The K-d tree data structure is a multidimensional binary search tree that divides the space in two half-spaces with a hyper-plane, so in each level it is generated two half-spaces that contains its child nodes, which are recursively subdivided. The search for the nearest neighbour uses one property of the K-d tree to eliminate the search in one half-space in each level. Therefore, if the number of points from Pc is too large the processing time does not increase a lot, once

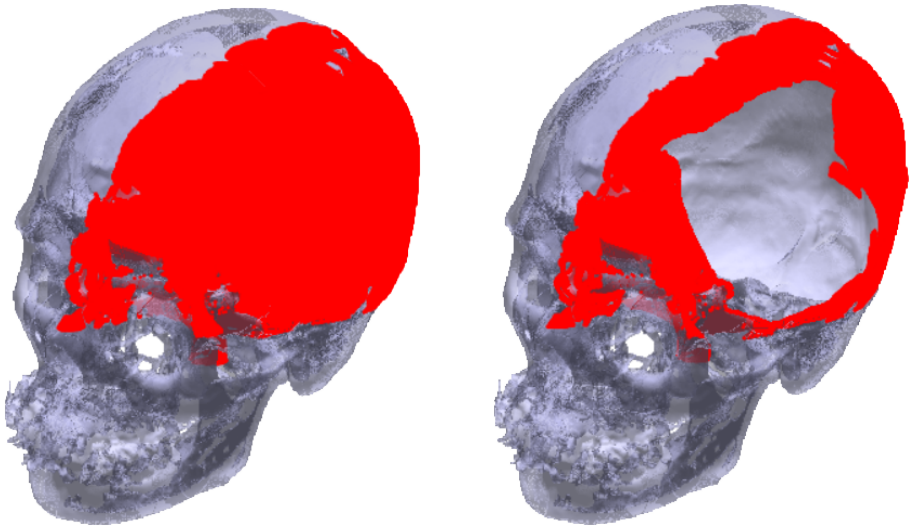


Figure 5.25 – Each red point of the left skull has a correspondent red point in the right skull.

the average cost of this algorithm is $O(log(n))$.

This algorithm was used by [TAKIMOTO et al., 2016] to make the registration of two point clouds, in which it was necessary to determine the distance between two point clouds to match both point clouds.

5.3 Initial Set of Point to Create a Curve Model

A list of pairs of points (one from Pc and another from Pc') and their respective distance was determined in the previous step. A threshold t determines an initial set of points to create the first curve model. This threshold will eliminate all the pair of points with a distance smaller than t. The selection of a good value for t is important. If the value of t is too big, the proposed method will have more iterations, resulting in longer processing time. Otherwise, if the value of t is too small, the initial date set will have points that are not from the injured zone. Thus, a good value of t can be selected with the help of a histogram of the distances. The value will be the one in which the number of points within a certain range of distance starts to decrease. Using as example Fig. 5.26 a good value for threshold t is 15.

Fig. 5.26 shows a histogram in which x axis is the distance between a pair of points determined in the distance map, and y axis is the number of times these distance is determined, i.e., the number of pair of points with similar distances. Note that the distances in Fig. 5.26 varies approximately from 27 to 0. These high distances can only be determined because the reference point cloud is the mirrored point cloud (Pc'). Therefore, all Pc' points have a respective closest point in Pc, and the Pc' points in the middle of injury with their respective Pc closest points determine these high distances. If the input point cloud (Pc) is used as reference, the Pc' points in the middle of injury would never

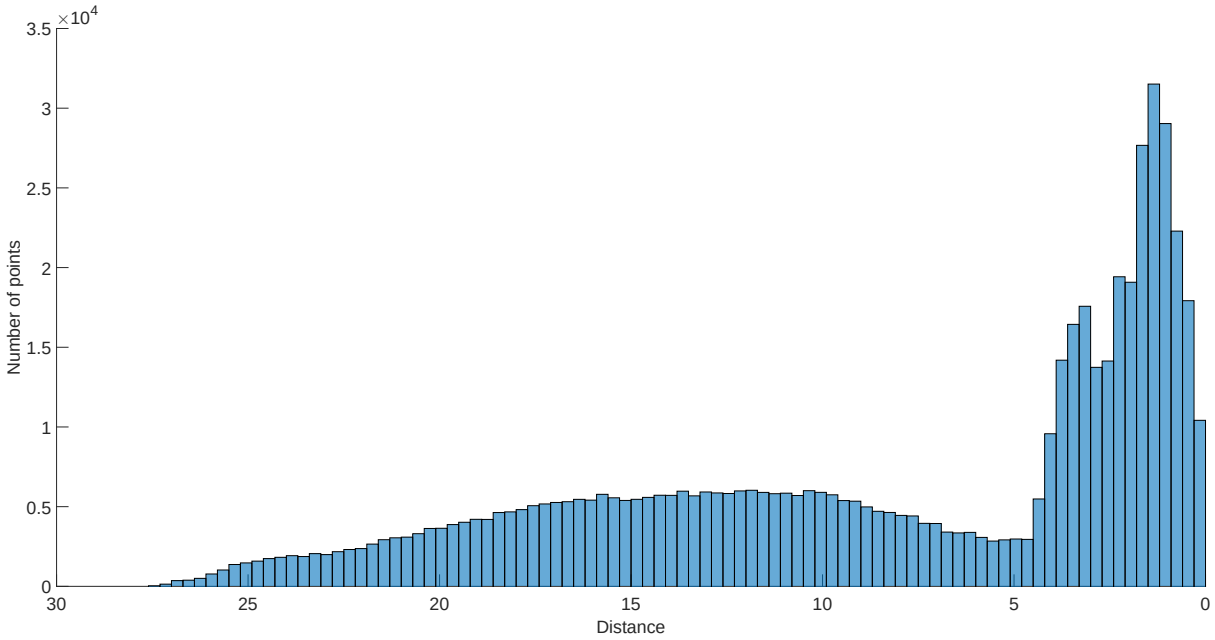


Figure 5.26 – Example of a distance histogram to select a threshold value.

be chosen and the high distances would never be determined. Thus, the operation $\ominus$ is not commutative. $Pc' \ominus Pc$ is different than $Pc \ominus Pc'$, and in the proposed method the distance map is $Pc' \ominus Pc$.

The initial points to create the first curve model are the remaining point from Pc, naming it as $\mathbf{d}_i, i = 1$. The remaining points from Pc' are not used in the proposed algorithm. However, these points can help to determine the prosthesis surface by informing the curvature of the skull.

The procedure of determining the initial set of points can also be used to determine the Deviation Zone Estimation (see Appendix A).

5.4 Injured Zone Curve Model Determination

The injured zone curve model will be determined by analyzing a set of points $\mathbf{d}_i$ selected from Pc. The initial set of points $\mathbf{d}_1$ was determined in the previous step. $\mathbf{d}_i$ will change in each iteration. The change procedure will be later explained.

The method presented in Chapter 4 determines a piecewise cubic Bézier curve as the injury curve model. In the first step, the K-means algorithm [LLOYD, 1982] divides $\mathbf{d}_i$ in several clusters, in which each point from $\mathbf{d}_i$ will be classified in k clusters. The K-means algorithm is an iterative method that divides the input data in k cluster or regions. The algorithm is based on the Voronoi diagram [VORONOI, 1908; AURENHAMMER, 1991] and each of the k points that defines the Voronoi diagram should be close to the mean of the Voronoi cell respective data. The user defines the number of clusters. The minimum required number of parameters to determine the fitting curve is still an open issue. The

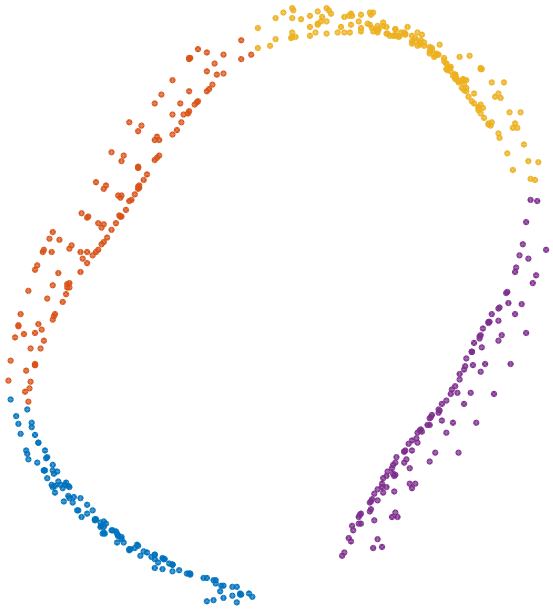


Figure 5.27 – Clustering the $\mathbf{d}_i$ point using K-means, each cluster is represented in a different color.

number of cluster is directly related to the number of parameters that the ANSA adjusts. Once the piecewise Bézier curve is used as curve model, a good guess is to use the number of parabolas needed to define the contour of the injury as initial number of clusters. The number of clusters can be modified throughout the iterations, and this modification will be later explained. Fig. 5.27 shows an example in which each cluster is represented in a different color.

Two points within each cluster are determined so that they are as far apart as possible from each other. Fig. 5.29(a) shows these points marked as diamonds, and these extreme points are used to order the clusters using the following criteria:

1. Determine the neighbour clusters for each cluster. There is a closest extreme point from another cluster for each extreme point of a cluster.

2. The starting cluster is the one with the biggest distance between its neighbour clusters. The starting point of this cluster is the one with longest distance to its neighbour cluster, the other extreme point is the cluster end point.

3. The second cluster is the closest cluster to the end point of the first cluster, the start point is the closest extreme point and the end point is the other one.

4. The procedure is repeated until the next cluster is the first cluster.

Each segment of the piecewise cubic Bézier curve approximates only one cluster. Thus, it is necessary to determine the junction points between the curve segments. The start and end points inside each cluster is not a good option. These points are close to the outer or inner part of the injury, determining a boundary curve model that will not approximate

the middle of the injury. Thus, a point in the middle of the border between two clusters is the the best junction point. The determination of this point is not an easy task. It is proposed a method to approximately determine this point.

The first step is to use the cluster start or end point and find the respective closest point in the neighbour cluster. Fig. 5.28(a) shows an example of this step, in which the red circles and blue circles are parts of two neighboring clusters and the blue circle with red contour are the start or end point of the blue cluster and the closest point in the other cluster is linked with a black line.

The second step determines a group of points that is close to that point. This group is determined by performing a range search using as radius the closest distance between the two clusters added to 5 times the average edge length of the input STL file. The mean point is determined within this group and the closes point of the mean point in the other cluster is determined as junction point. Fig. 5.28(b) shows an example of of this step; the closest distance between the two clusters determines the gray area, and the distance between the two cluster added to 5 times the average edge length of the input STL file determines the yellow area. The green circle shown in Fig. 5.28(c) is the mean point of the yellow area points, and the yellow circle shown in Fig. 5.28(d) is the blue cluster junction point.

This algorithm is performed twice in each cluster and the junction points of the clusters from the example shown in Fig. 5.27 is shown in Fig.5.29(b), in which each junction point is cyan or magenta diamond, the cyan is the start and the magenta is the end. Note that the proposed method just select a better junction point if the contact between two cluster have a lot of points, and if there are just a few points in the contact area, the initial extreme point is the junction point. This can be observed in Fig.5.29, in which the end points determined between blue and purple clusters (see Fig.5.29(a)) are the same of the ones determined with the proposed method (see Fig.5.29(b)). It is also possible to notice that the junction points between purple and yellow clusters are better selected with the proposed method.

The next step determines the injured zone curve model ($\mathbf{P}_i$) from the injured zone points ($\mathbf{d}_i$). The method to create the curve is the one described in Chapter 2 that uses the simulated annealing to fit a piecewise cubic Bézier curve to a data set. Each cluster is going to be approximated by a segment of the piecewise curve, and the junction point is the start point of each cluster. Using as example the clusters from Fig. 5.29(b), the piecewise curve will have 4 segments, and the first segment will start in the cyan mark of purple cluster and end in the cyan mark of yellow cluster. The second segment start in the cyan mark of yellow cluster and end cyan mark of orange cluster. The third and forth segments use the same logic, and the end of the last segment is the start of the first segment. The approximating curve determined from the points shown in Fig. 5.29(b) is shown in Fig. 5.30. In this figure the points $\mathbf{d}_i$ are shown in yellow points, the piecewise

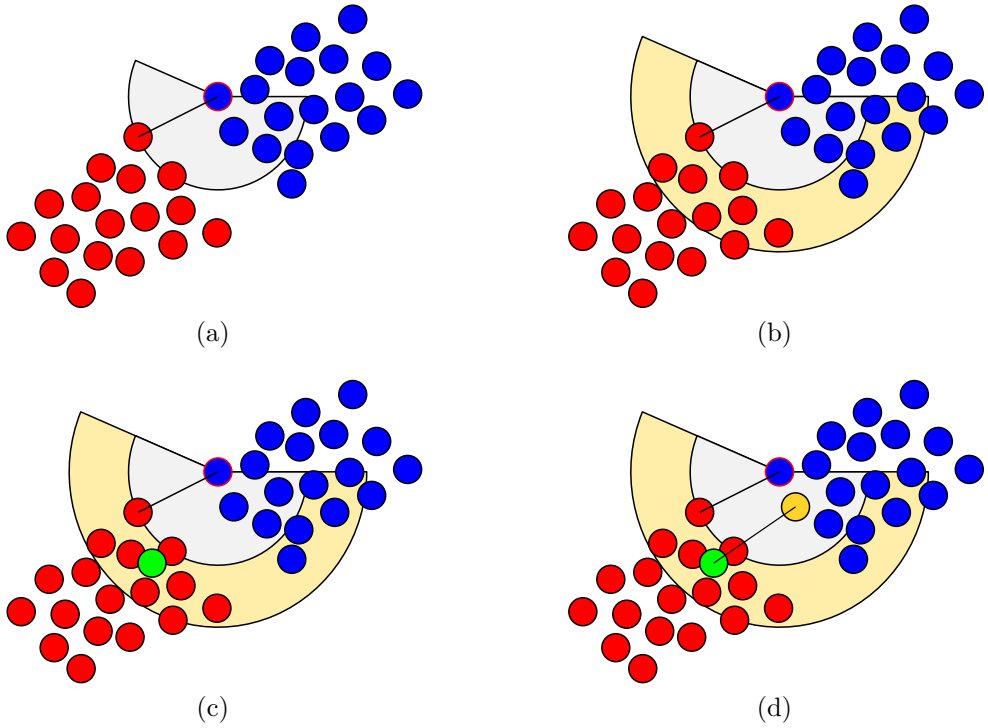


Figure 5.28 – Determination of junction points between two clusters. (a) red and blue circles are the extreme of two clusters, blue circle with red line is one of the extreme point determined in the search of the two furthest points inside a cluster; (b) There is a respective closest point in the other cluster linked with a black line. The gray arc is the region determined with the distance between these two points, and the gray arc radius added to 5 times the average edge length of the input STL file determines the yellow arc region; (c) The green circle is the mean point of all the red circles inside the yellow region; (d) The junction point of the blue cluster is the green circle nearest point from the blue cluster, and it is marked with a yellow circle.

curve is the red line and its control points are the black diamonds.

5.5 Stop Criteria and Determination of the Next Iteration Injured Zone Points

The next step verifies whether or not the injured zone points $\mathbf{d}_i$ determines a continuous surface without holes. It is possible to see that the points in Fig. 5.27 still needs some points to determine a continuous surface, there are still some missing points between blue and purple clusters. Thus, the algorithm uses the determined approximate curve as reference to determine the stop criteria. Each piecewise curve segment is sampled with 100 points and for each sampled point there is a respective nearest point in the injured zone points $\mathbf{d}_i$. If the distance between these points are greater than a limit, there is a hole in $\mathbf{d}_i$ and a new iteration is necessary. Fig. 5.31 shows an illustration, in which the

(a) (b)

Figure 5.29 – The junction points of each cluster, the junction points are shown in cyan and magenta diamond. (a) The junction points are the two furthest points inside a cluster; (b) the junction points are the ones determined by the proposed methodology.

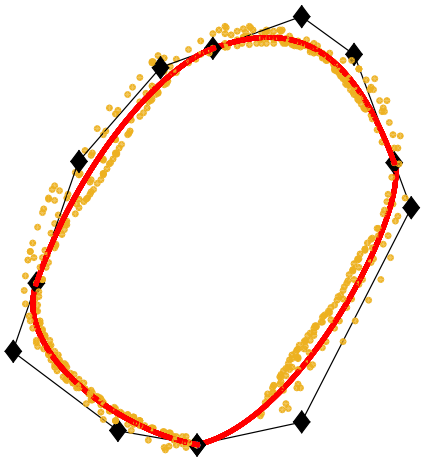


Figure 5.30 – Approximate curve from injured zone points $\mathbf{d}_i$, the red curve is the approximate curve, the black diamond are the curve control points and the yellow points are the injured zone points $\mathbf{d}_i$.

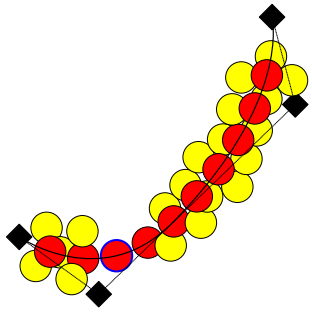


Figure 5.31 – The yellow circles are the current $\mathbf{d}_i$ points, the black line with black diamonds is the approximate curve segment and its control points, and the red circles are the curve sampled points. The red circle with blue contour line does not have a close yellow point, showing that the current $\mathbf{d}_i$ points still need to be completed.

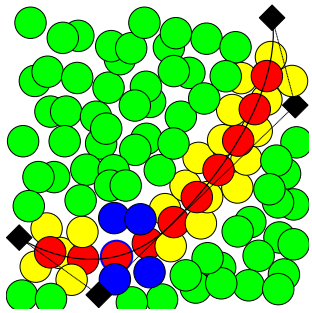


Figure 5.32 – The procedure to update $\mathbf{d}_i$. The yellow circles are the current $\mathbf{d}_i$, the black line with red circles are the approximate curve segment with its sampled points. The blue circles are the inlier, the new points to be added to $\mathbf{d}_i$; and the green circles are the outliers points filtered by the curve model.

approximating curve is the black line with its control point as black diamond, the curve sampled points are the red circles and part of the $\mathbf{d}_i$ points are the yellow circle. Note that the corresponding yellow circle closer to the red circle with blue contour line is greater than the others red circle, showing that there is a possible hole in this area.

The determination of the distance limit is locally done in each cluster. There is a corresponding nearest sampled point (red circle) for each point of the cluster (yellow circle). The average and the standard deviation of the distance between these points are used as stop criteria, let $\mu(dist)$ and $\sigma(dist)$ respectively be the average distance and the standard deviation. If the distance between a sampled curve point (red circle) and its respective neared point (yellow circle) be greater than $\mu(dist) + n.\sigma(dist)$ the algorithm must continue. n is an adjustable parameter that varied from $n = 3$ for easy examples (convex boundary) to $n = 1$ for difficult tests (concave boundary).

Therefore, the algorithm has a local stop criteria once each cluster has a different stop criteria, and the algorithm continues updating the injured zone points $\mathbf{d}_i$ only in the cluster that the stop criteria is not achieved.

In the procedure to determine the initial zone points ($\mathbf{d}_1$) a threshold t was used to

filter some points. This threshold t is set as 0 from the second iteration, meaning that $\mathbf{d}_i$ could be all points from Pc. However, the injured zone curve $\mathbf{P}_{i-1}$ of the previous iteration is used as reference model to determine the $\mathbf{d}_i$ points update. The update of $\mathbf{d}_i$ uses the curve segments that approximates the unfinished clusters as the filtration criteria. It is used the same limit of the stop criteria ($\mu(dist) + n.\sigma(dist)$). If the distance of a point from Pc is smaller than this limit, the point is inserted into $\mathbf{d}_i$ classified as inlier. Otherwise, it is classified as outlier and not included in $\mathbf{d}_i$. In the update of $\mathbf{d}_i$, the elimination of possible prior accepted outlier is performed. If a point of the cluster is greater than $\mu(dist) + n.\sigma(dist)$, it is eliminated from $\mathbf{d}_i$.

An illustration is shown in Fig. 5.32, the yellow circles are the $\mathbf{d}_i$ points from the previous iteration, the curve model is the black line with its control points in black diamonds; the red circles are the sampled points from the curve; the green circles are the outlier and the blue circle are the inliers. The $\mathbf{d}_i$ updated from Fig. 5.27 is shown in Fig. 6.35(a), the explanation of every element in the figure will be later done in the results.

Note that in the first iteration there is not a previous injured zone curve model ($\mathbf{P}_0 = null$). Therefore, the procedure to classify $\mathbf{d}_i$ as inlier and outlier is not performed in the first iteration. The process to filter the point is done by the modification of the threshold t to select the initial injured zone points from Pc.

Each new iteration will add new points to $\mathbf{d}_i$ only in clusters that the stop criteria is not achieved. However, a new cluster division is done after the classification of $\mathbf{d}_i$ in inliers and outliers. Thus, all $\mathbf{d}_i$ points with its respective cluster are stored in the end of every iteration, and a new cluster division is done after the addition of new points in the new iteration, erasing the current cluster division.

At the end of each iteration, it is verified if the number of k clusters is enough. The number of clusters is directly correlated to the number of curve segments that is used to create the injured zone model. If the number of k is high, the algorithm works well. However, the higher the number of k, the higher is the processing time of the curve fitting algorithm. On the other hand, if the number of k is low, the curve could not approximate the required points. Thus, the determination of a good number for k is important. However, this problem is still an open issue in the curve fitting problem. It is hard to determine the required minimum number of control points to solve the problem. Therefore, this verification is necessary at the end of each iteration, in which it is monitored weather the number of finished clusters decreases. This decreasing is an indication that the curve could not approximate the injured skull zone points. Thus, if the number of finished clusters decreases, the current iteration is discarded and k is increased by 1.

6 Results

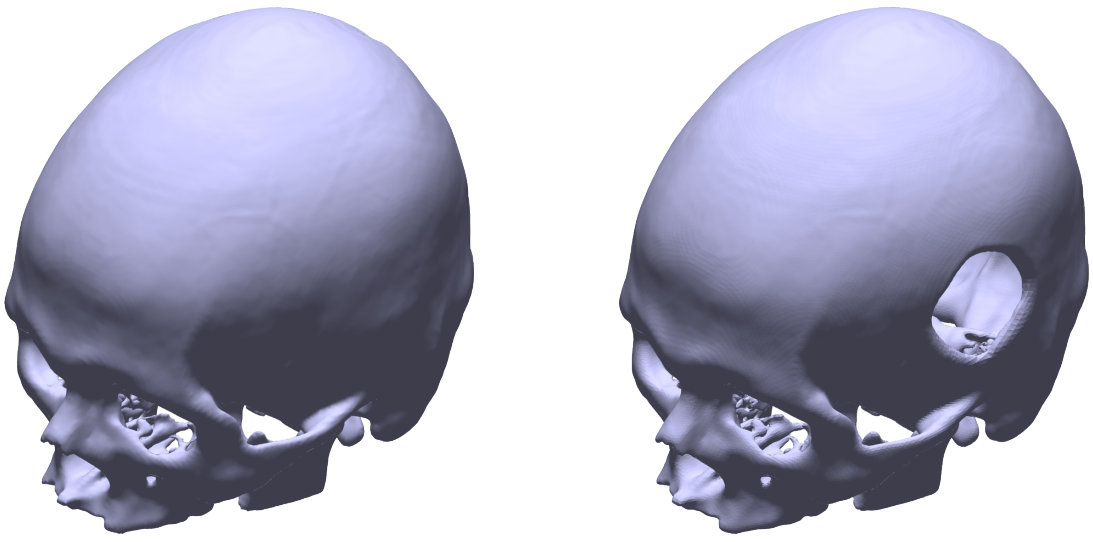
The proposed method to determine the injured skull zone is tested in 4 examples. The first three examples are artificially created injuries, in which a hole in a healthy skull was produced, and the last example is from a STL file with a injured skull.

6.1 Example 1

The first test is the simplest one. The injured skull is a convex zone created from a healthy skull, that are respectively shown in Figs. 6.33(b) and (a).

Fig. 6.34(a) shows the initial points ($\mathbf{d}_1$) of the injured zone. $\mathbf{d}_1$ has 557 points that are marked in red points. These points were determined using as threshold $t = 15$, and it is possible to see that in the lower region of the injury there are still some missing points. The number of clusters k is set as 4 and Fig. 6.34(b) shows the cluster division of $\mathbf{d}_1$, in which each cluster is represented in a different color and the junction points are the cyan diamond from each cluster. The injured zone curve model $\mathbf{P}_1$ determined from $\mathbf{d}_1$ is shown in Fig. 6.34(c), in which $\mathbf{d}_1$ is shown in yellow points and the curve in red line with its control points in black diamond.

Fig. 6.35(a) shows the second iteration and also shows that the algorithm is finished in 3 clusters using as stop criteria $\mu(dist) + 3.\sigma(dist)$, in which $\mu(dist)$ is the average distance of $\mathbf{d}_1$ to the curve and $\sigma(dist)$ is its standard deviation. Note that it is not verified if the number of clusters is enough in the first iteration, once there is no previous iteration to



(a) (b)

Figure 6.33 – First injured skull example. (a) Healthy skull; (b) Injured skull.

Table 1 – First example results. $N.Pts$ is the number of points of $\mathbf{d}_i$; $Rem.\mathbf{d}_{i-1}$ is the number of points removed from $\mathbf{d}_{i-1}$; *Inlier* and *Outlier* are respectively the number of points classified as inliers and outliers, in which the inliers are added to $\mathbf{d}_i$; $N.F.k$ is the number of cluster in which the stop criteria is not achieved; and $N.k$ is the number of clusters used to divide $\mathbf{d}_i$.

Iteration	N. Pts	Rem. $\mathbf{d}_{i-1}$	Inlier	Outlier	N.F. k	N. k
1	557	–	–	–	1	4
2	606	20	69	22,879	0	4

compare whether or not the number of finished clusters decreases.

Fig. 6.35(a) shows $\mathbf{d}_2$, in which the red points are the $\mathbf{d}_1$, the blue points are the inliers and the green points are the filtered outliers. These inliers and outliers were classified using as reference the curve shown in Fig. 6.34(c) with a threshold of $\mu(dist) + 3.\sigma(dist)$. $\mathbf{d}_2$ is composed by 606 points, in which 69 points were added and $22,879$ points were filtered in this iteration. Note that the difference between $\mathbf{d}_2$ and $\mathbf{d}_1$ are not the same number of added points, showing that 20 points were removed from $\mathbf{d}_1$. Fig. 6.35(b) and (c) are respectively the cluster division of $\mathbf{d}_2$ and the injured zone curve model. The algorithm achieved the stop criteria in all 4 clusters, and Fig. 6.36(a) shows the final result.

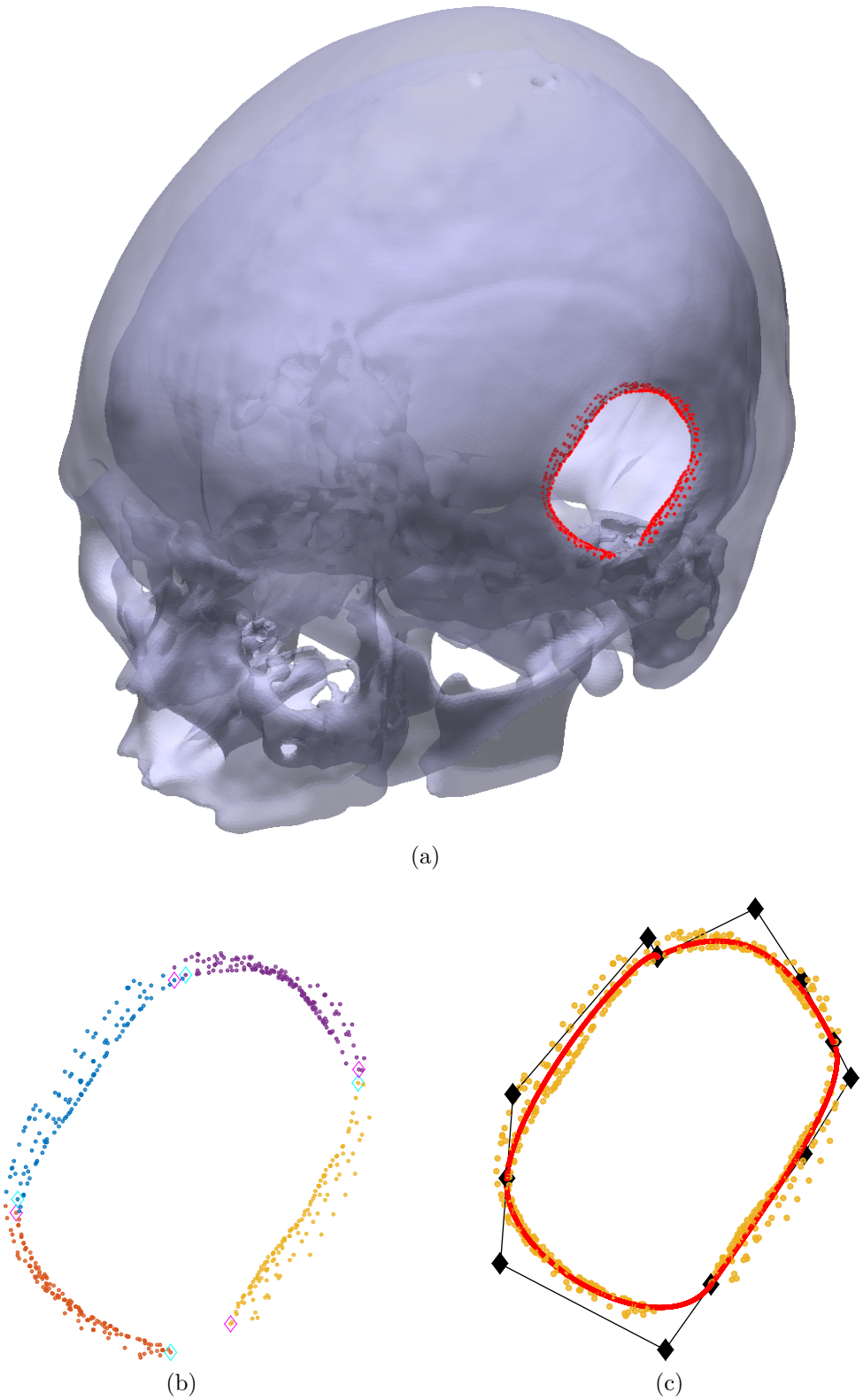
Once the injury is created from a healthy skull, it is possible to determine the geometry that fills the injury hole by a simple subtraction of the injured skull and the healthy skull. Fig. 6.36(b) shows the solid of this subtraction. The points of the injured skull zone are shown in red points in Fig. 6.36(a) and (b). Table. 1 shows the results of the first example, with $N.Pts$ being the number of points of $\mathbf{d}_i$, $Rem.\mathbf{d}_{i-1}$ the number of points removed from the previous iteration, $N.F.k$ the number of clusters that the stop criteria is not achieved and $N.k$ the number of clusters.

6.2 Example 2

The second example is another artificially created injury. This example is created from the same healthy skull of the first example. In this example, different from the previous example, the injury was created in the right side of the healthy skull with a concave zone, and the healthy and injured skull are shown respectively in Figs. 6.37(a) and (b).

The initial points ($\mathbf{d}_1$) of the injured zone are shown in Fig. 6.38(a), with $\mathbf{d}_1$ being the 569 red points marked in the figure. These points were determined using as threshold $t = 12$, and it is possible to see that there are still some missing points in the upper region of the injury. In this example it is also used $k = 4$ for the number of clusters. Figs. 6.38(b) and (c) are respectively the division in clusters of d_1 and the injury curve model P_1. It was used $\mu(dist) + 2.\sigma(dist)$ as stop criteria and as threshold to classify d_i in inlier or outlier, and with this stop criteria 3 clusters have achieved the stop criteria.

Fig. 6.39(a) shows the 647 points of $\mathbf{d}_2$, in which the red points are the points from

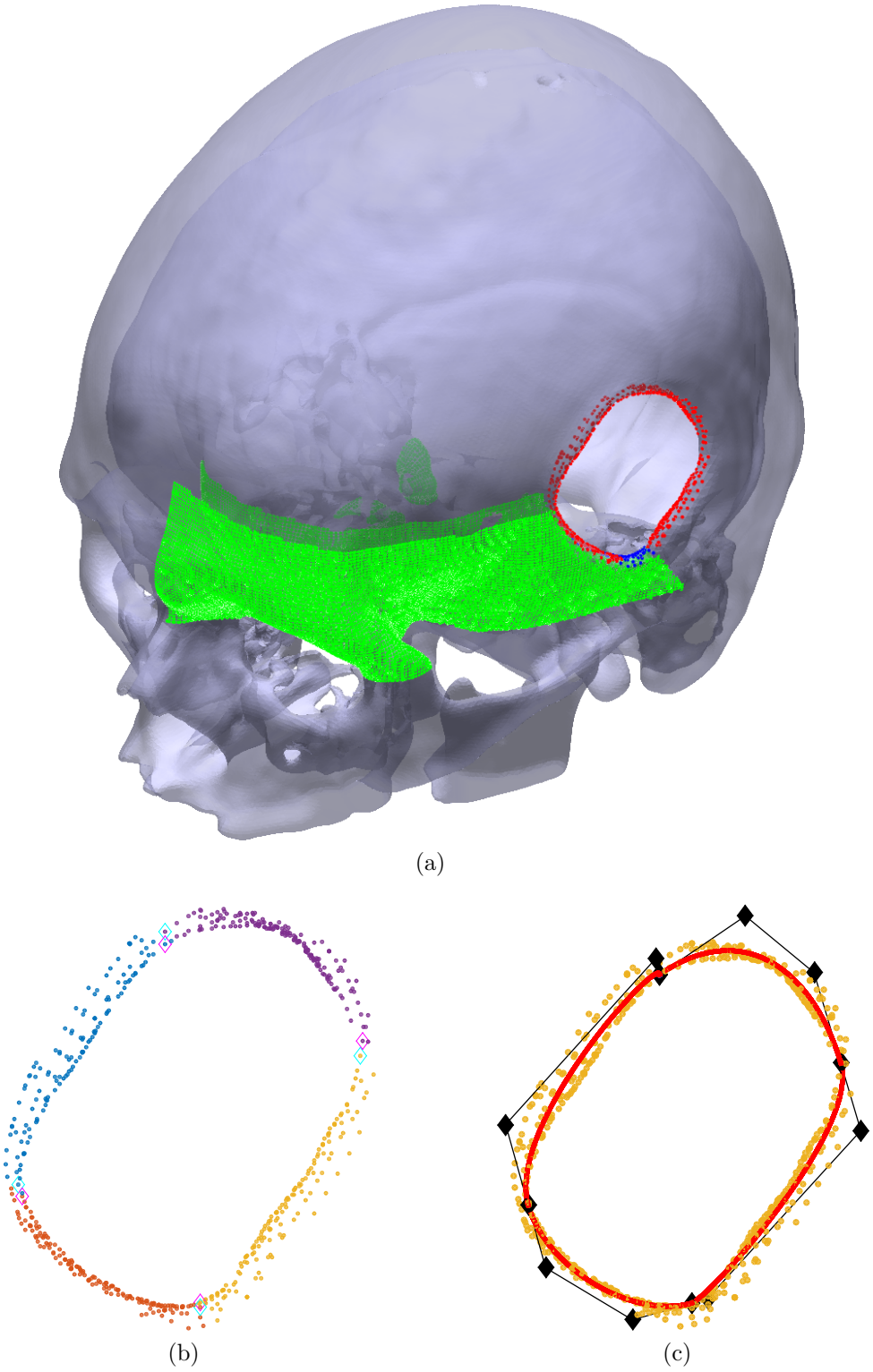


(a)

(b)

(c)

Figure 6.34 – Iteration 1. (a) $\mathbf{d}_1$ is represented with red points; (b) Each cluster of $\mathbf{d}_1$ is in different color; (c) Injured zone curve model $\mathbf{P}_1$ is the red curve, the control points are the black diamonds and $\mathbf{d}_1$ are the yellow points.

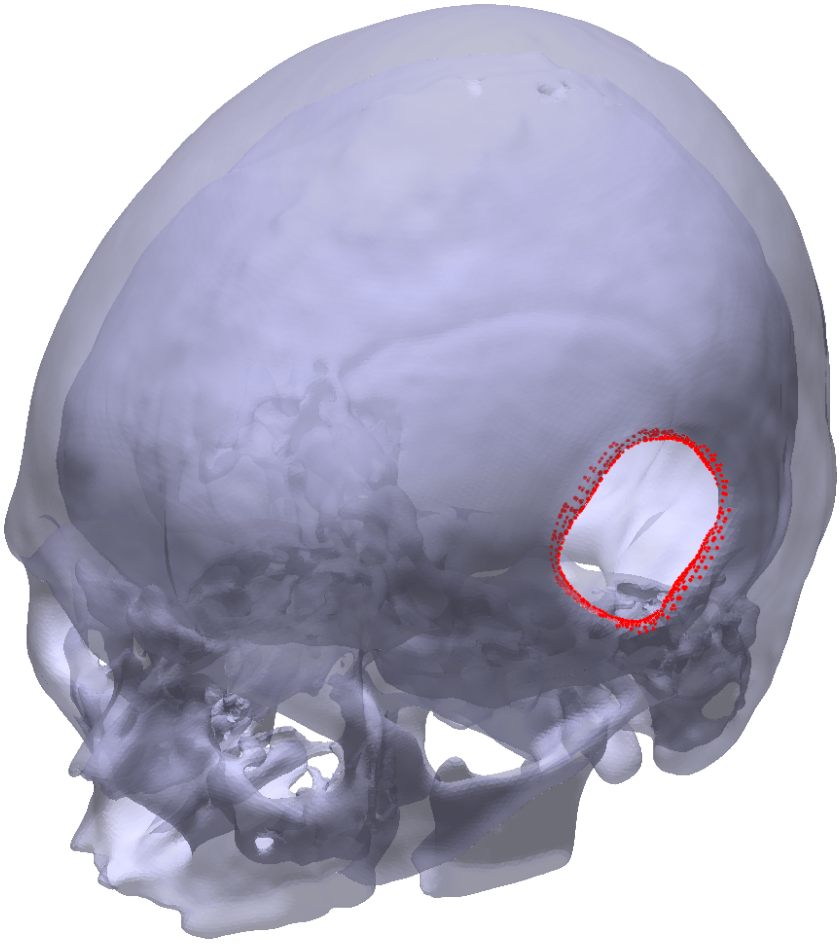


(a)

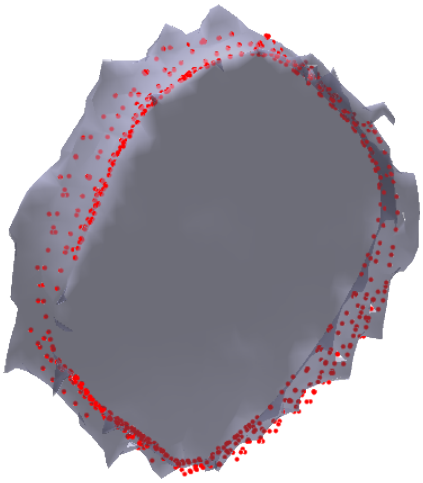
(b)

(c)

Figure 6.35 – Iteration 2. (a) $\mathbf{d}_1$ is represented with red points, the inliers added to $\mathbf{d}_2$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_2$ is in different color; (c) Injured zone curve model $\mathbf{P}_2$ is the red curve, the control points are the black diamonds and $\mathbf{d}_2$ are the yellow points.

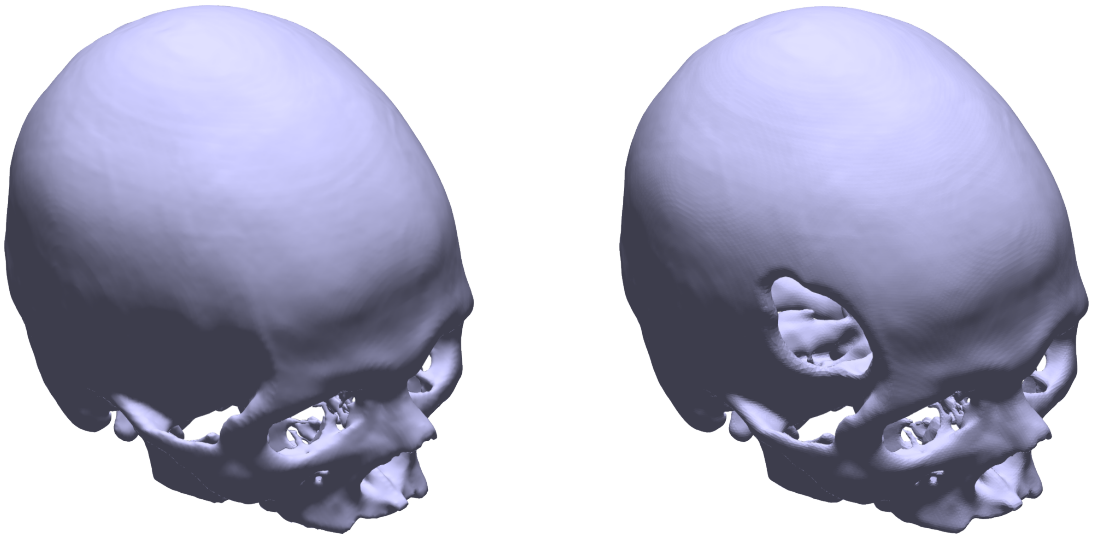


(a)



(b)

Figure 6.36 – Injured skull zone are the red points. (a) Injured skull; (b) Solid obtained by the subtraction of the injured skull and the healthy skull.



(a) (b)

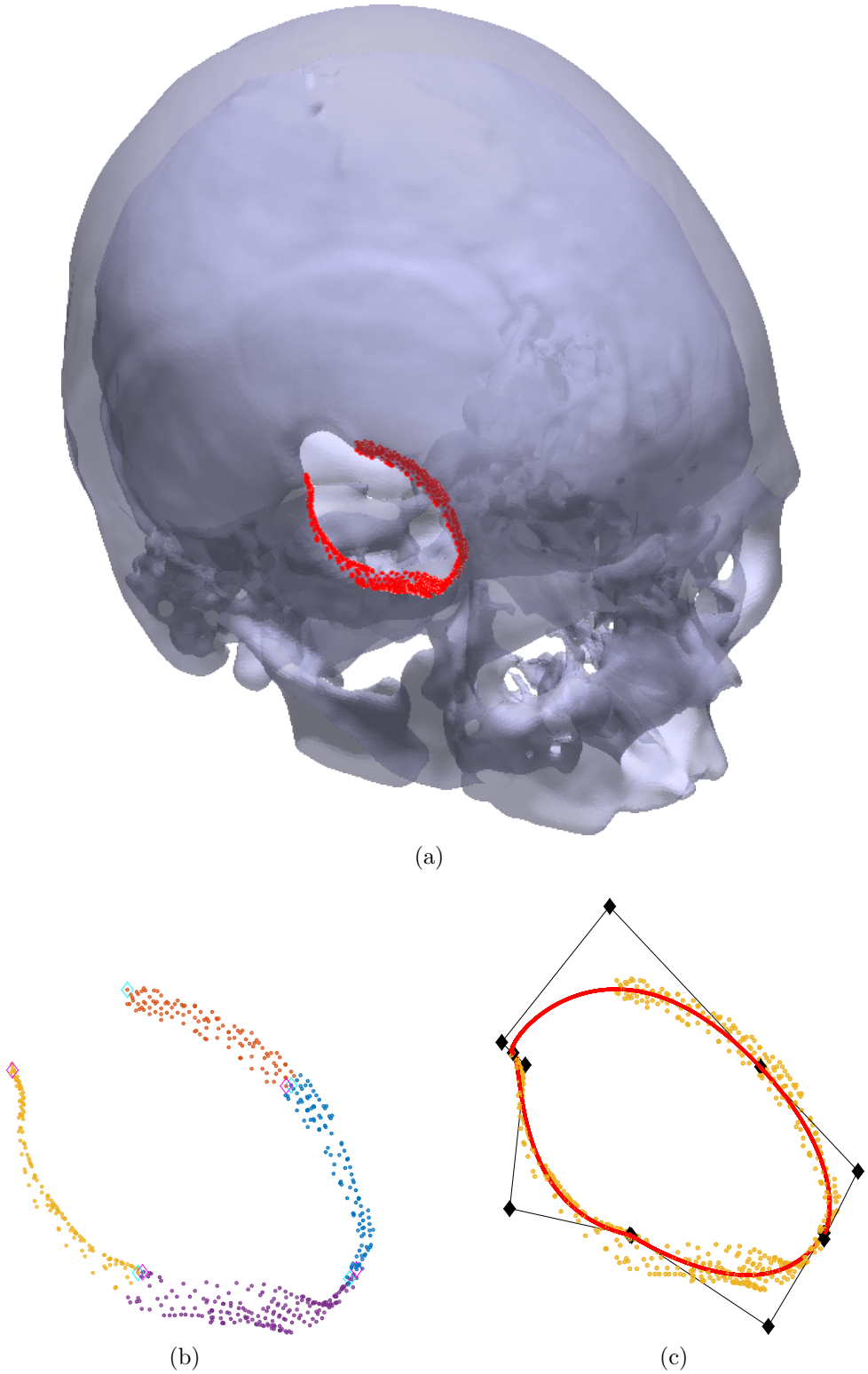
Figure 6.37 – Second injured skull example. (a) Healthy skull; (b) Injured skull.

$\mathbf{d}_1$, the blue points are the 78 inliers points added to $\mathbf{d}_2$, the green points are the $41,502$ filtered outliers points, and there are no points eliminated from $\mathbf{d}_1$. $\mathbf{d}_2$ clusters division and injury curve model $\mathbf{P}_2$ are shown respectively in Figs. 6.39(b) and (c). It was used the same stop criteria of the first iteration, and once again only one cluster is not finished, showing that it is necessary another iteration. The number of clusters is verified and it is appropriate.

The third iteration is shown in Fig. 6.40(a), in which $\mathbf{d}_2$ are the red points, the blue points are the 131 inliers points added to $\mathbf{d}_3$, the green points are the $63,288$ filtered outliers points, and one point was eliminated from $\mathbf{d}_2$. $\mathbf{d}_3$ clusters division and injury curve model $\mathbf{P}_3$ are shown respectively in Figs. 6.40(b) and (c). There is only one cluster to be completed, and it is possible to see that the yellow cluster is almost filled.

Fig. 6.41(a) shows the forth iteration and the 914 points of $\mathbf{d}_4$, in which the red points are the points from $\mathbf{d}_3$, the blue points are the 137 inliers points added to $\mathbf{d}_4$, the green points are the $33,551$ filtered outliers points, and there are no points eliminated from $\mathbf{d}_3$. $\mathbf{d}_4$ clusters division and injury curve model $\mathbf{P}_4$ are shown respectively in Figs. 6.41(b) and (c). All the four cluster are completed in this iteration and the final result is shown in Fig.6.42.

It is possible to determine the geometry of the missing part of the injured skull by the subtraction of the healthy and injured skull from Figs. 6.37(a) and (b). The injured skull zone ($\mathbf{d}_4$) is shown in red points in the injured skull and the missing part geometry respectively in Figs.6.42(a) and (b). Table. 2 shows the results of the second example, with $N.Pts$ being the number of points of $\mathbf{d}_i$, $Rem.\mathbf{d}_{i-1}$ the number of points removed from the previous iteration, $N.F.k$ the number of clusters that the stop criteria is not achieved, and $N.k$ the number of clusters.

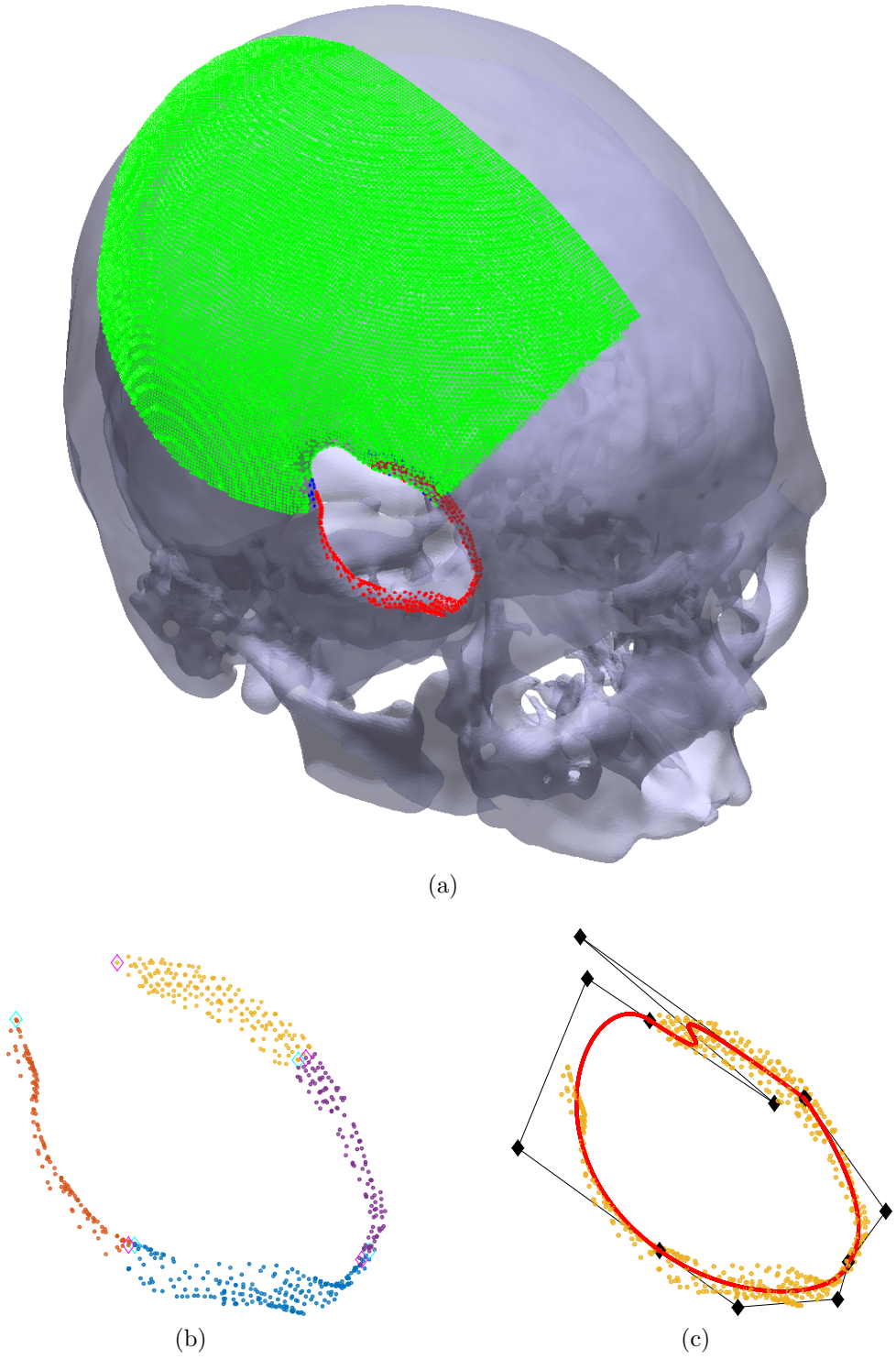


(a)

(b)

(c)

Figure 6.38 – Iteration 1. (a) $\mathbf{d}_1$ is represented with red points; (b) Each cluster of $\mathbf{d}_1$ is in different color; (c) Injured zone curve model $\mathbf{P}_1$ is the red curve, the control points are the black diamonds and $\mathbf{d}_1$ are the yellow points.

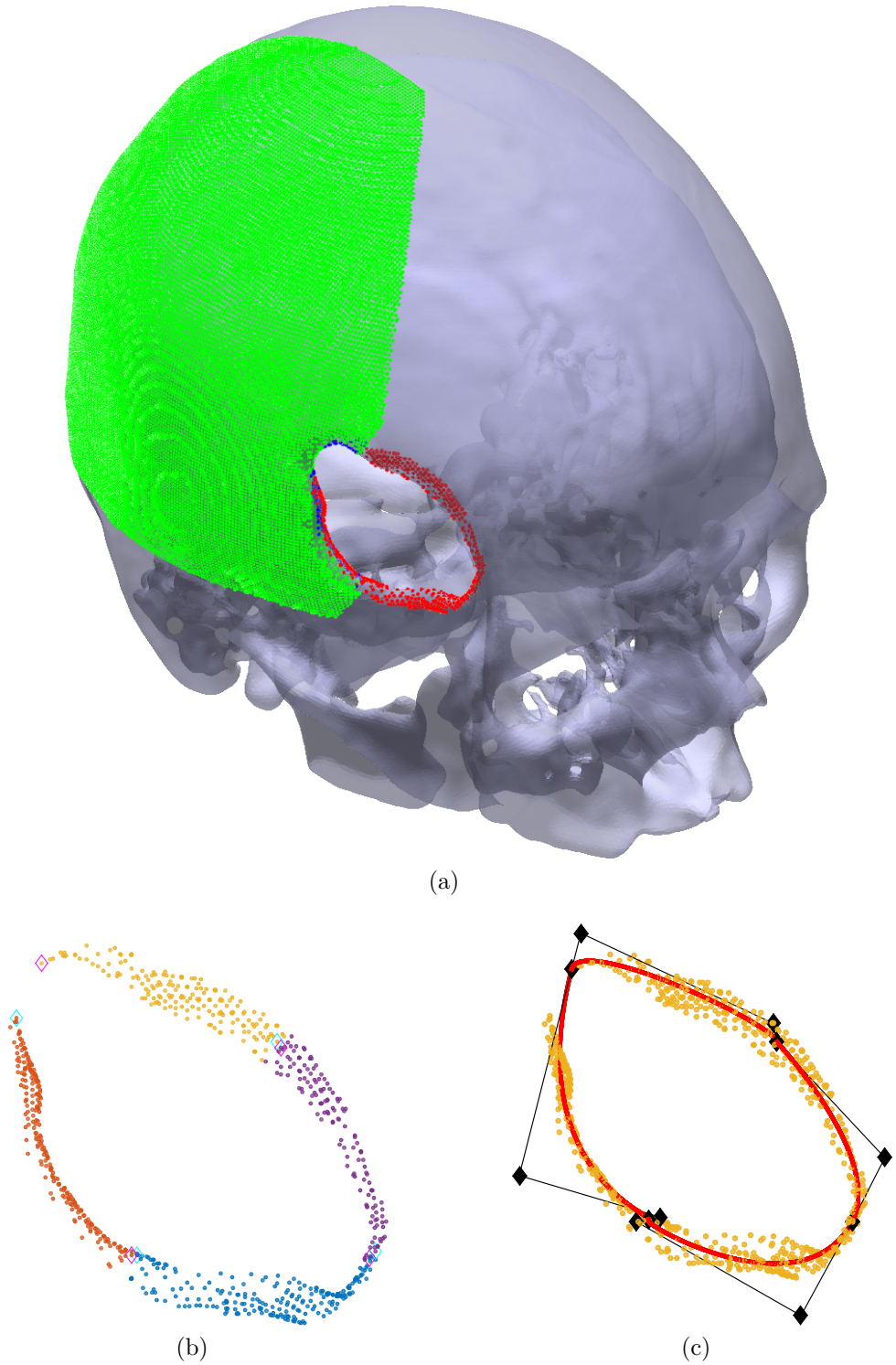


(a)

(b)

(c)

Figure 6.39 – Iteration 2. (a) $\mathbf{d}_1$ is represented with red points, the inliers added to $\mathbf{d}_2$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_2$ is in different color; (c) Injured zone curve model $\mathbf{P}_2$ is the red curve, the control points are the black diamonds and $\mathbf{d}_2$ are the yellow points.

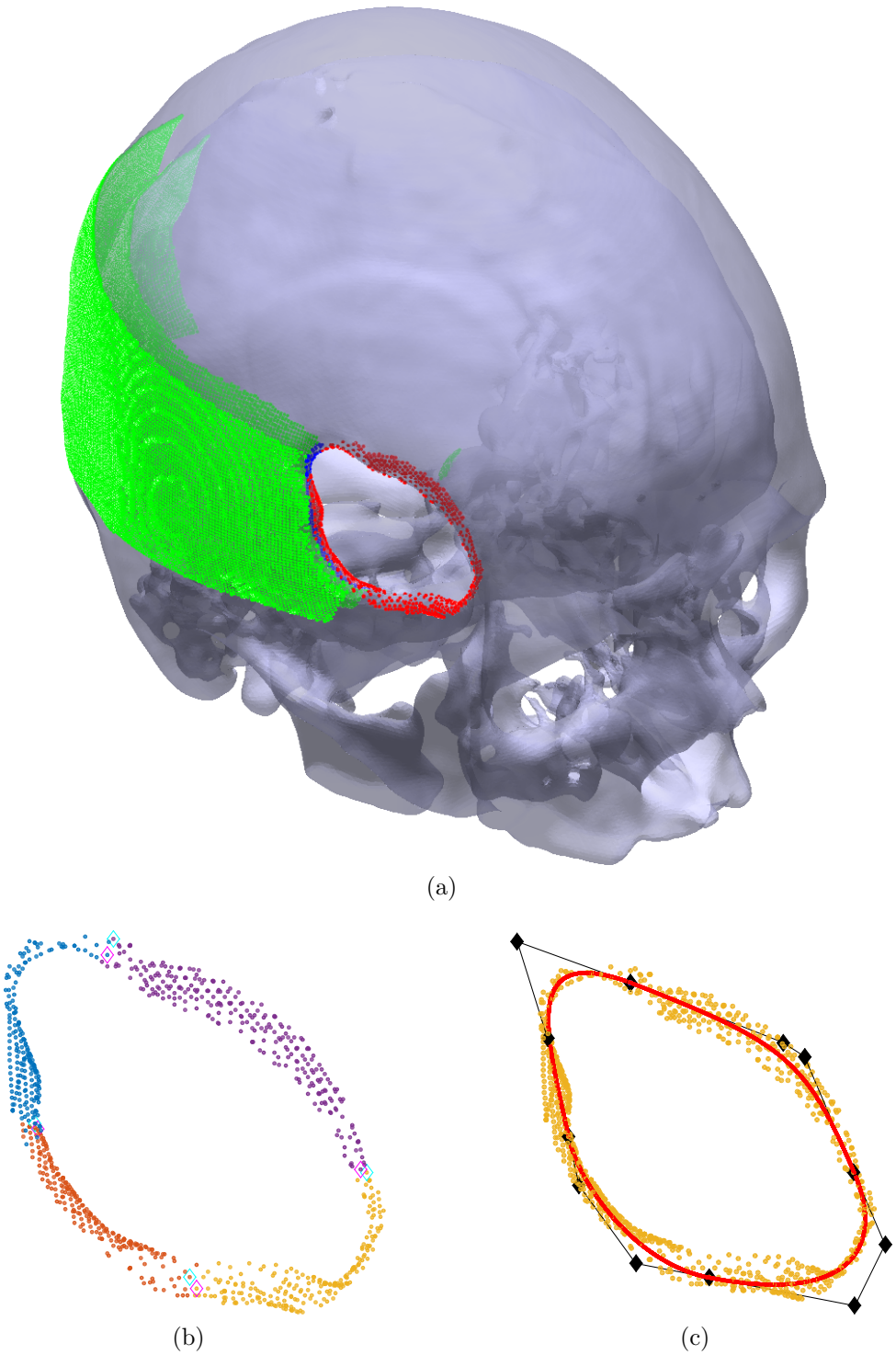


(a)

(b)

(c)

Figure 6.40 – Iteration 3. (a) $\mathbf{d}_2$ is represented with red points, the inliers added to $\mathbf{d}_3$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_3$ is in different color; (c) Injured zone curve model $\mathbf{P}_3$ is the red curve, the control points are the black diamonds and $\mathbf{d}_3$ are the yellow points.

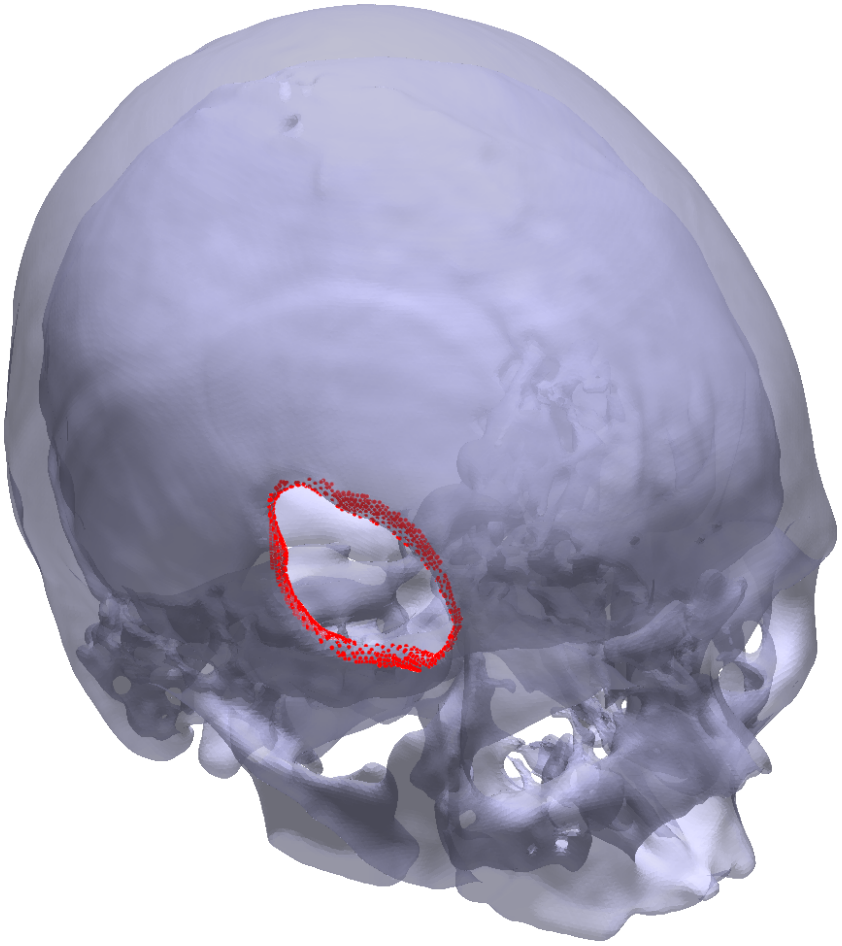


(a)

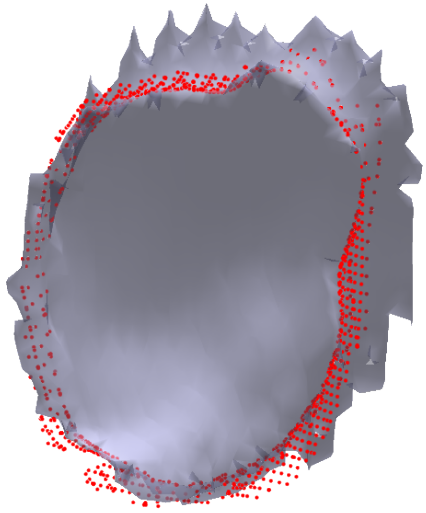
(b)

(c)

Figure 6.41 – Iteration 4. (a) $\mathbf{d}_3$ is represented with red points, the inliers added to $\mathbf{d}_4$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_4$ is in different color; (c) Injured zone curve model $\mathbf{P}_4$ is the red curve, the control points are the black diamonds and $\mathbf{d}_4$ are the yellow points.



(a)

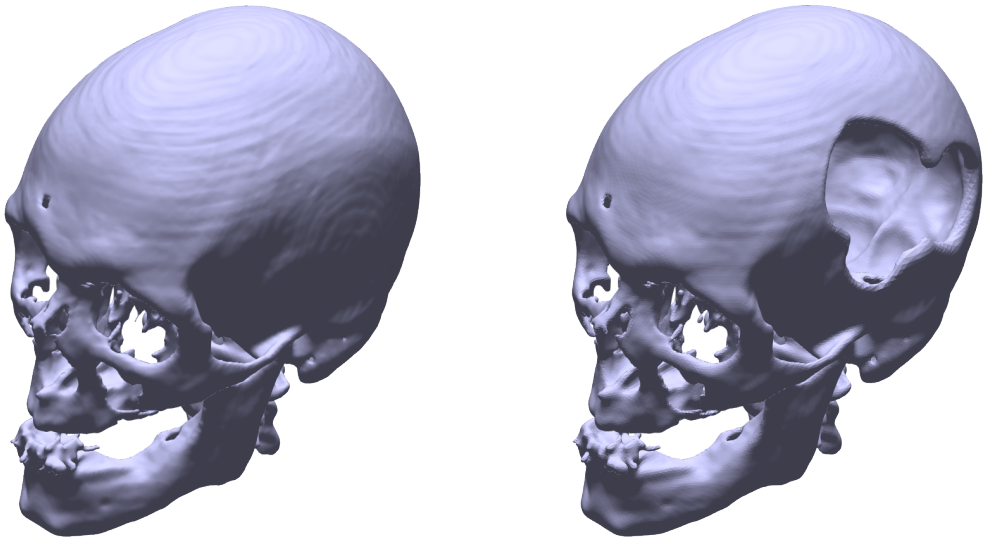


(b)

Figure 6.42 – Injured skull zone are the red points. (a) Injured skull; (b) Solid obtained by the subtraction of the injured skull and the healthy skull.

Table 2 – Second example results. $N.Pts$ is the number of points of $\mathbf{d}_i$; $Rem.\mathbf{d}_{i-1}$ is the number of points removed from $\mathbf{d}_{i-1}$; *Inlier* and *Outlier* are respectively the number of points classified as inliers and outliers, in which the inliers are added to $\mathbf{d}_i$; $N.F.k$ is the number of cluster in which the stop criteria is not achieved; and $N.k$ is the number of clusters used to divide $\mathbf{d}_i$.

Iteration	N. Pts	Rem. $\mathbf{d}_{i-1}$	Inlier	Outlier	N.F. k	N. k
1	569	–	–	–	1	4
2	647	0	78	41,502	1	4
3	777	1	131	63,288	1	4
4	914	0	137	33,551	0	4



(a) (b)

Figure 6.43 – Third injured skull example. (a) Healthy skull; (b) Injured skull.

6.3 Example 3

The third example is another artificially created injury. This example is created from another healthy skull. In this example, the injury was created in the left side of the healthy skull with a more complex concave zone, and the healthy and injured skull are shown respectively in Figs. 6.43(a) and (b).

The initial points ($\mathbf{d}_1$) of the injured zone are shown in Fig. 6.44(a), with $\mathbf{d}_1$ being the 1320 red points marked in this figure. These points were determined using as threshold $t = 15$, and it is possible to see that there are some missing points in the lower part of the injury. Different from the previous two examples, it is used $k = 8$ for the number of clusters in the third example. Figs. 6.44(b) and (c) are respectively the division in clusters of $\mathbf{d}_1$ and the injury curve model $\mathbf{P}_1$. It was used $\mu(dist) + 2.\sigma(dist)$ as stop criteria and as threshold to classify $\mathbf{d}_i$ in inlier or outlier, and with this stop criteria 7 clusters have achieved the stop criteria, just the burgundy cluster has some missing points.

Fig. 6.45(a) shows the 1361 points of $\mathbf{d}_2$, in which the red points are the points from

Table 3 – Third example results. $N.Pts$ is the number of points of $\mathbf{d}_i$; $Rem.\mathbf{d}_{i-1}$ is the number of points removed from $\mathbf{d}_{i-1}$; *Inlier* and *Outlier* are respectively the number of points classified as inliers and outliers, in which the inliers are added to $\mathbf{d}_i$; $N.F.k$ is the number of cluster in which the stop criteria is not achieved; and $N.k$ is the number of clusters used to divide $\mathbf{d}_i$.

Iteration	N. Pts	Rem. $\mathbf{d}_{i-1}$	Inlier	Outlier	N.F. k	N. k
1	1,320	–	–	–	1	8
2	1,361	26	67	14,852	0	8

$\mathbf{d}_1$, the blue points are the 67 inliers points added to $\mathbf{d}_2$, the green points are the $14,852$ filtered outliers points, and there are 26 points eliminated from $\mathbf{d}_1$. $\mathbf{d}_2$ clusters division and injury curve model $\mathbf{P}_2$ are shown respectively in Figs. 6.45(b) and (c). There are no points to input in $\mathbf{d}_2$, all the 8 clusters are completed. Therefore, the detection is completed.

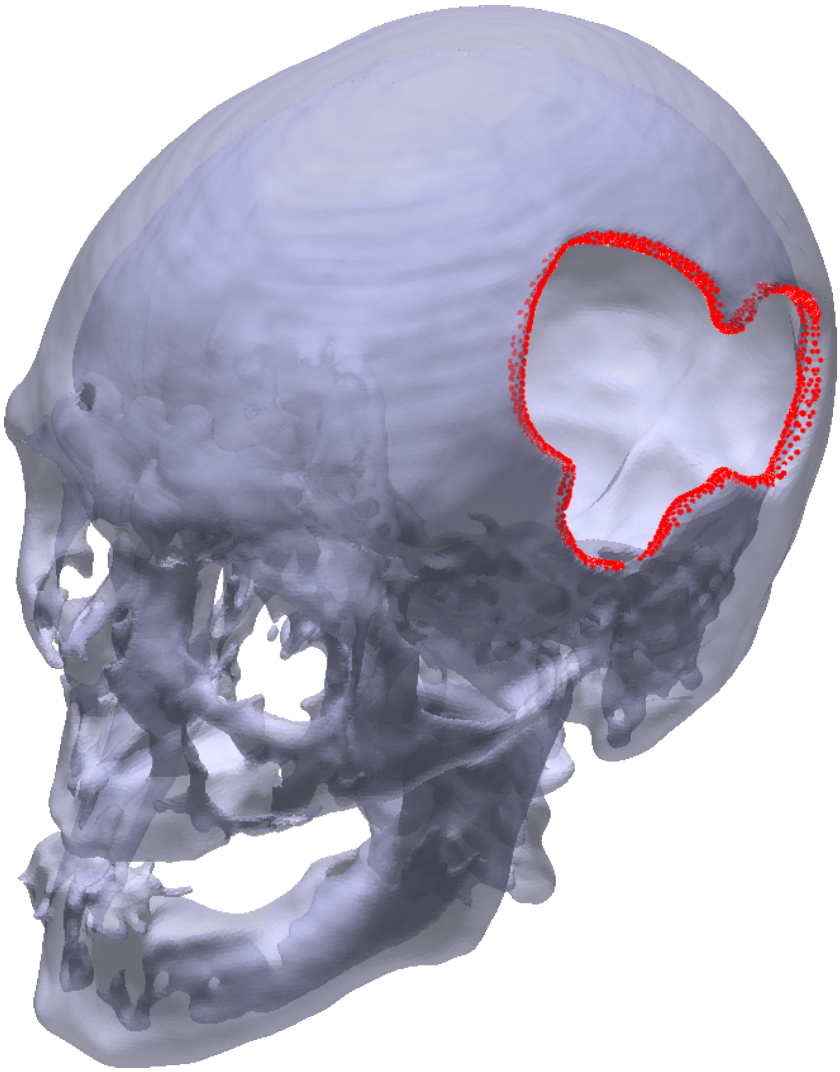
It is possible to determine the geometry of the missing part of the injured skull by the subtraction of the healthy and injured skull from Figs. 6.43(a) and (b). The injured skull zone ($\mathbf{d}_2$) is shown in red points in the injured skull and the missing part geometry respectively in Figs.6.46(a) and (b). Table. 3 shows the results of the third example, with $N.Pts$ being the number of points of $\mathbf{d}_i$, $Rem.\mathbf{d}_{i-1}$ the number of points removed from the previous iteration, $N.F.k$ the number of clusters that the stop criteria is not achieved and $N.k$ the number of clusters.

It was expected that this example could be more difficult than the second example, once the second example just had two concave areas and the third example had more than four. However, this example determines the injured zone in just 2 iterations, while it was necessary to have 4 iteration in the second example. This difference is especially due to the initial points $\mathbf{d}_1$. While these points are almost complete in the third example, there are a bigger region to be filled in the second example. Thus, determining a good initial points $\mathbf{d}_1$ speed up the algorithm, in which less iterations is necessary.

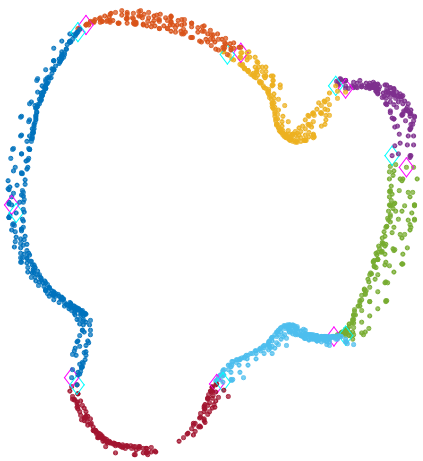
6.4 Example 4

In the previous three examples the injury was artificially created, while the last example is the one with a natural injury. The artificial injury is smoother than the natural injury, thus it is expected to have a higher number of iterations to solve this example. Fig. 6.47 shows the forth injured skull test. This example, as expected, is bigger than the previous three examples and the injured zone has more details.

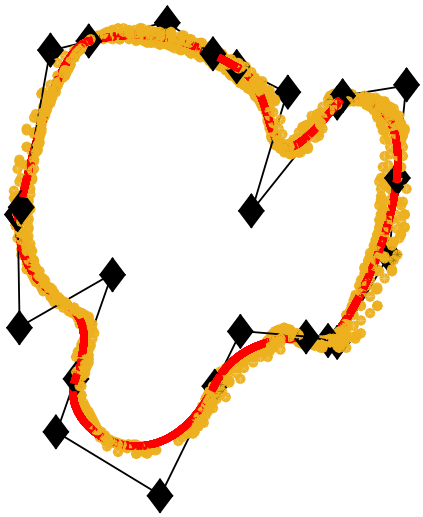
The initial points ($\mathbf{d}_1$) of the injured zone are shown in Fig. 6.48(a), with $\mathbf{d}_1$ being the $7,054$ red points marked in this figure. It is used $k = 10$ as number of clusters, and Figs. 6.44(b) and (c) are respectively the cluster division of $\mathbf{d}_1$ and the injury curve model $\mathbf{P}_1$. These points were determined using as threshold $t = 20$, and it is possible to see that



(a)

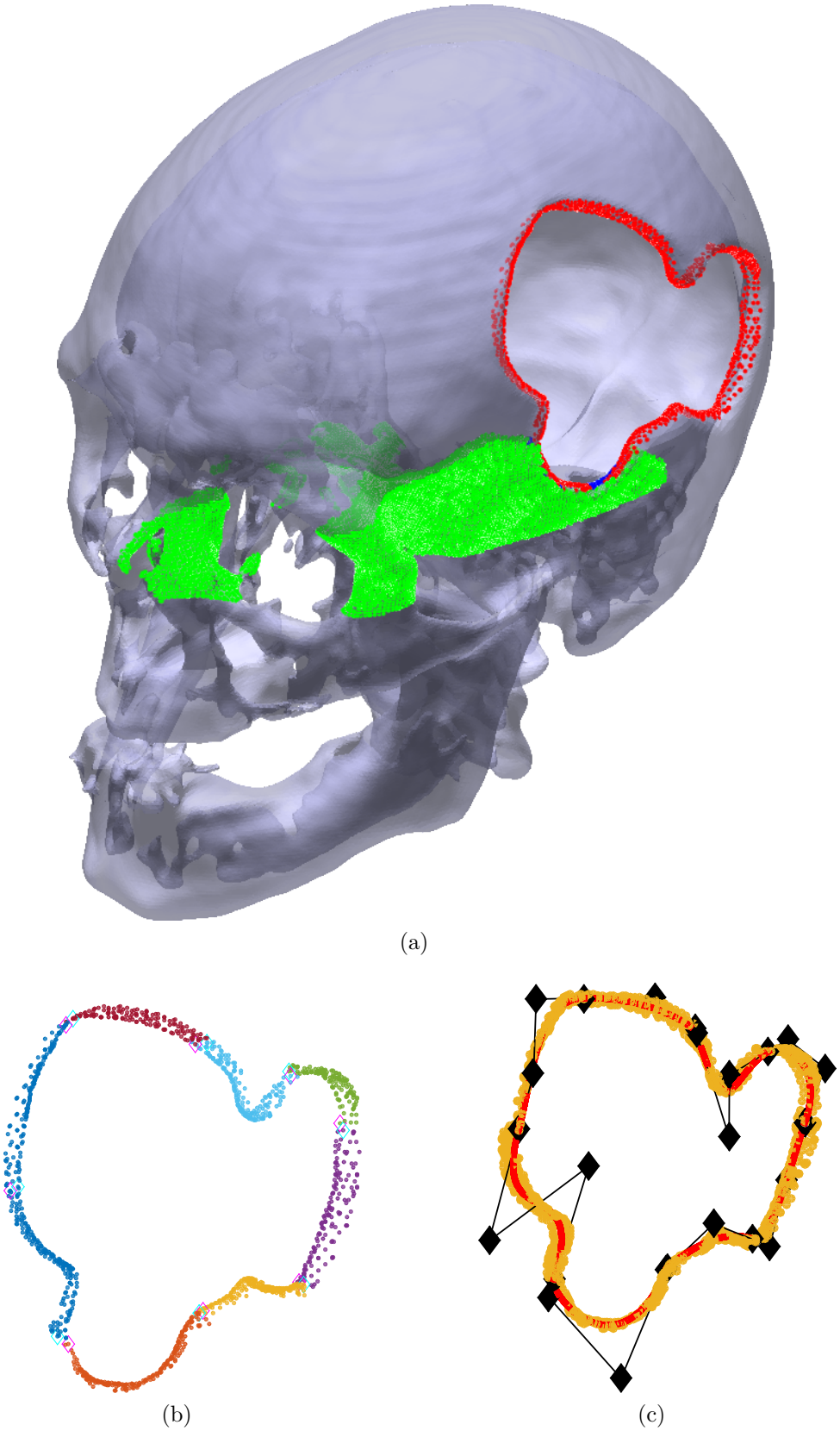


(b)



(c)

Figure 6.44 – Iteration 1. (a) $\mathbf{d}_1$ is represented with red points; (b) Each cluster of $\mathbf{d}_1$ is in different color; (c) Injured zone curve model $\mathbf{P}_1$ is the red curve, the control points are the black diamonds and $\mathbf{d}_1$ are the yellow points.

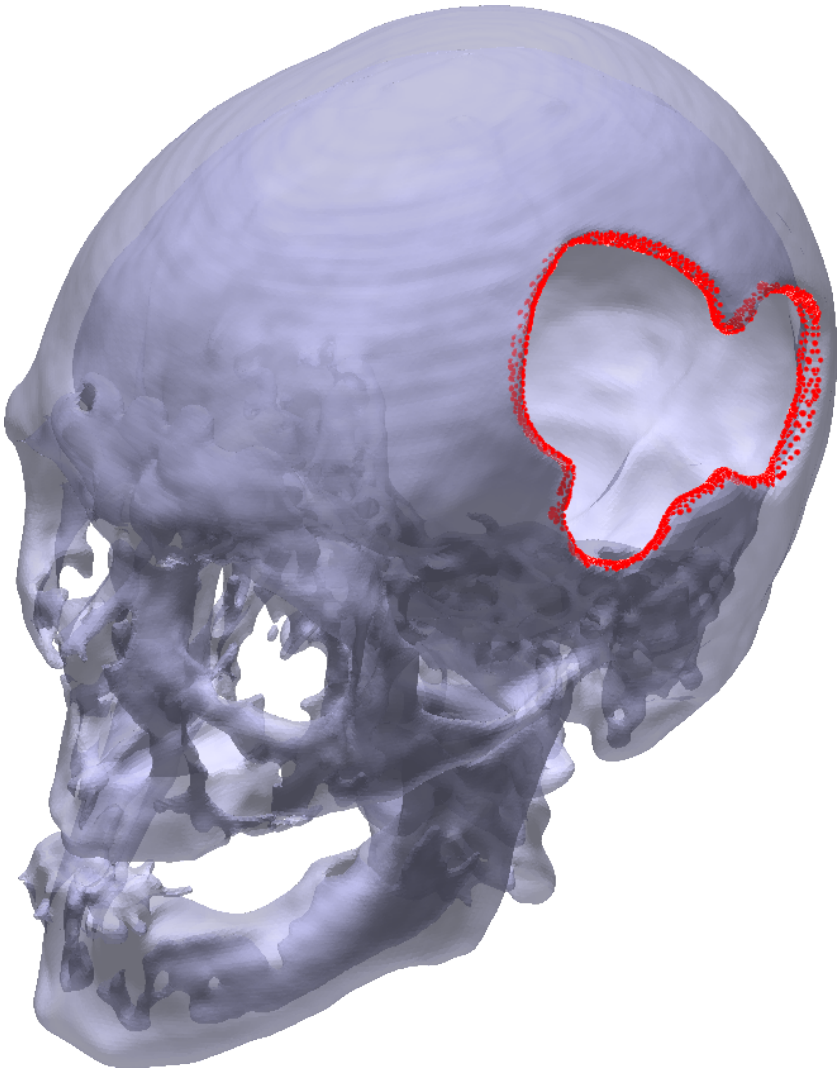


(a)

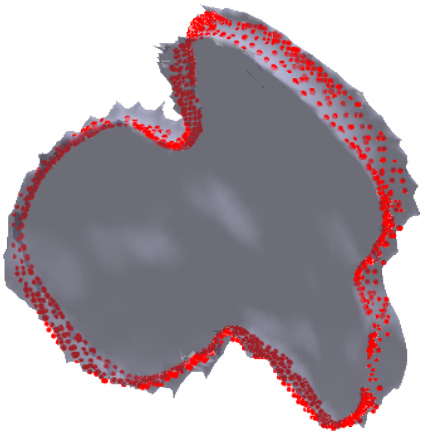
(b)

(c)

Figure 6.45 – Iteration 2. (a) $\mathbf{d}_1$ is represented with red points, the inliers added to $\mathbf{d}_2$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_2$ is in different color; (c) Injured zone curve model $\mathbf{P}_2$ is the red curve, the control points are the black diamonds and $\mathbf{d}_2$ are the yellow points.



(a)



(b)

Figure 6.46 – Injured skull zone are the red points. (a) Injured skull; (b) Solid obtained by the subtraction of the injured skull and the healthy skull.

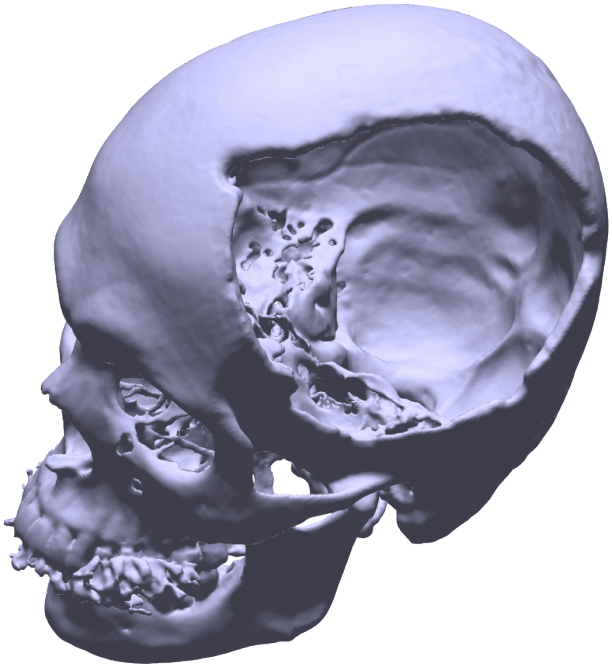


Figure 6.47 – Forth injured skull example.

there are several regions with missing points; in both yellow clusters, the division between the green and light blue clusters, the division between the yellow and purple cluster, and the division between the blue and orange clusters. It was used $\mu(dist) + 1.\sigma(dist)$ as stop criteria, and just 3 clusters have achieved such condition, remaining 7 clusters to be filled.

Fig. 6.49(a) shows the $6,719$ points of $\mathbf{d}_2$, in which the red points are the points from $\mathbf{d}_1$, the blue points are the 563 inliers points added to $\mathbf{d}_2$, the green points are the $134,691$ filtered outliers points, and there are 898 points eliminated from $\mathbf{d}_1$. $\mathbf{d}_2$ clusters division and injury curve model $\mathbf{P}_2$ are shown respectively in Figs. 6.49(b) and (c). The number of $\mathbf{d}_i$ decreases from the first to the second iteration, once 563 points were included and 898 were excluded. The included points are the points that filled the missing part of the injured zone, while the excluded points are the ones from the zone boundary, making it thinner. This behaviour is expected once the stop criteria is lower than the previous three examples. There are still 7 clusters to be filled and the next iteration is show in Fig. 6.50.

Fig. 6.50(a) shows the $6,109$ points of $\mathbf{d}_3$, in which the red points are the points from $\mathbf{d}_2$, the blue points are the 202 inliers points added to $\mathbf{d}_3$, the green points are the $91,295$ filtered outliers points, and there are 812 points eliminated from $\mathbf{d}_2$. $\mathbf{d}_3$ clusters division and injury curve model $\mathbf{P}_3$ are shown respectively in Figs. 6.50(b) and (c). Once again, there are more eliminated points than added points, and there are still 7 clusters to be completed.

Fig. 6.51(a) shows the forth iteration, in which the red points are the points from $\mathbf{d}_3$, the blue points are the 670 inliers points added to $\mathbf{d}_4$, the green points are the $131,726$ filtered outliers points, and there are 784 points eliminated from $\mathbf{d}_3$. Thus, $\mathbf{d}_4$ is composed by $5,995$ points whose cluster division is shown in Fig. 6.51(b). The injury curve model

$\mathbf{P}_4$ is shown in Fig. 6.51(c). There are still 7 clusters to be finished. There are clusters on the lower part (yellow and purple) almost completed, however there are still four big gaps in the injured zone.

Fig. 6.52(a) shows iteration 5, in which the red points are the points from $\mathbf{d}_4$, the blue points are the 724 inliers points added to $\mathbf{d}_5$, the green points are the $118,662$ filtered outliers points, and there are 745 points eliminated from $\mathbf{d}_4$, resulting $5,974$ points in $\mathbf{d}_5$. The clusters division and injury curve model $\mathbf{P}_5$ are shown respectively in Figs. 6.52(b) and (c). Once again there are 7 clusters to be finished. This iteration added a lot of points, but added points from the previous iteration that clearly are points of the injured zone were removed (the lower region between orange and blue cluster). This is due to the poor curve model $\mathbf{P}_4$ that in regions with sharp details created a smooth curve. There are two possibilities of this poor curve model; the $ANSA$ algorithm failed or the number of curve segments is not enough.

Fig. 6.53(a) shows iteration 6, in which the red points are the points from $\mathbf{d}_5$, the blue points are the 468 inliers points added to $\mathbf{d}_6$, the green points are the $121,058$ filtered outliers points, and there are 807 points eliminated from $\mathbf{d}_5$, resulting $5,635$ points in $\mathbf{d}_6$. The clusters division and injury curve model $\mathbf{P}_6$ are shown respectively in Figs. 6.53(b) and (c). There are just 3 not finished clusters in this iteration, showing that more clusters have achieved the stop criteria even though the number of $\mathbf{d}_6$ decreases. It was used the same number of cluster from the previous iteration showing that it is possible to create a good injury curve model with this number of clusters.

Fig. 6.54(a) shows iteration 7, in which the red points are the points from $\mathbf{d}_6$, the blue points are the 206 inliers points added to $\mathbf{d}_7$, the green points are the $66,482$ filtered outliers points, and there are 248 points eliminated from $\mathbf{d}_6$, resulting in $5,593$ points in $\mathbf{d}_7$. The clusters division and injury curve model $\mathbf{P}_7$ are shown respectively in Figs. 6.54(b) and (c). There are still 3 not finished clusters in this iteration. However, the purple cluster is now completed and the missing points are two region between 4 clusters.

Fig. 6.55(a) shows iteration 8, in which the red points are the points from $\mathbf{d}_7$, the blue points are the 224 inliers points added to $\mathbf{d}_8$, the green points are the $54,382$ filtered outliers points, and there are 261 points eliminated from $\mathbf{d}_7$, resulting $5,556$ points in $\mathbf{d}_8$. The clusters division and injury curve model $\mathbf{P}_8$ are shown respectively in Figs. 6.55(b) and (c). There are few modification from previous iteration, only some points in the middle of the yellow right cluster were added. There are still 2 not finished clusters, the yellow right cluster and the light blue left cluster.

Fig. 6.56(a) shows iteration 9, in which the red points are the points from $\mathbf{d}_8$, the blue points are the 47 inliers points added to $\mathbf{d}_9$, the green points are the $52,897$ filtered outliers points, and there are 165 points eliminated from $\mathbf{d}_8$, resulting $5,438$ points in $\mathbf{d}_9$. The clusters division and injury curve model $\mathbf{P}_9$ are shown respectively in Figs. 6.56(b) and (c). There are no visible modification between iteration 9 and iteration 8. However,

the number of unfinished clusters increases in iteration 10 as shown in Fig. 6.57, in which the cluster situated in the lower part of the injury is not finished anymore. Thus, it was necessary to increase the number of clusters from 10 clusters in iteration 9, to 12 clusters in iteration 10.

Fig. 6.58(a) shows iteration 10, in which the red points are the points from $\mathbf{d}_9$, the blue points are the 401 inliers points added to $\mathbf{d}_{10}$, the green points are the $36,330$ filtered outliers points, and there are 31 points eliminated from $\mathbf{d}_9$, resulting $5,808$ points in $\mathbf{d}_{10}$. The clusters division and injury curve model $\mathbf{P}_{10}$ are shown respectively in Figs. 6.58(b) and (c). The increase of the number of clusters helped to create a better injury curve model and less points are removed from the previous iteration. It is hard to evaluate whether the removed points are points from the injured zone or not. Thus, a bad curve model results in a high number of removal points; but a high number of removed points is not a indication of a bad curve model.

Iteration 10 have 2 unfinished clusters, and Fig. 6.59(a) shows iteration 11, in which the red points are the points from $\mathbf{d}_{10}$, the blue points are the 361 inliers points added to $\mathbf{d}_{11}$, the green points are the $40,308$ filtered outliers points, and there are 34 points eliminated from $\mathbf{d}_{10}$, resulting $6,135$ points in $\mathbf{d}_{11}$. The clusters division and injury curve model $\mathbf{P}_{11}$ are shown respectively in Figs. 6.59(b) and (c). There are few noticeable modifications from the previous iteration, only a few points in the middle of green and light blue clusters are added. There are still 2 unfinished clusters.

Fig. 6.60(a) shows iteration 12, in which the red points are the points from $\mathbf{d}_{11}$, the blue points are the 223 inliers points added to $\mathbf{d}_{12}$, the green points are the $45,562$ filtered outliers points, and there are 31 points eliminated from $\mathbf{d}_{11}$, resulting $6,327$ points in $\mathbf{d}_{12}$. The clusters division and injury curve model $\mathbf{P}_{12}$ are shown respectively in Figs. 6.60(b) and (c). Once again just a few points are visually added, there are a few points added between blue and burgundy clusters; and there are some points added in the side of green and blue clusters, but almost no points are added between it.

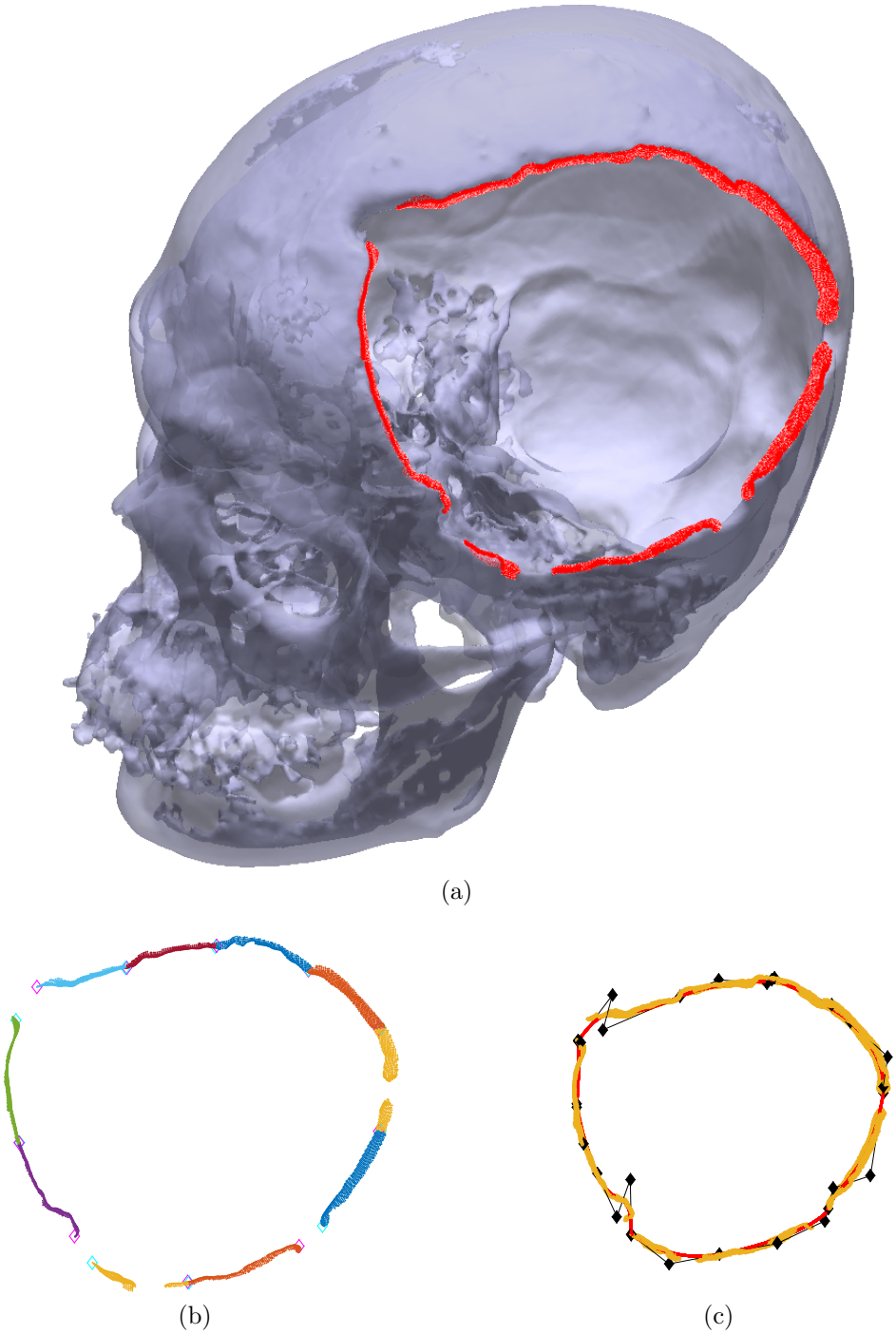
Fig. 6.61(a) shows iteration 13, in which the red points are the points from $\mathbf{d}_{12}$, the blue points are the 358 inliers points added to $\mathbf{d}_{13}$, the green points are the $16,321$ filtered outliers points, and there are 149 points eliminated from $\mathbf{d}_{12}$, resulting in $6,536$ points in $\mathbf{d}_{13}$. The clusters division and injury curve model $\mathbf{P}_{13}$ are shown respectively in Figs. 6.61(b) and (c). There are still 2 unfinished clusters, however it is possible to see that these two clusters are almost completed.

Fig. 6.62(a) shows iteration 14, in which the red points are the points from $\mathbf{d}_{13}$, the blue points are the 431 inliers points added to $\mathbf{d}_{13}$, the green points are the $27,651$ filtered outliers points, and there are 35 points eliminated from $\mathbf{d}_{13}$, resulting $6,932$ points in $\mathbf{d}_{14}$. The clusters division and injury curve model $\mathbf{P}_{14}$ are shown respectively in Figs. 6.62(b) and (c). The algorithm finished in this iteration and the final injured skull zone is shown by the red points in Fig. 6.63.

Table 4 – Forth example results. $N.Pts$ is the number of points of $\mathbf{d}_i$; $Rem.\mathbf{d}_{i-1}$ is the number of points removed from $\mathbf{d}_{i-1}$; *Inlier* and *Outlier* are respectively the number of points classified as inliers and outliers, in which the inliers are added to $\mathbf{d}_i$; $N.F.k$ is the number of cluster in which the stop criteria is not achieved; and $N.k$ is the number of clusters used to divide $\mathbf{d}_i$.

Iteration	N. Pts	Rem. $\mathbf{d}_{i-1}$	Inlier	Outlier	N.F. k	N. k
1	7,054	–	–	–	7	10
2	6,719	898	563	134,691	7	10
3	6,109	812	202	91,295	7	10
4	5,995	784	670	131,726	7	10
5	5,974	745	724	118,662	7	10
6	5,635	807	468	121,058	3	10
7	5,593	248	206	66,482	3	10
8	5,556	261	224	54,382	2	10
9	5,438	165	47	52,897	2	10
10	5,808	31	401	36,330	2	12
11	6,135	34	361	40,308	2	12
12	6,327	31	223	45,562	2	12
13	6,536	149	358	16,321	2	12
14	6,932	35	431	2,7651	0	12

The algorithm finished, however it is possible to see that there are some missing points in the left side of the injury. Fig.6.64 shows that there a hole in the input STL. Thus, the algorithm performed well in determining the injured zone, and the missing part is due to nonexistent points in the input data file. Table. 4 shows the results of the last example, with $N.Pts$ being the number of points of $\mathbf{d}_i$, $Rem.\mathbf{d}_{i-1}$ the number of points removed from the previous iteration, $N.F.k$ the number of clusters that the stop criteria is not achieved and $N.k$ the number of clusters.

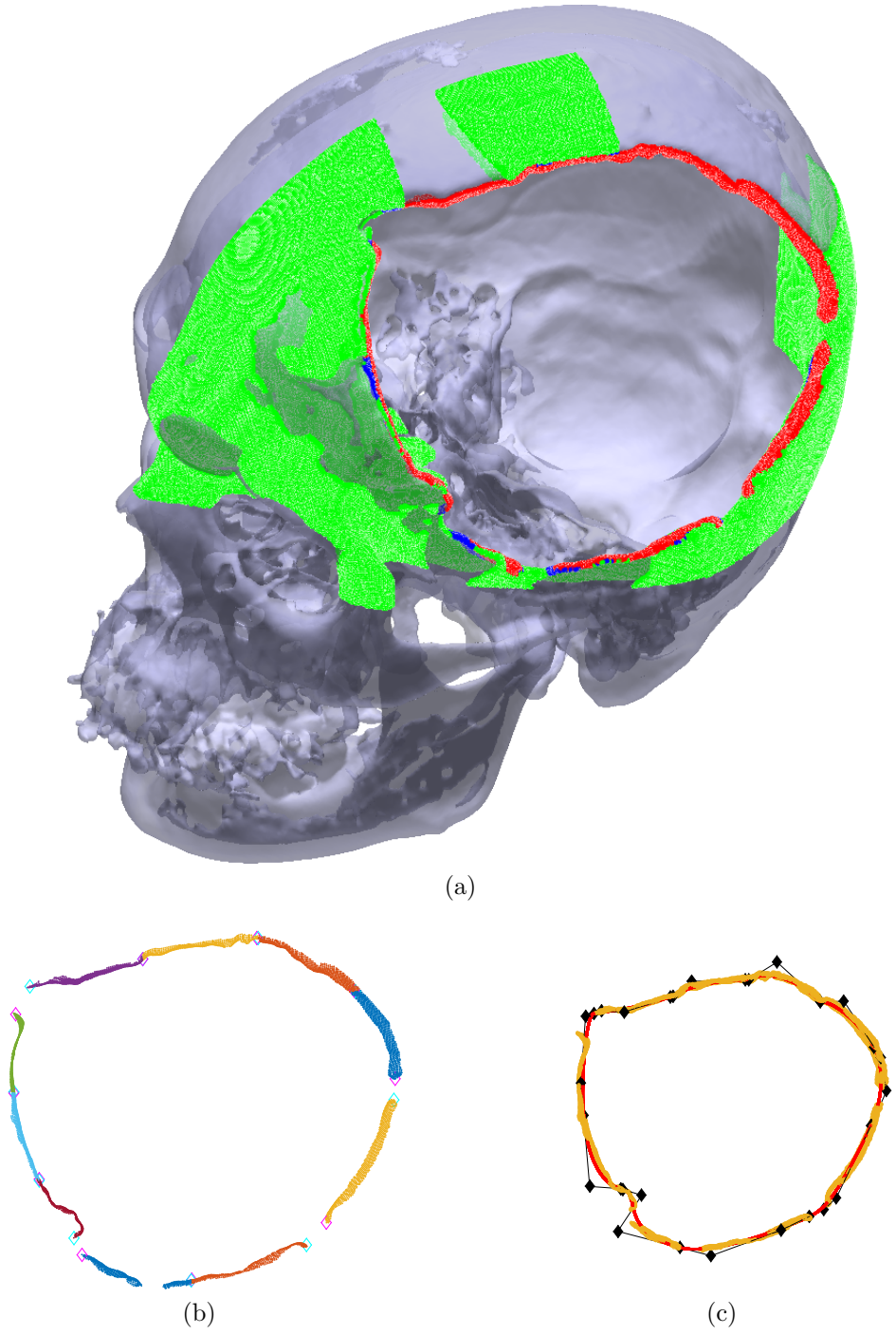


(a)

(b)

(c)

Figure 6.48 – Iteration 1. (a) $\mathbf{d}_1$ is represented with red points; (b) Each cluster of $\mathbf{d}_1$ is in different color; (c) Injured zone curve model $\mathbf{P}_1$ is the red curve, the control points are the black diamonds and $\mathbf{d}_1$ are the yellow points.

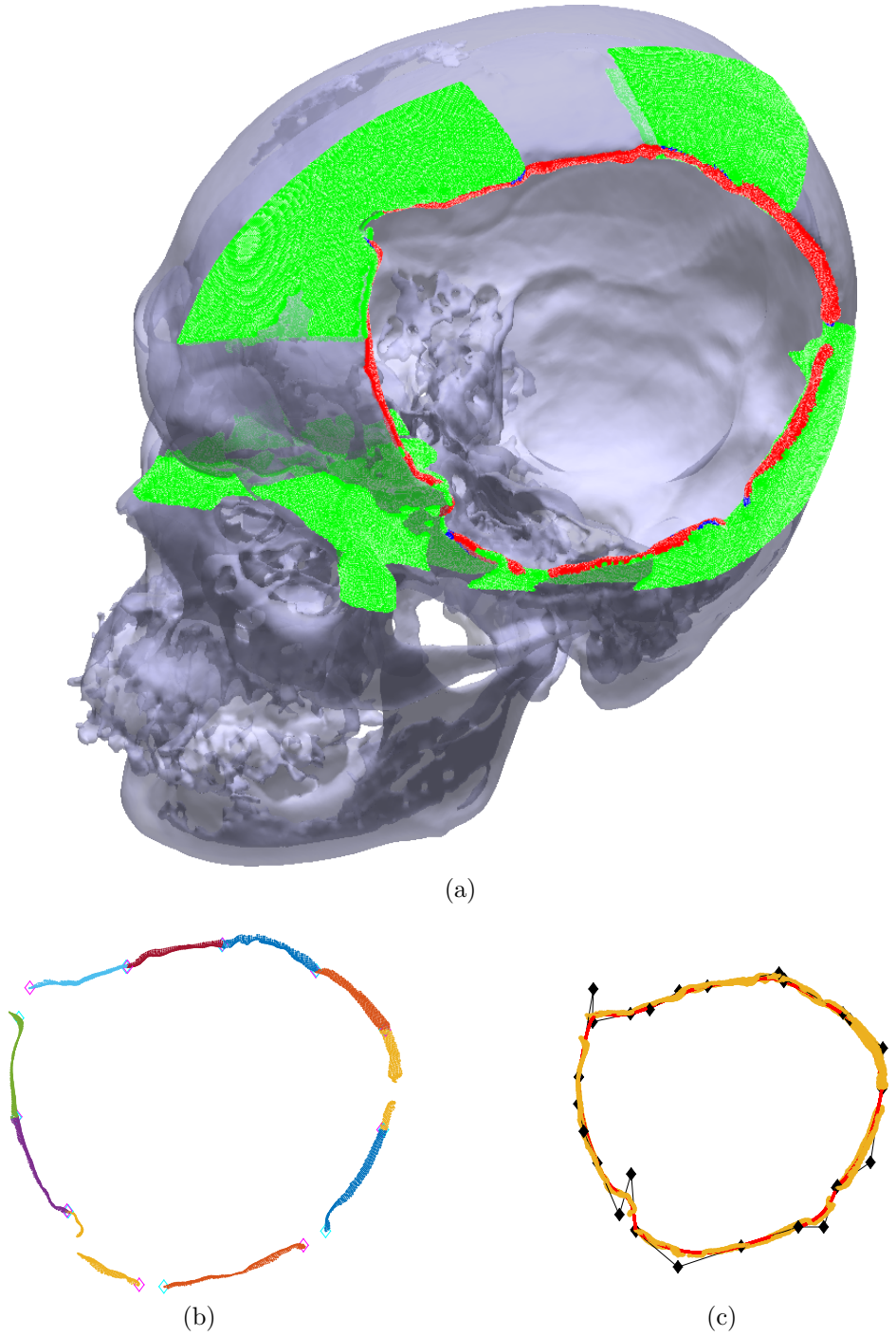


(a)

(b)

(c)

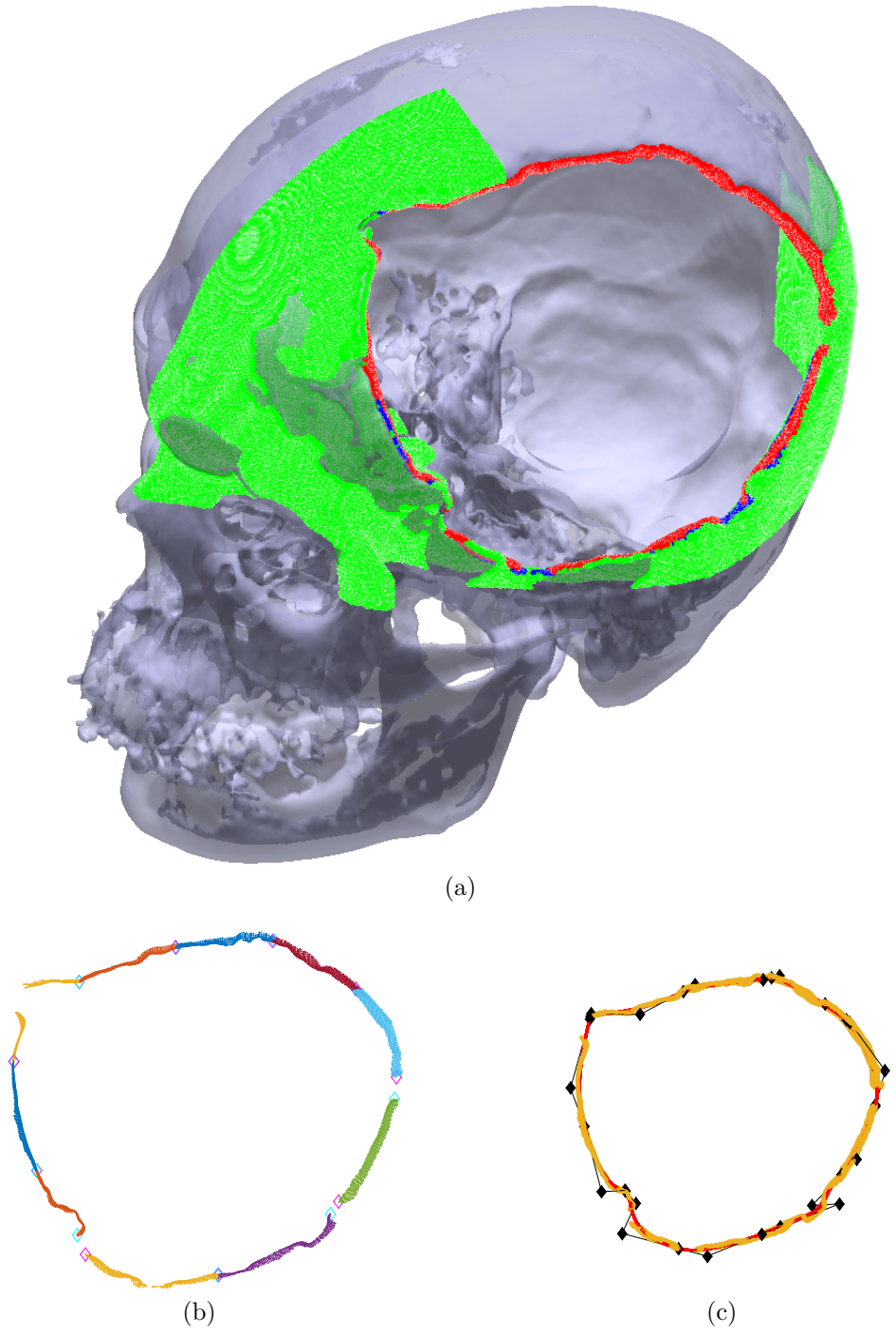
Figure 6.49 – Iteration 2. (a) $\mathbf{d}_1$ is represented with red points, the inliers added to $\mathbf{d}_2$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_2$ is in different color; (c) Injured zone curve model $\mathbf{P}_2$ is the red curve, the control points are the black diamonds and $\mathbf{d}_2$ are the yellow points.



(a)

(b) (c)

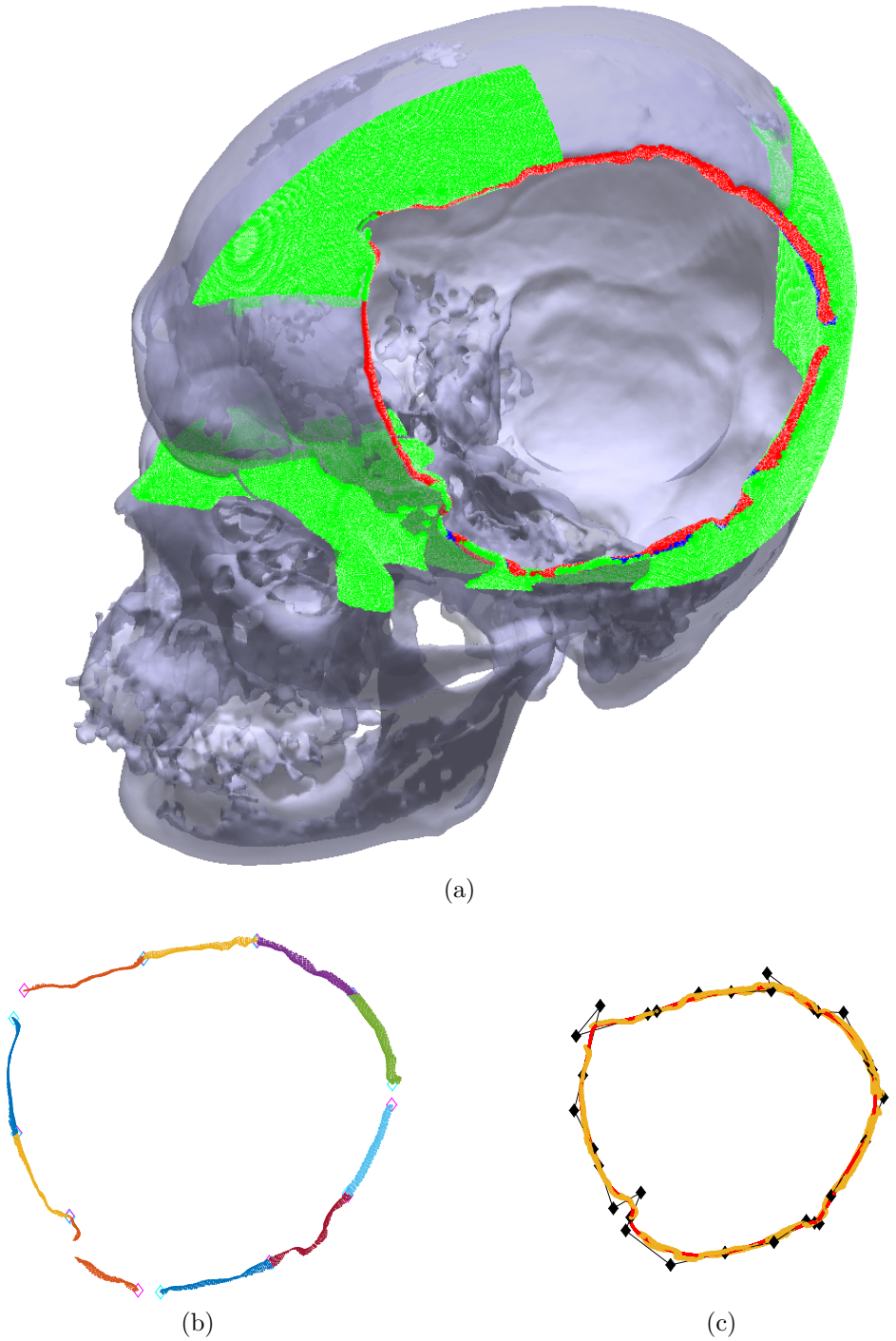
Figure 6.50 – Iteration 3. (a) $\mathbf{d}_2$ is represented with red points, the inliers added to $\mathbf{d}_3$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_3$ is in different color; (c) Injured zone curve model $\mathbf{P}_3$ is the red curve, the control points are the black diamonds and $\mathbf{d}_3$ are the yellow points.



(a)

(b) (c)

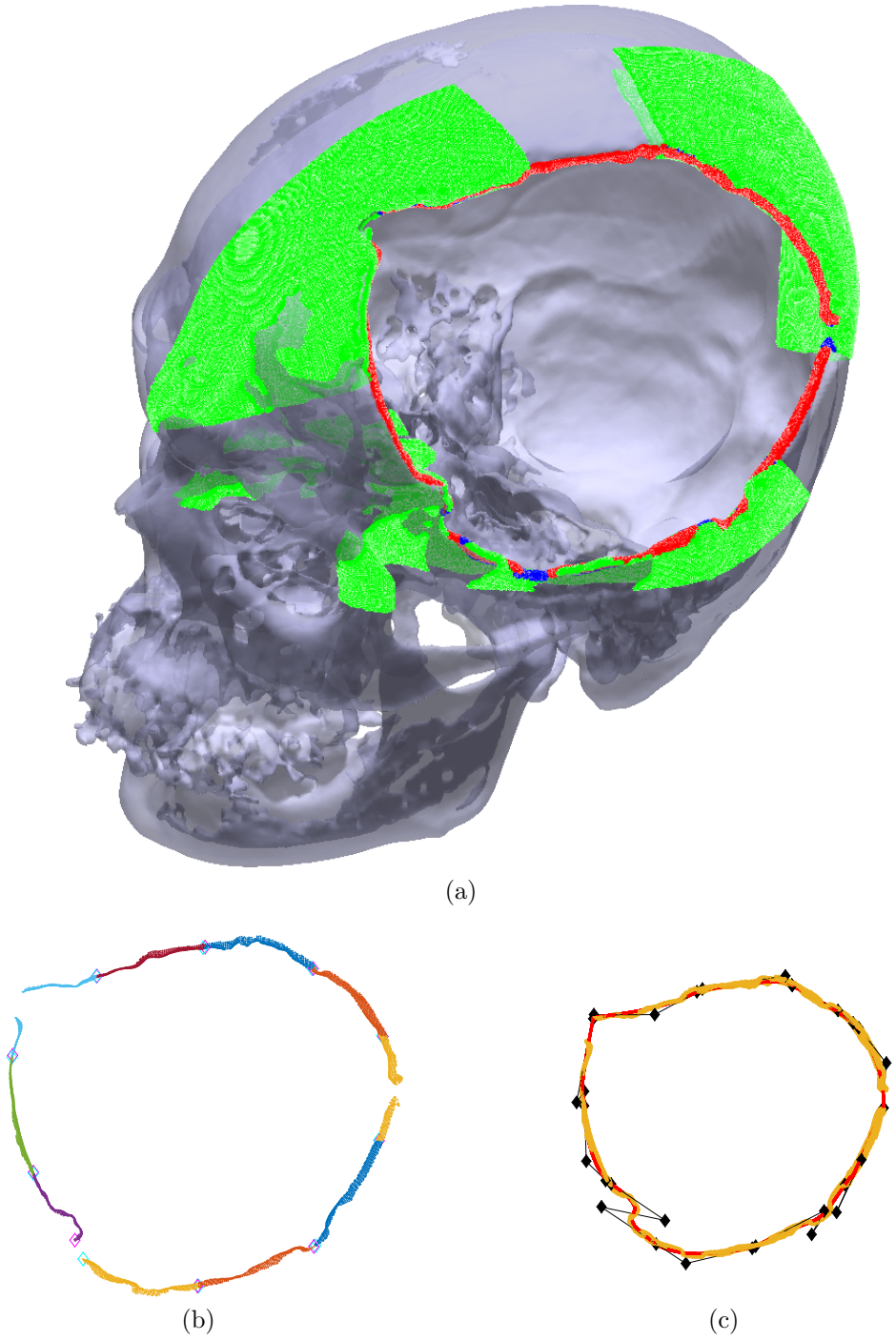
Figure 6.51 – Iteration 4. (a) $\mathbf{d}_3$ is represented with red points, the inliers added to $\mathbf{d}_4$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_4$ is in different color; (c) Injured zone curve model $\mathbf{P}_4$ is the red curve, the control points are the black diamonds and $\mathbf{d}_4$ are the yellow points.



(a)

(b) (c)

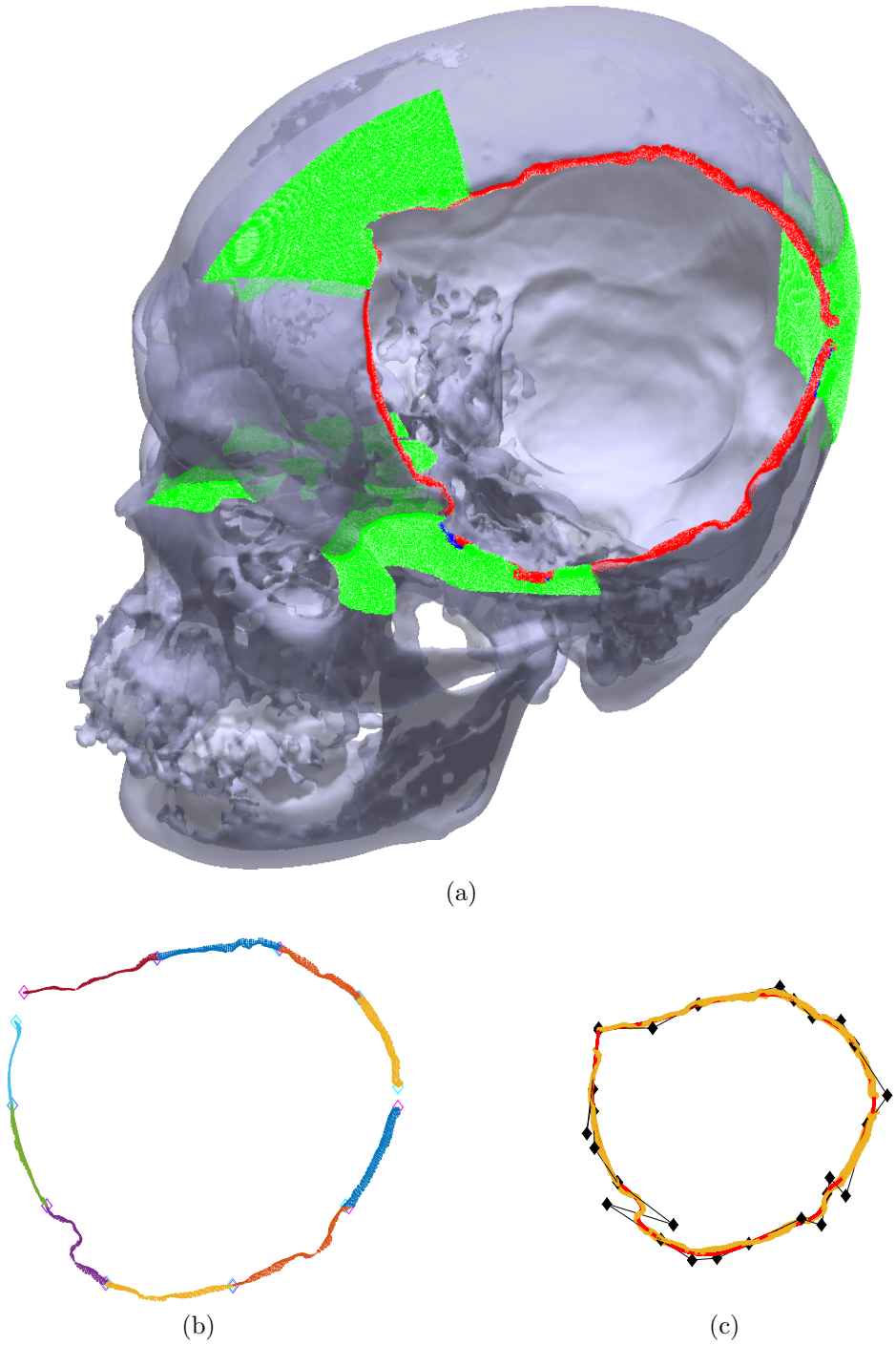
Figure 6.52 – Iteration 5. (a) $\mathbf{d}_4$ is represented with red points, the inliers added to $\mathbf{d}_5$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_5$ is in different color; (c) Injured zone curve model $\mathbf{P}_5$ is the red curve, the control points are the black diamonds and $\mathbf{d}_5$ are the yellow points.



(a)

(b) (c)

Figure 6.53 – Iteration 6. (a) $\mathbf{d}_5$ is represented with red points, the inliers added to $\mathbf{d}_6$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_6$ is in different color; (c) Injured zone curve model $\mathbf{P}_6$ is the red curve, the control points are the black diamonds and $\mathbf{d}_6$ are the yellow points.

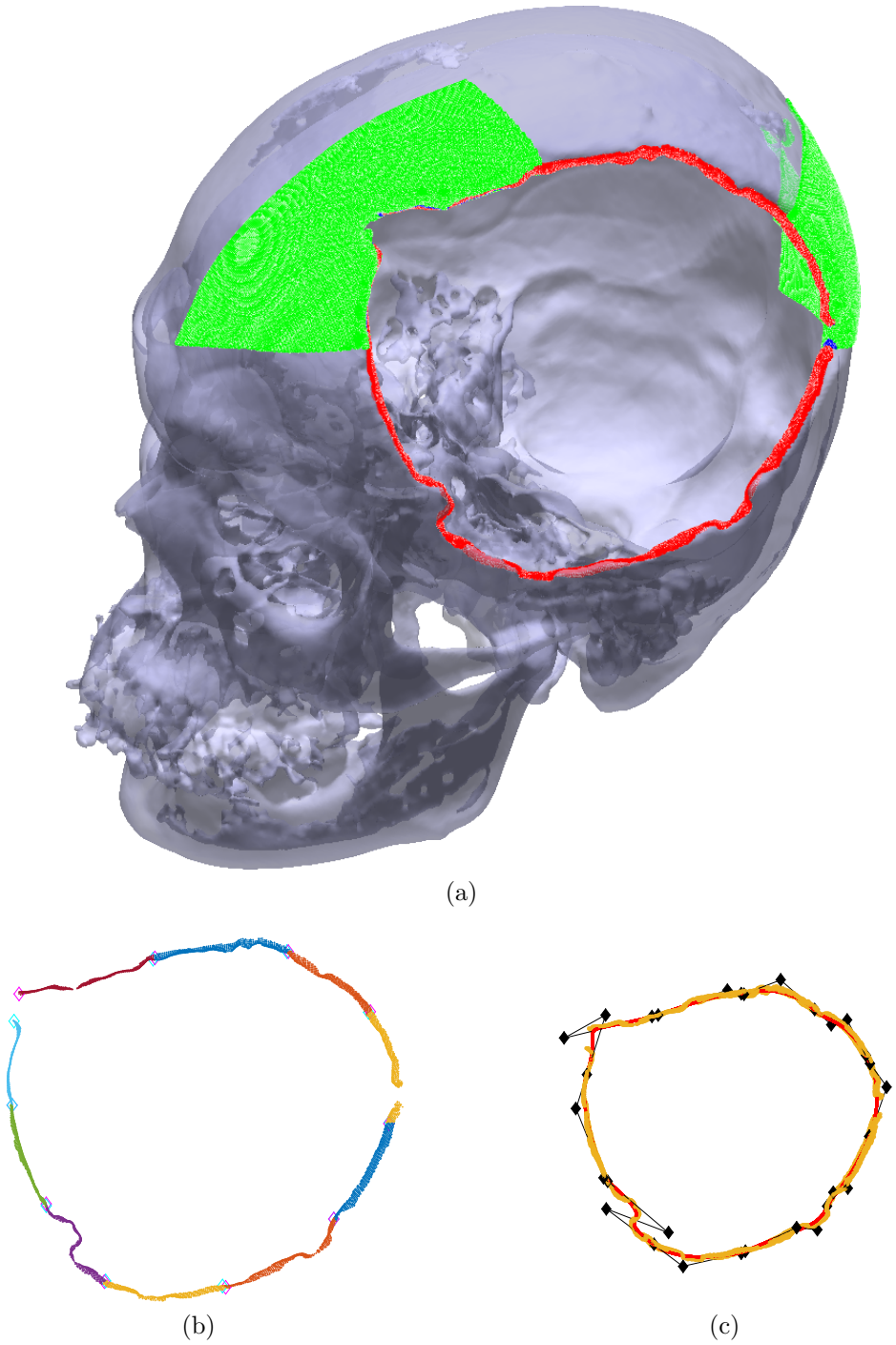


(a)

(b)

(c)

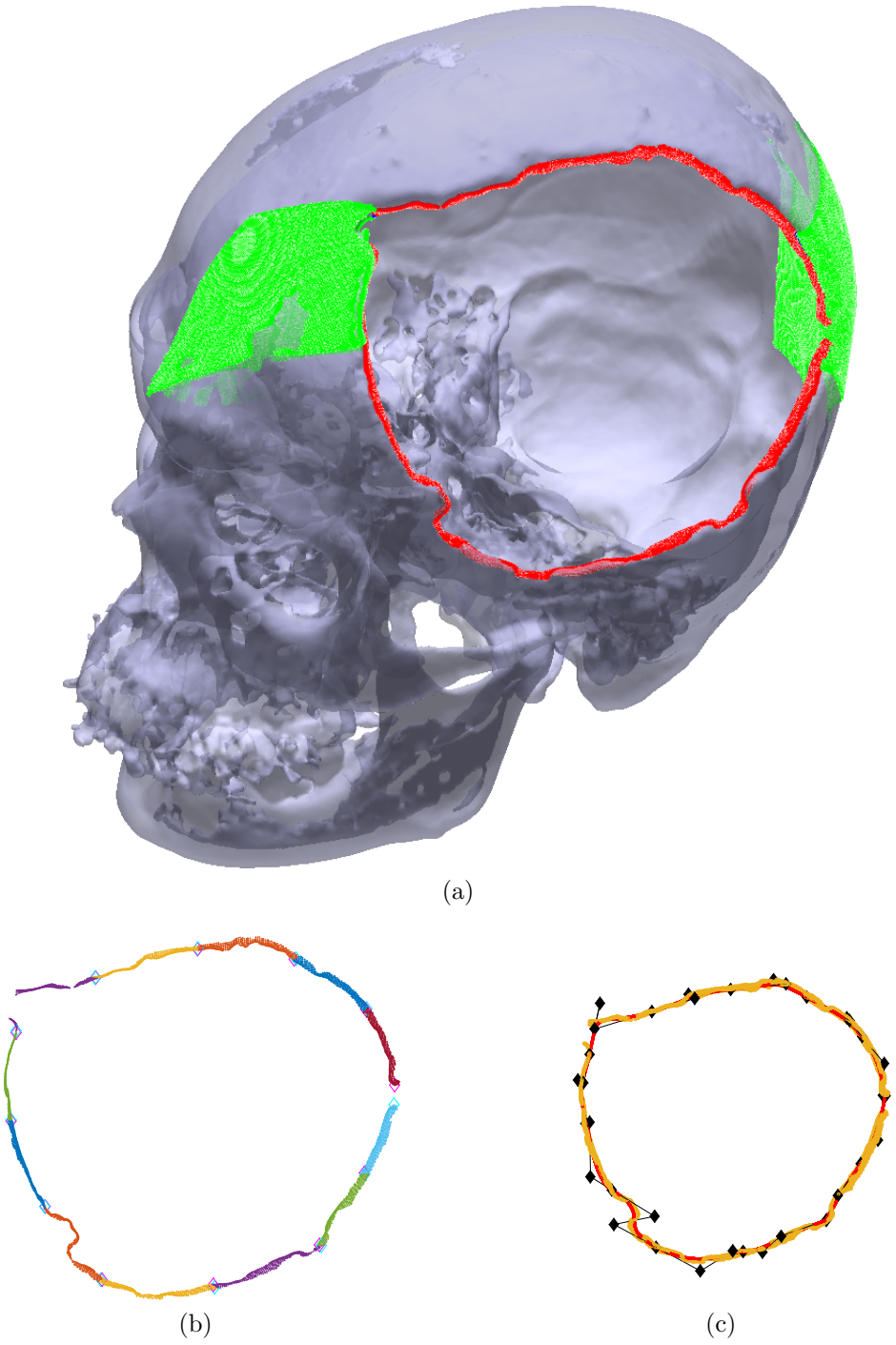
Figure 6.54 – Iteration 7. (a) $\mathbf{d}_6$ is represented with red points, the inliers added to $\mathbf{d}_7$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_7$ is in different color; (c) Injured zone curve model $\mathbf{P}_7$ is the red curve, the control points are the black diamonds and $\mathbf{d}_7$ are the yellow points.



(a)

(b) (c)

Figure 6.55 – Iteration 8. (a) $\mathbf{d}_7$ is represented with red points, the inliers added to $\mathbf{d}_8$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_8$ is in different color; (c) Injured zone curve model $\mathbf{P}_8$ is the red curve, the control points are the black diamonds and $\mathbf{d}_8$ are the yellow points.



(a)

(b) (c)

Figure 6.56 – Iteration 9. (a) $\mathbf{d}_8$ is represented with red points, the inliers added to $\mathbf{d}_9$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_9$ is in different color; (c) Injured zone curve model $\mathbf{P}_9$ is the red curve, the control points are the black diamonds and $\mathbf{d}_9$ are the yellow points.

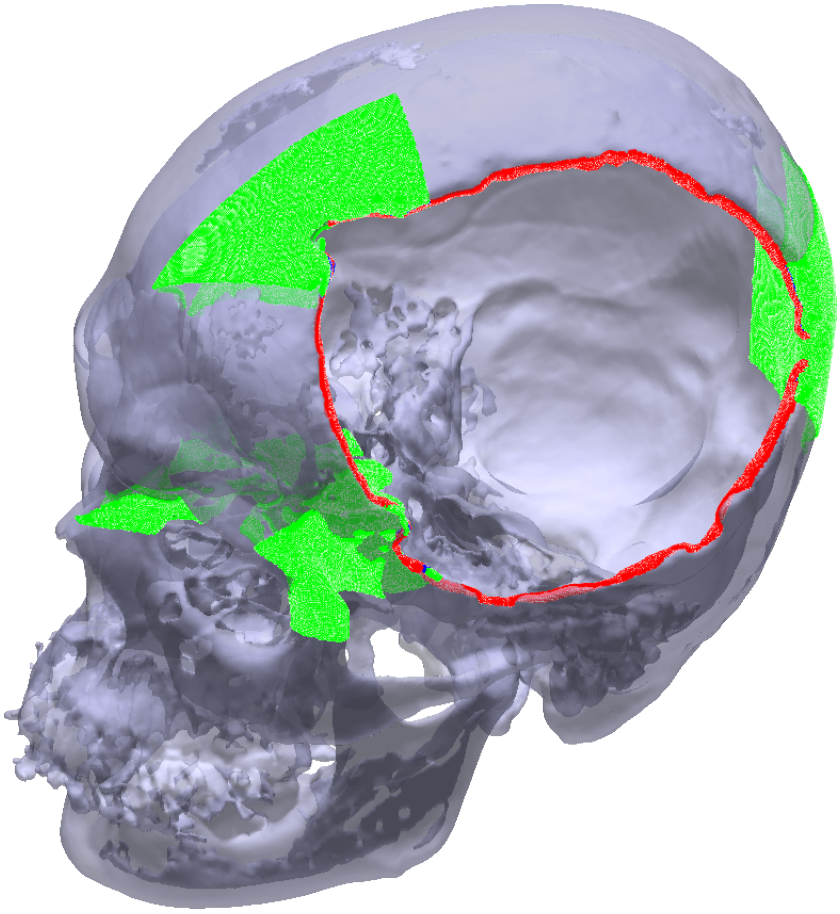
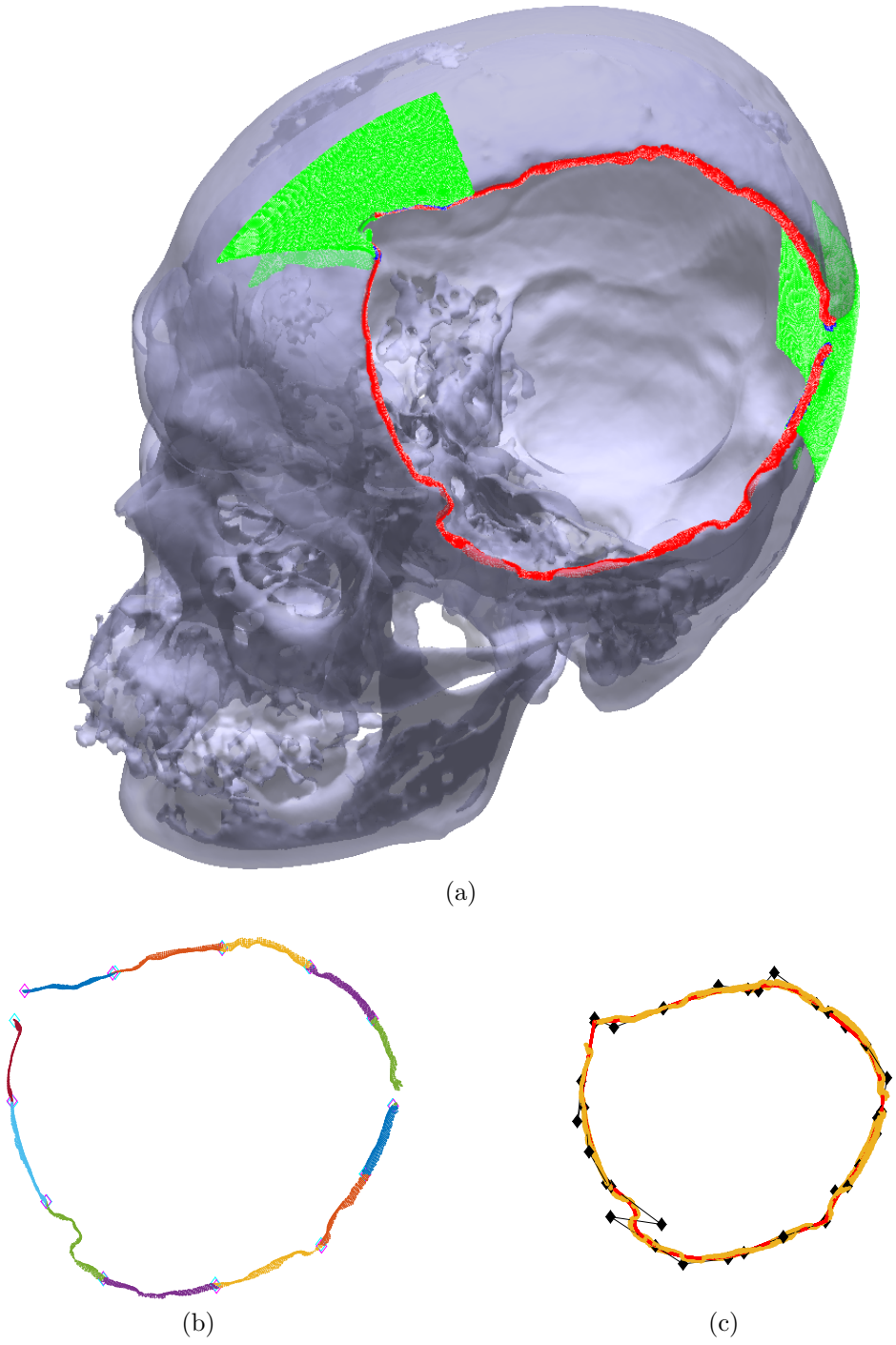


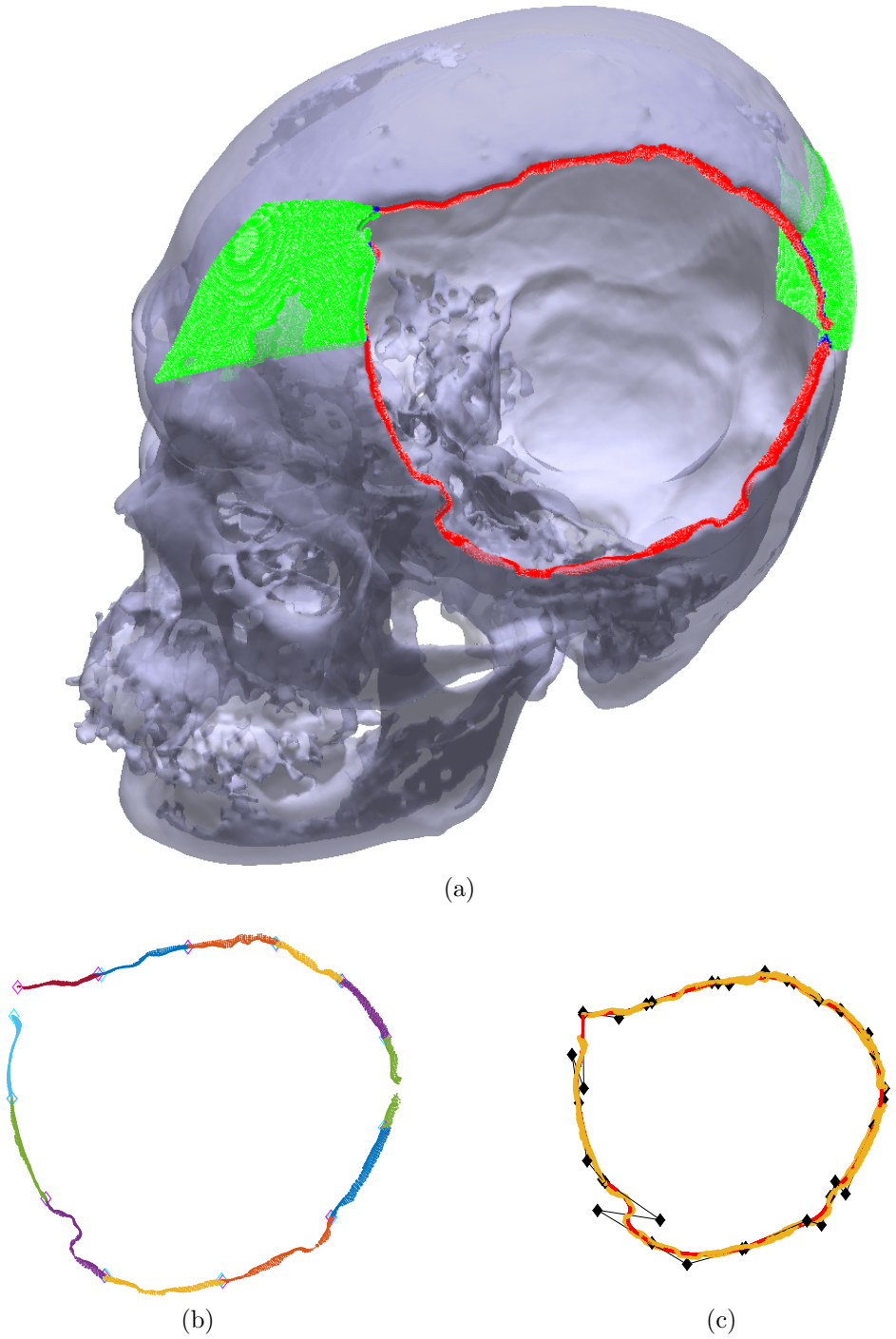
Figure 6.57 – Iteration 10 needed to be performed 3 times. The iteration was rejected twice due to the increase of unfinished clusters, specifically in the lower part of the injury.



(a)

(b) (c)

Figure 6.58 – Iteration 10. (a) $\mathbf{d}_9$ is represented with red points, the inliers added to $\mathbf{d}_{10}$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_{10}$ is in different color; (c) Injured zone curve model $\mathbf{P}_{10}$ is the red curve, the control points are the black diamonds and $\mathbf{d}_{10}$ are the yellow points.

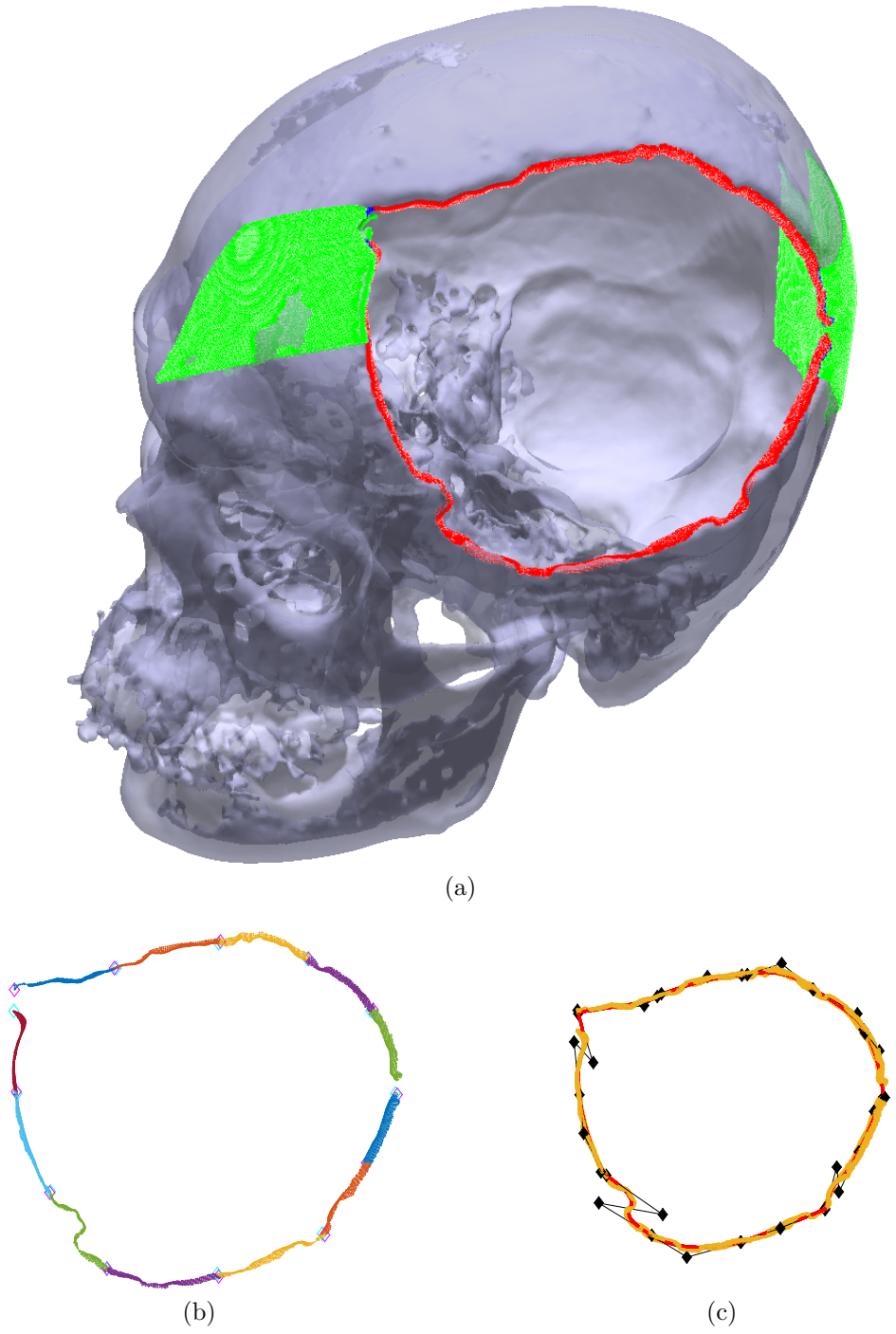


(a)

(b)

(c)

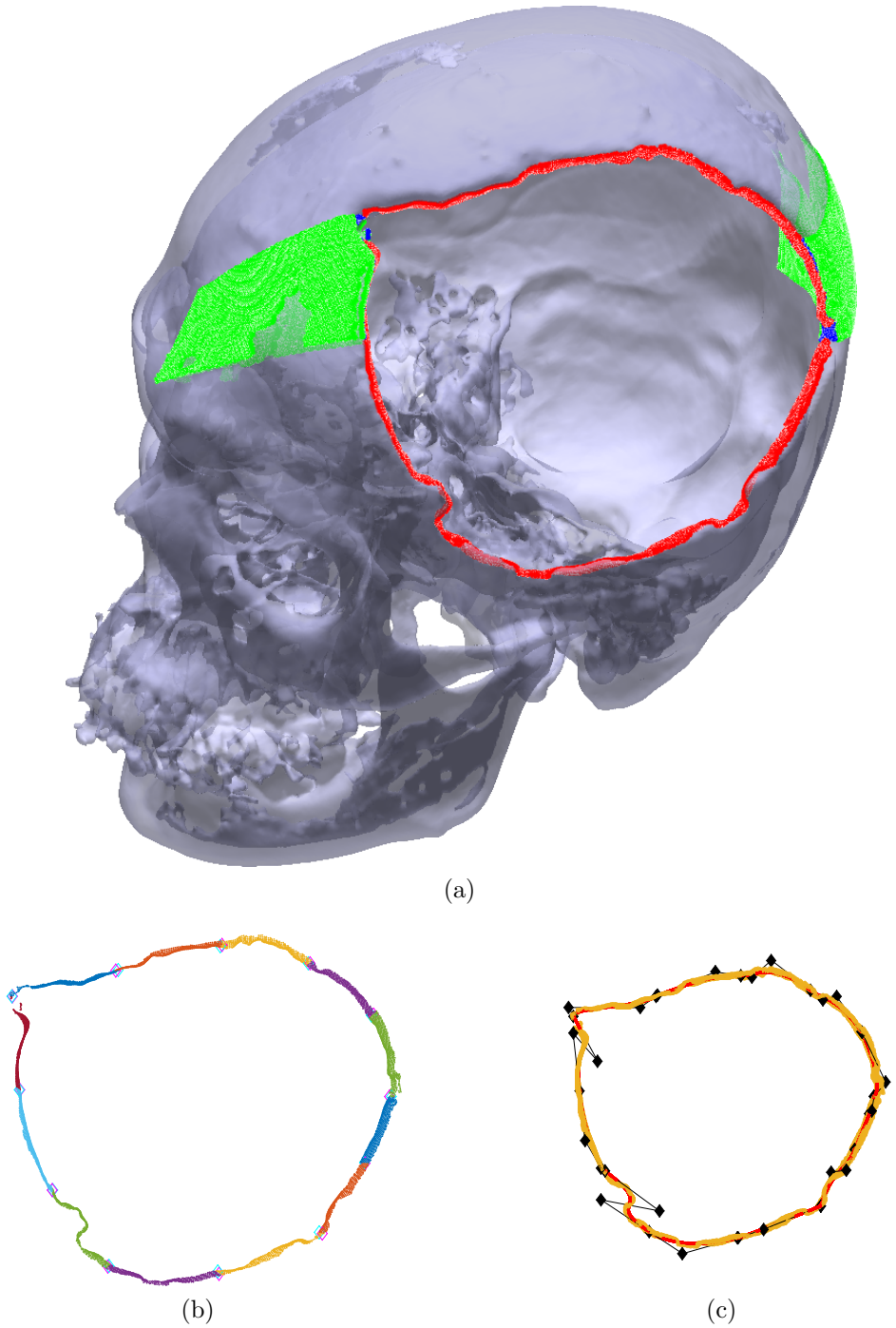
Figure 6.59 – Iteration 11. (a) $\mathbf{d}_{10}$ is represented with red points, the inliers added to $\mathbf{d}_{11}$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_{11}$ is in different color; (c) Injured zone curve model $\mathbf{P}_{11}$ is the red curve, the control points are the black diamonds and $\mathbf{d}_{11}$ are the yellow points.



(a)

(b) (c)

Figure 6.60 – Iteration 12. (a) $\mathbf{d}_{11}$ is represented with red points, the inliers added to $\mathbf{d}_12$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_{12}$ is in different color; (c) Injured zone curve model $\mathbf{P}_{12}$ is the red curve, the control points are the black diamonds and $\mathbf{d}_{12}$ are the yellow points.

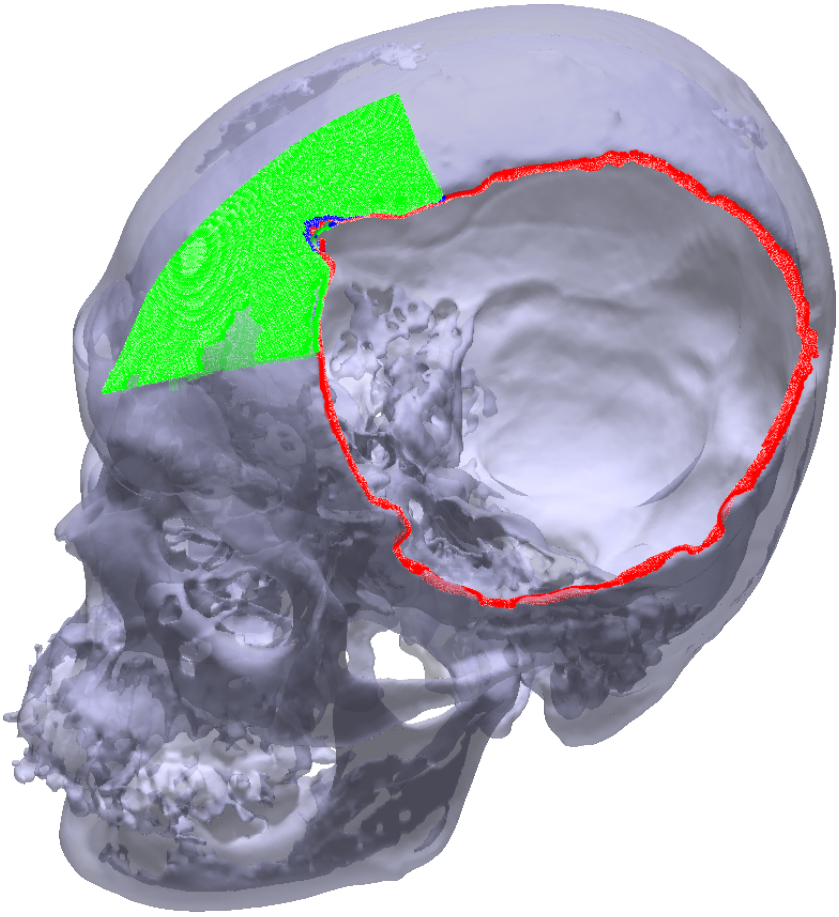


(a)

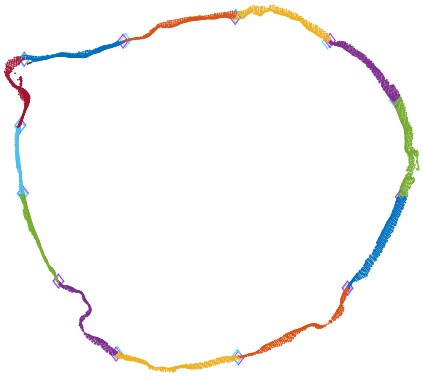
(b)

(c)

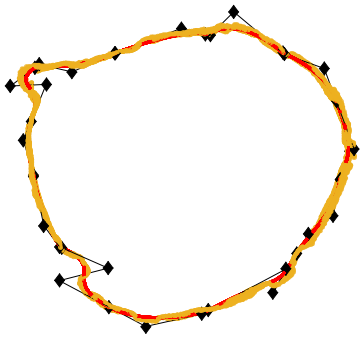
Figure 6.61 – Iteration 13. (a) $\mathbf{d}_{12}$ is represented with red points, the inliers added to $\mathbf{d}_{13}$ are the blue points and the filtered outliers are the green points; (b) Each cluster of $\mathbf{d}_{13}$ is in different color; (c) Injured zone curve model $\mathbf{P}_{13}$ is the red curve, the control points are the black diamonds and $\mathbf{d}_{13}$ are the yellow points.



(a)



(b)



(c)

Figure 6.62 – Iteration 14. (a) $\mathbf{d}_{13}$ is represented with red points, the inliers added to $\mathbf{d}_{14}$ are the blue points and the filtered outliers are the green points; (b) Each cluster of d_{14} is in different color; (c) Injured zone curve model $\mathbf{P}_{14}$ is the red curve, the control points are the black diamonds and $\mathbf{d}_{14}$ are the yellow points.

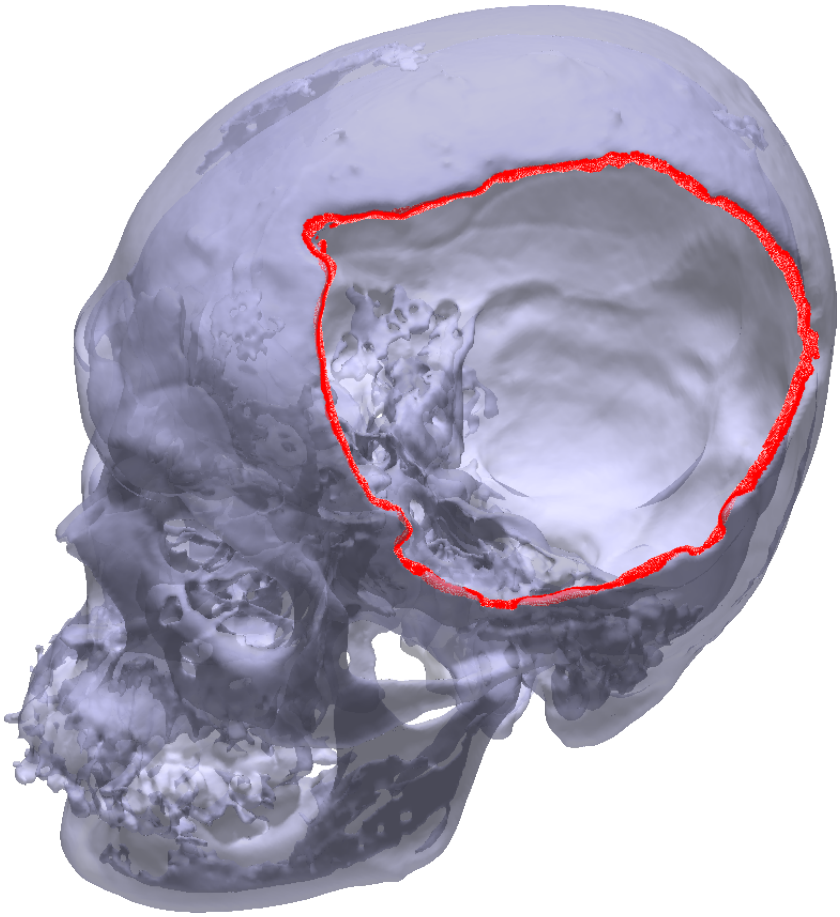


Figure 6.63 – Injured skull zone are the red points.

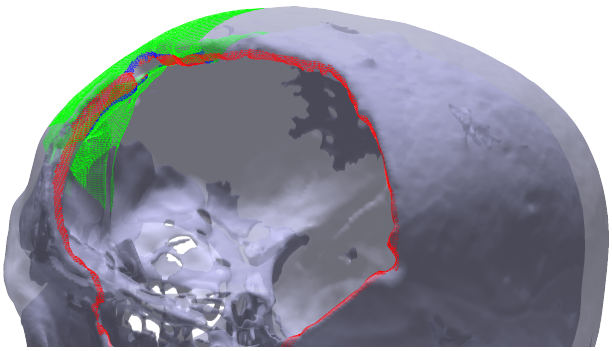


Figure 6.64 – There is a hole in the middle of the injured zone.

6.5 Parameters Sensibility

Throughout this thesis, several parameters were presented. The variation of these parameters changes the behavior and the convergence of the proposed method. In the curve fitting algorithm, there are two important parameters. The first parameter is the weight W that controls the regularization of the cost function. The higher the spread of the points the smaller the value for W, in the curves of Fig. 4.17 the value of W is higher than the one used in the four cases of the determination of the injured zone. The second parameter is the number of the points in which the curve is discretized. It was used 100 points for each curve segment. The increase of number of discretized points increases the accuracy of the determination the discrepancy between point and curve. However, this increase also increases the processing time.

In the automatic procedure, there are four important parameters. The first parameter is the value used in the range search of the distance map. It was used the minimum distance plus 10% of this value. The value was empirically determined. The increase of this value will make correspondence of points that will lead in a worst start for the injured zone, and the decrease of this value determines a smaller number of points in the initial injured zone.

The second parameter is the threshold t used to determine the initial injured zone. It is possible to use higher values for t, but this increase will lead a higher number of iteration to determine the injured zone. However, if the value for t is smaller, the initial injured zone will have point that are not from the injured, such as the region of the eyes or nose. The usage of t as the higher value of the histogram of distances is also very bad, once the initial injured zone will be just a few sparse points, and could determine a curve that will never be able to determine all the point of the injured zone.

The third parameter is n, the one used in the stop criteria as well as the classification of points in inliers and outliers. This parameter was also empirically determined. The determination of a good value for n is tricky. A higher number for n will classify further points of the curve as inliers. In simple cases, this increase is good because it is necessary less iterations to determine the injured zone. However, in more complex examples, this increase will lead in the classification of points as inliers that are not part of the injury. These points can be part of the inner or outer part of the skull.

The forth parameter is k, that defines the number of cluster and number of curve segments. The number of curve segments used in the curve fitting is the same problem to determine the number of control points necessary in the curve fitting, and this problem is still an open issue in the curve fitting problem. If the number for k is not enough the curve will never be close to the injured zone. On the other hand, if the number for k is too high, two problems can occur. The first one the overfitting problem, that is solved with the adopted cost function. The second problem is the increase of processing time, once there are more control points to be adjusted by the ANSA. Then, the adopted strategy

try to determine a good value for k by previous experience, and a method to increase this value if it is not suitable.

7 Conclusion

The proposed method consists in a semi automatic determination of the injured skull zone. The determination of this zone is important to design a skull prosthesis, considering the evaluated distances between the prosthesis and the skull. A semi-automatic iterative method determined the injured skull zone, in which points are added and removed to the injured zone with the aid of a injured zone curve model. In each iteration, the ANSA algorithm determined a piecewise Bézier curve and this curve is used as the model to add and remove points to and from the injured zone. The points of the current zone are divided in clusters and each cluster is approximated by only one curve segment. The usage of the a piecewise Bézier curve to determine the injured skull curve model makes the curve fitting algorithm more flexible and stable. The piecewise curve and the point cloud cluster division divided the problem in several curve fitting problems. Then, it is not necessary to order all the points of the point cloud to determine the approximating curve. it is necessary just to order the cluster and determine the start and end of the cluster. This is a huge advantage, once it is very difficult and costly to order a point cloud.

The procedure of adding and removing points are done until the stop criteria is achieved. This stop criteria is not global. Each cluster has its own stop criteria; and the proposed method will stop only if all the clusters achieve their own stop criteria. The proposed stop criteria is to verify whether every part of the curve model has a closest point from the injured skull zone within a certain distance limit. Once the determination of minimum number of required parameters to solve the curve fitting problem is still an open issue, there is also a method to increase the number of clusters in the proposed algorithm.

The proposed method was tested with 4 examples: 3 examples were artificially created and the last example is a natural injury. The algorithm was successful in all 4 the tests, the stop criteria works in all examples. In the first 3 artificially created examples, the injured zone is smoother, therefore less iteration was required; while in the last example, which is from a real injured skull, the injured skull zone has more details requiring more iterations. Thus, the harder the example, the more iterations it took. The mechanism to increase the number of clusters also was successful.This mechanism was not necessary in the first 3 examples due to their smooth geometries. However, in the last example, the increase of number of clusters was necessary to create a better model of the injured zone curve.

All the 4 examples have one feature in common, the injury always is in just one side of the skull. Thus, the proposed method has a limitation. If there are injuries in both sides of the skull or the injury is in the front or back of the skull, the proposed method will not work. This limitation is due to the way that the initial guess of the zone is determined, using the skull's symmetry. In this case, another way to produce the initial reference is

required to start the algorithm.

The proposed methodology was used to determine the injured skull zone that can be used in the cranioplastly to design better prosthesis. However, several parts of the proposed methodology can be used in another fields. It is possible to use the comparison of two point clouds to repair expansive parts such as a turbine blade as shown in Appendix A. The usage of the curve fitting in the reverse engineering field is known. The proposed curve fitting method could determine a curve of the injured skull. The shape of an injured skull is very complex, as can be seen in the fourth injury example, and the proposed curve fitting method could determine this approximating curve. Then, the developed curve fitting algorithm is more robust and can be used in several cases in the reverse engineering, in which its geometry is complex.

7.1 Future Works

The proposed method uses the symmetry to determine the initial reference for the injured zone. Thus, as future work it is necessary to create a method in which this symmetry is not used, enabling the proposed algorithm to be used in cases of frontal or back skull injury.

The usage of the injured zone in the design of a prosthesis also needs to be addressed in the future work to develop a closed surface which defines the prosthesis geometry with the three set of points (injured zone, inner surface and outer surface) and the injury boundary tangent vectors.

Bibliography

ADI, D.; SHAMSUDDIN, S.; HASHIM, S. Z. M. NURBS curve approximation using particle swarm optimization. *Computer Graphics, Imaging and Visualization (CGIV), 2010 Seventh International Conference on*, p. 73–79, 2010. Cited in page 34.

AKAIKE, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, v. 19, n. 6, p. 716–723, 1974. Cited in page 25.

AURENHAMMER, F. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 23, n. 3, p. 345–405, set. 1991. Cited in page 46.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *CACM*, ACM, New York, NY, USA, v. 18, p. 509–517, September 1975. Cited in page 44.

BERRY, C.; BARARI, A. Cyber-physical system utilizing work-piece memory in digital manufacturing. *IFAC-PapersOnLine*, v. 52, n. 10, p. 201–206, 2019. ISSN 2405-8963. 13th IFAC IMS. Cited in page 98.

BOHACHEVSKY, I. O.; JOHNSON, M. E.; STEIN, M. L. Generalized simulated annealing for function optimization. *Technometrics*, v. 28, n. 3, p. 209–217, 1986. Cited in page 27.

Carr, J. C.; Fright, W. R.; Beatson, R. K. Surface interpolation with radial basis functions for medical imaging. *IEEE Transactions on Medical Imaging*, v. 16, n. 1, p. 96–107, Feb 1997. ISSN 1558-254X. Cited in page 18.

CHEN, J.-J.; LIU, W.; LI, M.-Z.; WANG, C.-T. Digital manufacture of titanium prosthesis for cranioplasty. *The International Journal of Advanced Manufacturing Technology*, v. 27, n. 11, p. 1148–1152, Feb 2006. ISSN 1433-3015. Cited in page 18.

CHIYOKURA, H. *Solid Modeling with Designbase: Theory and Implementation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988. ISBN 0201192454. Cited in page 20.

CORANA, A.; MARCHESI, M.; MARTINI, C.; RIDELLA, S. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Transactions on Mathematical Software*, v. 13, p. 262–280, 1987. Cited 2 times in pages 27 and 28.

El Saddik, A. Digital twins: The convergence of multimedia technologies. *IEEE MultiMedia*, v. 25, p. 87–92, 2018. Cited in page 98.

EUFINGER, H.; WEHMÖLLER, M.; MACHTENS, E.; HEUSER, L.; HARDERS, A.; KRUSE, D. Reconstruction of craniofacial bone defects with individual alloplastic implants based on CAD/CAM-manipulated CT-data. *Journal of Cranio-Maxillofacial Surgery*, v. 23, n. 3, p. 175–181, jun 1995. Cited 2 times in pages 16 and 17.

GOFUKU, S.; TAMURA, S.; MAEKAWA, T. Point-tangent/point-normal B-spline curve interpolation by geometric algorithms. *Computer-Aided Design*, v. 41, p. 412–422, 2009. Cited in page 22.

GáLVEZ, A.; IGLESIAS, A. Efficient particle swarm optimization approach for data fitting with free knot b-splines. *Computer-Aided Design*, v. 43, n. 12, p. 1683 – 1692, 2011. ISSN 0010-4485. Cited in page 25.

HASEGAWA, A. Y.; ROSSO Jr., R. S. U.; TSUZUKI, M. S. G. Differential evolution optimization for Bezier curve fitting. *11th IFAC Workshop on Intelligent Manufacturing Systems*, p. 233–238, 2013. Cited 4 times in pages 10, 21, 22, and 34.

HIEU, L. C.; BOHEZ, E.; Vander Sloten, J.; ORIS, P.; PHIEN, H. N.; VATCHARAPORN, E.; BINH, P. H. Design and manufacturing of cranioplasty implants by 3-axis cnc milling. *Technology and health care : official journal of the European Society for Engineering and Medicine*, v. 10, n. 5, p. 413–23, 2002. Cited 3 times in pages 16, 17, and 18.

HU, S.-M.; WALLNER, J. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer-Aided Geometric Design*, v. 22, p. 251–260, 2005. Cited in page 22.

JAISINGHANI, S.; ADAMS, N. S.; MANN, R. J.; POLLEY, J. W.; GIROTTO, J. A. Virtual Surgical Planning in Orthognathic Surgery. *Eplasty*, Open Science Co., v. 17, p. ic1, 2017. Cited in page 17.

JARDINI, A. L.; LAROSA, M. A.; de Carvalho Zavaglia, C. A.; BERNARDES, L. F.; LAMBERT, C. S.; KHARMANDAYAN, P.; CALDERONI, D.; Maciel Filho, R. Customised titanium implant fabricated in additive manufacturing for craniomaxillofacial surgery: This paper discusses the design and fabrication of a metallic implant for the reconstruction of a large cranial defect. *Virtual and Physical Prototyping*, v. 9, n. 2, p. 115–125, apr 2014. Cited 3 times in pages 16, 17, and 18.

JOHNSON, N.; KOTZ, S.; BALAKRISHNAN, N. *Continuous univariate distributions*. Nova York: Wiley & Sons, 1995. (Wiley series in probability and mathematical statistics: Applied probability and statistics, v. 2). Cited in page 29.

KIM, B. J.; HONG, K. S.; PARK, K. J.; PARK, D. H.; CHUNG, Y. G.; KANG, S. H. Customized cranioplasty implants using three-dimensional printers and polymethyl-methacrylate casting. *Journal of Korean Neurosurgical Society*, v. 52, n. 6, p. 541–546, dec 2012. Cited 2 times in pages 16 and 18.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983. Cited in page 27.

LEE, M. Y.; CHANG, C. C.; LIN, C. C.; LO, L. J.; CHEN, Y. R. Custom implant design for patients with cranial defects. *IEEE Engineering in Medicine and Biology Magazine*, v. 21, n. 2, p. 38–44, mar 2002. Cited in page 16.

LIN, Y.-C.; CHENG, C.-Y.; CHENG, Y.-W.; SHIH, C.-T. Skull repair using active contour models. *Procedia Manufacturing*, v. 11, p. 2164 – 2169, 2017. ISSN 2351-9789. Cited in page 18.

LLOYD, S. P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, v. 28, p. 129–137, 1982. Cited in page 46.

MAEKAWA, T.; MATSUMOTO, Y.; NAMIK, K. Interpolation by geometric algorithm. *Computer-Aided Design*, v. 39, p. 313–323, 2007. Cited in page 22.

MARTINS, T.; SATO, A.; TSUZUKI, M. Adaptive neighborhood heuristics for simulated annealing over continuous variables. *Simulated Annealing - Advances, Applications and Hybridizations*, InTech, 2012. Cited in page 28.

MARTINS, T. C.; CAMARGO, E. D. L. B.; LIMA, R. G.; AMATO, M. B. P.; TSUZUKI, M. S. G. Image reconstruction using interval simulated annealing in electrical impedance tomography. *IEEE Transactions on Bio-medical Engineering*, v. 59, n. 7, p. 1861–1870, 2012. Cited in page 31.

MARTINS, T. C.; TSUZUKI, M. S. G. Placement over containers with fixed dimensions solved with adaptive neighborhood simulated annealing. *Bulletin of the Polish Academy of Sciences: Technical Sciences.*, v. 57, n. 3, p. 273—280, dez. 2010. Cited in page 31.

MARTINS, T. C.; TSUZUKI, M. S. G. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. *Expert Syst. Appl.*, Pergamon Press, Inc., v. 37, n. 3, p. 1955–1972, mar. 2010. Cited in page 31.

METROPOLIS, N.; ROSENBLUTH, A.; ROSENBLUTH, M.; TELLER, A.; TELLER, E. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, v. 21, p. 1087, 1953. Cited in page 27.

PANDUNATA, P.; SHAMSUDDIN, S. Differential evolution optimization for Bezier curve fitting. *Computer Graphics, Imaging and Visualization (CGIV), 2010 Seventh International Conference on*, p. 68–72, 2010. Cited 2 times in pages 21 and 34.

PARK, E. K.; LIM, J. Y.; YUN, I. S.; KIM, J. S.; WOO, S. H.; KIM, D. S.; SHIM, K. W. Cranioplasty enhanced by three-dimensional printing: Custom-made three-dimensional-printed titanium implants for skull defects. *Journal of Craniofacial Surgery*, v. 27, n. 4, p. 943–949, jun 2016. Cited 2 times in pages 16 and 18.

PARTHASARATHY, J. 3D modeling, custom implants and its future perspectives in craniofacial surgery. *Annals of Maxillofacial Surgery*, v. 4, n. 1, p. 9, jan 2014. Cited in page 16.

RAVARI, A. N.; TAGHIRAD, H. D. Reconstruction of b-spline curves and surfaces by adaptive group testing. *Computer-Aided Design*, v. 74, p. 32 – 44, 2016. Cited in page 25.

ROTARU, H.; STAN, H.; FLORIAN, I. S.; SCHUMACHER, R.; PARK, Y. T.; KIM, S. G.; CHEZAN, H.; BALC, N.; BACIUT, M. Cranioplasty with custom-made implants: Analyzing the cases of 10 patients. *Journal of Oral and Maxillofacial Surgery*, v. 70, n. 2, p. 169–176, feb 2012. Cited 2 times in pages 16 and 18.

SCHLEICH, B.; ANWER, N.; MATHIEU, L.; WARTZACK, S. Skin model shapes: A new paradigm shift for geometric variations modelling in mechanical engineering. *CAD*, v. 50, p. 1–15, 2014. Cited in page 98.

SING, C. C.; PUEH, L. H.; KUMAR, A. S. Automatic hole repairing for cranioplasty using Bézier surface approximation. *Journal of Craniofacial Surgery*, v. 16, n. 6, p. 1076–1084, nov 2005. ISSN 10492275. Cited in page 18.

TAKIMOTO, R. Y.; TSUZUKI, M. de S. G.; VOGELAAR, R.; MARTINS, T. de C.; SATO, A. K.; IWAO, Y.; GOTOH, T.; KAGEI, S. 3D reconstruction and multiple point cloud registration using a low precision RGB-D sensor. *Mechatronics*, v. 35, p. 11–22, 2016. Cited in page 45.

TAVARES, R.; MARTINS, T.; TSUZUKI, M. Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning. *Expert Systems with Applications*, v. 38, n. 4, p. 2951 – 2965, 2011. Cited in page 23.

TRAINOR, P. A.; RICHTSMEIER, J. T. Facing up to the challenges of advancing Craniofacial Research. *American Journal of Medical Genetics, Part A*, v. 167, n. 7, p. 1451–1454, jul 2015. Cited in page 16.

UEDA, E. K.; TSUZUKI, M. de S. G.; SATO, R. Y. T. anf A. K.; MARTINS, T. de C.; ROSSO Jr., R. S. U. Determinação da curva aproximadora utilizando trechos de curvas de Bézier e o recozimento simulado. *Anais do XXI congresso brasileiro de automática*, Vitória, Brazil, 2016. Cited 3 times in pages 23, 34, and 35.

VOLPE, Y.; FURFERI, R.; GOVERNI, L.; UCCHEDDU, F.; CARFAGNI, M.; MUSSA, F.; SCAGNET, M.; GENITORI, L. Surgery of complex craniofacial defects: A single-step AM-based methodology. *Computer Methods and Programs in Biomedicine*, v. 165, p. 225–233, oct 2018. Cited 2 times in pages 16 and 18.

VORONOI, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. premier mémoire. sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik*, v. 133, p. 97–178, 1908. Cited in page 46.

WEHMÖLLER, M.; EUFINGER, H.; KRUSE, D.; MASSBERG, W. CAD by processing of computed tomography data and CAM of individually designed prostheses. *International Journal of Oral and Maxillofacial Surgery*, v. 24, n. 1 PART 2, p. 90–97, feb 1995. Cited 2 times in pages 16 and 17.

YAMAGUCHI, F. *Curves and surfaces in computer aided geometric design*, Springer-Verlag, 1988. Cited in page 34.

ZIELINSKI, E.; JACOBS, R. J.; BARKER, E.; RODBY, K.; ANTONY, A. K. Virtual Surgical Planning in Craniomaxillofacial Reconstruction. In: *A Textbook of Advanced Oral and Maxillofacial Surgery Volume 2*. [S.l.]: InTech, 2015. p. Ch. 29. Cited in page 17.

Appendix

APPENDIX A – Deviation Zone Estimation

The Digital Twin is a replication of the living and the non living physical entities in the virtual world [El Saddik, 2018]. These replication enable the communication between the physical and digital world, in which physical and digital entities could exchange data, that can be widely used to optimize the manufacturing process by the use of artificial intelligence algorithms. The Deviation Zone Estimation (DZE) is an application of the Digital Twin, in which it is determined the difference of a measured object and the designed object. The DZE can be used to make a quality control of manufacturing task [SCHLEICH et al., 2014], a real-time quality control inspection [BERRY; BARARI, 2019] or preventing maintenance.

Part of the proposed method could be used to determine the DZE. The initial part of the algorithm in which determined the initial zone ($\mathbf{d}_1$) is the same of determining the DZE, as can be seen in Fig. A.65.

Thus, as an example it is possible to find the DZE of a worn turbine and monitor whether or not a maintenance is necessary. A turbine has a CAD model which is used as reference point cloud (Pc'). Therefore, if the only source of difference between the reference model and the measured point cloud (Pc) is just the worn, the DZE is easily determined by just subtracting both point clouds. However, there are at least two sources of deviation error. The first source is the sensor measurement error and the second one is manufacturing deviation. The worn region, the sensor error and the manufacturing deviation are all detectable if the same simple subtraction is done, and it is impossible to determine which one is which one.

The usage of a threshold t eliminates the sensor error and manufacturing deviations.

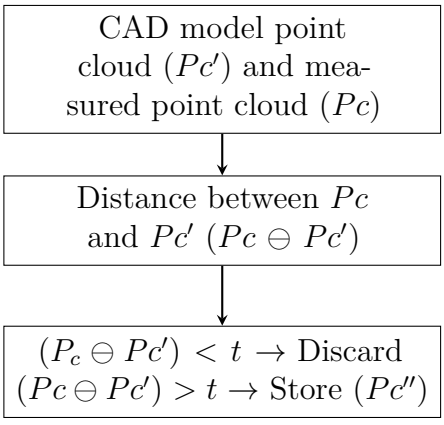


Figure A.65 – Part of the proposed method that can be used to DZE.

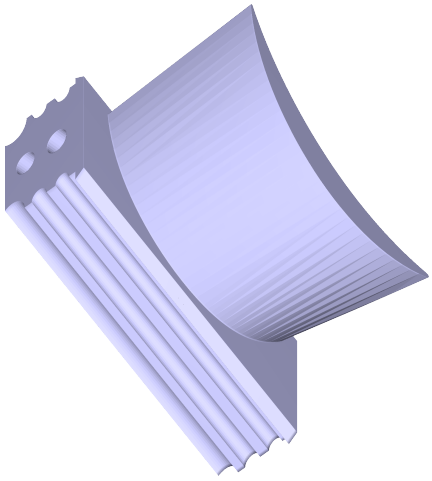
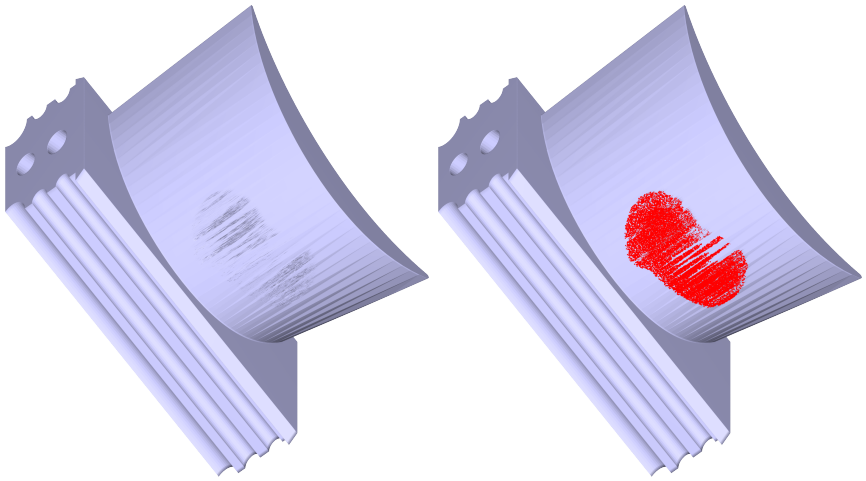


Figure A.66 – Turbine blade CAD model.



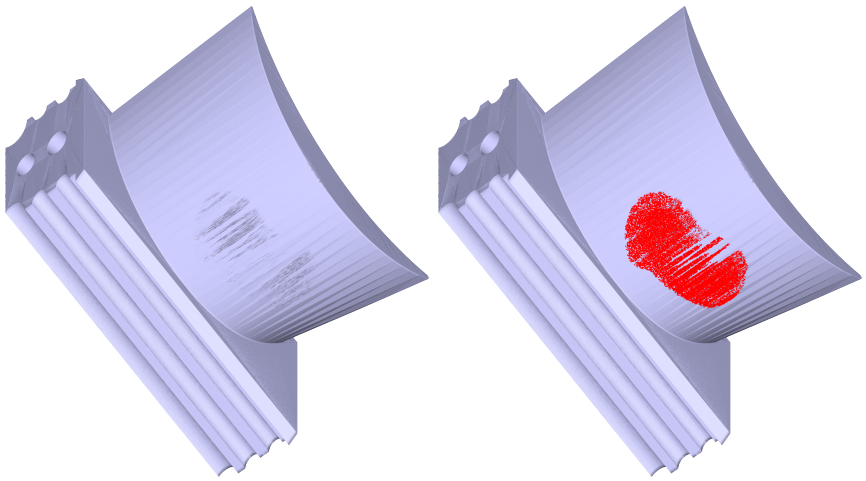
(a) (b)

Figure A.67 – Turbine blade created worn region. (a) The wear are in the middle of the blade with a max of 1 random dislocation in the normal direction; (b) worn region points are marked with red points.

This time the value of t is known, it is the summation of the manufacturing deviation and the sensor maximum error. Thus, let Fig. A.66 be the turbine blade CAD model. The turbine blade worn is created by adding a random noise at the normal direction with intensity between $[-1, 0]$. Fig. A.67 shows the created worn region, in which the noise is added to $16,193$ red points.

The errors and noisy simulation is deformed with the same procedure of the worn region creation. It is added a random noise to the points at the normal direction, however the intensity is between $[-0.01, 0.01]$ and it is applied to all the points excluding the red points. The resulting blade is shown in Fig. A.68.

If the DZE detection is done by a naive implementation by just determining all the points in which the distance is different than 0 ($t = 0$), the resulting point cloud will look like to the blade shown in Fig. A.68(a). Then, if it is used a threshold $t = 0.01$ to filter



(a) (b)

Figure A.68 – Turbine blade with noise, all the points excluding the worn region are added with a max of 0.01 random noise. (a) The worn region are the same of the previous image, however it is possible to see that there are some noise in the top and base of the blade; (b) worn region points are marked with red points.

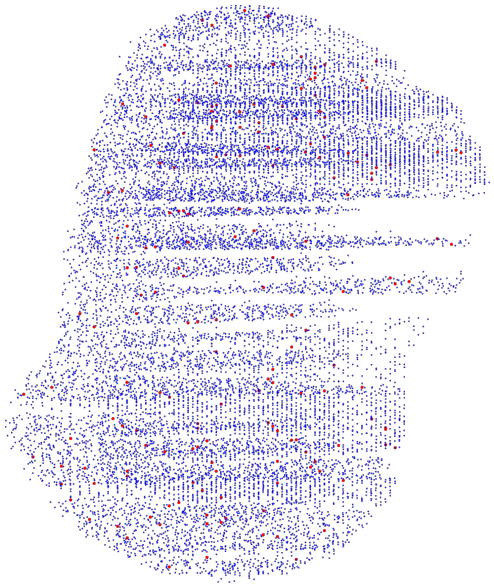


Figure A.69 – The blue points are the detected worn region points, and the red points are the not detected worn region points.

the errors, the resulting DZE is the one shown in Fig. A.69, in which the blue points are the detected worn region points, and the red points are worn region points which were not detected. It is possible to detect $16,015$ points corresponding 98.9% of the created worn region points.