Approach for CAD model Reconstruction from a deformed mesh

Aicha Ben Makhlouf[a], Borhen Louhichi[b], Mohamed Ali Mahjoub[a], Gérard Subsol[c]
[a]*LATIS laboratory of Advanced Technology and Intelligent Systems, University of sousse, Sousse 4023, Tunisia*
[b]*LMS, ENISo, University of Sousse, Sousse 4023, Tunisia*
[c]*Laboratory of Informatics, Robotics and Microelectronics of Montpellier CNRS/University of Montpellier, Montpellier, France*
Aychaa.benmakhlouf@gmail.com; borhen.louhichi@etsmtl.ca; mohamedali.mahjoub@eniso.rnu.tn; Gerard.subsol@lirmm.fr

Abstract

Geometric model reconstruction from a set of points is a difficult problem, which has been tackled with many different approaches. The reconstruction of the mechanical part is a necessity to visualize parts, simulate, assembly and detect interferences... The reconstruction of geometric entities (curves, edges, surfaces, faces) of these parts introduces particular difficulties. The most difficult problem to obtain the 3D geometric model from a cloud of points is the reconstruction of the faces. Many methods have been proposed to simplify the reconstruction of surfaces. There are two types of surface reconstruction: one of primitive shapes and the other of complex shapes like deformed mechanical parts and objects containing complex surfaces. In this paper, we present an algorithm to reconstruct the computer-aided design model from a deformed mesh. Then, we address a solution to reconstruct a 3D surface from a cloud of points extracted from a deformed mesh.

1. Introduction

The appearance of 3D scanning techniques such as the use of scanners in various fields is increasingly multiplied. These scanners are becoming a standard source for input data in many application areas, providing millions of 3D points. Consequently, the problem of reconstruction a CAD model from this point cloud is receiving more and more attention. This model is composed of many surfaces. So, to rebuild a CAD model, it is necessary to reconstruct its different surfaces. The CAD model is one of the most important models used throughout the product's entire life cycle. It represents the geometric support used in many other activities (analysis, manufacturing, assembly, etc.). This model is used to visualize 3D objects that were scanned and approximate their shapes by mathematical formulations. As we can simulate, test and virtually validate products, we will have a model that is faithful to its real object and easy to manipulate and modify. This paper is organized as follows: first, a literature review covering the reconstruction of the CAD model from a set of points is presented. Next, the general algorithm to reconstruct the deformed CAD model is exposed. Then, the algorithms utilized to reconstruct these different types of surfaces are detailed, followed by an illustration of the results. Conclusions is presented at the end.

2. State of the art

Computer-aided design is the use of computer systems to create three-dimensional graphical representations of physical objects. The benefits of CAD include visualization of 3D objects that were scanned, simulation, testing and virtually validate products.

A CAD model can be reconstructed from a mesh or from a set of 3D points. On the first hand, reconstruction of CAD model from a mesh, many researches has been developed to reconstruct this model. In [1], Volpin proposed a new method for mesh simplification and surface reconstruction. The method begins by simplifying the initial mesh model by first constructing restricted curvature deviation regions, generating a boundary conforming finite element quadrilateral mesh of the regions, and then fitting a smooth surface over the quadrilateral mesh using the plate energy method. After the construction of the quadratic mesh, Volpin provides continuity between the surfaces reconstructed from the regions.

To refine the meshes, Ren [2] developed a remeshing algorithm to iteratively refine a given mesh to be closer to the actual shape of the meshed object. This method inserts a new node between two consecutive nodes of the old mesh respecting the type and form of the surface. This subdivision divides each triangle by four. Refinement methods are applied to the mesh to evaluate the

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.181

triangulation of 3D surfaces. These methods add and improve information for better reconstruction.

In the same context as Ren, several methods have been presented for the approximation of a surface by successive subdivision of the corresponding mesh [3]. A new node is inserted between two successive nodes. So, each edge is divided into two and each triangle is divided into four triangles respecting the shape of the mesh object. Other methods have been based on different subdivisions known patterns to refine a mesh successively in order to obtain a smooth surface [4] [5].

To evaluate a surface from a mesh, many methods have been based on Bezier triangles. For example, Walton [6] proposed an evaluation algorithm of a Bezier surface from a triangle of a mesh. This algorithm was then improved by Owen [7]. This algorithm attributes for each node of the triangulation a vector which will be used then as the normal to the reconstructed surface. Subsequently, from the coordinates of the three vertices of the triangle and the three normal vectors on these vertices, other control points will be calculated.

In [8], Beniere and al., have proposed a new method to reconstruct a boundary representation (B-Rep) model based on the extraction of geometric primitives from a 3D mesh to detect geometric primitive types of the mesh and to compute the parameters that give the best fit. To define the topology of the object, intersections between primitives should be calculated.

Louhichi and al. [9] proposed a method to reconstruct a CAD model from a deformed mesh. This method consists of using the weighted displacement estimation (WDE) method to solve the problem of the inconstant density of information over the deformed surface and the un-organization of the set of points. The idea is to calculate the regular lattice of control points on the initial surface (before deformation). To update the position of the B-spline surface lattice control points, the weighted displacement of each control point should be estimated. Using this lattice, we can compute the parameters of the B-spline surface and we reconstruct the deformed CAD face.

On the other hand, reconstruction of CAD model from a point cloud, many methods have been proposed. The method of Ball-Pivoting (BPA) [10] computes a triangular mesh interpolating a given point cloud. Generally, the points are surface samples acquired with multiple range scans of object. The principle of BPA is very simple: Three points form a triangle if a ball of a user-specified radius ρ touches them without containing any other point. Starting with a seed triangle, the ball pivots around an edge until it touches another point, forming another triangle. The process continues until all reachable edges have been tried, and then starts from another seed triangle, until all points have been considered.

Also, the Poisson method [11] is one of the strongest methods in the domain of 3D surface reconstruction. It uses the Poisson equation to interpolate a set of oriented points. It computes the gradient of an indicator function χ which is a vector field equal to zero almost everywhere except at points near the surface, where it is equal to the inward surface normal. Thus, the oriented point samples can be viewed as samples of the gradient of the model's indicator function which is defined as 1 at points inside the model, and 0 at points outside, and then the reconstructed surface is obtained by extracting an appropriate iso-surface.

In the context of B-Splines surfaces, many methods have supposed that the surface has a simple topological type. For example, Dietz [12], Hoschek and Schneider [13], Rogers and Fog [14] and Sarkar and Menq [15] assume that the surface is a deformed quadrilateral region with trimmed boundaries. In [16], Hoppe and al. have developed an algorithm that takes a set of unorganized points $\{x_1, \ldots, x_n\}$ of an unknown manifold M and produces as output a simplicial surface that approximates M. Neither the topology, the presence of boundaries, nor the geometry of M are assumed to be known in advance, all are inferred automatically from the data.

3. The general algorithm for 3D model reconstruction

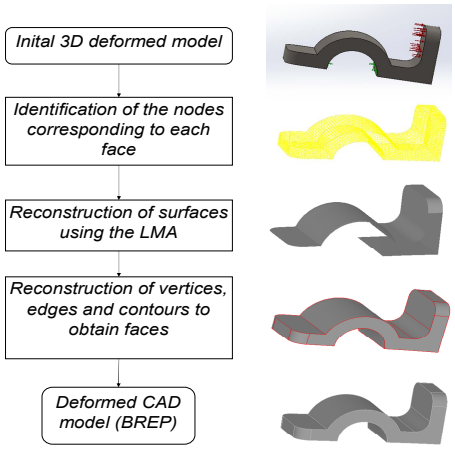


Figure 1: General algorithm of the reconstruction [9]

In general, the reconstruction algorithm follows the hierarchy of the BREP model (see Figure 1). In a first step, it begins with the identification of the topology (the identification of triangulations - or the point cloud corresponding to the faces). After, the surfaces are determined from the previous information. In the third stage, contours are added to surfaces to obtain faces in order to reconstruct the deformed model.

The third step, reconstruction of the surfaces from 3D points, is the most difficult step in this algorithm. In fact,

this problem has been investigated by various works. The surfaces have two types: simple classical entities which are described by mathematical formulas and can be modeled as perfectly as other complex objects. Among these primitives we mention: the plane, cylinder, cone, sphere, torus... or complex geometry entities (Bezier, B-Spline, NURBS ...). We have chosen to use the Levenberg Marquardt algorithm to approximate geometric entities.

4. Surface reconstruction

To reconstruct a geometric primitive from a set of 3D points, the Levenberg Marquardt algorithm [17] is applied to minimize the objective function which is the sum of non-linear least squares problems. The main idea of this algorithm is to find the vector p which minimizes the following objective function: $J(p) = \sum d_i^2(p)$, such as d_i is the distance from the point $\boldsymbol{x_i}$ to the geometry defined by the vector of parameters p.

The algorithm requires an initial estimation and the first derivatives of the distance function. In the case of a sphere, for example, $p = (\boldsymbol{x}, r) = (x, y, z, r)$, where x is the center of the sphere and r its radius. The distance equation $d_i = d_i(\boldsymbol{x_i}) = |\boldsymbol{x}_i - \boldsymbol{x}| - r$ and the objective function to minimize is $J(\boldsymbol{x}, \boldsymbol{r}) = \sum(|\boldsymbol{x}_i - \boldsymbol{x}| - r)^2$.

Derivatives of this objective function are defined as follows:

$$\frac{\partial J}{\partial x} = -2\sum d_i(x_i - x)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial J}{\partial y} = -2\sum d_i(y_i - y)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial J}{\partial z} = -2\sum d_i(z_i - z)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial J}{\partial r} = -2\sum d_i$$

And derivatives of the distance d_i are:

$$\frac{\partial d_i}{\partial x} = -(x_i - x)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial d_i}{\partial y} = -(y_i - y)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial d_i}{\partial z} = -(z_i - z)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial d_i}{\partial r} = -1$$

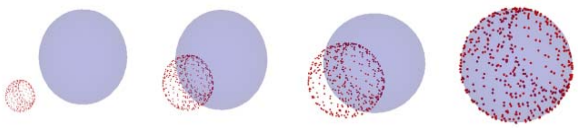


Figure 2: LM reconstruction of a sphere after 10 iterations

The literature has shown that it is a very reliable method which converges from the first iterations. The figure (Figure 2) represents an example of the reconstruction of a sphere although its initial parameters are so far off the exact parameters.

The red points are 3D points to fit. The blue sphere is the sphere with parameter p which is optimized at each iteration. After 10 iterations, the sphere passing through the 3D points is reconstructed.

However, to fit a torus, the vector $p = (\boldsymbol{x}, \boldsymbol{A}, r, R) = (x, y, z, A, B, C, r, R)$, such as $\boldsymbol{x}$ is the center of the torus, $\boldsymbol{A}$ is the direction numbers of the torus axis, r is the major radius and R is the minor radius. The equation of the distance: $d_i = d_i(\boldsymbol{x_i}) = \sqrt{g_i^2 + (f_i - r)^2} - R$, where $f_i = \sqrt{|\boldsymbol{x}_i - \boldsymbol{x}|^2 - g_i^2}$, $g_i = \boldsymbol{a}(\boldsymbol{x_i} - \boldsymbol{x})$ and the objective function: $J(\boldsymbol{x}, \boldsymbol{A}, r, R) = \sum\left[\sqrt{g_i^2 + (f_i - r)^2} - R\right]^2$

In the case of a plane, $p = (\boldsymbol{x}, \boldsymbol{a}) = (x, y, z, A, B, C)$, where $\boldsymbol{x}$ is a point on the plane and $\boldsymbol{a}$ is the direction cosines of the normal to the plane. The distance equation: $d_i = d_i(\boldsymbol{x_i}) = d(\boldsymbol{x_i}, \boldsymbol{x}, a) = \boldsymbol{a}(\boldsymbol{x_i} - \boldsymbol{x}) = a(x_i - x) + b(y_i - y) + c(z_i - z)$ and the objective function $J(\boldsymbol{x}, \boldsymbol{a}) = \sum {g_i}^2 = \sum[\boldsymbol{a}.(\boldsymbol{x_i} - \boldsymbol{x})]^2$.

To fit a cylinder given its 3D points, the initial vector $p = (\boldsymbol{x}, \boldsymbol{A}, r) = (x, y, z, A, B, C, r)$, such as $\boldsymbol{x}$ is a point on the cylinder axis, $\boldsymbol{A}$ is the direction numbers axis and r is the radius of this cylinder. The distance equation $d_i = d_i(\boldsymbol{x}_i) = f_i - r$, where $f_i = \sqrt{|\boldsymbol{x}_i - \boldsymbol{x}|^2 - g_i^2}$ and $g_i = \boldsymbol{a}(\boldsymbol{x_i} - \boldsymbol{x})$. The objective function $J(\boldsymbol{x}, \boldsymbol{A}, r) = \sum(f_i - r)^2$. To calculate the initial cylinder axis $\boldsymbol{A} = (\mathrm{A}, \mathrm{B}, \mathrm{C})$, the following inertia matrix which is symmetric should be calculated:

$$\begin{pmatrix} b^2 + c^2 & -a * b & -a * c \\ -a * b & a^2 + c^2 & -b * c \\ -a * c & -b * c & a^2 + b^2 \end{pmatrix}$$

Then, the values of the eigenvectors of this symmetric matrix which return a matrix should be determined to find the initial axis ($\mathrm{A} = \boldsymbol{A}_x$, $\mathrm{B} = \boldsymbol{A}_y$, $\mathrm{C} = \boldsymbol{A}_z$) which is its first colon.

Levenberg Marquardt algorithm is an iterative method which interpolates between the Gauss–Newton algorithm and the method of gradient descent. Its advantage is to find a solution even if it starts very far off the final minimum. When the current solution is far from the best solution, the algorithm behaves like the gradient algorithm: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes like the Gauss-Newton method. The solution $\boldsymbol{p}^*$ of each iteration can be expressed as follow:

$$\boldsymbol{p}^* = \boldsymbol{p}(\boldsymbol{\lambda}) = -\left(\boldsymbol{F_0^T F_0} + \boldsymbol{\lambda D^T D}\right)^{-1} \boldsymbol{F_0^T d}(\boldsymbol{p_0})$$

$\boldsymbol{F_0}$ is the matrix having the gradient of $\boldsymbol{d_i}(\boldsymbol{p_0})$ as its ith row, $\boldsymbol{D}$ is an appropriate weighting matrix, $\boldsymbol{d}(\boldsymbol{p_0})$ is the vector of residuals $\boldsymbol{d_i}(\boldsymbol{p_0})$, and $\boldsymbol{\lambda} \geq \boldsymbol{0}$ is a variable called the Levenberg Marquardt parameter. The matrix $\boldsymbol{F_0^T F_0} + \boldsymbol{\lambda D^T D}$ is named $\boldsymbol{H}$ and the vector $\boldsymbol{F_0^T d}(\boldsymbol{p_0})$ is called $\boldsymbol{V}$. The

algorithm includes a modification suggested by Nash [18] which is to use a weighting matrix defined so that $\boldsymbol{D^T D}$ is the identity matrix plus the diagonal of $\boldsymbol{F_0^T F_0}$. Nash's use of the identity in the definition of $\boldsymbol{D}$ forces $\boldsymbol{H}$ to be positive definite. Since $\boldsymbol{H}$ is also symmetric, the system $\boldsymbol{Hx = -v}$ can be reliably solved using the Cholesky decomposition [19]. Factors with which $\boldsymbol{\lambda}$ is incremented and decremented respectively should be chosen. Also, the parameter vector, $\boldsymbol{p}$, must be normalized at every iteration.

To reconstruct B-Spline curves and surfaces, the Levenberg Marquardt algorithm is applied to optimize the control points of each geometric entity. In the case of a B-Spline curve, each point is defined using the following equation:

$$\mathrm{S(u)} = \sum_{\mathrm{i=0}}^{\mathrm{M+1}} \mathrm{N_{i,k}}(u_l)\mathbf{p_i}$$

Given a set of points $\mathbf{q_l}$ $(\boldsymbol{l = 0, \dots, L+1})$, the objective is to find the curve that approximates these points. This curve interpolates the first and the last points $\mathbf{q_0}$ and $\mathbf{q_{L+1}}$ and it approximates the $\boldsymbol{L}$ points $\mathbf{q_l}$. This problem can be solved using the following least squares function:

$$J(p) = \sum_{l=1}^{L}\left(\sum_{i=1}^{M} N_{i,k}(u_l)\mathbf{p_i} - \mathbf{q_l}\right)^2$$

The goal is to find M control points that will be optimized. That's why, the differential of the objective function with respect to $\mathbf{p_j}$ must be defined as below:

$$\frac{\partial J(P)}{\partial p_j} = 2\sum_{l=1}^{L}\left(\sum_{i=1}^{M} N_{i,k}(u_l)\mathbf{p_i} - \mathbf{q_l}\right)N_{j,k}(u_l)$$
$$(j = 1,2,\dots,M).$$

And the derivative of the distance d_i is $\frac{\partial_{d_i}}{\partial_{p_j}} = N_{j,k}(u_l)$.

The algorithm requires the initialization of the degree of B-Spline curve and the control points that will be optimized in order to find the best control points that approximate the curve.

B-Spline basis functions are automatically calculated using the degree of the curve, the number of control points and the parameters $\boldsymbol{u_l}$ associated with each 3D point. At each iteration of the algorithm, new control points are calculated until the distance between the points of the B-spline curve defined by these control points and the set of 3D points is minimized.

In the case of B-Spline surface which is defined with its degree, the values of the knot vectors and its control points, the objective is to approximate the surface defined on each point by the following equation [20]:

$$S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)P_{i,j}$$

The knot vectors U and V have respectively $r+1$ and $s+1$ nodes, where $r = n+p+1$ and $s = m+q+1$. This surface is of $\boldsymbol{p}$ degree in the $\boldsymbol{u}$-direction and $\boldsymbol{q}$-degree in the $\boldsymbol{v}$-direction, such as $\boldsymbol{u}$ and $\boldsymbol{v}$ are the location parameters that locate a point in the surface [21].
Its node values must be in the interval [0,1]. So, the knot vectors should take the following form:

$$\underbrace{\mathbf{0}, \dots, \mathbf{0}}_{\mathrm{p+1}} < \boldsymbol{u_{p+1}} < \cdots < \boldsymbol{u_n} < \underbrace{\mathbf{1}, \dots, \mathbf{1}}_{\mathrm{p+1}}$$

The distance between two consecutive nodes of the subset $u_{p+1}, \dots, u_n$ for a uniform B-spline surface is identical and the vector is uniform where $i \in [p+1, n]$. $N_{i,p}$ and $N_{j,q}$ are the B-Spline basis functions of degree $\boldsymbol{p}$ in the $\boldsymbol{u}$-direction and degree $\boldsymbol{q}$ in the $\boldsymbol{v}$-direction. These basis functions can be computed in a recursive way:

$$N_{i,0}(u) = \begin{cases} 1 \; if \; u_i \le \mathrm{u} \le u_{i+1} \\ 0 \; otherwise \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} \cdot N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \cdot N_{i+p+1,p-1}(u)$$

The approximation of the surface is considered as an optimization problem which minimizes the distance between a point of the point cloud and the surface obtained by the initial control points.

The problem of surface reconstruction is described by Weiss and al. in [22] as below:

Given a set of points $\boldsymbol{S_{p_r}}$, for each point $\boldsymbol{p_r}$, there exists a point on the surface defined by the parameters $(\boldsymbol{u_r, v_r})$. Using the sum of the least squares, the surface passing through these points can be approximated.

Also in [23], Martin Aigner and Bert Jüttler, have explained that while the reconstruction of a surface from a set of points, certain distances between 3D points and the surface obtained by the initial control points should be minimized. These distances can be measured in different ways.

The geometric distance from a given point $\boldsymbol{p_r}$ to the surface is given by this Euclidean distance:

$$d_r = min\left\|\left(p_r - s_{u_r,v_r}\right)\right\|$$

The minimization of squares of the geometric distance leads to a least squares problem described as follows:

$$J(p) = \sum_{r=1}^{M} d_r^2$$

Due to the nonlinearity, an iterative method must be used to solve this problem. This method starts with a base surface. After, it calculates - for each data point - the closest associated points. By substituting these points [24], a least squares problem for surface parameters is obtained and can be determined using optimization techniques. Also, in [25], Xiao-Diao and al. described the problem of finding the minimum distance between a point and a B-Spline surface.

Given a set of points $\boldsymbol{q_l}$ and associated parameters $(\boldsymbol{u_l}, \boldsymbol{v_l})$ with $l = 0,1, \dots, L$, a B-Spline surface can be reconstructed. Fist, the parameters $(\boldsymbol{u_l}, \boldsymbol{v_l})$ of each point $\boldsymbol{q_l}$ should be initialized. To attend this objective, a base surface formed by initial control points should be defined. Then by projecting the points on the base surface, the parameters associated to each point are obtained. This base surface is defined as the average plane of the 3D points, and its boundaries are defined using the maximum parameter in the $\boldsymbol{u}$-direction and the maximum parameter in the $\boldsymbol{v}$-direction.

To reconstruct the surface passing through the $\boldsymbol{q_l}$ points the following approximation method of least squares is used:

$$J(\boldsymbol{P}) = \sum_{l=0}^{L} \left(\sum_{i=0}^{N} \sum_{j=0}^{M} N_{i,k_u}(u_l) N_{j,k_v}(v_l) \boldsymbol{P}_{i,j} - \boldsymbol{q}_l \right)^2 \rightarrow \min$$

$\boldsymbol{P}$: is a collection of $(N+1) \times (M+1)$ control points $\boldsymbol{P}_{i,j}$. Mathematically, this is equivalent to solve $(N+1) \times (M+1)$ linear equations:

$$\sum_{l=0}^{L} \left(\sum_{i=0}^{N} \sum_{j=0}^{M} N_{i,k_u}(u_l) N_{j,k_v}(v_l) \boldsymbol{P}_{i,j} - \boldsymbol{q}_l \right) N_{r,k_u}(u_l) N_{s,k_v}(v_l) = 0$$

with $(r = 0, \dots, N; s = 0, \dots, M)$.

Alternatively, the system of the above linear equations can be written as follows:

$$\sum_{l=0}^{L} \sum_{i=0}^{N} \sum_{j=0}^{M} N_{i,k_u}(u_l) N_{j,k_v}(v_l)\, N_{r,k_u}(u_l) N_{s,k_v}(v_l) \boldsymbol{P}_{i,j} = \sum_{l=0}^{L} N_{r,k_u}(u_l) N_{s,k_v}(v_l)\, \boldsymbol{q}_l$$

with $(r = 0, \dots, N; s = 0, \dots, M)$.

As shown, the above system has four matrixes of manipulation and multiplication. It is difficult to set the final matrix on the basis of the above representation.

However, if $\boldsymbol{q}_l$ are randomly sampled, the approximation algorithm of least squares of the set of points can be solved by deriving a simple representation. In fact, it is clear that the B-Spline surface can be represented like a B-Spline curve. This surface contains $(N+1) \times (M+1)$ control points.

By noting $K = (N+1) \times (M+1) - 1$, the following equation is obtained:

$$S(u, v) = \sum_{k=0}^{K} p_k B_k(u, v)$$

With $B_{j \times (N+1)+i} = N_i(u) N_j(v)$. Consequently, the problem of least squares can be written as follows:

$$J(\boldsymbol{P}) = \sum_{l=0}^{L} \left(\sum_{k=0}^{K} \boldsymbol{p}_k B_k(u_l, v_l) - \boldsymbol{q}_l \right)^2 \rightarrow \min$$

B: is a matrix of size $(L+1) \times (\mathrm{K}+1)$

$$\begin{pmatrix} B_0(u_0, v_0) & B_1(u_0, v_0) & \cdots & B_K(u_0, v_0) \\ B_0(u_1, v_1) & B_1(u_1, v_1) & \dots & B_K(u_1, v_1) \\ \vdots & \vdots & \vdots & \\ B_0(u_L, v_L) & B_1(u_L, v_L) & \dots & B_K(u_L, v_L) \end{pmatrix}$$

Using the same principle as B-Spline curves, the derivative of the distance function on each point is:

$\frac{\partial d_i}{\partial p_j} = B_j(u_l, v_l)$ and the differential of the objective function with respect to $\mathbf{p_j}$ is :

$$\frac{\partial J(P)}{\partial p_j} = 2 \sum_{l=0}^{L} \left(\sum_{i=0}^{K} \boldsymbol{p}_i B_i(u_l, v_l) - \boldsymbol{q}_l \right) B_j(u_l, v_l) \quad (j = 0,1, \dots, K).$$

Finally, by applying the algorithm of Levenberg Marquardt which minimizes the objective function of the sum of this non-linear least squares problem, the approximated control points of the reconstructed surface are obtained.

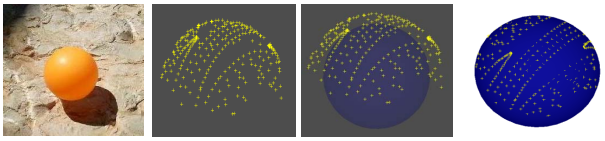
$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j}$$

5. Validation

The proposed approach is developed by using Microsoft Visual Studio C++ 6.0 and Open Cascade which is an open source software development platform for 3D CAD.

To evaluate the proposed method's performance, many case studies are used. First, we present the results of reconstructing four geometric primitives by approximating the point cloud using the Levenberg Marquardt algorithm. Then, we present the results of the reconstruction of a B-Spline curve and two B-Spline surfaces. After that, the reconstruction of the complete CAD model of a deformed mechanical piece is presented.

5.1. The reconstruction of geometric primitives



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

Figure 3: LM reconstruction of a sphere after 10 iterations

(a) Real sphere of radius 19.8; (b) Point cloud of 3D points; (c) approximation of the sphere; (d) Reconstructed sphere of radius 19.8 after 5 iterations.

- Reconstruction of a real sphere of radius 19.8 cm (see Figure 3)

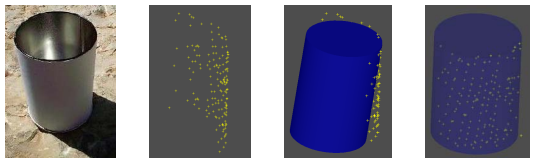


Figure 4: Reconstruction of a real cylinder after 2 iterations

(a) Real cylinder of radius 34, height 74 and axis (0,0,1); (b) point cloud of 3D scan; (c) Approximation of the cylinder ;(d) Reconstructed cylinder: radius 34, height 74, axis (0.2,0.4,1)

- Reconstruction of a real cylinder of radius 34 cm, height 74 cm and axis (0.2,0.4,1) (see Figure 4)

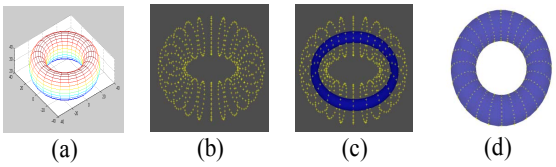


Figure 5: Reconstruction of torus after 5 iterations

(a) Torus of minor radius 10 cm and major radius 30 cm; (b) Point cloud of the torus; (c) approximation of the torus; (d) Reconstructed torus of major radius 30 and minor radius 10

- Reconstruction of a torus of minor radius 10 cm and major radius 30 cm (see Figure 5)

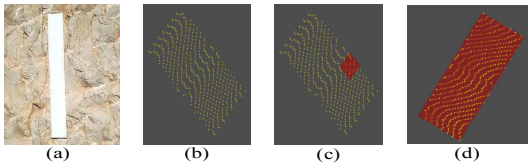


Figure 6: Reconstruction of a real plane after 2 iterations

(a) Real plane of axis (-0.1,-0.2,1) ; (b) Point cloud of the plane ; (c) approximation of the plane ; (d) Reconstructed plane from the point cloud of axis (-0.1,-0.2,1).

- Reconstruction of a real plane using its 3D point cloud of axis (-0.1, -0.2, 1) (see Figure 6)

5.2. The reconstruction of B-Spline curves and surfaces

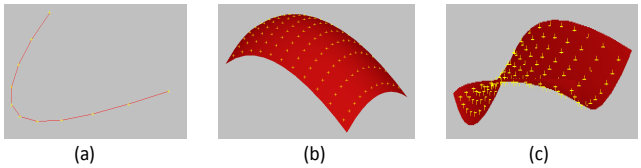


Figure 7: Optimization of control points of a B-Spline curve and two B-Spline surfaces

(a) Approximated B-Spline curve of degree 3; (b) and (c) Approximated B-Splines surfaces of degree 2

5.3. The reconstruction of mechanical parts

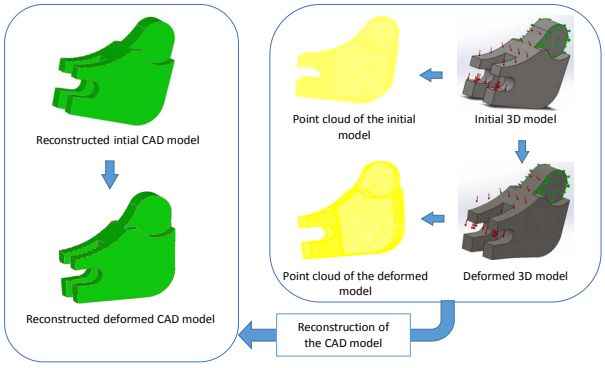


Figure 8: Reconstruction of a deformed mechanical part

The objective of our researchers is to reconstruct the CAD model from the point cloud of deformed mechanical parts which are reconstructed of planar and cylindrical faces. After its deformation, these faces became of complex type and have been modelized using B-Spline form.

The figure (Figure 8) presents a simple mechanical model containing 26 planar and cylindrical faces. These surfaces became complex B-Spline surfaces after their deformation.

6. Conclusion

In this paper, the iterative approximation method (LM) is used to reconstruct the CAD model specially deformed mechanical model. In the first, we have presented the method to reconstruct simple geometric primitives by defining the equation of the distance of each one. Then, we demonstrated the reconstruction of B-Spline surface by adjusting the control points iteratively. In each iteration, the distance between the 3D scanned points and the points of the reconstructed surface is minimized. The convergence of this method from the first iterations shows its performance to reconstruct CAD models.

7. References

[1] O. Volpin, A. Sheffer, M. Bercovier and L. Joskowicz, "Mesh simplification with smooth surface reconstruction," Computer-Aided Design, vol. 30, no. 11, pp. 875-882, 1998.

[2 B. Ren and I. Hagiwara, "Composite freeform surface reconstruction using recursive interpolating subdivision scheme," Computers in Industry, vol. 50, no. 3, pp. 265-275, 2003.

[3] X. Yang, "Surface interpolation of meshes by geometric subdivision," Computer-Aided Design, vol. 37, no. 5, pp. 497-508, 2005.

[4] Weiyin Ma and J P Kruth , "Parameterization of randomly measured points for least squares fitting of B-Spline curves and surfaces," Computer Aided Design, vol. 27, no. 9 , pp. 663-675, 1995.

[5] D. Rypl and Z. Bittnar, "Triangulation of 3D surfaces reconstructed by interpolating subdivision," Computers & Structures, vol. 82, no. 23-26, pp. 2093-2103, 2004.

[6] D. Walton and D. Meek, "A triangular G1 patch from boundary curves," Computer-Aided Design, vol. 28, no. 2, pp. 113-123, 1996.

[7] S. Owen and D. White, "International Journal for Numerical Methods in Engineering," Mesh-based geometry, vol. 58, no. 2, pp. 375-395, 2003.

[8] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton and W. Puech, "A comprehensive process of reverse engineering from 3D meshes to CAD models," Computer-Aided Design, vol. 45, no. 11, pp. 1382-1393, 2013.

[9] B. Louhichi, G. Abenhaim, A. Tahan, "CAD/CAE integration: updating the CAD model after a FEM analysis," Int J Adv Manuf Technol, vol. 76, no. 1-4, pp. 391-400, 2014.

[10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," IEEE Trans. Visual. Comput. Graphics, vol. 5, no. 4, p. 349–359, 1999.

[11] M. Kazhdan, M. Bolitho, H. Hoppe, "Poisson Surface Reconstruction," in Symposium on Geometry Processing, 2006.

[12] U. Dietz, "Erzeugung glatter Flachen aus MeBpunkten," Technical Report 1717, Department of Mathematics, University of Darmstadt, Germany, February 1995.

[13] J. Hoschek, F. Schneider and P. Wassum, "Optimal approximate conversion of spline surfaces," Computer Aided Geometric Design, vol. 6, no. 4, p. 293–306, 1989.

[14] D. Rogers and N. Fog , "Constrained B-spline curve and surface fitting," Computer-Aided Design, vol. 21, no. 10, p. 641–648, 1989.

[15] B. Sarkar and C. Menq, "Parameter optimization in approximating curves and surfaces to measurement data," Computer Aided Geometric Design, vol. 8, no. 4, p. 267–290, 1991.

[16] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganized points," ACM SIGGRAPH Computer Graphics, vol. 26, no. 2, pp. 71--78, 1992.

[17] C. Shakarji, "Least-squares fitting algorithms of the NIST algorithm testing system," J. Res. Natl. Inst. Stand. Technol., vol. 103, no. 6, p. 633, 1998.

[18] J. Nash, "Compact Numerical Methods for Computers," in Linear Algebra and Function Minimisation, Bristol, England, 1979.

[19] G. Golub and C. van Loan, "Matrix Computations," in The Johns Hopkins University Press, Baltimore, 1983.

[20] L. Piegl, W. Tiller, "The NURBS Book," New York, 1997.

[21] W. Ma and J.P. Kruth, "Parametrization of randomly measured points for least squares fitting of B-spline curves and surfaces," CAD, vol. 27, p. 663–675, 663–675 1995.

[22] V. Weiss, L. Andor, G. Renner, T. Várady, "Advanced surface fitting techniques," Computer Aided Geometric Design, vol. 19, no. 1, pp. 19-42, January 2002.

[23] M. Aigner and B. Jüttler, "Gauss-Newton type Techniques for Robustly Fitting Implicitly Defined Curves and Surfaces to Unorganized Data Points", in: Shape Modeling International, pp. 121–130, 2008.

[24] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," in SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques, New York, NY, USA, 2001.

[25] X. Chen, G. Xu, J. Yong, G. Wang, J. Paul, "Computing the minimum distance between a point and a clamped B-spline surface," Graphical Models, vol. 71, no. 3, pp. 107-112, 2009.