Analysis of clustering algorithms for by-zone machining of free-form surfaces

Mahfoud Herraz, Jean-Max Redonnet, Marcel Mongeau, Mohammed Sbihi

Analysis of clustering algorithms for by-zone machining of free-form surfaces

Mahfoud Herraz[a], Jean-Max Redonnet[a], Marcel Mongeau[b] and Mohammed Sbihi[b]

[a] Université Paul Sabatier and Institut Clément Ader, Université de Toulouse, France
[b] ENAC, Université de Toulouse, France

February 12, 2022

Abstract

End-milling of free-form surfaces on multi-axis CNC machines are complex and expensive operations involved in the production of many high-value parts, such as molds and stamping dies. To carry out such time-expensive operations, partitioning the surface into zones, in order to mill each zone with the most suitable parameters, is an approach gaining credit in recent years.

Data clustering algorithms provide tools that appear to be naturally adapted to perform such a procedure. In this paper, is provided a fine analysis of various clustering algorithms, as well as the feature vectors and the metrics commonly used. Furthermore, the specific problems that may arise when applying clustering algorithms to free-form surfaces partitioning are presented and original solutions to deal with them are proposed.

Keywords: free-form surfaces; end-milling; toroidal cutter; clustering; K-means, RPCL, HAC

Contents

1 Introduction

Tool path planning is an important issue for the end-milling of free-form surfaces, due to the high manufacturing cost commonly observed for parts such as molds, stamping dies and others high added value parts. Therefore, any way to decrease machining time may lead to significant gains in terms of production costs. In this context, partitioning the surface into zones is a very interesting approach that is more and more studied and used. Indeed, by-zone machining allows one to mill each zone with different, more efficient, tailored parameter values (which include, among others, the milling direction).

Besides, the choice of by-zone machining is directly related to the choice of the cutter shape. Indeed, it is well established that a toroidal cutter may lead to better results than its ball-end counterpart [1]. Actually, a toroidal cutter provides better results than a ball-end one when machining along the steepest-slope direction; but when used perpendicularly to the steepest-slope di-

rection, its performances are worse than those of a ball-end cutter. Because the steepest-slope direction may vary a lot across a free-form surface, partitioning this surface into several zones (each of which will then be machined along an appropriate direction) is considered as a promising approach to improve efficiency of the toroidal cutter choice.

2 Bibliography

Defining zones for end-milling of free-form surfaces is a complicated problem which has been addressed by many authors. Most studies focus on 3-axis machining. One of them is presented in [2]. Using a ball-end cutter, this method is based on the preferred machining direction field, which maximizes the width of the machining strip (or step-over distance) respecting the scallop height constraint. This field has the properties of symmetric tensor of order 2, which makes it possible to identify the critical points of this tensor corresponding to degenerate points, where all the directions are equivalent. Then, depending on the type of each critical point, boundaries between zones are identified around this point, which define a partition of the surface. Finally, each zone is machined using an iso-scallop strategy. A very similar method is proposed in [3]: an analytical approach is presented, where the machining direction that maximizes the strip width is identified.

Heuristic optimization methods inspired by the vehicle routing problem are proposed in [4]. The aim is to partition the surface into zones with small variations of the steepest-slope direction in order to increase the effective radius of the toroidal cutter. This should maximize the step-over distances and minimizes the tool-path length. The best results are obtained by the Clarke and Wright algorithm (described in [5]), the parallel implementation of this algorithm being faster.

The same goal, looking for an optimal partition, is pursued in the approach presented in [6]. This approach is based on adaptive multiagent systems (AMAS) [7]. The idea is to partition the part surface into zones according to a criterion based on optimal machining direction which is, for each point, the steepest-slope direction. The part surface is meshed, and the optimal direction is calculated on each mesh unit. The problem is then solved (using the AMAS approach) by considering an agent on each mesh unit. The zones are characterized by two antagonistic criteria: the number of elements (agents) in the zone and the maximum difference between any two directions in the zone. The optimization process requires, however, numerous iterations to converge towards a stable state in which no agent makes a new decision.

In [8], the authors propose a partitioning approach for 3-axis machining with a toroidal cutter guaranteeing the efficiency of the toroidal cutter. The first zone contains all the points that could be machined along the steepest-slope direction of the point at which the slope angle is maximal, while satisfying at least a minimal improvement of the effective radius compared to the same radius ball-end cutter. The second zone is build similarly considering the remaining points. This process is iterated until no point remains.

Other studies addresses 3+2-axis machining, which is the same as 3-axis machining, adding two rotation axes used for positioning the part between two milling phases. Among them, the method proposed in [9] discretizes the surface with a regular isoparametric mesh and calculates the Gaussian curvature and the mean curvature for each node of the mesh. Then, a curvature type: convex, concave, or saddle-like, is assigned to each point. After that, the points are divided into interference-free points, and points that are not accessible. This partition is then refined using the fuzzy C-means algorithm, and the number of zones is decided by the subtractive clustering method [10].

One of the most noticeable work in partitioning methods for 3+2-axis machining is the free-form surface partitioning method proposed in [11]. They first conduct a study of the parameters to be used for the feature vector in order to obtain a better partition of the surface by using the fuzzy C-means algorithm. Calling $\mathbf{S}(u, v)$ the parametric function defining a free-form surface to be machined, and $\mathbf{n}(u, v)$ the vector normal to this surface at point (u, v); this study, carried out on two test surfaces, shows that the parameter combination $(u, v, \mathbf{n}(u, v))$ is effective. The orientation of the tool, denoted by the vector $\mathbf{T}$, is calculated such that: the axis of the tool is coplanar with the feed rate, $\mathbf{F}$ and the axis, $\mathbf{Z_m}$, of the machine. The tilt angle is specified by the user. For points in regions where the surface is convex, the tilt angle can be null, and it can be very small at concave points such as local gouging is avoided. The parallel-plane machining strategy adopted for one test case shows that generally the tool-path length decreases as the number of zones increases. In fact, the machining time decreases until an optimal number of zones is reached; it then increases due to the time necessary for the tool to withdraw and to travel from zone to zone.

In [12], the K-means algorithm is used to partition a free-form surface for 3+2 axis machining. Their feature vector is composed of the 3D coordinates of $\mathbf{S}(u, v)$ and $\mathbf{n}(u, v)$, which gives a six-parameter feature vector (S_x, S_y, S_z, n_x, n_y, n_z). The choice of 3D coordinates may however cause problems (see Section 4).

Partitioning surfaces for 5-axis machining has also been addressed in the literature. A method based on the Normalized-cut algorithm is proposed in [13]. This algorithm is used in image segmentation and vector field classification. First, a field of optimal directions is projected onto a regular grid, which defines the interpolation points of the tool-path in the parametric space. Then, the

surface is partitioned using the Normalized-cut algorithm, for which the dissimilarity measure is a convex combination of proximity parameters (the parametric coordinates u and v), and the optimal direction of machining. Each zone is machined using either a "zigzag" iso-scallop strategy or an iso-contour strategy, according to the nature of the critical points in the zone.

To sum up, the objective of such approaches is to find an appropriate partitioning of the surface to optimize machining. Some approaches consider the partitioning as an optimization problem where the objective function is a machining criterion (milling time or tool-path length). Although it is computationally expensive, this kind of approach has the advantage of considering a partitioning that is directly related to machining. Other approaches consider the partitioning as a clustering problem with the aim of finding a partition of the surface with homogeneous and distinguished zones.

The common point of most of these approaches is that they are based on local criteria and properties: local geometry of the surface, effective radius and step-over distance at the contact point, etc. Given the complexity of free-form surfaces, a more global view of the context may lead to more effective partitions (in terms of machining criteria). A PCA-based free-form surface analysis has been presented in [14] for the whole surface. It can be used in the same way for each single zone to build a quick approximation of the zone shape and orientation. The partitioning process can then rely on this approximation to exploit well-known results about milling processes. For example, it is straightforward to understand that the longer the tool-path in a given direction is, the more efficient the milling will be, because the slowdowns due to the kinematic limits at both ends of the path are less influential. It is worth mentioning that local-only approaches cannot integrate this kind of results.

3 Preliminary considerations

Let S be a parametric surface defined by $\mathbf{S}(u, v)$. A *partition* of S is a finite family of K subsets $(z_i)_{i=1,\dots,K}$ of S that are pairwise disjoint and whose union is equal to S. In other terms:

$$z_i \cap z_j = \emptyset, \text{ for any pair of integers } 1 \leqslant i < j \leqslant K, \text{ and}$$

$$\bigcup_{j=1}^{K} z_j = S$$

In machining of free-form surfaces, zones are defined through a set of predetermined sample points. Any set of sample points can be used. However, the best results are obtained when these points are equally spread over the whole surface. To achieve this condition, a regular mesh of isoparametric curves is defined onto the surface. This way, a set of elementary meshes, defined by their parametric boundaries, is also defined. For each mesh unit, a datapoint located at its center is defined. Clustering algorithms operate on these center sample points to define zones (Figure 1).

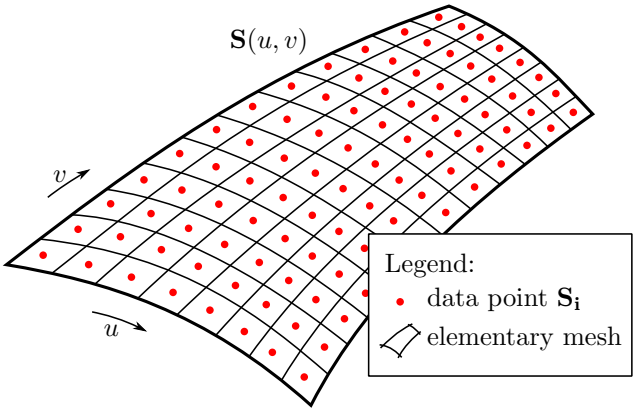


Figure 1: Sample point definition.

Classification algorithms are commonly used in data mining procedures that aims to extract knowledge from large databases. An extensive review on unsupervised learning and classification can be found in [15]

A distinction must be done between supervised and unsupervised classification. Supervised classification algorithms are based on a set of pre-classified data (learning set) to predict the class of a new data. In this case, the class number is known and not decided by the algorithm. Supervised algorithms include: decision trees, artificial neural networks, naive Bayesian classifiers, and support vector machines. A literature review on classification and combining algorithms can be found in [16].

On the other hand, unsupervised classification algorithms are not based on any prior knowledge. At the opposite of supervised algorithms, the purpose is not to predict the class of a new point but to classify existing, but unordered, data into homogeneous classes. Unsupervised classification algorithms are also called clustering algorithms. Most of the time, the number of classes (*clusters*) is initially unknown. However, some algorithms allow the user to set this value as an input, in regard of the specific data to be classified. Clustering algorithms include: K-means, fuzzy C-means, hierarchical classification, and competitive learning. A survey of clustering algorithms can be found in [17].

Any clustering algorithms rely on 3 components:

- the feature vector that represent data
- the metric that defines a method to compute distances between data
- the algorithm itself

The following study focuses on the analysis of each of these three components in the particular case of partitioning of free-form surfaces.

4 The feature-vector

For most clustering problems, the data to be processed is obvious and is part of the definition of the problem itself. But in the case of partitioning free-form surfaces, it is necessary to precisely define the most relevant parameters in order to obtain a convincing result. These parameters constitute what is called the feature vector. The choice of this vector has a great impact on the result of the clustering process.

In free-form surfaces end-milling, two different types of parameters may be taken into consideration:

- spatial proximity parameters: 3D coordinates of datapoints or their parametric coordinates (u, v) could be meaningful choices for this type.
- machining related parameters: the normal vector $\mathbf{n}(u, v)$, the curvatures (Gaussian, mean or principal) could be meaningful choices for this type.

A good analysis of feature vector component is presented in [11]. Finally, authors of this paper choose to use the parametric coordinates (u, v) and the normal vector $\mathbf{n}(u, v)$. In [12] the 3D space coordinates (S_x, S_y, S_z) and the normal vector $\mathbf{n}(u, v)$ have been chosen.

In the present study a slightly different choice have been made. First, remark that the use of 3D coordinates as parameters in the feature vector may not be completely suitable for free-form surfaces. Indeed, two 3D points may be far from each other on a part' surface while having a small Euclidean distance. This case is illustrated in Figure 2: the points P_1, P_2 and P_3 are defined such that P_3 is closer to P_1 than P_2 in terms of Euclidean distance while P_2 is closer according to geodesic distance.

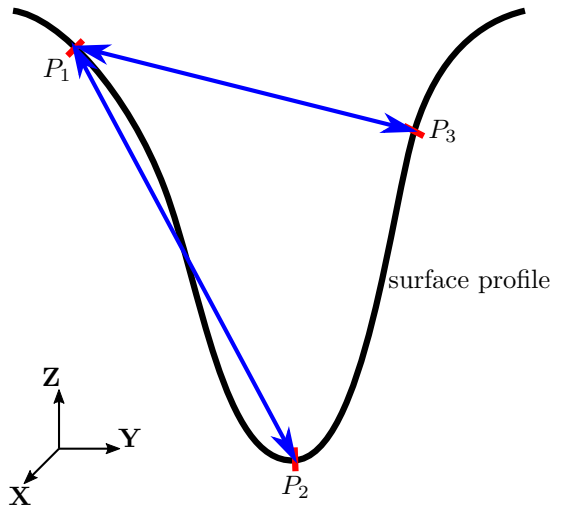


Figure 2: The problem using 3D coordinates as components of the feature vector.

Ideally, the geodesic distance should be considered instead. Yet, it is very expensive to compute and could not be taken into consideration for real-world application.

Thus, a better alternative is the (u, v) parametric coordinates. It is a convenient and computational-cheap solution that can be embedded in an Euclidean distance metric (see Section 5.1).

Besides, instead of the $\mathbf{n}(u, v)$ coordinates, parameters directly related to machining process have been chosen in this paper. This choice is based on two well-known results about machining with a toroidal cutter: first, toroidal cutter efficiency is better when machining along the steepest-slope direction; and second, lower is the steepest slope, greater is the efficiency of the toroidal cutter. Since using the toroidal cutter involves directly the steepest slope s and its direction θ, it is more suitable to use them rather than the normal vector $\mathbf{n}$ in the feature vector. As illustrated in Figure 3, these angles are defined for each mesh i:

- s_i: the value of the steepest slope at the center of the mesh i
- θ_i: the angle between the direction of the steepest slope projected on the plane $\mathbf{X_m}, \mathbf{Y_m}$ of the machine reference and the axis $\mathbf{X_m}$.

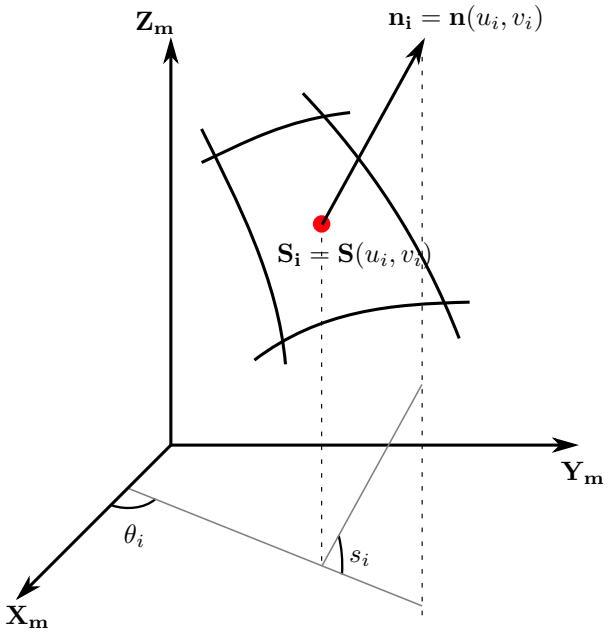


Figure 3: The four components (u_i, v_i, s_i and θ_i) of the feature vectors.

Furthermore, they are directly employed in the analytical formula used to calculate the effective radius of a toroidal cutter [18].

To sum up, like in [11], the parametric coordinates u and v of the datapoints are used to express the spatial proximity of datapoints, while $s(u, v)$ and $\theta(u, v)$ have been chosen as machining related parameters.

These choices have been made to reduce the size of the

feature vector, and to be more specific on parameters that are really influential on the machining process.

From a practical point of view, once the relevant parameters are selected, feature vectors must be calculated for each point of the dataset. As described in Section 3, the set of points the algorithms operate on is defined by discretizing the surface S with a regular isoparametric curves meshing. Let n be the number of meshes (or elementary surfaces). For each mesh $i \in [1, n]$, the feature vector is defined at the mesh center point: $x_i = (u_i, v_i, s_i, \theta_i)^T$. These points represent our dataset D. In what follows, the mesh number i is denoted m_i, and the datapoint attached to it is x_i. Furthermore, given a partition of S in K zones, the centroid of the zone z_k, $k \in [1, K]$, is denoted c_k.

5 The metric

For any clustering algorithm, it is necessary to define the notion of distance or, more precisely, dissimilarity between points represented be feature vectors. Hereafter, these distances measurement methods are called metrics. We present below the most commonly encountered metrics.

5.1 Euclidean distance

In the four-dimensional space of parameters (u, v, s, θ), the Euclidean distance between a mesh m_i and the center of a zone z_k is defined by:

$$d^2(x_i, c_k) = (u_i - \overline{u}_k)^2 + (v_i - \overline{v}_k)^2 + (s_i - \overline{s}_k)^2 + (\theta_i - \overline{\theta}_k)^2 \quad (1)$$

This distance gives the same importance to all the variables, which may not be relevant. In addition, several terms which are not homogeneous (different units and orders of magnitude) are added together, which lacks physical meaning. The advantage of the Euclidean metric lies in its simplicity of implementation.

5.2 Mahalanobis distance

This metric is widely used in statistics and signal processing [19]. It differs from Euclidean distance because it takes into account the variance and correlation of the data set. Thus, unlike the Euclidean distance where all the components of the vectors are treated independently and in the same way, this metric gives a lower weight to the most dispersed variables and takes into account the correlation between the variables.

Let Σ be the covariance matrix of the feature vectors of the whole dataset:

$$\Sigma = \begin{pmatrix} \mathrm{Var}(u) & \mathrm{Cov}(u,v) & \mathrm{Cov}(u,s) & \mathrm{Cov}(u,\theta) \\ \mathrm{Cov}(u,v) & \mathrm{Var}(v) & \mathrm{Cov}(v,s) & \mathrm{Cov}(v,\theta) \\ \mathrm{Cov}(u,s) & \mathrm{Cov}(v,s) & \mathrm{Var}(s) & \mathrm{Cov}(s,\theta) \\ \mathrm{Cov}(u,\theta) & \mathrm{Cov}(v,\theta) & \mathrm{Cov}(s,\theta) & \mathrm{Var}(\theta) \end{pmatrix}$$

Then, the distance d between a mesh m_i and the center of the zone z_k is defined by:

$$d^2(x_i, c_k) = (x_i - c_k)^T \Sigma^{-1} (x_i - c_k)$$

5.3 Standard deviation-based distance

This metric is mostly used in statistics for processing heterogeneous data (*i.e.* with various magnitudes and variances). It is based on Euclidean distance, but each term is divided by the corresponding standard deviation. Thus, the distance between the mesh m_i and the center of the zone z_k is defined by:

$$d^2(x_i, c_k) = \frac{(u_i - \overline{u}_k)^2}{\sigma_u} + \frac{(v_i - \overline{v}_k)^2}{\sigma_v} + \frac{(s_i - \overline{s}_k)^2}{\sigma_s} + \frac{(\theta_i - \overline{\theta}_k)^2}{\sigma_\theta}$$

where $\sigma_u = \sqrt{\mathrm{Var}(u)}$ is the standard deviation of the variable u over the whole surface S. Using a similar definition of standard deviation for the other variables, this metric can also be defined as:

$$\begin{aligned} d^2(x_i, c_k) &= (x_i - c_k)^T \begin{pmatrix} \sigma_u & 0 & 0 & 0 \\ 0 & \sigma_v & 0 & 0 \\ 0 & 0 & \sigma_s & 0 \\ 0 & 0 & 0 & \sigma_\theta \end{pmatrix}^{-1} (x_i - c_k) \\ &= (x_i - c_k)^T {D_\sigma}^{-1} (x_i - c_k) \end{aligned}$$

This distance could be considered as a special case of the Mahalanobis distance, where the covariance of the different pairs of variables is not taken into account. Actually, the distance based on the standard deviation does not take into account the correlation between the variables. However, it allows the summation of different terms since the weight assigned to each term is proportional to its variance, which is not the case for the Euclidean distance.

It is worth to be noted that the three previously presented distances are derived from the following general distance form d on $\mathbb{R}^4$, $d^2(x, y) = (x-y)^T M (x-y)$ where M is a symmetric positive definite matrix.

5.4 Ward distance

Unlike previous metrics, Ward distance is intended to measure dissimilarity between clusters rather than dissimilarity between datapoints. This means that among the algorithms presented in Section 6, the Ward distance can only be used with the HAC algorithm (Section 6.3).

Let C_k be a cluster containing n_k datapoints; its centroid c_k is defined by:

$$c_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i$$

Let $\varsigma(C_k)$ be the sum of the squared Euclidean distances calculated from each point of C_k to its centroid c_k; $\varsigma(C_k)$ is defined by:

$$\varsigma(C_k) = \sum_{i=1}^{n_k} \|x_i - c_k\|^2$$

For any pair of distinguished clusters C_i and C_j, which contains respectively n_i and n_j datapoints, the Ward distance indicates by how much this sum of squares will increase when the two clusters are merged:

$$d(C_i, C_j) = \varsigma(C_i \cup C_j) - \varsigma(C_i) - \varsigma(C_j) = \frac{n_i\, n_j}{n_i + n_j} \|c_i - c_j\|^2$$

When two clusters C_i and C_j are merged, the distance between $C_i \cup C_j$ and any other cluster C_k (which size is n_k) can be updated using the following relation:

$$d(C_i \cup C_j, C_k) = \frac{1}{n_i + n_j + n_k} \left((n_i + n_k)\, d(C_i, C_k) + (n_j + n_k)\, d(C_j, C_k) - n_k\, d(C_i, C_j) \right) \quad (2)$$

6 The algorithm

Numerous clustering algorithms and variants have been developed. Unsupervised algorithms can be divided into three main categories: deterministic centroid-based algorithms (K-means-like algorithms), competitive learning algorithms and hierarchical algorithms. To be as representative as possible of these three categories, one algorithm of each family has been tested. First, the K-means algorithm represents a large family of algorithms, including K-medoids and fuzzy C-means algorithms. It is based on the computation of distances from cluster centroids. Second, we select the Rival Penalized Competitive Learning [20] (RPCL) algorithm, which is one of the most common competitive learning algorithms. This algorithm has randomness features and requires few centroid calculations. The third algorithm is the Hierarchical Agglomerative Clustering [21] (HAC) algorithm which, like any hierarchical clustering algorithm, breaks free from the notion of cluster centroids by defining directly distances between clusters.

6.1 The K-means algorithm

The K-means algorithm is one of the most commonly used clustering algorithm. It was introduced in [22]. Its theoretical framework and its proof of convergence are presented in [15]. Various parallelized versions of this algorithm have also been proposed [23]. The K-means algorithm is presented in Algorithm 1.

The K-means algorithm generally results in partitions with smooth borders. The number of clusters is fixed and specified by the user. Note that in the special case of partitioning free-form it may be augmented after the convergence due to the existence of disconnected clusters — see Section 8. Most of the time, the Euclidean distance is used as metric, but a user defined metric can also be used. It is straightforward to define a stopping criterion (stationary point with no changes) for K-means algorithm and convergence is ensured for any Euclidean metric deriving from an inner product. However, K-means algorithm may diverge for non-Euclidean metrics, and it is important for convergence that centers of clusters are exactly the average points (and not the closest mesh center to average).

Algorithm 1: The K-means algorithm.

Data: Dataset $(x_i)_{1 \leqslant i \leqslant n}$, number of clusters K
Initialization:
Choose initial centroids $(c_k)_{1 \leqslant k \leqslant K}$
repeat
  `/* Step 1: class assignment */`
  for *i=1* **to** n
    $j = \arg\min_{1 \leqslant k \leqslant K} (\|c_k - x_i\|)$
    set $x_i \in C_j$
  `/* Step 2: centroids calculation */`
  for $k = 1$ **to** K
    $c_k = \frac{1}{n_k} \sum_{x \in C_k} x$
until *convergence (no more changes)*
return clusters C_k, $k = 1, 2, \ldots, K$

6.2 The RPCL algorithm

The Rival Penalized Competitive Learning algorithm (RPCL — Algorithm 2), introduced by [20], is a non-deterministic classification algorithm. Initially, a number of centroids are defined. At each iteration, a point of the dataset is randomly chosen, and the distance between each centroid and this point is calculated. The winning centroid (the closest one) is brought closer to the point in question, while its rival (the second closest one) is moved away (*i.e.* penalized). Unlike K-means algorithm, this algorithm may result in a partition containing fewer clusters than the initial number. Furthermore, the RPCL algorithm takes into account the history of the winning center (*i.e.* the number of times it won) in order to reduce the sensitivity of the algorithm to the initial position of the centroids.

RPCL is known to be faster than other algorithms, but its non-deterministic nature is likely to generate different solutions for the same problem. This means, from a practical point of view, that running the same algorithm again on the same surface can lead to a better (or worse) result. In the context of an industrial process, this may be very inconvenient.

Algorithm 2: The Rival Penalized Competitive Learning algorithm (RPCL).

Data: Dataset $(x_i)_{1\leqslant i\leqslant n}$, number of clusters K, α_w learning ratio of the winner, α_r learning ratio of the rival

Initialization:

Choose randomly the K centroids $(c_k)_{1\leqslant k\leqslant K}$

Set $m_k = 1$ for $k = 1, \cdots, K$ `/* wins count */`

repeat

  Choose randomly a point x_i, $1 \leqslant i \leqslant n$

  `/* Step 1: find winner and rival */`

  $w = \arg\min_k \gamma_k \|x_i - c_k\|^2$ `/* winner */`

  $r = \arg\min_{k\neq w} \gamma_k \|x_i - c_k\|^2$ `/* rival */`

  where $\gamma_k = \frac{m_k}{\sum_{k=1}^{K} m_k}, \forall\, 1 \leqslant k \leqslant K$

  `/* Step 2: update centroids */`

  for *k=1* **to** K

    if k=w **then**

      $m_k = m_k + 1$

      $c_k = c_k + \alpha_w (x_i - c_k)$

    if k=r **then**

      $c_k = c_k - \alpha_r (x_i - c_k)$

  `/* adaptatives learning rates */`

  Update α_w and α_r

until *convergence*

return Centroids $(c_k)_{1\leqslant k\leqslant K}$

Algorithm 3: The Hierarchical Agglomerative Clustering (HAC).

Data: Dataset $(x_i)_{1\leqslant i\leqslant n}$

Initialization :

$C_i = \{x_i\}\ \forall\, 1 \leqslant i \leqslant n$ `/* clusters */`

$d_{ij} = \|x_i - x_j\|\ \forall\, 1 \leqslant i < j \leqslant n$ `/* distances */`

$n_c = n$

repeat

  `/* Step 1: Merge the closest clusters */`

  $(i^*, j^*) = \arg\min_{i<j}(d_{ij})$

  $C_{i^*} = C_{i^*} \cup C_{j^*}$

  $C_{j^*} = \emptyset$

  $n_c = n_c - 1$

  `/* Step 2: Update distances */`

  for $k = 1$ **to** n_c

    if $k < i$ **then** Calculate d_{ki^*}

      `/* using Ward distance */`

    if $k > i$ **then** Calculate d_{i^*k}

      `/* using Ward distance */`

    $d_{j^*k} = d_{kj^*} = +\infty$

until *convergence*

return clusters C_i, $i = 1, 2, \ldots, n_c$

6.3 The HAC algorithm

The hierarchical agglomerative clustering (HAC), detailed in algorithm 3, is also widely used in unsupervised classification. At the beginning, each datapoint of the dataset is contained in its own individual class, thus the algorithm is initialized with n classes. The number of classes is iteratively reduced to $n_c < n$. At each step, the two closest (most similar) classes are merged. More details can be found in [21]. Several dissimilarity measures exist in the literature, the best known is the minimum Ward variance distance (see Section 5.4).

The advantage of HAC, over K-means and RPCL, is that it is not necessary to define the number of classes. Actually, the number of classes commonly used corresponds to the iteration with the maximum distance jump. However, hierarchical classification takes more computation time than other algorithms. Indeed, because the mesh of the surface must be quite fine for sake of precision, this algorithm operates on a very large number of classes at the beginning. In addition, the hierarchical classification does not allow any possibility of going back, the merger of two classes being indeed irreversible.

7 Testing clustering methods for free-form surfaces partitioning

In this section, the previously presented algorithms are compared in context of partitioning free-form surfaces. The main comparison criterion is, of course, the total time machining the partition they provide leads to. This machining time is calculated using the machine-tool cinematic model published in [24], that is proven to be very accurate. Following analysis presented in Section 4, the best choice concerning the feature vector is (u, v, s, θ). Therefore, only this feature vector is taken into account in the following analysis. Also, all metrics and algorithms presented in Sections 5 and 6 respectively, are tested and analyzed.

7.1 Tests protocol

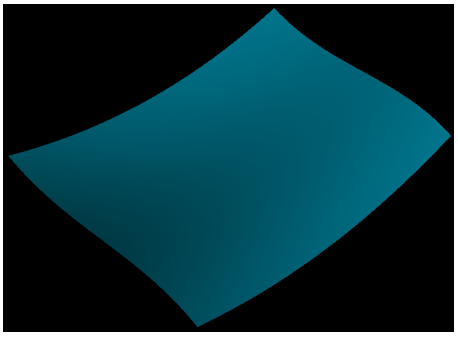
In order to carry out fair comparisons, clustering methods are carried out on two surfaces (Figure 4). The first one is used in [25], while the second one is used in [8]. For this reason they are thereafter called Choi surface and Rubio surface respectively. Both surfaces are discretized with a regular isoparametric meshing sizing 80×80.

Additionally, clusters number is fixed to $K = 3$ and number of iterations is limited to 300 for K-means and RPCL algorithms.

For each zone provided by clustering algorithms two machining directions are taken into account:

- the steepest-slope direction

Choi surface:



Rubio surface:

Figure 4: Tests surfaces.

- the principal direction, *i.e.* the direction that the zone is the most extended. This direction is calculated using principal component analysis (PCA) [14].

These both two directions have been taken into account because they are typical of machining processes. Indeed, it is well-known that the toroidal cutter provides better results than its ball-end counterpart when machining along steepest-slope direction. However, the study published in [14] highlights that, in some cases, machining along the principal direction using a ball-end cutter may lead to better results. This case may arise especially when the two directions are close to the perpendicular, and the zone is very extended along its principal direction in regard of the steepest-slope direction. However, to be sure no case is neglected these two typical directions have been included into this study. In case of steepest-slope direction, only the toroidal cutter has been tested, while in case of principal direction, both toroidal cutter and same outer radius cylindrical cutter have been tested, despite only the best result is presented in results below.

7.2 Results

Beside numerical results, the test process provides graphics for both clustering result and machining simulation (Figure 5). This is useful to check results are suitable for real world machining.

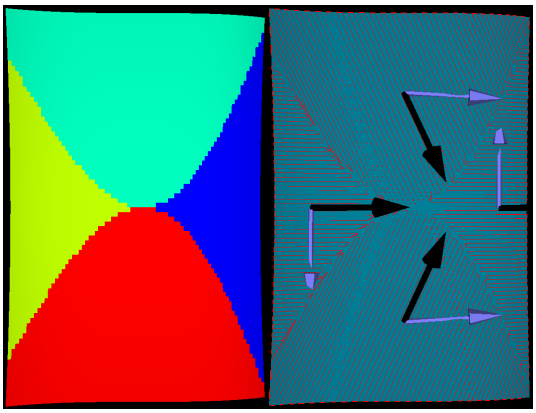


Figure 5: Surface partition and by-zone machining simulation example.

Numerical results are presented in Table 1 for the Choi surface, and in Table 2 for the Rubio surface. For each test case, are considered:

- the clustering algorithm
- the metric
- the machining direction

For each test case, results are:

- the machining simulation duration
- the total toolpath length

Furthermore, despite initial number of clusters is set to 3, the full process applied to the Choi surface results in a 4-zones partition, due to the connected components search presented in Section 8. This post-processing operation has no influence on clustering process itself. Its sole purpose is to ensure clustering result is composed of connected zones only.

Numerical results in Table 1 and 2 show that the HAC algorithm never provide the solution the fastest to machine. Besides, for K-means and RPCL algorithms results are often close. Indeed, the average difference between results is less than 8%. Concerning metrics, results are also quite close. Euclidean distance provides the best result for 3 test cases, Mahalanobis distance provides the best result for 2 test cases, and standard deviation-based distance provides the best result for 3 test cases.

Given the wide variety of free-form surfaces that can be encountered in real-world applications, it is difficult to conclude as to which algorithm and metric might give the best results in any case.

7.3 Discussion

As said above, the numerical results are not completely conclusive, except the fact that the HAC algorithm is not recommended creating a surface partition intended for machining by zone. At least to get the final partitioning. Indeed, the HAC algorithm has a big advantage over the other algorithms: it does not need a preliminary

Table 1: Tests results for the Choi surface.

direction	metric	*K-means*		RPCL		HAC (Ward distance)	
		duration	length	duration	length	duration	length
steepest-slope	Euclidean	104.7	4763	98.7	4861	117.1	4454
	Mahalanobis	125.2	6179	105.9	5777		
	standard deviation	100.1	4800	101.2	4780		
principal	Euclidean	107.8	6230	102.2	6333	136.8	6302
	Mahalanobis	108.0	6593	104.1	6652		
	standard deviation	111.1	6426	110.7	6447		

durations in seconds; lengths in millimeters

Table 2: Tests results for the Rubio surface.

direction	metric	*K-means*		RPCL		HAC (Ward distance)	
		duration	length	duration	length	duration	length
steepest-slope	Euclidean	91.6	4634	97.0	4726	96.2	4630
	Mahalanobis	88.7	4935	99.8	6277		
	standard deviation	91.5	4612	94.3	4966		
principal	Euclidean	84.3	5881	90.7	5940	85.3	5850
	Mahalanobis	89.0	5519	85.7	5874		
	standard deviation	83.0	5808	90.1	6009		

durations in seconds; lengths in millimeters

guess of the number of clusters. Therefore, it can be used to determine the number of clusters to use as an input parameter for K-means or RPCL algorithms. From this point of view, the RPCL algorithm also has an advantage, because it can "push away" some supernumerary centroids, thereby reducing the number of clusters.

The main drawback of the RPCL algorithm is the part of randomness it includes. Indeed, at each iteration, the point with respect to which the whole procedure is carried out is chosen at random. Therefore, for the same surface, different runs of the same algorithm can lead to different results. Indeed, according to the tests which have been carried out, this does not happen very often; but nevertheless, it does happen, and this is a serious drawback in the context of an industrial process. In addition, the RPCL algorithm requires the input of two parameters (the learning ratios) which can be difficult to define efficiently. Indeed, the relationship between these parameters and the quality of the result cannot be anticipated. Therefore, only multiple trials can determine the best values. On the other hand, the RPCL algorithm has the advantage of being generally faster than other algorithms.

Regarding the K-means algorithm, the number of clusters must be entered beforehand, which can be a disadvantage. But several parallel executions of the algorithm, each with a different initial number of clusters is a good workaround for this problem. The tests which have been carried out in this direction show that the additional cost in terms of computation time is negligible.

Another matter of interest is the various metrics commonly used in clustering processes that have been tested. On this side, no solution emerges clearly. Therefore, since the Euclidean distance is the easiest to calculate, it may be the best choice at first sight. Other metrics may provide better results, but this is very difficult to anticipate.

To summarize, even if other algorithms and other metrics can give good results, in the context of industrial use, it appears that the K-means algorithm used with a Euclidean metric seems to be the safest choice and the most versatile. However, given that the clustering algorithms applied to the partitioning of free-form surfaces are generally very fast in comparison with the rest of the procedure, it may be beneficial, for some special cases, to test different algorithms and different metrics.

8 Problem specifically related to the use of clustering algorithms for partitioning free-form surfaces

An issue that may appear when applying clustering algorithm to free-form surface partitioning is non-connected zones. A special algorithm, dedicated to deal with this problem is presented here.

Actually, the clustering algorithms do not guarantee partitioning with connected clusters only. Indeed, if the feature vector contains only proximity parameters such as: u, v or $\mathbf{S}(u, v)$, the resulting clusters would rather be connected. However, using a feature vector containing other parameters, such as machining-related parameters, issues may arise. Indeed, since two geometrically far points can have similar machining-related parameters (slope s and steepest-slope direction θ), a classification using a feature vector containing such parameters can lead to disconnected clusters (Figure 6). Of course, a disconnected zone cannot be machined at once and must be split into as many as needed connected zones.

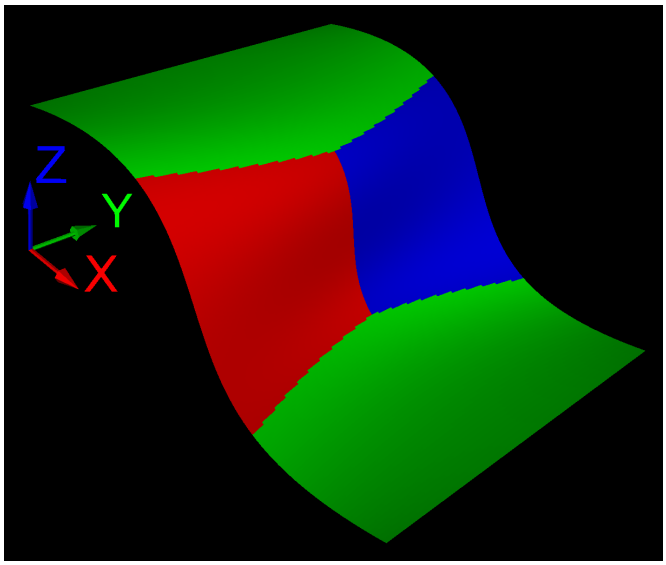


Figure 6: Example of disconnected zone (in green).

It is thus necessary to check, before machining, if all zones are connected and be able to identify, in disconnected zones, the connected components and consider them as independent zones.

This can be achieved using algorithms coming from graph theory. Actually, this problem could be considered as a well-known Connected Components Search (CCS) [26] in a non-oriented graph. However, the computation time required to identify connected components might be not negligible when a CCS algorithm is used, especially if the meshing is fine, which corresponds to numerous vertices in the graph. In this case, CCS algorithm may take a lot of time.

A faster algorithm, inspired by Connected Components Labeling (CCL) [27] algorithms has been developed. Also called blob extraction or region labeling, the CCL algorithms are originally image processing and analysis techniques, aiming at grouping the pixels of an image into components using a given heuristic.

The developed algorithm works on the raw result of the clustering process, *i.e.* a two-dimensional array zID, which sizes $tessU$ and $tessV$ are the numbers of elementary mesh unit in each parametric dimension. Inside this array, are stored the unique zone identifiers of each elementary mesh unit. At this stage, unconnected zones have not been detected yet, thus unconnected regions of the same zone have the same number. To store the result of the procedure, a same size two-dimensional array, called *labels*, is also created. The principle of the developed algorithm is described in Algorithm 4.

Algorithm 4: CCL-like algorithm for connected components analysis (principle).

```
Data: array zID, array labels, tessU, tessV
Function label(i, j, l):
    labels[i][j] = l
    if zID[i - 1][j] = zID[i][j] and labels[i - 1][j] =
     null
        label (i − 1, j, l)
    if zID[i][j - 1] = zID[i][j] and labels[i][j - 1] =
     null
        label (i, j − 1, l)
    if zID[i + 1][j] = zID[i][j] and labels[i + 1][j]
     = null
        label (i + 1, j, l)
    if zID[i][j + 1] = zID[i][j] and labels[i][j + 1]
     = null
        label (i, j + 1, l)
Function main:
    l = 0
    label (0, 0, l)
    for i = 0 to tessU - 1
        for j = 0 to tessV - 1
            if labels[i][j] = null
                l = l + 1
                label (i, j, l)
```

The result of the application of CCL-like algorithm on the example of Figure 6 is presented in Figure 7. Based on a recursive function, it is very fast. For example, applied to the Choi surface discretized with an 80×80 mesh, it took less than 20 milliseconds to achieve the full processing of the connected components.

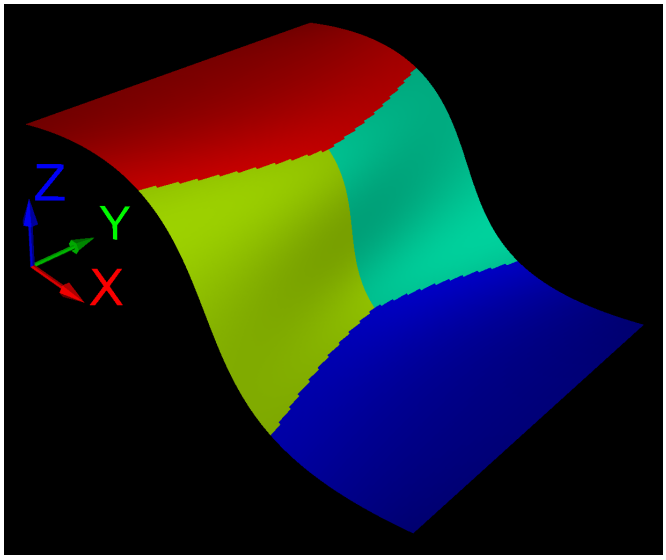


Figure 7: Result of the connected component search.

9 Conclusion

In this paper, an analysis of various clustering algorithms applied to free-form surface partitioning for by-zones machining is presented. In this particular context, both feature vectors, similarity measurement methods and algorithms themselves are studied and their assets and drawbacks are highlighted. In order to help the reader to make a choice, a bunch of tests are carried out on two surfaces found in the literature. Furthermore, an algorithm dealing with the non-connected zones problem that may arise in this context is provided.

Given the wide variety of free-form surfaces that can be encountered in industrial applications, a conclusive choice is difficult to make, but using the K-means algorithm with a Euclidean metric and a dedicated vector of features can be a good first choice. Based on this conclusion, in further work, other algorithms, such as k-medoids or fuzzy C-means, may be compared to K-means.

References

[1] Bedi S, Ismail F, Mahjoob M, Chen Y (1997) Toroidal versus ball nose and flat bottom end mills. The International Journal of Advanced Manufacturing Technology 13:326–332

[2] Kumazawa GH, Feng HY, Barakchi Fard MJ (2015) Preferred feed direction field: A new tool path generation method for efficient sculptured surface machining. Computer-Aided Design 67-68:1–12

[3] Liu X, Li Y, Ma S, Lee Ch (2015) A tool path generation method for freeform surface machining by introducing the tensor property of machining strip width. Computer-Aided Design 66:1–13

[4] Djebali S, Segonds S, Redonnet JM, Rubio W (2015) Using the global optimisation methods to minimise the machining path length of the free-form surfaces in three-axis milling. International Journal of Production Research 53(17):5296–5309

[5] Altınel ÍK, Öncan T (2005) A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. Journal of the Operational Research Society 56(8):954–961

[6] Djebali S, Perles A, Lemouzy S, Segonds S, Rubio W, Redonnet JM (2015) Milling plan optimization with an emergent problem solving approach. Computers & Industrial Engineering 87:506–517

[7] Bernon C, Gleizes MP, Peyruqueou S, Picard G (2003) ADELFE: A methodology for adaptive multi-agent systems engineering. In: Petta P, Tolksdorf R, Zambonelli F (eds) Engineering Societies in the Agents World III, Springer, pp 156–169

[8] Duc V, Monies F, Segonds S, Rubio W (2020) Automatic minimal partitioning method guaranteeing machining efficiency of free-form surfaces using a toroidal tool. The International Journal of Advanced Manufacturing Technology 107:4239–4254

[9] Chen ZC, Dong Z, Vickers GW (2003) Automated surface subdivision and tool path generation for $3\frac{1}{2}\frac{1}{2}$-axis CNC machining of sculptured parts. Computers in Industry 50(3):319–331

[10] Pal NR, Chakraborty D (2000) Mountain and subtractive clustering method: Improvements and generalizations. International Journal of Intelligent Systems 15(4):329–341

[11] Roman A, Bedi S, Ismail F (2006) Three-half and half-axis patch-by-patch NC machining of sculptured surfaces. The International Journal of Advanced Manufacturing Technology 29(5):524–531

[12] Liu X, Li Y, Li Q (2018) A region-based 3+2-axis machining toolpath generation method for freeform surface. The International Journal of Advanced Manufacturing Technology 97(1-4):1149–1163

[13] Makhanov S (2007) Optimization and correction of the tool path of the five-axis milling machine: Part 1. Spatial optimization. Mathematics and Computers in Simulation 75(5):210–230

[14] Herraz M, Redonnet JM, Mongeau M, Sbihi M (2020) A new method for choosing between ball-end cutter and toroidal cutter when machining free-form surfaces. The International Journal of Advanced Manufacturing Technology 111(5):1425–1443

[15] Duda RO, Hart PE, Stork DG (2012) Pattern Classification. John Wiley & Sons

[16] Kotsiantis SB, Zaharakis ID, Pintelas PE (2006) Machine learning: A review of classification and combining techniques. Artificial Intelligence Review 26(3):159–190

[17] Xu R, Wunsch D (2005) Survey of clustering algorithms. IEEE Transactions on Neural Networks 16(3):645–678

[18] Redonnet JM, Djebali S, Segonds S, Senatore J, Rubio W (2013) Study of the effective cutter radius for end milling of free-form surfaces using a torus milling cutter. Computer-Aided Design 45(6):951–962

[19] McLachlan GJ (1999) Mahalanobis distance. Resonance 4(6):20–26

[20] Xu L, Krzyzak A, Oja E (1993) Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. IEEE Transactions on Neural Networks 4(4):636–649

[21] Szekely GJ, Rizzo ML (2005) Hierarchical Clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method. Journal of Classification 22(2):151–183

[22] MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, University of California Press, Berkeley, CA, pp 281–297

[23] Zhao W, Ma H, He Q (2009) Parallel K-means clustering based on MapReduce. In: Jaatun MG, Zhao G, Rong C (eds) Cloud Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 674–679

[24] Pessoles X, Landon Y, Rubio W (2010) Kinematic modelling of a 3-axis NC machine tool in linear and circular interpolation. International Journal of Advanced Manufacturing Technology 47(5-8):639–655

[25] Choi YK (2007) Tool path generation and 3D tolerance analysis for free-form surfaces. International Journal of Machine Tools and Manufacture 47

[26] Skiena SS (1998) The Algorithm Design Manual. Springer-Verlag, Berlin, Heidelberg

[27] He L, Ren X, Gao Q, Zhao X, Yao B, Chao Y (2017) The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition 70:25–43