Segmentation of shot peening patterns using k-means clustering

Vladislav Sushitskii[a,b], Hong Yan Miao[a,b], Martin Lévesque[a], Frédérick P. Gosselin[a,b]

[a] *Laboratory for Multiscale Mechanics (LM2), Department of Mechanical Engineering, Polytechnique Montreal, 2500 Chemin de Polytechnique, Montreal, H3T1J4, Quebec, Canada*

[b] *Aluminium Research Centre – REGAL, 1065 Avenue de la Medecine, Quebec, G1V0A6, Quebec, Canada*

Abstract

Shot peen forming is an industrial process for shaping thin and large metal plates. The process consists in treating a plate with a stream of rigid shot that are projected through a moving nozzle according to a peening pattern. The existing methods for the peening pattern computation either prescribe a continuously varying peening intensity, which is not reproducible in practice, or require having pre-defined intensities. Here, we present a segmentation strategy that enforces practical constraints to any peening pattern by dividing it into uniformly treated zones. In addition, the strategy automatically identifies the best treatment parameters for each segment. The strategy consists of the clustering algorithm, which splits the pattern into segments, and of the noise filtering algorithm, which removes too small segments from the pattern. The clustering algorithm is inspired by the k-means method, and the filtering algorithm is based on cellular automata. Both algorithms were tested numerically using 200 randomly generated test cases.

Keywords: Segmentation, Clustering, Filtering, k-means, Optimization, Modeling, Peen forming

Email addresses: vladislav.sushitskii@polymtl.ca (Vladislav Sushitskii), hong-yan.miao@polymtl.ca (Hong Yan Miao), martin.levesque@polymtl.ca (Martin Lévesque), frederick.gosselin@polymtl.ca (Frédérick P. Gosselin)

1. Introduction

Fabricating aircraft and rockets requires forming large metal plates with high precision. These plates constitute, for example, wing skins, fuselage shells or fuel tank segments. The outlined components often exhibit a nonuniform double curvature, meaning that they are locally curved in two directions and that the local radii of curvature vary over the plate. A cost-effective shaping method that delivers such forms is shot peen forming. This treatment consists in projecting a stream of rigid shot towards the surface of the plate. The shot stream plastically deforms the surface and induces local bending of the component [1, 2]. The stream width is smaller than the plate size, and the treatment can be applied to both sides of the plate with a variable intensity. This means that the developed curvatures can be locally controlled by altering the peened segments and the peening parameters.

Numerical simulation of shot peen forming involves the solution of the *forward* and of the *inverse* problems [3]. The forward problem solution means computing the deformation resulting from a predefined peening treatment. The inverse problem solution means defining a necessary treatment to achieve a predefined target shape. A numerical inverse problem solver is necessary for industrial companies to avoid a long and costly trial-and-error design phase.

The solution to the inverse problem in peen forming is a map of local peening parameters (regimes) mapped over the initial geometry of the treated plate, which is called the peening pattern. For instance, Figure 1 shows two peening patterns that allow to shape a flat square plate into a spherical patch. The first steps towards numerical resolution of the inverse problem were made by VanLuchene et al. [4, 5], who simulated the effect of peening using the peening-induced forces. Later, two inverse problem solvers operating directly with the peening parameters were presented [6, 7]. In all these cases, the peening parameters were experimentally related to the peening-induced forces [5] or curvatures [6, 7]. The inverse problem can also be solved through formulating the peening pattern in terms of the peening-induced residual stresses, which are related to the peening parameters through numerical impact simulations [8, 9].

The approaches that rely on the peening-induced forces [4, 5] and stresses [8, 9] as loads lose their accuracy in case of large peening-induced rotations. Thus, the large rotations provoke geometrically nonlinear behavior of the

plate, which breaks the experimentally developed relations between these loads and the peening parameters. On the other hand, a direct empirical relation between the peening parameters and the induced curvatures [6, 7] does not hold for complex peening patterns, because the local curvatures are influenced by the curvatures in the neighboring zones.

Another strategy for the peen forming simulation involves formulating the peening pattern in terms of local eigenstrains [10, 11]. The eigenstrain represents the plastic strain induced by the treatment, which triggers deformation when introduced in the component. The advantage of eigenstrains is that they are independent of the plate geometry, unlike the peening-induced forces, stresses or curvatures [12, 13]. Moreover, the eigenstrains can be assumed constant during reconfiguration of the component, which was proven by means of numerical simulations and experiments with 76×19 mm shot peened metal coupons [14, 15]. With the eigenstrain approach, the inverse problem can be solved using a numerical optimization algorithm [16], an artificial neural network [17] or the theory of non-Euclidean plates [3]. In turn, a one-to-one correspondence between the peening regimes and the induced eigenstrain can be established using experiments [12], small-scale numerical simulations [18], or a combination of both [19]. It is, however, imperative to hold the plate flat during peening and to ascertain the absence of stresses and strains in the initial state of the plate, because the established correspondence can be altered otherwise [16].

The eigenstrain approach also allows to simulate other forming processes that induce plastic strain in metal components, such as laser peen forming [20] or English wheel [21]. Moreover, eigenstrains allow to represent a peen formed plate as a laminate subjected to incompatible expansion or contraction of its layers [22], so this approach bridges the gap between peen forming and fabrication of laminated shape-shifting structures. For example, the inverse problem solver presented in Ref. [16] works both for peen forming and for patterned multilayer films [23], and the solver examined in Ref. [3] uses the results developed in the field of 4D printed elastic sheets [24, 25].

The numerical inverse problem resolution involves the discretization of the initial geometry with shell finite elements. The computed peening pattern is therefore a discrete map. In the general case, the eigenstrains prescribed to each element vary from one element to another, as it is illustrated in Figure 1 b). On the other hand, the practical conditions of physically applying the peening treatment limit this spatial variation. Thus, the maximal eigenstrain

that can be prescribed to a segment is limited by the equipment capacity. In addition, the peening parameters cannot be varied gradually over the plate. Moreover, each modification of peening parameters during treatment slows down the shaping process, which is especially the case for changing the media [26]. These reasons suggest that the pattern must consist of segments formed by multiple elements that are prescribed with the same eigenstrain. For a simpler and easier process, the segmented peening pattern must involve the fewest possible peening regimes while preserving the precision of the final shape.

Figure 1 c) presents an example of a segmented pattern. In practice, such segmented patterns are usually applied using masks [27, 28]. If the peening equipment admits only uniform treatment of the component during one cycle, then a separate mask is produced for each peening regime involved.

In particular, the problem of segmentation of the eigenstrain pattern is addressed in Refs. [16, 17]. The presented inverse problem solvers operate with one available eigenstrain magnitude, so the peening pattern is a set of treated and untreated segments. This effect is achieved in Ref. [16] by limiting the overall treated area. This condition forces the optimization algorithm to assign the maximal eigenstrain magnitude to the treated elements and thus

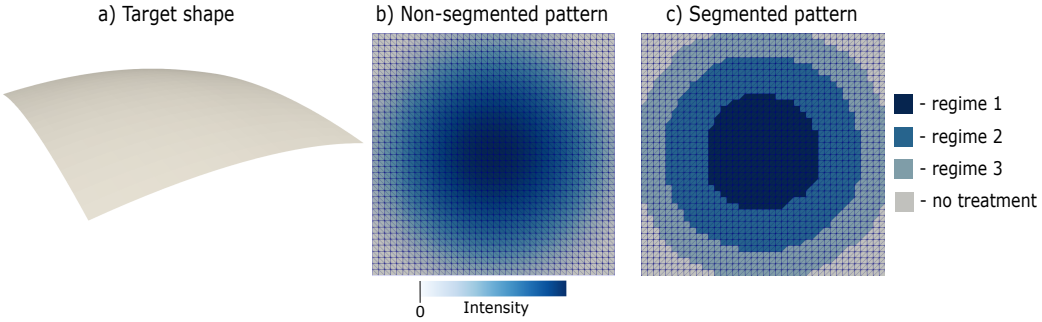

Figure 1. Examples of a non-segmented and of a segmented peening pattern. Both patterns make a flat square plate deform into a spherical patch (a) when applied from both sides of the plate. The non-segmented pattern (b) prescribes a gradual variation of the peening intensity over the plate, which is hardly reproducible with peening equipment. The segmented pattern (c) is reproducible because it consists of uniformly treated segments. Both patterns are discretized with triangular finite elements, and the effect of their application can be simulated using the eigenstrain approach. For this, one must establish in advance a one-to-one correspondence between the peening regimes or the intensities, and the induced eigenstrains.

penalizes the intermediate eigenstrain values. On the other hand, the neural network presented in Ref. [17] naturally operates with one fixed eigenstrain magnitude, which is set during the training phase. Thus, the neural network computes a treatment probability for each element, and the elements with high probabilities are then assigned with the fixed eigenstrain magnitude. Although the examined solvers provide ready-to-use patterns, the use of only one eigenstrain magnitude limits the range of achievable target shapes.

The inverse problem solver presented in Ref. [3] does not restrict the variation of the eigenstrain assigned to each element. The division of the pattern into segments is done during post-processing with the help of grouping. This approach allows any number of regimes in the pattern, which means that it is flexible in terms of target shapes. However, the original and the grouped patterns induce different final shapes, so the peening regimes have to be thoroughly adjusted to minimize this difference.

The strategies for segmentation of the peening pattern presented in Refs. [3, 16, 17] share a common drawback: they require having predefined peening regimes. This suggests that if the current peening regimes do not allow to achieve the target shape according to the simulations, then the best suitable regimes must be found by trial-and-error. Thus, if the segmentation is embedded in the inverse problem solver [16, 17], then each new trial means restarting the inverse problem solver. Otherwise, if the segmentation is executed after the inverse problem resolution by means of grouping (see Ref. [3]), then the grouping algorithm must be relaunched with a new combination of peening regimes, which are defined manually. Hence, the necessity for having pre-defined peening regimes elongates the inverse problem solution and makes it less accurate, because the trial-and-error solution is not guaranteed to be the optimal.

Conversely, the problem of segmentation of a peening pattern is similar to the problem of clustering [29, 30]. Clustering algorithms split a set of points into groups based on the point coordinates. In terms of the peening pattern, this means splitting the mesh into segments based on the eigenstrain values assigned to each finite element. Moreover, clustering algorithms are able to find a centroid for each cluster. With regard to the peening pattern, this allows to find the mean eigenstrain in the given segment and to homogenize the eigenstrain in this segment. Each eigenstrain value corresponds to a peening regime, so a clustering algorithm is able to prescribe the best suitable peening regime for each segment. This technique allowed, for example, to

compute the pattern shown in Figure 1 (c) starting from the non-segmented pattern presented in Figure 1 (b).

An optimal clustering algorithm for the peen forming case must minimize the difference between the original and the segmented patterns in order to reduce the difference in the developed shapes. This can be accomplished using the `k`-means clustering algorithm [31], which groups points in space based on their proximity in terms of squared Euclidean distance. This algorithm is also advantageous in terms of the computation speed and adaptability to sparse point sets [29, 30].

We present in this paper a numerical strategy for segmenting the peening pattern without predefined peening regimes, which is designed as a post-processing tool for the inverse problem solver. The segmentation strategy relies on a modified `k`-means clustering algorithm, which is adapted for the peen forming case. The clustering algorithm is followed by a noise filtering algorithm that corrects local clustering errors and thus makes the pattern fully applicable.

2. Theoretical background

2.1. The bilayer eigenstrain formulation of the peening pattern

The eigenstrain $\boldsymbol{\varepsilon}$ prescribed to each finite element by an eigenstrain-based inverse problem solver is constant in the in-plane direction inside the element, but it varies along the through-thickness coordinate z. We assume that the treated material is isotropic and not pre-stressed, so that $\varepsilon_{xx}(z) = \varepsilon_{yy}(z) = \gamma(z)$ and $\varepsilon_{zz}(z) = -2\gamma(z)$, where x and y are the in-plane lagrangian coordinates. The peening-induced isotropic eigenstrain profile $\gamma(z)$ is a continuous function, but it can be idealized, i.e., discretized, in the way that the final shape induced by the idealized profile is the same as that induced by the original profile (when assuming plate theory) [3]. In the general case of treatment from both sides, the simplest idealized profile involving the fewest variables is the bilayer profile γ_{bi}. This profile assumes that the plate consists of two equally thick layers assigned with the eigenstrains ε^t and ε^b as:

$$\gamma_{bi}(z) = \begin{cases} \varepsilon^t & \text{for} \quad 0 < z < \frac{h}{2}, \\ \varepsilon^b & \text{for} \quad -\frac{h}{2} < z < 0, \end{cases} \tag{1}$$

where h is the total plate thickness and z is the through-thickness coordinate having its origin at the mid-surface level. Figure 2 illustrates the idealization of the through-thickness eigenstrain profile.

The bilayer eigenstrain profile is locally described with two variables $(\varepsilon^t, \varepsilon^b)$, and each of the two peening regimes applied from the top and bottom sides influences both ε^t and ε^b. Thus, the treatment from the top side induces positive ε^t and negative ε^b, while the treatment from the bottom side induces negative ε^t and positive ε^b. In case of simultaneous treatments from both sides, the values of ε^t and ε^b are superpositions of contributions made by the top and the bottom side peening regimes.

Under the assumptions of plate theory, the final shapes induced by the profiles $\gamma(z)$ and γ_{bi} coincide if the local forces and moments induced by the two profiles are equal, meaning that the conditions

$$\int_{-h/2}^{h/2} \gamma(z) dz = \frac{h}{2}(\varepsilon^t + \varepsilon^b), \tag{2}$$

$$\int_{-h/2}^{h/2} \gamma(z) z dz = \frac{h^2}{8}(\varepsilon^t - \varepsilon^b) \tag{3}$$

are met. Therefore, Eqns. (2, 3) allow to determine ε^t and ε^b element-wise and thus to reduce any through-thickness eigenstrain profile $\gamma(z)$ to the bilayer formulation.

2.2. The general approach to `k`*-means clustering of the eigenstrain pattern*

In terms of clustering, the variables $(\varepsilon^t, \varepsilon^b)$ act as coordinates of points on a plane, where each point corresponds to the peening state of a finite element in the peening pattern. Typical of a `k`-means method, our clustering algorithm uses a predefined number of centroids, or in other words, a predefined number of peening regimes. When the points are clustered, the algorithm assigns each finite element with the eigenstrains corresponding to the centroid of the cluster containing this element. This strategy allows, in particular, to constrain the maximal eigenstrain in the pattern by imposing limits on the coordinates of the centroids. The cluster centroids are related to the peening regimes, so the clustering algorithm outputs top- and bottom-side peening patterns formulated in terms of peening regimes.

The `k`-means clustering is an iterative process involving the relocation of

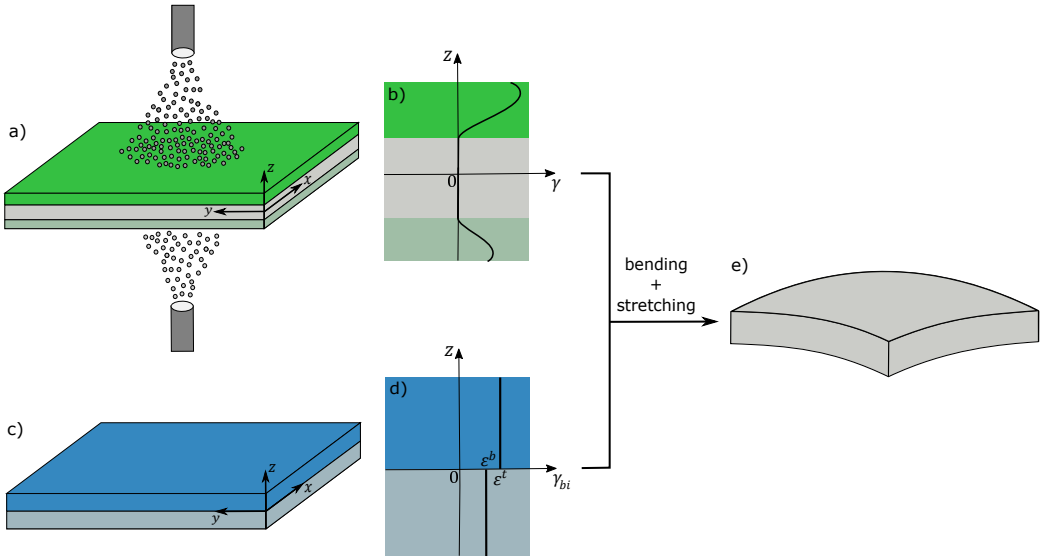

Figure 2. Idealization of the through-thickness eigenstrain profile. In this example, we suppose that the plate is uniformly peened from both sides and that the top side peening regime is more intense. a) The treatment induces eigenstrains in the outer layers of the plate (bright green and pale green), while the internal layer (gray) is not affected. b) The induced eigenstrain prescribes an in-plane expansion to the outer layers, and the expansion magnitude γ is non-uniform along the through-thickness lagrangian coordinate z. c) To idealize the eigenstrain profile, we represent the plate as a bilayer consisting of equally thick layers (bright blue and pale blue). d) We prescribe a uniform eigenstrain to both layers and thus obtain an idealized through-thickness eigenstrain profile γ_{bi}, which is described with two variables: $(\varepsilon^t, \varepsilon^b)$. The values of ε^t and ε^b are computed in the way that γ_{bi} induces the same final shape as $\gamma(z)$ (see Eqns. 2, 3), in the sense of plate theory. e) The final shape induced by γ_{bi} and $\gamma(z)$ is stretched with respect to the initial state and convex towards the side where a more intense treatment was applied.

the centroids on each iteration. The result of clustering is dependent on the initial guess of the centroid positions, which are assigned randomly. For this reason, the algorithm must be relaunched several times with different initial centroid positions. Each trial, in turn, comprises several iterations. At the end of each trial, the algorithm evaluates the quality of clustering by computing the sum of squared Euclidean distances between the points and the centroids of clusters to which the points are assigned. The trial providing the lowest sum of distances is chosen as the best clustering. The number of trials is chosen the balance between the computational cost and the desired accuracy. For example, performing 100 trials of the `k`-means algorithm allows

to decrease the clustering error by 60% with respect to a single trial in average [32]. However, the number of all possible partitions of a given set of points is finite, so, if the algorithm is guaranteed to try all partitions in a certain number of trials, then further increasing of this number does not bring any positive effect.

3. Methodology

3.1. The clustering algorithm

We characterize a peening regime with a couple $\left(\varepsilon^{(1)}, \varepsilon^{(2)}\right)$, where, by convention, $\varepsilon^{(1)}$ is positive and $\varepsilon^{(2)}$ is negative. In other words, we describe the forming effect of a treatment using the bilayer model, where the treated layer expands by $\varepsilon^{(1)}$ and the other layer contracts by $\varepsilon^{(2)}$. Hence, if a plate is treated from the top side with regime i inducing $\left(\varepsilon_i^{(1)}, \varepsilon_i^{(2)}\right)$ and from the bottom side with regime j inducing $\left(\varepsilon_j^{(1)}, \varepsilon_j^{(2)}\right)$, then their combination results in $\left(\varepsilon_i^{(1)} + \varepsilon_j^{(2)}, \varepsilon_i^{(2)} + \varepsilon_j^{(1)}\right)$. An unclustered pattern consists of such eigenstrain *combinations* assigned to each element, but the aim of the clustering algorithm is to find the best suitable *regimes*. Consequently, the clustering algorithm must split the combinations into contributions made by the top and bottom peening regimes independently.

The clustering algorithm starts by initializing the points that are to be clustered on the coordinate plane $(\varepsilon^t, \varepsilon^b)$. Each finite element e generates one point with coordinates $(\varepsilon_e^t, \varepsilon_e^b)$, where $(\varepsilon_e^t, \varepsilon_e^b)$ is the eigenstrain prescribed to this element by the inverse problem solver. Therefore, the elements and the points are directly associated. Subsequently, the clustering algorithm initializes the *void* regime $(0, 0)$ corresponding to the absence of treatment, and it also randomly initializes N peening regimes in terms of $\left(\varepsilon^{(1)}, \varepsilon^{(2)}\right)$ under the following constraints:

$$\begin{cases} 0 < \varepsilon^{(1)} < \varepsilon_{max}^{(1)}\,, \\ \varepsilon_{min}^{(2)} < \varepsilon^{(2)} < 0\,, \end{cases} \tag{4}$$

where $\varepsilon_{max}^{(1)}$ and $\varepsilon_{min}^{(2)}$ are, correspondingly, the maximal positive and the minimal negative eigenstrains achievable with the peening equipment. These values must be experimentally characterized in advance.

Afterwards, the clustering algorithm initializes the cluster centroids, that

Centroid	Peening regime (top)	Peening regime (bottom)
00	0	0
01	0	1
02	0	2
⋮	⋮	⋮
ij	i	j
⋮	⋮	⋮
NN	N	N

Table 1
Formation of the cluster centroids. Each centroid represents a combination of peening regimes applied from the top and bottom sides. N available peening regimes form $(N+1)^2$ centroids because the absence of treatment is considered as an additional void regime numbered as 0.

are computed as all possible combinations of the peening regimes. Hence, there are $\mathtt{k} = (N+1)^2$ centroids initialized, and each of them represents a cluster. A centroid formed by regime i applied from the top side and by regime j applied from the bottom side is marked as ij (Table 1). The void regime is numbered as 0, so the centroids marked as $i0$ and $0i$ correspond to treatment with regime i only from the top side and only from the bottom side, respectively. Figure 3 a) shows the points and the centroids initialized on the plane.

The next stage is the deletion of all centroids that lie above the line $\varepsilon^t = \varepsilon^b$. Indeed, the centroids ij and ji are induced by the same peening regimes applied conversely, so they are symmetrical about the line $\varepsilon^t = \varepsilon^b$. However, the algorithm aims at optimizing the peening regimes, and the range of regimes is the same for both sides of the plate. Consequently, the algorithm deletes one centroid from each pair of symmetrical centroids.

The symmetry principle also applies to the points corresponding to the elements. Thus, the eigenstrains $(\varepsilon_e^t, \varepsilon_e^b)$ and $(\varepsilon_e^b, \varepsilon_e^t)$ represent the same treatment combination applied from different sides. However, the points cannot be deleted like the centroids because all points contain information on the given pattern. Accordingly, the clustering algorithm reflects all points that lie above the line $\varepsilon^t = \varepsilon^b$ across this line instead of deletion. With this, the points lying on either side of the line $\varepsilon^t = \varepsilon^b$ are fused and clustered

simultaneously. Figure 3 b) traces the reflected points and the preserved centroids.

The iterative process begins next. Similarly as during the standard `k`-means clustering, each point, i.e., element, is assigned to the cluster represented by the closest centroid, which is shown in Figure 3 c). However, the centroids cannot be relocated freely for every iteration because they are coupled. Indeed, the relocation of centroid ij means altering the eigenstrains induced by both regimes i and j. This, in turn, induces a simultaneous relocation of all centroids formed by regimes i and j in combination with another regime. For this reason, the clustering algorithm does not relocate centroids but adjusts the peening regimes directly. The new centroid positions are then computed as a function of the new peening regimes.

The peening regimes are adjusted using the notion of *contributions*. Let us consider a point $(\varepsilon_e^t, \varepsilon_e^b)$ generated by element e, which is assigned to cluster ij on the current iteration. The centroid of this cluster represents the combination of treatments with regimes i and j. In turn, point e also represents a combination of treatments, but the contributions made by the top and bottom peening are not defined. However, the clustering algorithm derives these contributions, which have coordinates $\left(\varepsilon_{e,i}^{(1)}, \varepsilon_{e,i}^{(2)}\right)$ and $\left(\varepsilon_{e,j}^{(1)}, \varepsilon_{e,j}^{(2)}\right)$, and uses them to adjust regimes i and j, respectively. The contributions are derived in the way that they are as close as possible to the eigenstrains induced by regimes i and j.

We formulate the expression for the coordinates of the contributions using the method of Lagrange multipliers [33]. These coordinates must minimize the sum of squared Euclidean distances f to the regimes i and j:

$$f = \left(\varepsilon_{e,i}^{(1)} - \varepsilon_i^{(1)}\right)^2 + \left(\varepsilon_{e,i}^{(2)} - \varepsilon_i^{(2)}\right)^2 + \left(\varepsilon_{e,j}^{(1)} - \varepsilon_j^{(1)}\right)^2 + \left(\varepsilon_{e,j}^{(2)} - \varepsilon_j^{(2)}\right)^2. \tag{5}$$

In addition, the combination of contributions $\left(\varepsilon_{e,i}^{(1)}, \varepsilon_{e,i}^{(2)}\right)$ and $\left(\varepsilon_{e,j}^{(1)}, \varepsilon_{e,j}^{(2)}\right)$ must equal $(\varepsilon_e^t, \varepsilon_e^b)$, which leads to:

$$\begin{cases} \varepsilon_{e,i}^{(1)} + \varepsilon_{e,j}^{(2)} = \varepsilon_e^t, \\ \varepsilon_{e,i}^{(2)} + \varepsilon_{e,j}^{(1)} = \varepsilon_e^b. \end{cases} \tag{6}$$

With this, the Lagrangian function $\mathcal{L}$ takes the form:

$$\begin{aligned}\mathcal{L} = &\left(\varepsilon_{e,i}^{(1)} - \varepsilon_i^{(1)}\right)^2 + \left(\varepsilon_{e,i}^{(2)} - \varepsilon_i^{(2)}\right)^2 + \left(\varepsilon_{e,j}^{(1)} - \varepsilon_j^{(1)}\right)^2 + \left(\varepsilon_{e,j}^{(2)} - \varepsilon_j^{(2)}\right)^2 \\ &- \lambda_1 \left(\varepsilon_{e,i}^{(1)} + \varepsilon_{e,j}^{(2)} - \varepsilon_e^t\right) - \lambda_2 \left(\varepsilon_{e,i}^{(2)} + \varepsilon_{e,j}^{(1)} - \varepsilon_e^b\right).\end{aligned} \tag{7}$$

The stationary point of the Lagrangian function is:

$$\begin{cases} \varepsilon_{e,i}^{(1)} = 0.5\left(\varepsilon_i^{(1)} - \varepsilon_j^{(2)} + \varepsilon_e^t\right), \\ \varepsilon_{e,i}^{(2)} = 0.5\left(\varepsilon_i^{(2)} - \varepsilon_j^{(1)} + \varepsilon_e^b\right), \\ \varepsilon_{e,j}^{(1)} = 0.5\left(\varepsilon_j^{(1)} - \varepsilon_i^{(2)} + \varepsilon_e^b\right), \\ \varepsilon_{e,j}^{(2)} = 0.5\left(\varepsilon_j^{(2)} - \varepsilon_i^{(1)} + \varepsilon_e^t\right), \end{cases} \tag{8}$$

$$\begin{cases} \lambda_1 = \varepsilon_e^t - \varepsilon_i^{(1)} - \varepsilon_j^{(2)}, \\ \lambda_2 = \varepsilon_e^b - \varepsilon_j^{(1)} - \varepsilon_i^{(2)}. \end{cases} \tag{9}$$

The two contributions computed according to Eqns. (8) minimize the sum of squared Euclidean distances (5) and satisfy conditions (6), simultaneously.

Let $\mathcal{T}_i$ be the set of points assigned to all the centroids formed using regime i on the current iteration. The algorithm computes the new eigenstrain induced by regime i as the mean of all contributions $\left(\varepsilon_{e,i}^{(1)}, \varepsilon_{e,i}^{(2)}\right)$ for $e \in \mathcal{T}_i$. The void regime is, however, fixed at point $(0, 0)$. When all regimes are adjusted, the algorithm recalculates the centroids and reassigns the points to the clusters. Same as with the standard `k`-means algorithm, this iterative process is repeated until stabilization of the peening regimes and, consequently, of the centroids. The clustering algorithm is summarized as Algorithm 1. Figure 3 d) illustrates the result of clustering.

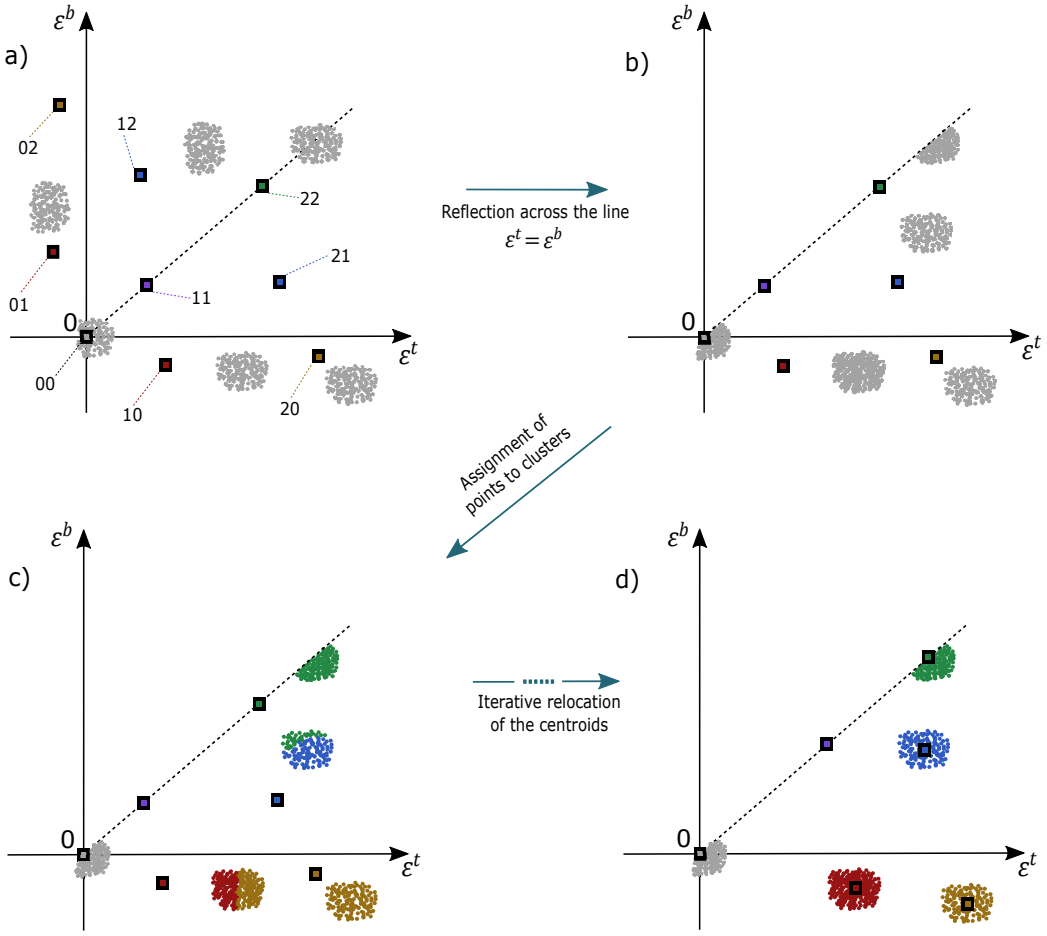

Figure 3. Example of clustering with $N = 2$ available peening regimes.
a) At the beginning, the clustering algorithm traces the points (gray dots) that are to be clustered at coordinates $(\varepsilon^t, \varepsilon^b)$. Each point represents a finite element, and its coordinates reflect the eigenstrain assigned to the element by the inverse problem solver. The clustering algorithm randomly initializes the peening regimes and traces the cluster centroids (colored squares), each of which is a combination of two peening regimes for all possible permutations. The coordinates of centroid ij represent the eigenstrain induced by the application of regime i from the top side and of regime j from the bottom side.
b) Next, the algorithm eliminates the symmetry by reflecting the points that lie above the line $\varepsilon^t = \varepsilon^b$ across this line. The centroids that lie above this line are deleted.
c) Subsequently, each point is assigned to the closest centroid (cluster) in terms of squared Euclidean distance.
d) The algorithm iteratively adjusts the peening regimes, derives new centroid positions and reassigns the points until the stabilization of the centroids. The centroid $(0, 0)$ is fixed at the origin during all iterations. In this example, no points were attributed to cluster 11 (purple), which corresponds to the treatment with regime 1 from both sides.

Algorithm 1 The clustering algorithm

for each finite element e **do**
    Initialize one point with coordinates $(\varepsilon_e^t, \varepsilon_e^b)$
end for
Initialize the void peening regime $(0, 0)$
Randomly initialize N peening regimes in terms of $\left(\varepsilon^{(1)}, \varepsilon^{(2)}\right)$ under constraints (4)
Compute the centroids in coordinates $(\varepsilon^t, \varepsilon^b)$ as all possible combinations of two peening regimes
for each centroid that lies above the line $\varepsilon^t = \varepsilon^b$ **do**
    Delete the centroid
end for
for each point that lies above the line $\varepsilon^t = \varepsilon^b$ **do**
    Swap its coordinates ε^t and ε^b
end for
while at least one centroid changes its position **do**
    for each point **do**
        Compute the squared Euclidean distance to each centroid
        Assign the point to the cluster formed by the closest centroid
        Compute the two contributions using Eqns. (8)
    end for
    for each regime except for the void regime **do**
        Compute the mean of all contributions belonging to this regime
        Assign the mean as the new peening regime
    end for
    Compute the centroids as the combinations of the adjusted peening regimes
end while
for each finite element e **do**
    Assign the element with the eigenstrain corresponding to the centroid of the cluster containing point $(\varepsilon_e^t, \varepsilon_e^b)$
end for

3.2. Filtering of the peening pattern using cellular automata

The clustering algorithm groups the finite elements based on the eigenstrain prescribed during the inverse problem resolution. After clustering, each finite element in the peening pattern is assigned with a peening regime. The elements assigned with the same regime can be physically situated in different parts of the plate and thus form separate segments. In practice, however, the minimal size of the segments is limited by the shot stream width or the masking resolution. For example, a segment consisting of only one element is not typically reproducible in practice. The presence of such small-scale segments in the clustered pattern is called the *checkerboard* problem, which is illustrated in Figure 4. This problem arises when the eigenstrain prescribed by the inverse problem solver is on the border between two clusters, so the elements assigned with different peening regimes are mixed together after clustering.

We cope with the checkerboard problem with a filtering algorithm, which is executed after clustering for the top and bottom patterns separately. More precisely, we use an image noise filter based on cellular automata (CA) described in Ref. [34]. The filter is initially conceived for images consisting of square pixels that are assigned with a gray level, but we reformulate it for the case of triangular meshes where each element is assigned with a peening regime. The filtering algorithm scans the elements and notes the regimes assigned to each element and to its three neighbors, i.e., to its triangular Von Neumann neighborhood [35]. If the algorithm detects that the element

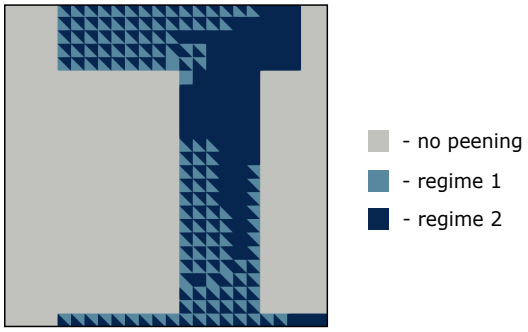

Figure 4. Example of a peening pattern heavily affected by the checkerboard problem after clustering. The pattern is represented on a square plate, which is meshed with triangular finite elements. The elements assigned with different peening regimes of a close intensity are locally mixed, so the proposed pattern is not reproducible in practice.

is mostly surrounded by elements prescribed with another regime, then it assigns the element with the regime that constitutes the majority. In case when none of the peening regimes constitutes a majority in the local neighborhood, then the element is assigned with the regime corresponding to its first neighbor. The same principle applies to the edge elements that have only two neighbors. The corner elements are strictly assigned with the regime corresponding to their single neighbor. The filtering algorithm is iterative and lasts until the stabilization of the assigned regimes. All triangular elements are scanned and corrected simultaneously at each iteration. This filter can also be applied after the grouping of the peening pattern [3], because the grouped patterns can also face the checkerboard problem.

The filtering algorithm is summarized as Algorithm 2. We formulate the algorithm in terms of the peening regime ρ_e assigned to element e. The regimes assigned to the neigboring elements are denoted as ρ_{e1}, ρ_{e2}, ρ_{e3}.

4. Results

The clustering and the filtering algorithms were validated numerically using 200 random target shapes. For each test case, we firstly solved the inverse problem using the algorithm presented in Ref. [3] and obtained the *free* peening pattern. Secondly, we applied the clustering algorithm (Algorithm 1) and thus obtained the *clustered* peening pattern. Thirdly, we applied the filtering algorithm (Algorithm 2) and obtained the *filtered clustered* peening pattern. Finally, we simulated the application of each of these three patterns using the method presented in Ref. [3] to evaluate discrepancy between this and that. In other words, we solved the forward problem for each pattern and thus obtained three final shapes. We computed the dimensionless error Ω in each case as the Hausdorff distance between the final shape and the target shape divided by the square root of the total area of the plate.

The target shapes were generated as a result of the application of random peening patterns on a 1×1 m square plate. The plate thickness and the Poisson's ratio were randomly assigned in each test case. The plates were meshed with 1152 triangular elements, and the random peening patterns were generated on this mesh using the algorithm presented in Ref. [3]. The first 100 test cases (series 1) involved only one available peening regime, which was applied in randomly situated rectangular segments of the plate. The

Algorithm 2 The filtering algorithm

while the eigenstrain is reassigned at least for one element **do**
  for each finite element e **do**
    if the element has three neighbors e_1, e_2, e_3, **then**
      if $\rho_e \neq \rho_{e1}$ and ($\rho_{e1} = \rho_{e2}$ or $\rho_{e1} = \rho_{e3}$) **then**
        Assign the value of ρ_{e1} to ρ_e
      else if $\rho_e \neq \rho_{e2}$ and $\rho_{e2} = \rho_{e3}$ **then**
        Assign the value of ρ_{e2} to ρ_e
      else if
        ($\rho_e \neq \rho_{e1}$ and $\rho_e \neq \rho_{e2}$ and $\rho_e \neq \rho_{e3}$) and
        ($\rho_{e1} \neq \rho_{e2}$ and $\rho_{e1} \neq \rho_{e3}$ and $\rho_{e2} \neq \rho_{e3}$) **then**
        Assign the value of ρ_{e1} to ρ_e
      end if
    end if
    if the element has two neighbors e_1, e_2, **then**
      if $\rho_e \neq \rho_{e1}$ and $\rho_e \neq \rho_{e2}$ **then**
        Assign the value of ρ_{e1} to ρ_e
      end if
    end if
    if the element has one neighbor e_1, **then**
      if $\rho_e \neq \rho_{e1}$ **then**
        Assign the value of ρ_{e1} to ρ_e
      end if
    end if
  end for
end while

second 100 cases (series 2) involved four available regimes applied in random segments.

Figure 5 illustrates the numerical validation workflow using one test case from series 2. The presented clustered pattern (Figure 5 d) leads to the highest Ω in series 2 because the pattern is significantly affected by the checkerboard problem. The filtered clustered pattern (Figure 5 e) contains irregularly shaped peening segments, but it is not subjected to the checkerboard problem, so it is applicable in practice.

Figure 6 summarizes the validation results in terms of Ω. The histograms demonstrate that the dimensionless error did not exceed 0.35% for the final shapes provided by the free patterns. Both clustering and filtering algorithms

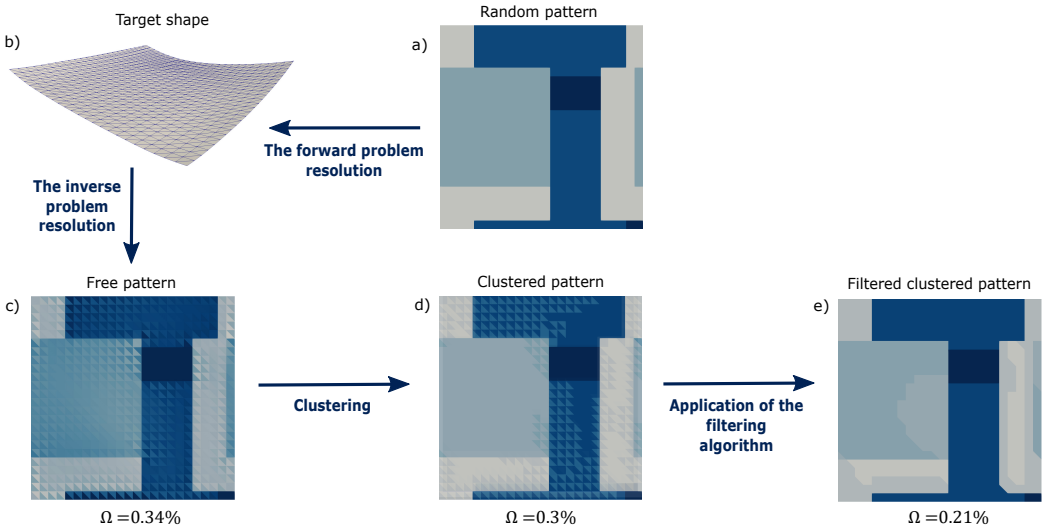

Figure 5. One of the test cases from series 2 that demonstrated a marked checkerboard problem. a) The random peening pattern generated on a 1×1 m sized plate using the algorithm presented in Ref. [3]. Different colors on the peening pattern denote different peening regimes. b) The target shape generated by application of the random pattern. The plate is 6 mm thick, and the Poisson's ratio equals 0.36. The deflections are at the original scale. c) The free pattern computed using the inverse problem resolution algorithm (see Ref. [3]). d) The clustered peening pattern computed by application of Algorithm 1 to the free pattern. e) The filtered clustered algorithm obtained by application of Algorithm 2 to the clustered pattern. We solved the forward problem for the patterns c), d) and e) and computed the dimensionless error Ω for each case. In the presented case, the filtered clustered pattern reproduces the original random pattern better than the free pattern, so it leads to a lower dimensionless error Ω.

made a slightly negative impact in terms of Ω, but Ω remained inferior to 0.4% for all test cases. Moreover, Ω remained inferior to 0.1% for more than 80% of cases in series 1 and for more than 70% of cases in series 2 after filtering.

During the clustering phase, we fixed the number of available regimes as one for series 1 and as four for series 2. No other information on the peening regimes was provided to the clustering algorithm, and the centroid positions were not constrained. The number of trials for the clustering algorithm was fixed as 100. The influence of clustering in terms of Ω is plotted in Figure 7 a). Thus, the clustered pattern decreased Ω with respect to the free pattern for 78 cases in series 1 and for 32 cases in series 2. The decrease in Ω in these cases is explained by the segmented structure of the random patterns that were used to generate the target shapes. Hence, the clustering removed the gradual eigenstrain variation from the free patterns and made the clustered patterns more similar to the original ones.

On the other hand, the difference in the influence of clustering on the two series is explained by the fact the clustered patterns in series 1 were less subjected to the checkerboard problem. Indeed, the only available peening regime was situated far from the origin in terms of $(\varepsilon^t, \varepsilon^b)$, so the majority of elements was assigned to the correct clusters. The main source of error in this case was the peening regime itself, which was slightly different from the peening regime applied in the original random pattern. On the other hand, the regimes in series 2 were situated close to each other, so the quantity of neighboring elements that were assigned to different clusters was more significant. Also, the four computed peening regimes did not exactly correspond to the original regimes.

It should be noted that although we fixed the quantity of peening regimes N for each series of tests, in practice N is not always pre-determined. In a such case, the clustering algorithm can be launched several times for different N, and the best suitable N can be determined based on the minimal Ω. A larger N provides more flexibility in terms of target shapes but extends the time needed for application of the pattern in practice.

The filtered clustered pattern increased Ω with respect to the clustered pattern for 77 cases in series 1 and for 67 cases in series 2, which is shown in Figure 7 b). The reason for this increase in most of the cases is that the filtering algorithm often enlarges the segments assigned to wrong clusters instead

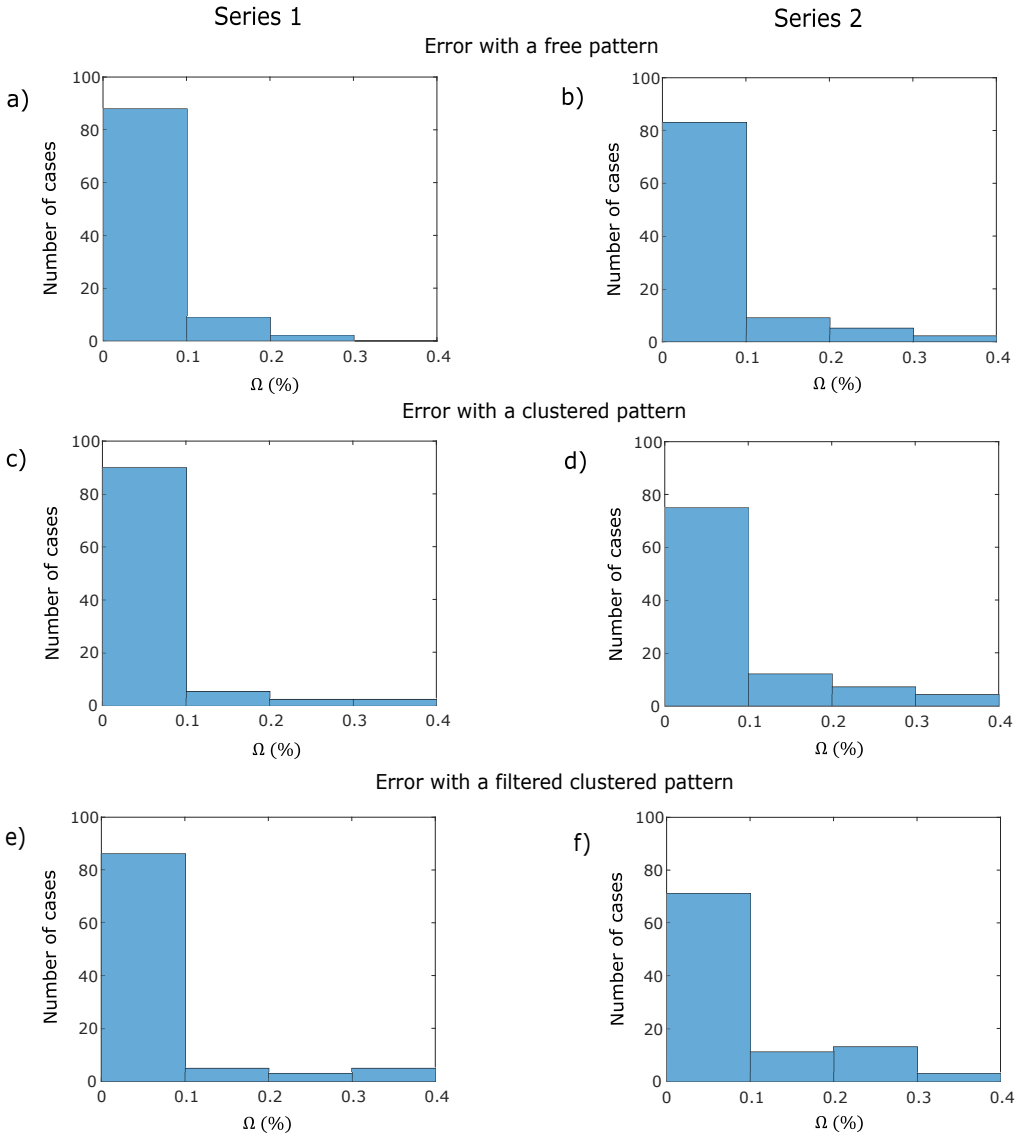

Figure 6. Histograms plotting the dimensionless error Ω between the target shapes and the final shapes computed during the validation of the clustering and filtering algorithms. We considered 200 test cases that were divided in two series of 100 cases each. In each case, we solved the inverse problem, clustered the peening pattern and applied the filtering algorithm. One peening regime was available for clustering in series 1, and four regimes were available in series 2. We simulated the application of the peening patterns at each stage and computed the dimensionless error Ω. Figures (a) and (b) plot the simulated Ω provided by the unclustered free patterns in series 1 and 2 respectively. Figures (c) and (d) trace the simulated Ω provided by the clustered patterns, and Figures (e) and (f) show Ω provided by the filtered clustered patterns. The filtered clustered patterns induce a slightly larger Ω than the free patterns in general, but Ω does not exceed 0.4% for all the types of patterns.

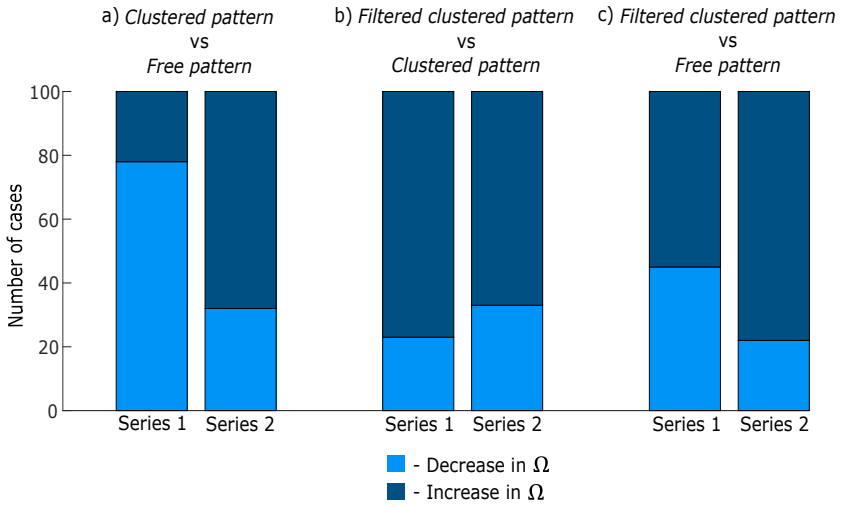

Figure 7. Influence of the clustering and the filtering algorithms on the dimensionless error Ω. The clustered patterns were obtained by application of the clustering algorithm to the free patterns, and the filtered clustered patterns were obtained by application of the filtering algorithm to the clustered patterns. Hence, the bars a) represent the influence of the clustering algorithm, the bars b) represent the influence of the filtering algorithm, and the bars c) represent the influence of the successive application of the two algorithms. The first series of tests implied one peening regime during generation of the training examples and clustering, while the second series implied four peening regimes. Both series consisted of 100 test cases. The results show that the influence of the successive application of the two algorithms is majorly negative.

of eliminating them. All in all, the successive application of the clustering and the filtering algorithms increased Ω with respect to the free pattern for more than a half of the test cases (see Figure 7 c). This is explained by the fact that both algorithms do not measure Ω, so their output depends only on the input pattern. One potential way to cope with this problem is to implement a black-box optimization strategy with respect to Ω, but this iterative approach would reduce the computational speed.

5. Conclusion

We developed a general strategy for the segmentation of the peening pattern, which makes every pattern applicable with shot peening equipment. The segmentation strategy computes the best suitable peening regimes and the segments where they should be applied. It includes a clustering algo-

rithm, which is inspired by the k-means method, and a filtering algorithm. The clustering algorithm is designed for the bilayer eigenstrain formulation, and any other formulation of the trough-thickness eigenstrain profile can be reduced to the bilayer. When applied consecutively, the clustering and the filtering algorithms remove from the pattern all eigenstrain variations that are not reproducible in practice. Hence, the strategy provides a ready-to use peening pattern, which is suitable for industrial reproduction using masking or a narrow peening stream.

The clustering and the filtering algorithms were validated numerically using the existing bilayer shell model. The validation showed that the clustering algorithm had a weak influence on the quality of the peening pattern, so that the original and the clustered patterns induced similar final shapes. A clustered pattern involving one peening regime is less affected by the checkerboard problem than this involving multiple regimes, but a large number of regimes provides more flexibility in terms of target shapes. The filtering also has a slight but majorly negative influence on the final shape. However, it efficiently deals with the checkerboard problem and rapidly eliminates local clustering errors.

The described segmentation strategy can be applied as a post-processing tool for any eigenstrain-based inverse problem solver that deals with laminated shape-shifting structures. Such structures are present in the fields of Micro-Electro-Mechanical Systems (MEMS) or 4D printed composites. Moreover, metal plates shaped with laser peening or English wheel can also be virtually represented as laminates and modeled using the eigenstrain approach. The segmentation strategy eliminates the need for trial-and-error determination of the best suitable eigenstrains and thus accelerates the inverse problem resolution.

6. Acknowledgments

The authors gratefully acknowledge financial support from The Natural Sciences and Engineering Research Council of Canada (NSERC), from The Fonds de Recherche du Québec – Nature et Technologies (FRQNT) and The Ministère de l'Économie et de l'Innovation du Québec, from The Aluminium Research Centre – REGAL, and from the industrial partner of this project – Aerosphere Inc. The authors also acknowledge the consulting support provided by Aerosphere Inc.

7. Funding and Conflicts of interests/Competing interests

This study was funded by:

- The Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference numbers RGPIN-2019-07072 and RGPIN-06412-2016);
- The Fonds de Recherche du Québec – Nature et Technologies (FRQNT) and The Ministère de l'Économie et de l'Innovation du Québec (funding reference number LU-210888);
- The Aluminium Research Centre – REGAL;
- Aerosphere Inc.

The authors have no relevant financial or non-financial interests to disclose. The authors have no conflicts of interests/competing interests to declare that are relevant to the content of this article.

Appendix A. k-means clustering of the trialyer eigenstrain pattern

Another common way to idealize the through-thickness eigenstrain profile is to represent it as a trilayer $\gamma_{tri}(z)$. The trilayer formulation implies the assignment of eigenstrain to the two outer layers of a variable thickness:

$$\gamma_{tri}(z) = \begin{cases} \varepsilon_*^t & \text{for} \quad \left(\frac{h}{2} - h_*^t\right) < z < \frac{h}{2}, \\ 0 & \text{for} \quad \left(h_*^b - \frac{h}{2}\right) < z < \left(\frac{h}{2} - h_*^t\right), \\ \varepsilon_*^b & \text{for} \quad -\frac{h}{2} < z < \left(h_*^b - \frac{h}{2}\right), \end{cases} \tag{A.1}$$

where ε_*^t and ε_*^b stand for the eigenstrains assigned to the top and bottom layers respectively, while h_*^t and h_*^b denote the thicknesses of these layers. Hence, in this formulation, the eigenstrain assigned to each finite element e is described with four variables $(\varepsilon_{*e}^t, h_{*e}^t, \varepsilon_{*e}^b, h_{*e}^b)$.

The range of peening regimes that can be applied from the top and bottom sides is the same, so the eigenstrains assigned to the top and bottom layers are clustered simultaneously. This means that each element generates two

points on a coordinate plane with axes ε_* and h_*. The coordinates of these points are $(\varepsilon^t_{*e}, h^t_{*e})$ and $(\varepsilon^b_{*e}, h^b_{*e})$. In this case, the points can be clustered using the standard `k`-means algorithm [31], except for the fact that the zero centroid $(0, 0)$ corresponding to the absence of treatment is fixed at the origin (Algorithm 3). The presence of this centroid suggests that N peening regimes actually induce $\mathtt{k} = N + 1$ cluster centroids. The other N centroids are initialized using the *Maxmin* method, which proves itself as the most efficient initialization technique for the `k`-means algorithm [32]. With this method, the centroids are initialized in sequence, and each subsequent centroid is placed at the location of the furthest point from the previous centroid. The first centroid in this sequence is the zero centroid.

Algorithm 3 The clustering algorithm in the trilayer case

for each finite element e **do**
    Initialize two points with coordinates $(\varepsilon^t_{*e}, h^t_{*e})$ and $(\varepsilon^b_{*e}, h^b_{*e})$
end for
Initialize the zero centroid at the origin
Initialize N centroids at random positions in coordinates (ε_*, h_*) using the Maxmin method
while at least one centroid changes its position **do**
    for each point **do**
        Compute the squared Euclidean distance to each centroid
        Assign the point to the cluster formed by the closest centroid
    end for
    for each cluster except for the zero cluster **do**
        Compute the mean of all points in the cluster
        Assign the mean as the new cluster centroid
    end for
end while
for each finite element e **do**
    Assign the top layer with a couple (ε^t_*, h^t_*) corresponding to the centroid of the cluster containing point $(\varepsilon^t_{*e}, h^t_{*e})$
    Assign the bottom layer with a couple (ε^b_*, h^b_*) corresponding to the centroid of the cluster containing point $(\varepsilon^b_{*e}, h^b_{*e})$.
end for

References

[1] D. Baughman, Peen forming, Machine Design 42 (1970) 156–160.

[2] S. Ramati, S. Kennerknecht, G. Levasseur, Single piece wing skin utilization via advanced peen forming technologies, Proceedings of the 7th International Conference on Shot Peening (ICSP7), Warsaw (1999).

[3] V. Sushitskii, W. M. van Rees, M. Lévesque, F. Gosselin, Determination of optimal shot peen forming patterns using the theory of non-euclidean plates (2021).

[4] R. VanLuchene, J. Johnson, R. Carpenter, Induced stress relationships for wing skin forming by shot peening, Journal of Materials Engineering and Performance 4 (1995) 283–290.

[5] R. VanLuchene, E. Cramer, Numerical modeling of a wing skin peen forming process, Journal of materials engineering and performance 5 (1996) 753–760.

[6] R. Kopp, J. Schulz, Optimising the double-sided simultaneous shot peen forming, Proceedings of the 8th International Conference on Shot Peening (ICSP8), Garmisch-Partenkirchen (2002) 227–233.

[7] T. Wang, M. Platts, J. Wu, The optimisation of shot peen forming processes, Journal of Materials Processing Technology 206 (2008) 78–82.

[8] A. Gariépy, J. Cyr, A. Levers, C. Perron, P. Bocher, M. Lévesque, Potential applications of peen forming finite element modelling, Advances in Engineering Software 52 (2012) 60–71.

[9] Y. Essa, M. Laspalas, E. Garcia, A. Escolán, B. Hernández-Gascón, F. Martin de la Escalera, Numerical analysis to optimize peen-forming process parameters, Berlin, Germany, 2015, pp. 1–10.

[10] T. Mura, Micromechanics of defects in solids. vol. 3, Springer Science & Business Media 580 (1987) 21.

[11] A. M. Korsunsky, The modelling of residual stresses due to surface peening using eigenstrain distributions, The Journal of Strain Analysis for Engineering Design 40 (2005) 817–824.

[12] S. Zhang, A. Venter, W. Vorster, A. Korsunsky, High-energy synchrotron X-ray analysis of residual plastic strains induced in shot-peened steel plates, The Journal of Strain Analysis for Engineering Design 43 (2008) 229–241.

[13] S. Coratella, M. Sticchi, M. Toparli, M. Fitzpatrick, N. Kashaev, Application of the eigenstrain approach to predict the residual stress distribution in laser shock peened AA7050-t7451 samples, Surface and Coatings Technology 273 (2015) 39–49.

[14] Z. Chen, F. Yang, S. Meguid, Realistic finite element simulations of arc-height development in shot-peened almen strips, Journal of Engineering Materials and Technology 136 (2014).

[15] T. Chaise, J. Li, D. Nélias, R. Kubler, S. Taheri, G. Douchet, V. Robin, P. Gilles, Modelling of multiple impacts for the prediction of distortions and residual stresses induced by ultrasonic shot peening (USP), Journal of Materials Processing Technology 212 (2012) 2080–2090.

[16] H. Y. Miao, M. Lévesque, F. P. Gosselin, Shot peen forming pattern optimization to achieve cylindrical and saddle target shapes: The inverse problem, CIRP Journal of Manufacturing Science and Technology 36 (2022) 67–77.

[17] W. Siguerdidjane, F. Khameneifar, F. P. Gosselin, Efficient planning of peen-forming patterns via artificial neural networks, Manufacturing Letters 25 (2020) 70–74.

[18] T. Wang, J. Platts, A. Levers, Finite element impact modelling for shot peen forming, Proceedings of the 8th international conference on shot peening, Garmisch-Partenkirchen, Germany (2002).

[19] A. M. Korsunsky, Residual elastic strain due to laser shock peening: modelling by eigenstrain distribution, The Journal of Strain Analysis for Engineering Design 41 (2006) 195–204.

[20] M. Luo, Y. Hu, L. Hu, Z. Yao, Efficient process planning of laser peen forming for complex shaping with distributed eigen-moment, Journal of Materials Processing Technology 279 (2020) 116588.

[21] K. Fann, Finite element study on forming metal sheets with an english wheel, in: IOP Conference Series: Materials Science and Engineering, volume 1222, IOP Publishing, 2022, p. 012006.

[22] T. Wang, M. Platts, A. Levers, A process model for shot peen forming, Journal of Materials Processing Technology 172 (2006) 159–162.

[23] J. M. Pajot, K. Maute, Y. Zhang, M. L. Dunn, Design of patterned multilayer films with eigenstrains by topology optimization, International journal of solids and structures 43 (2006) 1832–1853.

[24] W. M. van Rees, E. Vouga, L. Mahadevan, Growth patterns for shape-shifting elastic bilayers, Proceedings of the National Academy of Sciences 114 (2017) 11597–11602.

[25] W. M. van Rees, E. A. Matsumoto, A. S. Gladman, J. A. Lewis, L. Mahadevan, Mechanics of biomimetic 4D printed structures, Soft matter 14 (2018) 8771–8779.

[26] P. A. Faucheux, Simulating shot peen forming with eigenstrains, Phd dissertation, Polytechnique Montreal, 2019.

[27] D. Kirk, Shot peening, Aircraft engineering and aerospace technology (1999).

[28] K. Hornauer, W. Kohler, Development of the peen forming process for spherical shaped components, in: Proceedings of the 4th International Conference on Shot Peening (ICSP4), Tokyo, volume 500, 1990, pp. 585–594.

[29] L. Rokach, O. Maimon, Clustering methods, in: Data mining and knowledge discovery handbook, Springer, 2005, pp. 321–352.

[30] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2015) 165–193.

[31] S. Lloyd, Least squares quantization in pcm, IEEE transactions on information theory 28 (1982) 129–137.

[32] P. Fränti, S. Sieranoja, How much can k-means be improved by using better initialization and repeats?, Pattern Recognition 93 (2019) 95–112.

[33] L. D. Hoffmann, G. L. Bradley, K. H. Rosen, Calculus for business, economics, and the social and life sciences, McGraw-Hill, 2004.

[34] P. J. Selvapeter, W. Hordijk, Cellular automata for image noise filtering, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009, pp. 193–197.

[35] M. Zawidzki, Application of semitotalistic 2D cellular automata on a triangulated 3D surface, International Journal of Design & Nature and Ecodynamics 6 (2011) 34–51.