Research Article

Fast Point and Element Search Method in Adaptive Remeshing Procedure and Its Applications

Jie Zhang, Abel Cherouat, and Houman Borouchaki

*Gamma3 Project Team (ICD/GAMM3), Charles Delaunay Institute, University of Troyes,
12 Marie Curie street, BP2060, 10010 Troyes Cedex, France*

Correspondence should be addressed to Abel Cherouat, abel.cherouat@utt.fr

Received 5 April 2011; Accepted 3 May 2011

Academic Editor: X. Xue

Copyright © 2011 Jie Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The FPES (fast point and element search) method is a useful and efficient strategy for node field transfer from old mesh to the new mesh in adaptive remeshing procedure. The FE mesh after adaptive remeshing with various error estimates will be refined at local region, and the mesh after adaptive remeshing has the heterogeneous density distribution. The FPES has the capacity to define the nearest search path adapting to the mesh with heterogeneous density distribution. It is a point location process which includes point searching, point location in element, and weight factor distribution. This strategy has been integrated to our finite element adaptive remeshing simulations, and it works well and rapidly. The three-dimension finite element numerical simulation of simply tensile test, orthogonal cutting, and metal milling process is given out to study its accuracy and efficiency.

1. Introduction

Many physical phenomena in science and engineering can be modeled by partial differential equations and solved by means of the finite element method. In R^3 vectorial problems, the accuracy of numerical results and the amount of computational memory are critical, especially if quantitative results are required [1]. Adaptive remeshing procedure based on various prior and posterior error estimations is proposed by different researchers to control the mesh quality and then improve the accuracy of the numerical simulations. For some complex nonlinear problems, we even need to adaptive re-mesh numbers of times to ensure discrete model has the fine mesh. Johnson and Hansbo [2] has discussed the applications of adaptive finite element methods in commotional mechanics. Pittalá and Monno [3] used two dimension adaptive remeshing procedure to simulate the metal orthogonal cutting process. Some 3D advance adaptive remeshing procedure have been shown in thermal-mechanic simulations [4, 5] and electromagnetic simulations [6]. Field transfer of state variable is

a necessary procedure and one of the key technologies in adaptive remeshing procedure. The transfer accuracy affects numerical results directly, and advanced transfer algorithm can reduce computing time dramatically.

Generally, state variables are stored at both nodal and integration points of element. In [7], the displacement, strain tensor, plastic strain tensor, and stress tensor are transferred in 2D small-strain rate-independent elasto-plasticity adaptive remeshing procedure. The state variables are transferred with Gauss point from old mesh to the new one. In the work [8], the transfer process executes from old Gauss points to the new one in 2D ductile fracture simulations. To our adaptive remeshing scheme, we define a state array $\Lambda_i(S_{11}^i, S_{22}^i, S_{33}^i, S_{12}^i, S_{23}^i, S_{13}^i, \bar{\epsilon}_p^i, t^i)$. The first seven are state variables stored on Gauss point and we transferred directly using advanced interpolation method during adaptive remeshing procedure. The eighth state variable is temperature on node. If we transfer the nodal field relying on Gauss points, it will fetch in some errors easily. Firstly, shape function is used to transfer field from nodes to Gauss points in old mesh. Secondly, the adaptive remeshing procedure implements and field transfer from old mesh to the Gauss points of new mesh. Finally, the temperature on Gauss points which share the node are querying. The field on this node is obtained through average value of all fields on relative Gauss points. It is obviously that these three necessary processes affect transfer accuracy directly. So, a directly transfer scheme from node-to-node is requisite to our adaptive remeshing procedure. In this paper, we focus on node field (temperature for example) transfer from old mesh to the new one. The fast point and element search (FPES) code is developed to implement our node field transfer accurately and quickly.

The querying efficiency is the major problem to be considered in FPES algorithm, especially, in 3D topology spatial with heterogeneous mesh density. The proposed FPES algorithm has the ability of defining the search path adapt to the spatial mesh density. It uses vector calculation to judge the spatial position of the nodes. After locating, distance-based weight factor replace volume calculation to compute the field value on nodes. These operations improve the transfer efficiency greatly and ensure the transfer accuracy strictly. In this paper, three main processes of FPES algorithm are presented which include point-searching, point location in element and weight factor distributing. For testing, the proposed fast point and element search algorithm is integrated into our 3D finite element adaptive remeshing procedure. The numerical applications of simply tensile test, orthogonal cutting and metal milling process are given out to study its accuracy and efficiency. The interpolation method is also proposed to transfer nodal field for comparing the transfer accuracy with FPES method.

2. Fast Point and Element Search Algorithm

FPES (fast point and element search) method executes node-to-node field transfer accurately and quickly. We consider iteration i in adaptive remeshing procedure, $V_{(m,i+1)}$ is the value of state variable at node $N_{(m,i+1)}$ in new mesh H_{i+1} , $V_{(n,i)}$ is the value of state variable at node $N_{(n,i)}$ and $E_{(n,i)}$ is a element in old mesh H_i . The point search algorithm in FPES locates the node $N_{(m,i+1)}$ on $N_{(n,i)}$, whose node position have not been changed after remshing. The point position judgment in element algorithm locates the node $N_{(m,i+1)}$ in detected element $E_{(n,i)}$, whose node position have been changed after remshing. The weight factor process will distribute the weight factors for $N_{(n,i)}$ which relate to $N_{(m,i+1)}$.

2.1. Point Search Algorithm

Some classical searching algorithms, such as counter clock wise [9] and barycentric coordinates search [10], can define a fast search path in R^2 space and have extended in R^3 space for querying points in the space constructed by finite element nodes and elements. These algorithms strongly rely on neighbor elements of each element. When millions of elements exist or not easy to obtain the information of neighbor elements, container-based nearest-neighbor-lookup must be considered, such as k - d trees [11, 12] and uniform grid [13] search algorithm. The main problem is defined as follows: given a container (fixed set of N points) and a query position P in R^n , find the point P in the container that has the minimum Euclidean distance from point P . To our adaptive remeshing procedure, the querying space has the heterogeneous mesh density distribution and we have to define search path adapting to the mesh density. In this paper, we define the containers adapted to the mesh density. Indeed, we make the range (upper and lower bound) of the container adapting to the mesh density, and the capacity of the container is given and fixed. This operation for point search algorithm makes querying time descends drastically in our problems.

There are two process in point search algorithm: preprocessing and querying process. Preprocessing read the mesh space and create adaptive container, include:

- (i) define adaptive orientation (X , Y , or Z). It is better to chose the orientation through the mesh density changing, X orientation is chosen in our program for example,
- (ii) store all nodes label and its x coordinates value, then sort it,
- (iii) define container capacity (N , number of nodes in container), then store these N nodes in every container and calculate x range (upper and lower bound for x) using sorted x list. We note C_i as the label of containers and M_1 as the total number of containers which are created by x coordinates,
- (iv) based on C_i containers, create new containers through both y and z directions. Every x containers will be divided into $2 \cdot n$ sub container and every sub container have its capacity of $N/2 \cdot n$. We note C_i^s as the label of the sub containers. In this paper, $n = 2$ and $M = 2 \cdot 2 \cdot M_1$ containers obtained at last. These containers have the range in all directions.

Point querying process will locate $P(x, y, z)$ in defined container rapidly, it includes:

- (i) locate $P(x, y, z)$ in C_i with x coordinate. It can query speedy using defined X which follows the changing of mesh density,
- (ii) based on C_i , locate $P(x, y, z)$ in C_i^s which defined by its y and z range respectively,
- (iii) searching $P(x, y, z)$ in C_i^s which has $N/2 \cdot n$ points.

Some point querying tests have been done to test the querying efficiency. Testing in the environment of 2.5 GHz CPU and 1.98 GB memory for example, when we query one point in the mesh space with 21354 nodes and 109371 elements, 462 ms need for preprocessing and 0.284 ms (it is the average value when we search 10000 points in 2840 ms) need for point querying process. To querying one point, it is obviously that the preprocessing occupy almost all of the searching time. But when millions of points querying, the preprocessing time can be neglected. From this point, it is very suitable for large-scale finite element analysis coupled with adaptive remeshing procedure.

Figure 1: Time searching in different mesh spaces with the increasing number of nodes.

Figure 2: Point searching in various mesh space.

This is the first step of FPES algorithm and some advanced work have been done to research the relationship between querying efficiency and different distributions of mesh density. Figure 1 gives out the calculating time for preprocessing Figure 1(a) and point querying process Figure 1(b), respectively. With the increasing of searching size (N_t : total number of mesh nodes), the time-searching in both preprocessing and querying process increase.

In our adaptive remeshing procedure, the mesh density is controlled by two error estimations, especially the prior estimation. After changing the prior estimation parameter, different mesh spaces with the increasing number of nodes and changed distribution of mesh density are generated to study the relationship between querying efficiency and different distributions of mesh density. Firstly, we define the container factor $R = N/N_t$ for each mesh space, where N is the container capacity. Then, we can obtain various R through the different given values of N . Finally, the querying times for one point are calculated with various R in a mesh space. The mesh spaces with different N_t are calculated and shown in Figure 2. It is obvious that the container factor R affects the querying time greatly, as shown in Figure 2(a). The values r which need the shortest querying time will be obtained. The curve

Figure 3: The judgement of the point in element.

of the relationship between r and N_i is fitted by exponentiation interpolation function, as shown in Figure 2(b). Based on this equation, the algorithm firstly calculates the optimized container capacity N and the highest querying efficiency are reached finally.

2.2. Point Position Judgment in Element

Some points of new mesh are located in the points of old mesh and others will be located in elements of old mesh. The general method of judging the relation of spatial location between point and tetrahedron is volumetric discriminance. Two kinds of volume must be calculated with this method, which include: the volume of tetrahedron; the sub-tetrahedron which formed by the point and surface of tetrahedron. When volumetric summation of sub tetrahedrons equals to the tetrahedron volume, the point will be judged in tetrahedron. For large-scale calculating, this method is obviously limited by its calculating efficiency. In this paper, the vectorial calculation is proposed to judge the point in element.

In our mesh space, the tetrahedral elements are constructed by four nodes which have the given orders. Four regular surfaces are defined to make the normal of surfaces point at the the same direction (out or in element). As in Figure 3(a), we consider a tetrahedral element which is connected by N_1 , N_2 , N_3 , and N_4 . Surface F_3 , formed by N_2 , N_3 , and N_4 for example, define the normal vector $\mathbf{n}_3$ which is perpendicular to F_3 and point at out of element. P is the point which need to be judged in this element. Focusing on P and F_3 as in Figure 3(b), $\mathbf{m}$ is the vector from N_2 to point P , and the cosine of angle between $\mathbf{n}_3$ and $\mathbf{m}$ can be calculated as follows:

$$\cos \alpha = \frac{\mathbf{m} \cdot \mathbf{n}_3}{|\mathbf{m}| \times |\mathbf{n}_3|}, \quad \mathbf{n}_3 = N_2N_3 \times N_2N_4,$$

$$p = \begin{cases} \text{upside of surface} & \text{when } \cos \alpha < 0, \\ \text{on surface} & \text{when } \cos \alpha = 0, \\ \text{downside of surface} & \text{when } \cos \alpha > 0. \end{cases} \quad (2.1)$$

Based on (2.1), the necessary condition of point P locating in tetrahedron is that the cosine values of F_1 , F_2 , F_3 , and F_4 are all negative. Further, the point P locates on surface when there is only one $\cos \alpha = 0$ appearing and locates on edge when there are two $\cos \alpha = 0$ existing. A special condition must be considered when some nodes are out of the mesh space because of the adaptive remeshing procedure. Deal with this problem, we detected the nearest tetrahedron which has only one $\cos \alpha > 0$ and others are $\cos \alpha < 0$. Approximately, this node will be located on the nearest surface which satisfy $\cos \alpha > 0$.

In brief, the algorithm locates point P at the mesh space with four conditions. Firstly, P will be located at point using proposed point-searching algorithm. When p can not be located at point, the nearest points will be queried and sorted in S_n through the distance between P and points. Subsequently, P will be found in a element which includes in the elements set S_e , using above judgement method. S_e is the set of elements which connect to the nodes in S_n . Finally, p locates at surface or edge can be also categorized. The searching efficiency of first condition have been discussed in point search algorithm. The efficiency of judging point in element, surface and edge has not relation with mesh density and is difficult to modeling But actually, it need more computing time for this judging process and there are some searching examples in Section 3.

2.3. Weight Factor

All of the nodes $N_{(m,i+1)}$ in new mesh are located in old mesh like point P . The distance-based weight factors for $N_{(n,i)}$ in old mesh which relate to the queried node, tetrahedron, edge and surface are calculated as follows:

- (i) locate at node $N_{(n,i)}$ and weight factor for $k = 1$ node,
- (ii) locate at edge of element $E_{(n,i)}$ and weight factor for $k = 1, 2$ nodes,
- (iii) locate at surface of element $E_{(n,i)}$ and weight factor for $k = 1, 2, 3$ nodes,
- (iv) locate in element $E_{(n,i)}$ and weight factor for $k = 1, 2, 3, 4$ nodes.

We note α_k as the distance-based weight factor for node k and d_k is the distance between $N_{(m,i+1)}$ and k . When the related nodes number is n ($n = 1, 2, 3, 4$) and j is any one of related nodes, then α_k is defined as

$$\alpha_k = \begin{cases} 1 & \text{when } n = 1, \\ \frac{1 - d_k / \sum_{j=1}^n d_j}{n - 1} & \text{when } n > 1. \end{cases} \quad (2.2)$$

Finally, the state variable $V_{(m,i+1)}$ in new mesh can be transferred from old mesh like this:

$$V_{(m,i+1)} = \sum_{k=1}^n \alpha_k V_{(k,i)}. \quad (2.3)$$

In this paper, we are concentrated in searching and locating the points at the R^3 space constructed by numbers of nodes and tetrahedrons. But, the algorithm is a dependent code which does not rely on the adaptive remeshing strategy. It provides a stable and accurate node-to-node transfer scheme. After optimizing, it can well adapt to mesh density in various

Figure 4: 3D finite element simulation coupled with adaptive remeshing procedure.

mesh space and prompt the efficiency directly. So, the algorithm can be extended and applied to other problems. For us, the proposed algorithm is as the part of source code included in FPES.py and it serves the node filed transfer in adaptive remeshing procedure.

3. Adaptive Remeshing Procedure

The adaptive remeshing procedure integrates ABAQUS/Explicit solver, OPTIFORM adaptive mesher and field transformations, as shown in Figure 4. Borouchaki et al. [14] makes great contributions to this adaptive remeshing mesher. The proposed procedure is widely used in elastic-plastic-damage [4, 5] and electromagnetic [1, 6] FE simulations. The initial geometry and mesh are prepared before script executing. When script running, the whole analysis is divided into several iterations to execute the remeshing step by step. At each iteration, the meshes are refined and coarsed automatically; the physical fields (such as stress, equivalent plastic strain and temperature) are transferred from old to the new mesh using advanced algorithm; the elements will be killed when damage occurring.

In this paper, FPES algorithm is focused on transferring the temperature filed from old mesh to the new one. It is a dependent code which just relates to the spatial position of the nodes. Element killing and new boundary reconstructing in adaptive remeshing procedure have no effects on FPES. By contrary, the state variables, like temperature, will be removed following with the killed elements. This makes FPES become a useful tool of field transfer for simulating the metal removing process.

4. Test Results

In this section, the transfer accuracy of FPES algorithm is discussed through a simply tensile test. Some comparatione of transfer accuracy are presented using orthogonal cutting test. At last, the transfer efficiency is described through metal milling test.

Figure 5: Temperature transfer for simply tensile test using FPES code.

4.1. Transfer Accuracy in Simply Tensile Test

A simply tensile test using adaptive remeshing procedure is given out to research the transfer accuracy of FPES method. Initially, all of the nodes give the temperature of 200°C . As in Figure 5(a) shows us the temperature distribution in iteration 2 of tensile test before remeshing. The mesh has only 12 nodes and 20 tetrahedral elements. Figure 5(b) shows us

the temperature distribution after the adaptive remeshing and field transfer for iteration 2. The refined mesh includes 84 nodes and 271 tetrahedral elements. We control the adaptive remeshing parameter as $h_{\min} = 25$ mm to remeshing this $50 \text{ mm} \times 50 \text{ mm} \times 100 \text{ mm}$ specimen at this iteration. Figure 5(c) is the calculated results of iteration 2 and the temperature field before field transfer of iteration 3. We do adaptive remshing and field transfer with $h_{\min} = 15$ mm at iteration 3 as shown in Figure 5(d). The same operation is done for iteration 4 with $h_{\min} = 10$ mm as shown in Figures 5(e) and 5(f).

Comparing the temperature distributions before and after temperature transfer using FPES code in iteration 2, the temperature on 12 nodes in Figure 5(a) are all transferred to the nodes in Figure 5(b) which are searched no movement of spatial position (includes nodes from N_1 to N_{12}). To these nodes, the temperature values are obtained from old nodes directly. The nodes which are added into the mesh during adaptive remeshing are searched in the edges (9 nodes), surfaces (30 nodes) and elements (33 nodes) of mesh before adaptive remeshing. The distance-based weight factor are calculated for these nodes. Finally, the temperature values of these interpolated nodes (includes nodes from N_{13} to N_{84}) are computed with the weight of nodes in the edges, surfaces, and elements, respectively.

4.2. Transfer Accuracy Comparing with Interpolation Method

The interpolation method is also proposed to compare the transfer accuracy with FPES method. Three steps are included in node filed transfer using interpolation function. Firstly, shape function (as in equation) is used to transfer node field from nodes to interpolation points of element in the mesh before adaptive remeshing. Secondly, as one field of elements, the temperature are introduced into adaptive remeshing process and transferred to the interpolation points of new mesh. At last, the temperature on nodes are calculated in two parts: searching all of the elements which share this node; averaging the temperature on interpolation points of these elements. The shape function used is shown in (4.1), where T_e is temperature on element, T_i is temperature on node which belong to this element. Tetrahedral element has $n = 4$ nodes and $N_i = 1/4$ for each nodes

$$T_e = \sum_{i=1}^n N_i T_i. \quad (4.1)$$

The adaptive remeshing procedure for metal orthogonal cutting are presented to compare the transfer accuracy between FPES method and interpolation method. Initial, the workpiece has the very coarse mesh of 736 deformed tetrahedral elements. The cutting tool has 636 rigid hexahedral elements and only temperature degree is available. During the remeshing iterations, the mesh size of workpiece is controlled by geometrical and physical error estimations through the adaptive discrete parameters: $h_{\min} = 0.1$ mm and $h_{\max} = 2.0$ mm. The mesh of tool keeps on its shape and size without remeshing. In this paper, the Johnson-cook model [15] is considered to simulate the material thermal-mechanical behavior. The fractured elements are killed when Johnson-cook damage occurs. Figure 7(a) is orthogonal cutting process and chip shape at iteration 240 with cutting depth of 6.2 mm. Figure 6(a) is the temperature distribution at iteration 10 before adaptive remeshing, and Figure 6(b) are the temperature distribution after field transfer for new mesh with FPES method. Figures 6(c) and 6(d) is the same picture using proposed interpolation method for temperature

Figure 6: Iteration 10: temperature transfer for orthogonal cutting using both FPES method and interpolation method.

Table 1: Temperature lost using both FPES and interpolation methods.

i	T_1	T_2	T_{lost}	T'_1	T'_2	T'_{lost}	Element kill
1	41.97	41.97	0%	42.23	29.56	30.00%	No
10	176.51	176.51	0%	148.68	86.02	27.10%	No
50	806.75	806.75	0%	413.20	184.45	55.36%	Yes
100	1220.76	1220.76	0%	760.36	197.07	74.08%	Yes
150	519.12	519.12	0%	512.12	183.96	64.08%	Yes
200	502.83	484.10	3.72%	884.36	215.11	75.68%	Yes

transferring. It is obviously that the temperature values have no lost using FPES method and temperature values have 27.10% loss using proposed interpolation method.

Some other iterations, shown in Table 1, the maximum temperature value before (T_1 for FPES transfer and T'_1 for interpolation transfer) and after (T_2 for FPES transfer and T'_2 for interpolation transfer) field transfer iterations are given out. We can see that the temperature values have no lost after transferring using FPES method when there is no element killing. The temperature also have a 100% transfer when the element included maximum temperature is not moved, like iteration 100 and 150. There are 3.72% temperature value lost at iteration 200. It is because that the damaged elements are killed and they include the node which have maximum temperature. To comparing, there are huge lost of maximum temperature with interpolation method at both element killing and no killing. And the higher the maximum temperature the more temperature lost. The results have proved that FPES algorithm have its advantages in node field transfer whether there exists element killing or not.

Figure 7: Adaptive remeshing procedure and temperature transfer in orthogonal cutting: left, and metal up milling: right.

4.3. Transfer Efficiency in Simulation of Metal Milling

The up (conventional) milling adaptive remeshing procedure is presented to study the transfer efficiency of FPES algorithm. Initial, the workpiece has the very coarse mesh of 276 deformed tetrahedral elements. The milling tool has 2160 rigid hexahedral elements and only temperature degree is available. The finite meshes are refined and coarsen automatically in adaptive remeshing procedure. For this model, the adaptive discrete parameters are configured as: $h_{\min} = 0.05$ mm and $h_{\max} = 2.0$ mm. The mesh of tool keeps on its shape and size without remeshing also. Figure 7(b) is milling finite element model at iteration 150 which have milling depth of 3.0 mm.

In Table 2, field transfer efficiency using FPES code is illustrated. For example, at iteration 100, there are 9017 nodes and 39536 elements in mesh space. The optimized capacity

Table 2: Efficiency of node field transfer using FPES in different iterations.

i	N_t	E_t	N	S_{node}	$T_p(s)$	S_{edge}	S_{surf}	S_{elem}	$T_{\text{total}}(s)$
50	7946	37549	172	6697	0.999	4	79	1166	9.396
100	9017	39536	180	7286	1.142	7	139	1585	10.941
150	11644	51065	208	10143	1.540	3	149	1349	11.352
200	14456	62221	232	12340	2.019	7	174	1935	15.847

of container is 180. There are 7286 nodes are located at nodes in old mesh within 1.142 s. There are 7, 139 and 1585 nodes are located at edges, on surface and in element respectively within 9.799 s. The total searching time in this iteration is 10.941 s. During transfer process, almost all of the time is occupied by locating the nodes in elements of old mesh. This is the necessary time because we give the querying process a strict constraint for ensuring the transfer accuracy. For large finite element analysis, like metal forming process, this time for field transfer is acceptable.

5. Conclusion

The FPES (fast point and element search) method is a useful and efficient strategy to node field transfer from old mesh to the new mesh in adaptive remeshing procedure. The algorithm includes point-searching, point location in element and weight factor distributing. The nearest search path which adapts to the heterogeneous density distribution of mesh is defined. And the vector calculation and distance-based weight factor are used to improve querying efficiency and accuracy. The three-dimension finite element simulation of simply tensile test, orthogonal cutting and metal milling process is illustrated to discuss the transfer efficiency and accuracy.

References

- [1] H. Borouchaki, T. Grosgees, and D. Barchiesi, "Improved 3D adaptive remeshing scheme applied in high electromagnetic field gradient computation," *Finite Elements in Analysis and Design*, vol. 46, no. 1-2, pp. 84–95, 2010.
- [2] C. Johnson and P. Hansbo, "Adaptive finite element methods in computational mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 101, no. 1–3, pp. 143–181, 1992.
- [3] G. M. Pittalà and M. Monno, "A new approach to the prediction of temperature of the workpiece of face milling operations of Ti-6Al-4V," *Applied Thermal Engineering*, vol. 31, no. 2-3, pp. 173–180, 2011.
- [4] A. Cherouat, L. Moreau, and H. Borouchaki, "Advanced numerical simulation of metal forming processes using adaptive remeshing procedure," *Materials Science Forum*, vol. 614, pp. 27–33, 2009.
- [5] A. Cherouat, H. Borouchaki, K. Saanouni, and P. Laug, "Numerical methodology for metal forming processes using elastoplastic model with damage occurrence," *Journal of Materials Science and Technology*, vol. 22, no. 2, pp. 279–283, 2006.
- [6] G. Thomas, B. Houman, and B. Dominique, "Three dimensional adaptive remeshing scheme applied to the control of the spatial representation of complex feld pattern in electromagnetic," *Applied Physics B*, vol. 1, pp. 883–889, 2010.
- [7] D. Perić, Ch. Hochard, M. Dutko, and D. R. J. Owen, "Transfer operators for evolving meshes in small strain elasto-plasticity," *Computer Methods in Applied Mechanics and Engineering*, vol. 137, no. 3-4, pp. 331–344, 1996.
- [8] J. Mediavilla, R. H. J. Peerlings, and M. G. D. Geers, "A robust and consistent remeshing-transfer operator for ductile fracture simulations," *Computers and Structures*, vol. 84, no. 8-9, pp. 604–623, 2006.

- [9] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of voronoi," *ACM Transactions on Graphics*, vol. 4, pp. 74–123, 1985.
- [10] R. Sundareswara and P. Schrater, "Extensible point location algorithm," in *Proceedings of the International Conference on Geometric Modeling and Graphics (GMAG '03)*, London, UK, July 2003.
- [11] M. Murphy and S. S. Skiena, Ranger-nearest neighbor search in higher, 1996.
- [12] S. S. Skiena, *The Algorithm Design Manual-Nearest Neighbor Search*, Springer, New York, NY, USA, 2001.
- [13] W. R. Franklin, "Nearest point query on 184,088,599 points in 3 dimension with a uniform grid," Tech. Rep., 2006.
- [14] H. Borouchaki, A. Cherouat, P. Laug, and K. Saanouni, "Adaptive remeshing for ductile fracture prediction in metal forming," *Comptes Rendus—Mecanique*, vol. 330, no. 10, pp. 709–716, 2002.
- [15] G. R. Johnson and W. H. Cook, "Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures," *Engineering Fracture Mechanics*, vol. 21, no. 1, pp. 31–48, 1985.