2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Construction of Bounding Volume Hierarchies for Triangle Meshes with Mixed Face Sizes

Yi Li[1], Evan Shellshear[1], Robert Bohlin[1] and Johan S. Carlson[1]

***Abstract*— We consider the problem of creating tighter-fitting bounding volumes (more specifically rectangular swept spheres) when constructing bounding volume hierarchies (BVHs) for complex 3D geometries given in the form of unstructured triangle meshes/soups with the aim of speeding up our IPS Path Planner for rigid bodies, where the triangles often have very different sizes. Currently, the underlying collision and distance computation module (IPS CDC) does not take into account the sizes of the triangles when it constructs BVHs using a top-down strategy. To split triangles in a BVH node into two BVH nodes, IPS CDC has to compute both the split axis and the split position. In this work, we use the principal axes of the tensor of inertia as the potential split axes and the center of mass as the split position, where the computations of both the tensor of inertia and the center of mass require knowledge of the areas of the triangles. We show that our method improves performance (up to $20\,\%$ faster) of our IPS Path Planner when it is used to plan collision-free disassembly paths for three different test cases taken from manufacturing industries.**

I. Introduction

When design engineers working in manufacturing industries are designing a mechanical part, they may have to modify the design numerous times before it is approved for production. After each modification, the engineers have to ensure that this part can be assembled (i.e., fitting it together with other parts) and disassembled (i.e., extracting it out). The manual generation of detailed assembly/disassembly paths is both tedious and time-consuming. Our IPS Path Planner can be used to solve this problem during the virtual prototyping process to compute collision-free paths for part assembly/disassembly, where the part in question (hereinafter referred to as the rigid body) has six degrees of freedom (DOF) and the other parts are considered to be stationary obstacles (hereinafter referred to as the obstacles). During the planning process, we do not take into account kinematic and dynamic constraints on the rigid body's motion (i.e., the simpler problem of path planning is addressed instead of the problem of motion planning).

In order to speed up IPS Path Planner so that it can synthesize collision-free paths for rigid bodies faster, IPS CDC has to be as fast as possible. We assume in this paper that all mechanical parts are given in the form of unstructured triangle meshes/soups. To perform collision and distance queries on 3D models approximated by triangle meshes, some of the most efficient algorithms use Bounding Volume Hierarchies (BVH) as their acceleration structure, because BVHs are simple to construct and have a low memory consumption [1]. For example, RAPID [2] and PQP [3] use Oriented Bounding Box (OBB) trees [4] and Rectangular Swept Sphere (RSS) trees [5], receptively. IPS CDC also uses RSS trees. We refer the readers to [6] for an excellent overview of different collision or proximity queries and the underlying algorithms.

The bounding volume (BV) of a BVH node can almost never fit the primitives (i.e., triangles in this paper) inside the BV perfectly, because a good BV is the one that provides a good trade-off between tightness and speed of proximity query. We want BVs to fit the original primitives as tightly as possible. At the same time, a collision or proximity query on a pair of such BVs should be done as fast as possibly. Consequently, the leafs and internal nodes of a BVH often overlap substantially, especially when the 3D models of the parts contain large triangles with high side length ratios [7]. In this paper, we present a simple solution that speeds up the traversal of BVHs, and hence rigid-body path planning, by taking into account the size of triangles when constructing a BVH. We chose not to subdivide large triangles into smaller and uniform ones in a preprocess step because these additional triangles will lead to larger tree size and hence slower performance.

The rest of the paper is organized as follows. A short description of BVH construction in IPS CDC is given in Section II. We describe related work in Section III. In Section IV, we explain in detail how to compute the principal axes of the tensor of inertia (to be used as the potential split axes) and the center of mass (to be used as the split position). The experimental results, as well as a discussion of the results, are given in Section V. We conclude in Section VI.

II. Background

IPS Path Planner is a rigid body path planner. The configuration space ($\mathcal{C}$-space) of a 3D rigid body is identified with the Lie group $SE(3)$, the special Euclidean group in three-dimensions. IPS Path Planner searches the configuration space for a collision-free path (for a 3D rigid body) that connects a start configuration to a goal configuration. Unlike sampling-based path planning algorithms (e.g., Probabilistic Roadmap (PRM) [8] and Rapidly-exploring Random Tree (RRT) [9]) that compute a path by connecting independently and identically distributed random points in the configuration space, IPS Path Planner employs a deterministic path planning method so that it produces the same path every time.

[1]Yi Li, Evan Shellshear, Robert Bohlin, and Johan S. Carlson are with the Geometry and Motion Planning Department, Fraunhofer-Chalmers Centre for Industrial Mathematics, SE-412 88 Gothenburg, Sweden (e-mail address of the corresponding author: yi.li@fcc.chalmers.se)

IPS CDC constructs a BVH (as a binary tree) by recursively dividing a given list of triangles into disjoint sets. There are many kinds of BVs such as Axis-Aligned Bounding Box (AABB), Oriented Bounding Box (OBB) [4], and Rectangle Swept Spheres (RSS) [5], where a RSS is a volume covered by a sphere whose center is swept over a 3D rectangle. We chose RSS for IPS CDC, because it is not only more suitable for distance queries compared to OBB, but also performs well for collision queries [5], even though a RSS BVH can be slower to construct than an AABB BVH. Despite the fact that RSSs provide a tight fit to the underlying geometry, the RSSs of the sets of primitives (e.g., triangles) can still have arbitrary overlap, and hence slow down the tree traversal. This is especially true when a set contains one or multiple triangles with a high ratio between triangle side lengths.

A. BVH Construction

Given a list of n triangles, IPS CDC encloses them in BVs and groups the nested BVs into a BVH. Since IPS CDC uses a top-down strategy (i.e., it builds the BVH from the root node downward, where the root node contains all the triangles), the triangles in each node of the tree are split into two subsets. IPS CDC subdivides the nodes recursively until the subsets contain only a single triangle and the corresponding nodes are the leaf nodes of the BVH.

IPS CDC utilizes the same statistical techniques used by [10], [4], [5] to split triangles and compute BVs (i.e., defining the split point with the mean μ of vertex coordinates and using the covariance matrix $\mathbf{C}$ of the vertices for the split axis). The mean and covariance matrix are defined as:

$$\mu = \frac{1}{3n}\sum_{i=1}^{n} \mathbf{p}^i + \mathbf{q}^i + \mathbf{r}^i, \tag{1}$$

$$\mathbf{C}_{jk} = \frac{1}{3n}\sum_{i=1}^{n} \bar{\mathbf{p}}_j^i \bar{\mathbf{p}}_k^i + \bar{\mathbf{q}}_j^i \bar{\mathbf{q}}_k^i + \bar{\mathbf{r}}_j^i \bar{\mathbf{r}}_k^i, 1 \leq j,k \leq 3 \tag{2}$$

where $\mathbf{p}^i$, $\mathbf{q}^i$, $\mathbf{r}^i$ represent the vertices of the i'th triangle and $\bar{\mathbf{p}}^i = \mathbf{p}^i - \mu$, $\bar{\mathbf{q}}^i = \mathbf{q}^i - \mu$, $\bar{\mathbf{r}}^i = \mathbf{r}^i - \mu$. IPS CDC uses the mean μ and one of the three eigenvectors of the covariance matrix $\mathbf{C}$ as the split position and the split axis, respectively. The split plane is then defined as the plane that passes through the split position and is orthogonal to the split axis. Whenever the projection of a triangle's centroid onto the split axis is less than the mean μ, the triangle is placed in the left subset. The other triangles are placed in the right subset.

The surface area heuristic (SAH) [11], [12] cost function is widely used to estimate the cost of a particular split. The basic idea behind the SAH is to recursively separate a 3D volume into two sub-volumes by splitting the triangles in such a way as to minimize a cost function based on the sub-volumes' surface areas. The following SAH cost function is taken from [13] and is used in ray tracing scenarios to minimize the probability of a ray colliding with a given bounding volume:

$$K = K_T + K_I\left(\frac{SA(V_L)}{SA(V)}N_L + \frac{SA(V_R)}{SA(V)}N_R\right) \tag{3}$$

where K denotes the cost of dividing the volume V into two halves (i.e., L and R), K_T is the cost of one traversal step through the hierarchy, K_I is the cost of a single triangle proximity test, $SA(V)$ is the surface area of V, and N_L and N_R are the number of triangles in the left and right halves, respectively.

However, IPS CDC uses the following simpler but more computationally efficient cost function instead of (3):

$$K = SA(V_L) + SA(V_R) \tag{4}$$

which does not evaluate $SA(V)$. In fact, our experiments [14] show that this simple cost function allows for faster tree traversal than simply splitting based on the largest volume of the BV or many other more complicated splitting heuristics. IPS CDC always uses μ as the split position in order to speed up BVH construction.

III. Related Work

Given a collection of n triangles, IPS CDC computes the covariance matrix $\mathbf{C}$ by only taking into account the positions of the triangle vertices and then evaluates the three eigenvectors of $\mathbf{C}$ as potential split axes as shown in Section II-A. This procedure takes $O(n)$ time.

Instead of working on the given triangles directly, the $O(n \log n)$ time algorithm presented in [4], [15] computes the convex hull of the vertices of the triangles first and then uses the triangles of the resulting convex hull and their areas to compute the covariance matrix.

To do this, one firstly computes the surface area of the convex hull A_H

$$A_H = \textstyle\sum_i A^i, \tag{5}$$

where A^i is the area of the i'th triangle of the convex hull, which, given vertices $\mathbf{p}^i, \mathbf{q}^i$ and $\mathbf{r}^i$, can be calculated as

$$A^i = \tfrac{1}{2}|(\mathbf{p}^i - \mathbf{q}^i) \times (\mathbf{p}^i - \mathbf{r}^i)| \tag{6}$$

Secondly, we compute the centroid of the convex hull $\mathbf{m}_H$ as a weighted mean of the triangle centroids

$$\mathbf{m}_H = \frac{\sum_i A^i \mathbf{m}^i}{\sum_i A^i} = \frac{\sum_i A^i \mathbf{m}^i}{A_H}, \tag{7}$$

where $\mathbf{m}^i$ is the centroid of the i'th triangle of the convex hull

$$\mathbf{m}^i = \tfrac{1}{3}(\mathbf{p}^i + \mathbf{q}^i + \mathbf{r}^i) \tag{8}$$

Finally, the covariance matrix $\mathbf{C}'$ is defined as

$$\mathbf{C}'_{jk} = \sum_{i=1}^{n} \frac{A^i}{12}[9\mathbf{m}_j^i \mathbf{m}_k^i + \mathbf{p}_j^i \mathbf{p}_k^i + \mathbf{q}_j^i \mathbf{q}_k^i + \mathbf{r}_j^i \mathbf{r}_k^i] - \mathbf{m}_{H_j} \mathbf{m}_{H_k} A_H, \tag{9}$$

where the subscripts refer to which coordinate (i.e., x, y, or z) is being taken. The eigenvectors of $\mathbf{C}'$ are the potential split axes.

As shown in [15], this procedure takes $O(n)$ time if we skip the convex hull step and work directly with the original triangles. Unfortunately, this may lead to slower tree traversal resulting from bad triangle distributions leading to bad $\mathbf{C}'$ matrices.

IV. Construction of BVHs for Triangle Meshes with Mixed Triangle Sizes

Given two objects and their triangle meshes/soups, IPS CDC has to construct one BVH for each object before it is able to perform collision or distance computations to determine whether these two objects collide or if the minimum distance between them is less than a given threshold.

Currently, IPS CDC uses the mean μ and one of the three eigenvectors of the covariance matrix $\mathbf{C}$ as the split position and the split axis, respectively. As shown in (1) and (2), these two equations do not take into account the areas of the triangles when computing μ and $\mathbf{C}$. Instead of using the eigenvectors of $\mathbf{C}$ as the potential split axes, we propose using the principal axes of the tensor of inertia which is defined as

$$\begin{aligned}\mathbf{I}_{jk} = \frac{1}{3n}\sum_{i=1}^{n}\Big(&A^i(\mathbf{p}_j^i - CM_j)(\mathbf{p}_k^i - CM_k) + \\ &A^i(\mathbf{q}_j^i - CM_j)(\mathbf{q}_k^i - CM_k) + \\ &A^i(\mathbf{r}_j^i - CM_j)(\mathbf{r}_k^i - CM_k)\Big), \\ &1 \le j,k \le 3\end{aligned} \tag{10}$$

where $\mathbf{p}^i$, $\mathbf{q}^i$, $\mathbf{r}^i$ represent the vertices of the i'th triangle, A^i is the area of the triangle, and CM is the center of mass of the object approximated by a triangular mesh (with n triangles). In this paper, we assume that all physical objects have uniform density and we approximate the center of mass of each object with the centroid of the object's shell (i.e., the centroid of the object's triangle mesh). To compute the centroid of a triangle mesh, Algorithm 1 can be used.

Algorithm 1: The centroid of a triangular mesh

Input: A mesh with n triangles $T = \{t^1, t^2, \ldots, t^n\}$
Output: The centorid of the triangular mesh

1 $A_{sum} \leftarrow 0$ `// the sum of areas`
2 $centroid \leftarrow \mathbf{0}$
3 **for** $i \leftarrow 1$ ***to*** n **do**
4 $\quad \mathbf{p}^i$, $\mathbf{q}^i$, $\mathbf{r}^i \leftarrow$ the three vertices of t^i
5 $\quad center \leftarrow \frac{1}{3}(\mathbf{p}^i + \mathbf{q}^i + \mathbf{r}^i)$
6 $\quad A^i \leftarrow \frac{1}{2}|(\mathbf{p}^i - \mathbf{q}^i) \times (\mathbf{p}^i - \mathbf{r}^i)|$
7 $\quad centroid \leftarrow centroid + A^i \cdot center$
8 $\quad A_{sum} \leftarrow A_{sum} + A^i$
9 $centroid \leftarrow \frac{1}{A_{sum}} centroid$
10 **return** $centroid$

Next, we find the eigenvectors ($\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$) and eigenvalues (λ_1, λ_2, and λ_3) of the tensor, where the eigenvectors are sorted according to their eigenvalues (i.e., $\lambda_1 \ge \lambda_2 \ge \lambda_3$). Consequently, $\mathbf{e}_1$ has the highest eigenvalue λ_1, whereas $\mathbf{e}_3$ has the lowest eigenvalue λ_3. The eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are the principal axes of the tensor of inertia.

We use these results in our IPS CDC by replacing μ with CM as the split position, where CM is equal to $centroid$ as computed in Algorithm 1 (line 9). In addition, we also replace the eigenvectors of $\mathbf{C}$ with the principal axes (i.e., $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$) as the potential split axes.

When $centroid$ is finalized in Algorithm 1, the inverse of A_{sum} is used, and hence Algorithm 1 cannot cope with triangles that are degenerate or nearly degenerate (i.e., their surface area is very small compared to their edge lengths). Therefore, whenever A_{sum} is smaller than a predefined threshold ϵ, IPS CDC uses the mean and covariance matrix defined in (1) and (2), respectively, to compute the split position and the potential split axes.

The algorithm presented in this section requires two passes. The center of mass and the tensor of inertia are computed in the first pass and the second pass, respectively. Since both passes take $O(n)$ time, the entire procedure takes $O(n)$ time. In addition to the center of mass, we also compute the minimum triangle area A_{min} in the first pass, where $A_{min} \ge \epsilon$. If the area of a triangle is less than ϵ, it is ignored (i.e., A_{min} remains unchanged). In the second pass, we use $\frac{A^i}{A_{min}}$ instead of A^i when computing the tensor of inertia (i.e., replacing A^i with $\frac{A^i}{A_{min}}$ in (10)) in order to improve numerical stability.

The computation of a good $centroid$ is not just important for splitting the triangle mesh into submeshes, but it also plays a role in the fitting of a BV to a triangle mesh. This is because the split axis is used as the principle axis of the BV when fitting a RSS to the triangle mesh that is to be split. This process is presented in Fig. 1.

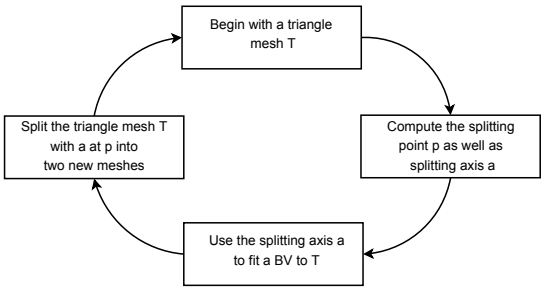

Fig. 1. The process of continual subdivision and fitting of triangle meshes during BVH construction.

In conclusion, the method we presented in this section is similar to the one in Section II. The differences are that we now use the centroid computed in Algorithm 1 as well as the axes computed from the tensor of inertia as defined by (10) and, most importantly, we now take into account the size of the triangles during the computations.

V. Experimental Results

In this paper, we have presented three different *fitters* to compute the split position and the split axis (which is used as the fitting axis too). These fitters are detailed in Section II, Section III, and Section IV, respectively. We name the fitter in Section II PQP_FIT, because it is used in both *PQP – A Proximity Query Package* [3] and IPS CDC. Next, the fitter in Section III is named RAPID2_FIT[1], because it is used in *RAPID – Robust and Accurate Polygon*

[1]The convex hull step presented in Section III is skipped and hence we work directly with the original triangles.

Interference Detection [2]. In fact, *PQP – A Proximity Query Package* implements both RAPID2_FIT and PQP_FIT with the second one as the default fitter. Finally, we name the fitter in Section IV INERTIA_FIT.

Currently, IPS CDC uses PQP_FIT. To demonstrate the effectiveness of our algorithm (i.e., INERTIA_FIT), we implemented it on top of IPS CDC. The code was written in C++ and compiled in Microsoft Visual Studio Professional 2017 (Version 15.9.12) on Windows 10 Pro 64 bit. The computer that we used contained an Intel® Core™ i7-7700K CPU @ 4.20 GHz (4 cores, 8 threads), a NVIDIA GeForce GTX 970 (4 GB) GPU, and 32 GB RAM. For comparison, we have also implemented RAPID2_FIT on top of IPS CDC.

We studied the performance of IPS Path Planner using IPS CDC with INERTIA_FIT by using it to solve three different test cases (TC) from the manufacturing industries: TC Air Cleaner Subassembly, TC Cylinder Head, and TC Tunnel Console. In each test case, parts to be disassembled are grouped into a single rigid body and the goal of IPS Path Planner is to compute a collision-free disassembly path for the rigid body as fast as possible. For comparison, we also solved these path planning problems using both IPS CDC with PQP_FIT and IPS CDC with RAPID2_FIT.

In TC Air Cleaner Subassembly, an air cleaner subassembly is located inside a metal frame made up of several round tubes when the subassembly is at the start configuration. The goal is to move the air cleaner subassembly from the start configuration to the goal configuration (outside the metal frame) while avoiding collision with the metal frame.

In TC Cylinder Head, a cylinder head is located between a cylinder block and a metal frame made up of several round tubes when the cylinder head is at the start configuration, where the cylinder head and the cylinder block are not in contact with each other since the cylinder head has already been lifted up from its assembled configuration. The cylinder head's goal configuration is located outside the metal frame. Consequently, the goal is to disassemble the cylinder head while avoiding collisions with both the cylinder block and the metal frame.

In TC Tunnel Console, the tunnel console has to be moved from the start configuration to the goal configuration as shown in Fig. 2 and Fig. 3, respectively, while avoiding collisions with the dashboard, the gear selector, the hand brake, the roof, and all other static geometries.

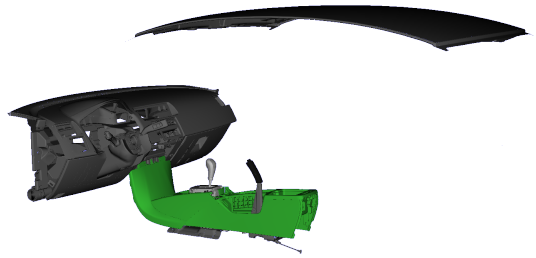

Fig. 2. TC Tunnel Console: The tunnel console (in green) is at the start configuration.

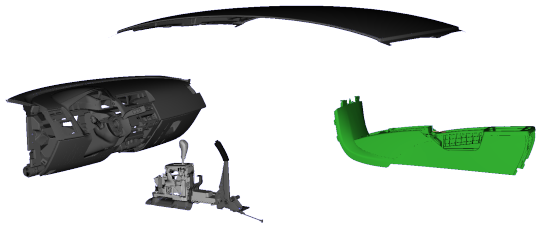

Fig. 3. TC Tunnel Console: The tunnel console (in green) is at the goal configuration.

The number of vertices and triangles in all 3D triangular meshes used by the three test cases are listed in Table I. Furthermore, the wireframe of the rigid body of TC Tunnel Console shown in Fig. 4 clearly demonstrate that meshes we have obtained from the manufacturing industries contain triangles with vastly different sizes and ratios between side lengths.

TABLE I

THE NUMBER OF VERTICES AND TRIANGLES IN ALL 3D TRIANGULAR MESHES USED BY THE THREE TEST CASES. TC AIR CLEANER SUBASSEMBLY†, TC CYLINDER HEAD†, AND TC TUNNEL CONSOLE† REPRESENT THE TRIANGULAR MESHES OF THE RIGID BODIES, WHEREAS TC AIR CLEANER SUBASSEMBLY‡, TC CYLINDER HEAD‡, AND TC TUNNEL CONSOLE‡ REPRESENT THE TRIANGULAR MESHES OF THE OBSTACLES.

3D Models	No. of Vertices	No. of Triangles
TC Air Cleaner Subassy†	797,076	1,253,476
TC Air Cleaner Subassy‡	290,433	415,378
TC Cylinder Head†	264,495	246,584
TC Cylinder Head‡	423,742	413,742
TC Tunnel Console†	65,997	73,469
TC Tunnel Console‡	288,484	300,567

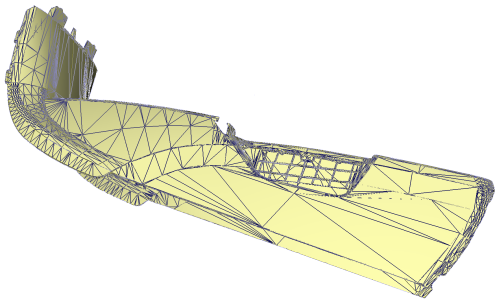

Fig. 4. TC Tunnel Console: Wireframe of the tunnel console.

After solving the three path planning problems (i.e., TC Air Cleaner Subassembly, TC Cylinder Head, and TC Tunnel Console) with IPS Path Planner using IPS CDC with different fitters (i.e., PQP_FIT, RAPID2_FIT, and INERTIA_FIT), the experimental running times and the speedups (compared to IPS Path Planner using IPS CDC with PQP_FIT) are listed in Table III. IPS Path Planner using IPS CDC with INERTIA_FIT is up to 20 % faster than IPS Path Planner using IPS CDC with PQP_FIT, whereas IPS Path Planner

TABLE II

TOTAL BV SURFACE AREA AND TOTAL BV VOLUME. SEE CAPTION OF TABLE I FOR ADDITIONAL EXPLANATION.

3D Models	PQP_FIT		RAPID2_FIT		INERTIA_FIT	
	Surface Area	Volume	Surface Area	Volume	Surface Area	Volume
TC Air Cleaner Subassy†	59.28	0.40	94.54	0.73	54.76	0.35
TC Air Cleaner Subassy‡	70.71	0.88	181.26	3.68	72.20	0.81
TC Cylinder Head†	23.92	0.14	33.14	0.22	22.13	0.13
TC Cylinder Head‡	125.72	1.73	187.59	3.41	111.69	1.34
TC Tunnel Console†	87.19	1.39	89.05	1.37	76.69	0.99
TC Tunnel Console‡	635.41	26.91	862.44	49.07	554.54	19.64

TABLE III

THE EXPERIMENTAL RUNNING TIMES (IN SECONDS) OF IPS PATH PLANNER USING IPS CDC WITH THREE DIFFERENT FITTERS. THE SPEEDUPS (COMPARED WITH PQP_FIT) ARE LISTED IN THE PARENTHESES.

Test Cases	PQP_FIT	RAPID2_FIT	INERTIA_FIT
Air Cleaner Subassy	21 (1.00)	77 (0.27)	19 (1.11)
Cylinder Head	161 (1.00)	403 (0.40)	151 (1.07)
Tunnel Console	47 (1.00)	68 (0.69)	40 (1.18)

TABLE IV

THE BVH BUILD TIMES (IN MILLISECONDS) OF IPS CDC WITH THREE DIFFERENT FITTERS. SEE CAPTION OF TABLE I FOR ADDITIONAL EXPLANATION.

3D Models	PQP_FIT	RAPID2_FIT	INERTIA_FIT
TC Air Cleaner Subassy†	3609	4069	4115
TC Air Cleaner Subassy‡	1024	1132	1145
TC Cylinder Head†	617	674	676
TC Cylinder Head‡	1058	1183	1174
TC Tunnel Console†	169	193	192
TC Tunnel Console‡	751	854	864

using IPS CDC with RAPID2_FIT is significantly slower than the other two. Moreover, we list both total BV surface area and total BV volume in Table II. Clearly, INERTIA_FIT is faster because it provides tighter fitting BVs than both PQP_FIT and RAPID2_FIT as can be seen by the lower volumes and surface areas in almost all cases. Moreover, IPS Path Planner using IPS CDC with INERTIA_FIT only takes slightly longer than the planner using IPS CDC with PQP_FIT to build BVHs as shown in Table IV.

VI. CONCLUSION

We have presented a method for creating tighter-fitting bounding volumes (more specifically RSSs) in order to speed up our IPS CDC module and IPS Path Planner for rigid bodies. The new method works with complex 3D geometries given in the form of unstructured triangle meshes/soups, which are commonly used in manufacturing industries. The speedup is achieved by taking into account the sizes of the triangles when computing the principal axes of the tensor of inertia (to be used as the potential split axes) and the center of mass (to be used as the split position). We have demonstrated the effectiveness of this new method by integrating it into IPS CDC and then use IPS Path Planner to plan collision-free disassembly paths for three different test cases taken from manufacturing industries.

ACKNOWLEDGMENTS

This work is part of the Sustainable Production Initiative and the Production Area of Advance at the Chalmers University of Technology. We would like to acknowledge the generosity of Volvo Cars in providing some models that were used for the experiments in this work.

REFERENCES

[1] M. Stich, H. Friedrich, and A. Dietrich, "Spatial splits in bounding volume hierarchies," in *Proceedings of the Conference on High Performance Graphics 2009*. ACM, 2009, pp. 7–13.

[2] "RAPID – Robust and Accurate Polygon Interference Detection," http://gamma.cs.unc.edu/OBB, 1997, [Online; accessed 7-July-2019].

[3] "PQP – A Proximity Query Package," http://gamma.cs.unc.edu/SSV, 1999, [Online; accessed 7-July-2019].

[4] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH 1996. ACM, 1996, pp. 171–180.

[5] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, vol. 4, 2000, pp. 3719–3726.

[6] Y. J. Kim, M. C. Lin, and D. Manocha, *Collision Detection*. Springer Netherlands, 2019, pp. 1933–1956.

[7] M. Ernst and G. Greiner, "Early split clipping for bounding volume hierarchies," in *Proceedings of 2007 IEEE Symposium on Interactive Ray Tracing*, 2007, pp. 73–78.

[8] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996.

[9] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[10] G. Barequet, B. Chazelle, L. J. Guibas, J. S. B. Mitchell, and A. Tal, "BOXTREE: A hierarchical representation for surfaces in 3d," *Comput. Graph. Forum*, vol. 15, pp. 387–396, 1996.

[11] J. Goldsmith and J. Salmon, "Automatic creation of object hierarchies for ray tracing," *IEEE Computer Graphics and Applications*, vol. 7, no. 5, pp. 14–20, May 1987.

[12] J. D. MacDonald and K. S. Booth, "Heuristics for ray tracing using space subdivision," *The Visual Computer*, vol. 6, no. 3, pp. 153–166, May 1990.

[13] I. Wald, "On fast construction of sah-based bounding volume hierarchies," in *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*. IEEE Computer Society, 2007, pp. 33–40.

[14] R. Ytterlid and E. Shellshear, "BVH split strategies for fast distance queries," *Journal of Computer Graphics Techniques*, vol. 4, no. 1, pp. 1–25, January 2015.

[15] S. Gottschalk, "Good-fit box for 3D triangles in $O(n \log n)$ time," http://gamma.cs.unc.edu/OBB/cov.ps, 1997, [Online; accessed 7-July-2019].